axiom™

The 30 Year Horizon

Manuel Bronstein *William Burge* *Timothy Daly*
James Davenport *Michael Dewar* *Martin Dunstan*
Albrecht Fortenbacher *Patrizia Gianni* *Johannes Grabmeier*
Jocelyn Guidry *Richard Jenks* *Larry Lambe*
Michael Monagan *Scott Morrison* *William Sit*
Jonathan Steinbach *Robert Sutor* *Barry Trager*
Stephen Watt *Jim Wen* *Clifton Williamson*

Volume 7.1: Axiom Hyperdoc Pages

Inclusion of names in the list of credits is based on historical information and is as accurate as possible. Inclusion of names does not in any way imply an endorsement but represents historical influence on Axiom development.

Michael Albaugh	Cyril Alberga	Roy Adler
Christian Aistleitner	Richard Anderson	George Andrews
S.J. Atkins	Henry Baker	Martin Baker
Stephen Balzac	Yurij Baransky	David R. Barton
Gerald Baumgartner	Gilbert Baumslag	Michael Becker
Nelson H. F. Beebe	Jay Belanger	David Bindel
Fred Blair	Vladimir Bondarenko	Mark Botch
Raoul Bourquin	Alexandre Bouyer	Karen Braman
Peter A. Broadbery	Martin Brock	Manuel Bronstein
Stephen Buchwald	Florian Bundschuh	Luanne Burns
William Burge	Ralph Byers	Quentin Carpent
Robert Caviness	Bruce Char	Ondrej Certik
Tzu-Yi Chen	Cheekai Chin	David V. Chudnovsky
Gregory V. Chudnovsky	Mark Clements	James Cloos
Jia Zhao Cong	Josh Cohen	Christophe Conil
Don Coppersmith	George Corliss	Robert Corless
Gary Cornell	Meino Cramer	Jeremy Du Croz
David Cyganski	Nathaniel Daly	Timothy Daly Sr.
Timothy Daly Jr.	James H. Davenport	David Day
James Demmel	Didier Deshommes	Michael Dewar
Jack Dongarra	Jean Della Dora	Gabriel Dos Reis
Claire DiCrescendo	Sam Dooley	Lionel Ducos
Iain Duff	Lee Duhem	Martin Dunstan
Brian Dupee	Dominique Duval	Robert Edwards
Heow Eide-Goodman	Lars Erickson	Richard Fateman
Bertfried Fauser	Stuart Feldman	John Fletcher
Brian Ford	Albrecht Fortenbacher	George Frances
Constantine Frangos	Timothy Freeman	Korrinn Fu
Marc Gaetano	Rudiger Gebauer	Van de Geijn
Kathy Gerber	Patricia Gianni	Samantha Goldrich
Holger Gollan	Teresa Gomez-Diaz	Laureano Gonzalez-Vega
Stephen Gortler	Johannes Grabmeier	Matt Grayson
Klaus Ebbe Grue	James Griesmer	Vladimir Grinberg
Oswald Gschnitzer	Ming Gu	Jocelyn Guidry
Gaetan Hache	Steve Hague	Satoshi Hamaguchi
Sven Hammarling	Mike Hansen	Richard Hanson
Richard Harke	Bill Hart	Vilya Harvey
Martin Hassner	Arthur S. Hathaway	Dan Hatton
Waldek Hebisch	Karl Hegbloom	Ralf Hemmecke

Henderson
Antoine Hersen
Roger House
Gernot Hueber
Pietro Iglio
Alejandro Jakubi
Richard Jenks
William Kahan
Kai Kaminski
Grant Keady
Wilfrid Kendall
Tony Kennedy
Ted Kosan
Paul Kosinski
Klaus Kusche
Bernhard Kutzler
Tim Lahey
Larry Lambe
Kaj Laurson
George L. Legendre
Franz Lehner
Frederic Lehobey
Michel Levaud
Howard Levy
Ren-Cang Li
Rudiger Loos
Michael Lucks
Richard Luczak
Camm Maguire
Francois Maltey
Alasdair McAndrew
Bob McElrath
Michael McGettrick
Edi Meier
Ian Meikle
David Mentre
Victor S. Miller
Gerard Milmeister
Mohammed Mobarak
H. Michael Moeller
Michael Monagan
Marc Moreno-Maza
Scott Morrison
Joel Moses
Mark Murray
William Naylor
Patrice Naudin
C. Andrew Neff
John Nelder
Godfrey Nolan
Arthur Norman
Jinzhong Niu
Michael O'Connor
Summat Oemrawsingh
Kostas Oikonomou
Humberto Ortiz-Zuazaga
Julian A. Padget
Bill Page
David Parnas
Susan Pelzel
Michel Petitot
Didier Pinchon
Ayal Pinkus
Frederick H. Pitts
Jose Alfredo Portes
Gregorio Quintana-Orti
Claude Quitte
Arthur C. Ralfs
Norman Ramsey
Anatoly Raportirenko
Albert D. Rich
Michael Richardson
Guilherme Reis
Huan Ren
Renaud Rioboo
Jean Rivlin
Nicolas Robidoux
Simon Robinson
Raymond Rogers
Michael Rothstein
Martin Rubey
Philip Santas
Alfred Scheerhorn
William Schelter
Gerhard Schneider
Martin Schoenert
Marshall Schor
Frithjof Schulze
Fritz Schwarz
Steven Segletes
V. Sima
Nick Simicich
William Sit
Elena Smirnova
Jonathan Steinbach
Fabio Stumbo
Christine Sundaresan
Robert Sutor
Moss E. Sweedler
Eugene Surowitz
Max Tegmark
T. Doug Telford
James Thatcher
Balbir Thomas
Mike Thomas
Dylan Thurston
Steve Toleque
Barry Trager
Themos T. Tsikas
Gregory Vanuxem
Bernhard Wall
Stephen Watt
Jaap Weel
Juergen Weiss
M. Weller
Mark Wegman
James Wen
Thorsten Werther
Michael Wester
R. Clint Whaley
John M. Wiley
Berhard Will
Clifton J. Williamson
Stephen Wilson
Shmuel Winograd
Robert Wisbauer
Sandra Wityak
Waldemar Wiwianka
Knut Wolf
Liu Xiaojun
Clifford Yapp
David Yun
Vadim Zhytnikov
Richard Zippel
Evelyn Zoernack
Bruno Zuercher
Dan Zwillinger

Contents

New Foreword

On October 1, 2001 Axiom was withdrawn from the market and ended life as a commercial product. On September 3, 2002 Axiom was released under the Modified BSD license, including this document. On August 27, 2003 Axiom was released as free and open source software available for download from the Free Software Foundation's website, Savannah.

Work on Axiom has had the generous support of the Center for Algorithms and Interactive Scientific Computation (CAISS) at City College of New York. Special thanks go to Dr. Gilbert Baumslag for his support of the long term goal.

The online version of this documentation is roughly 1000 pages. In order to make printed versions we've broken it up into three volumes. The first volume is tutorial in nature. The second volume is for programmers. The third volume is reference material. We've also added a fourth volume for developers. All of these changes represent an experiment in print-on-demand delivery of documentation. Time will tell whether the experiment succeeded.

Axiom has been in existence for over thirty years. It is estimated to contain about three hundred man-years of research and has, as of September 3, 2003, 143 people listed in the credits. All of these people have contributed directly or indirectly to making Axiom available. Axiom is being passed to the next generation. I'm looking forward to future milestones.

With that in mind I've introduced the theme of the "30 year horizon". We must invent the tools that support the Computational Mathematician working 30 years from now. How will research be done when every bit of mathematical knowledge is online and instantly available? What happens when we scale Axiom by a factor of 100, giving us 1.1 million domains? How can we integrate theory with code? How will we integrate theorems and proofs of the mathematics with space-time complexity proofs and running code? What visualization tools are needed? How do we support the conceptual structures and semantics of mathematics in effective ways? How do we support results from the sciences? How do we teach the next generation to be effective Computational Mathematicians?

The "30 year horizon" is much nearer than it appears.

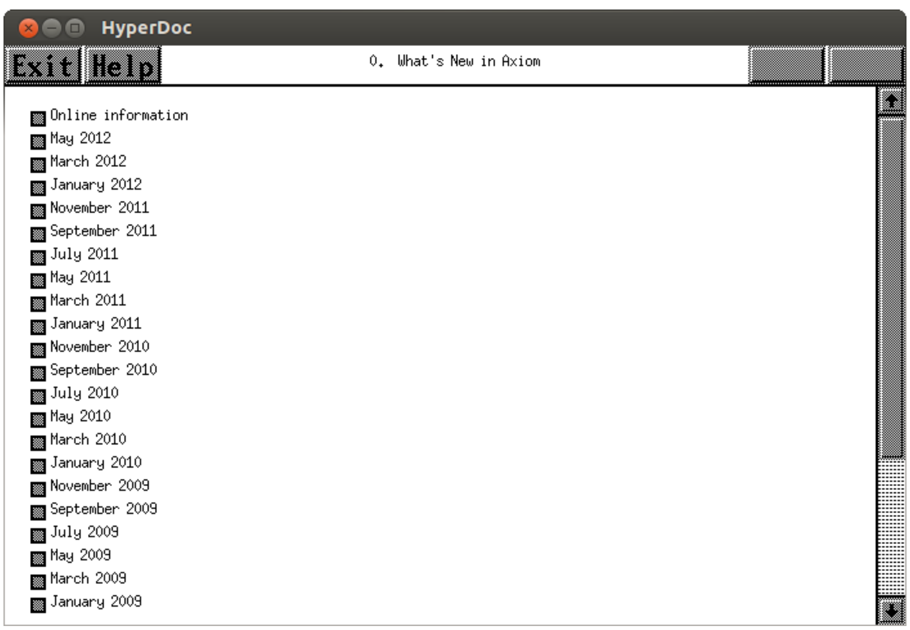
Tim Daly
CAISS, City College of New York
November 10, 2003 ((iHy))

Chapter 1

Release Notes

1.1 releasenotes.ht

What is new in Axiom



⇐ "Root Page" (RootPage) 3.1 on page 117
⇒ "Online information" (onlineInformation) 1.1 on page 3
⇒ "August 2014" (aug2014) 1.1 on page 4
⇒ "May 2012" (may2012) 1.1 on page 14

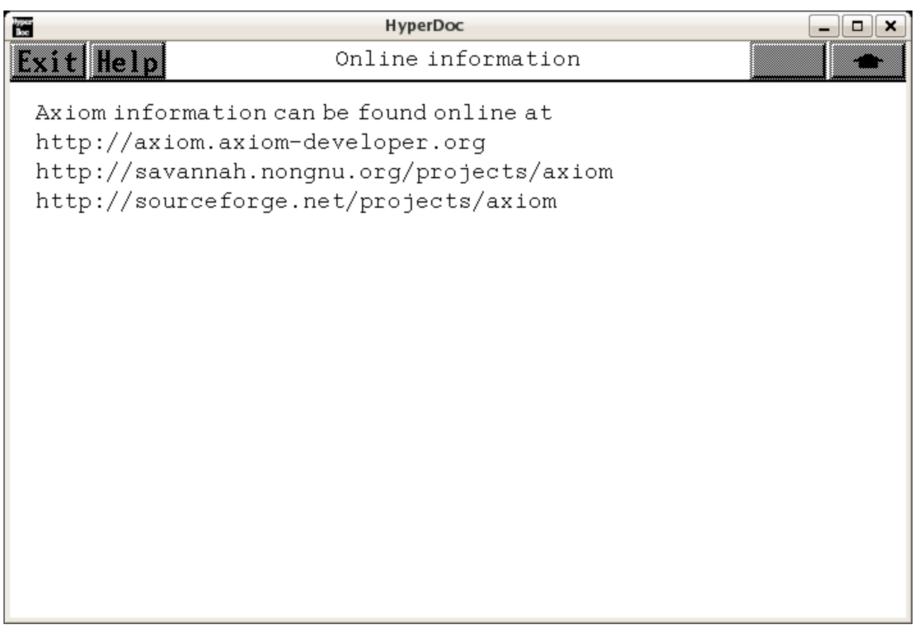
— **releasenotes.ht** —

```
\begin{page}{releaseNotes}{0. What's New in Axiom}
\beginscroll
\beginmenu
    \menudownlink{Online information}{onlineInformation}
    \menudownlink{August 2014}{aug2014}
    \menudownlink{May 2012}{may2012}
    \menudownlink{March 2012}{mar2012}
    \menudownlink{January 2012}{jan2012}
    \menudownlink{November 2011}{nov2011}
    \menudownlink{September 2011}{sept2011}
    \menudownlink{July 2011}{july2011}
    \menudownlink{May 2011}{may2011}
    \menudownlink{March 2011}{mar2011}
    \menudownlink{January 2011}{jan2011}
    \menudownlink{November 2010}{nov2010}
    \menudownlink{September 2010}{sept2010}
    \menudownlink{July 2010}{july2010}
```

```
    \menudownlink{May 2010}{may2010}
    \menudownlink{March 2010}{mar2010}
    \menudownlink{January 2010}{jan2010}
    \menudownlink{November 2009}{nov2009}
    \menudownlink{September 2009}{sept2009}
    \menudownlink{July 2009}{july2009}
    \menudownlink{May 2009}{may2009}
    \menudownlink{March 2009}{mar2009}
    \menudownlink{January 2009}{jan2009}
    \menudownlink{November 2008}{nov2008}
    \menudownlink{September 2008}{sept2008}
    \menudownlink{July 2008}{july2008}
    \menudownlink{May 2008}{may2008}
    \menudownlink{March 2008}{march2008}
    \menudownlink{January 2008}{january2008}
    \menudownlink{November 2007}{november2007}
    \menudownlink{February 2005}{feb2005}
\endmenu
\endscroll
\autobuttons
\end{page}
```

————-

Online Information

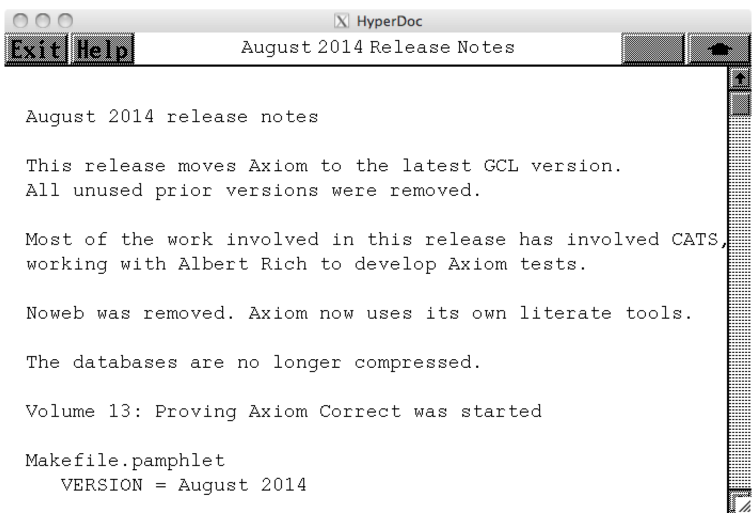


⇐ "What's New in Axiom" (releaseNotes) 1.1 on page 1

— **releasenotes.ht** —

```
\begin{page}{onlineInformation}{Online information}
\beginscroll
Axiom information can be found online at
{http://axiom.axiom-developer.org}
{http://savannah.nongnu.org/projects/axiom}
{http://sourceforge.net/projects/axiom}
\endscroll
\autobuttons
\end{page}
```

August 2014 Release Notes



⇐ "What's New in Axiom" (releaseNotes) 1.1 on page 1

— **releasenotes.ht** —

```
\begin{page}{aug2014}{August 2014 Release Notes}
\beginscroll
\begin{verbatim}
August 2014 release notes

This release moves Axiom to the latest GCL version.
All unused prior versions were removed.

Most of the work involved in this release has involved CATS,
```

```
working with Albert Rich to develop Axiom tests.

Noweb was removed. Axiom now uses its own literate tools.

The databases are no longer compressed.

Volume 13: Proving Axiom Correct was started

Makefile.pamphlet
   VERSION = August 2014
   TESTSET defaults to regresstests
   fix missing dvipdfm
   remove noweb

book/
   new directory containing axiom-developer emails
   reduce and categorize emails for history
   create timeline of people joining/leaving

books/
   axiom.sty
      document
      remove all other sources
      add additional macros
   axbook.tgz
      fix sinCosExpandRules
   dvipdfm.def
      added
   tanglec.c
      new C function to handle chunks
   tangle.lisp
      add fastxhtml extraction from bookvol11
   extract (rename from document)
      support new chunk syntax
   ps/
      move .ps to .eps
      add new images

books/Makefile
   handle bookheader.tex
   fix broken sed regular expression
   add bookvol13 (Proving Axiom Correct)

books/bookheader.tex
   user underline, colored links
   standardize headers, table of contents

books/bookvol0 (Jenks)
   add condata for MAGCDOC
   add graphics, fix typos, add text
```

```
books/bookvol1 (tutorial)
   use bookheader.tex
   fix typos

books/bookvol2 (users guide)
   use bookheader.tex
   add category theory chapter
   remove noweb

books/bookvol3 (programmers guide)
   use bookheader.tex
   remove noweb

books/bookvol4 (developers guide)
   use bookheader.tex
   section on changing GCL versions
   document makeHelpFiles
   remove autoload files
   remove noweb
   add HACKPI section

books/bookvol5 (interpreter)
   use bookheader.tex
   fix GETL for GCL
   add Attributes as Categories
   add Comparable
   add LeftOreRing
   add U32Matrix support macros
   add bc-matrix code
   add support code for U8Vector
   add support code for U32Vector
   add support code for U8Matrix
   add support code for U16Matrix
   add U32VectorPolyunomialOperations
   expose FiniteFieldFactorization
   expose MAGCDOC
   expose MatrixManipulation
   rename functionp to canFuncall?
   incremental rewrite of br-con.lisp
   merge ht-util.lisp

books/bookvol6 (commands)
   use bookheader.tex

books/bookvol7 (hyperdoc)
   apply Camm's patches
   remove bibliography
   standardize the table of contents
   update credits
```

```
   use bookheader.tex

books/bookvol7.1 (hyperdoc pages)
   standardize the table of contents
   update What's New Page
   update credits
   use bookheader.tex

books/bookvol8 (graphics)
   apply Camm's patches
   convert bookvol8 from noweb to chunks
   remove crc test cases
   standardize the table of contents
   update credits
   use bookheader.tex
   remove noweb

books/bookvol8.1 (graphics gallery)
   add CRC standard test cases, section 2.3-2.9

books/bookvol9 (compiler)
   use bookheader.tex
   functionp -> canFuncall?

books/bookvol10 (algebra implementation)
   absorb src/algebra/Makefile
   use bookheader.tex

books/bookvol10.1 (algebra theory)
   add interval arithmetic
   add section on interpolation formulas
   use bookheader.tex

books/bookvol10.2 (categories)
   AGG bug fixed 40355
   CLAGG bug fixed 40021
   MONAD bug fix 40350, add 7236
   ORDSET bug fix 40346
   PTRANFN bug fixed 40357
   REAL bug fix 40345
   add Attributes as Categories
   add Comparable
   add LeftOreRing
   add MAGCDOC
   add enumerate to FINITE
   add matrix initializer function
   fix failing test cases
   fix help documentation and regression tests
   use bookheader.tex
   write help documentation for all categories
```

```
   FiniteSetAggregate subset? fixed
   BinaryTreeCategory remove finiteAggregate, bug fix 7255

books/bookvol10.3 (domains)
   AFFSP bug fix 30230, 20495; introduce 60076
   BLAS1 bug fixed 40304
   CHAR bug fix 40022
   DFLOAT bug fix 20073, add 20570
   DFLOAT bug fixed 40019
   FPS bug fixed 40013
   INBFF bug fixed 20465, add 20571
   INTRVL bug fix 20480, 30143
   ITAYLOR bug fix 20542 pretend
   LIST bug fix 20063, add, add 20569
   MLIFT bug fix 20001, add 20568
   OPTPROB bug fix 40347
   RNUMP removed bug fix 30047
   SINT bug fix 20057 pretend
   SMTS bug fix 20383, 50005, add 20567
   SYMBOL bug fix 20056 pretend
   add U16Matrix
   add U16Vector
   add U32Matrix
   add U8Matrix
   add U8Vector
   add information to SingleInteger
   SEXOF bug fix 20501
   fix failing test cases
   fix help documentation and regression tests
   move EFULS EFUPXS from package to domain
   update U32Vector domain
   use bookheader.tex
   write help documentation for all domains

books/bookvol10.4 (packages)
   ATTREG bug fix 40362
   ELTAB bug fix 40360
   FSPECF bug fix 7235, 40077
   GRAY bug fixed 20467
   MAPPKG1 bug fix 40351
   MSYSCMD bug fix 40359
   SOLVERAD bug fix 40043
   SREGSET bug fix 50001
   add FiniteFieldFactorization
   add MatrixManipulation, add Raoul Bourquin
   add U32VectorPolynomialOperations
   apply Waldek's changes to imposelc in PGCD
   comment out bad code in GUESS
   document htrigs function in TRMANIP
   expand hyperbolic identities
```

```
   bug fix "bad reduction" in multivariate poly gcd
   bug fix ++X function example in gcd
   bug fix OUT.spad warning (bug 329)
   bug fix failing test cases
   bug fix help documentation and regression tests
   move EFULS EFUPXS from package to domain
   use bookheader.tex
   write help documentation
   add reportInstantiations to API package

books/bookvol10.5 (numerics)
   add netlib cephes function covers
   use bookheader.tex

books/bookvol11 (browser)
   use bookheader.tex
   remove noweb

books/bookvol12 (crystal)
   remove noweb

books/bookvol13 (proving axiom correct)
   created

books/bookvolbib (bibliography)
   add references and abstracts
   add subsection Ore Rings
   add subsection Partial Fraction Decomposition
   add subsection Integration
   add subsection Expression Simplification
   add subsection Differential Equations
   add subsection Advanced Documentation
   add subsection Numerics
   add subsection Interval Arithmetic
   add subsection Proving Axiom Correct
   add subsection Category Theory
   add subsection Polynomial GCD
   add subsection Exponenetial Integral
   add subsection Special Functions
   add subsection Numerical Algorithms
   add subsection Solving Systems of Equations

buglist
   MLIFT fix 20001, add 20568
   AGG fixed 40355
   ATTREG fix 40362
   BLAS1 fixed 40304
   CHAR fix 40022
   CLAGG fixed 40021
   DFLOAT fix 20073, add 20570
```

```
   DFLOAT fixed 40019
   ELTAB fix 40360
   FPS fixed 40013
   FSPECF fix 7235, 40077
   GRAY fixed 20467
   INBFF fixed 20465, add 20571
   INTRVL fix 20480, 30143
   ITAYLOR fix 20542 pretend
   LIST fix 20063, add, add 20569
   MAPPKG1 fix 40351
   MONAD fix 40350, add 7236
   MSYSCMD fix 40359
   OPTPROB fix 40347
   ORDSET fix 40346
   PTRANFN fixed 40357
   REAL fix 40345
   SINT fix 20057 pretend
   SMTS fix 20383, 50005, add 20567
   SOLVERAD fix 40043
   SREGSET fix 50001
   SYMBOL fix 20056 pretend
   fix bug 7072/19 D(0^z, z) closed
   fix bug 7248: differentiation fix bug in D(1,z)
   fix bug 7249: radicalSolve(z^7=1, z)
   fix bug 7249: radicalSolve(z^7=1, z) update
   fix bug 7251: integrate(1/(1+z^4), z=0..1) :: Complex Float
   fix bug 7252: argument(-%i)
   fix bug 7253: There are no library operations named 'when'
   fix bug 7254: f==n+->sum(sum(1/i,i=1..j),j=1..n) complains
   fix bug 7256: acot(-1) values differ
   fix bug 7257: normalize(subst(asin(z),z=-1)) division by zero
   fix bug 7258: acosh(0.0) invalid argument to acosh
   fix bug 20501, SEXOF pretend, 30047 RNUMP
   fix bug 30230, 20495; introduce 60076
   fix bug 7236
   fix bug 329, add todo 330
   remove dead items
   remove error message for deleted files
   add 7142 exp^log(x) Cannot take first of an empty list update
   add 7233: fill! operation from U8Vector does not show up
   add 7242/7243
   add 7244, 7245
   add bug 7241, fix the integration routines
   add bugs found in tests
   add kamke4, kamke6 missing bug
   add todo 335: add packages to )d op gcd
   add 7232
   wish 1011: sum(1/(k+a), k=1..n) by Gosper's method

Wikipedia
```

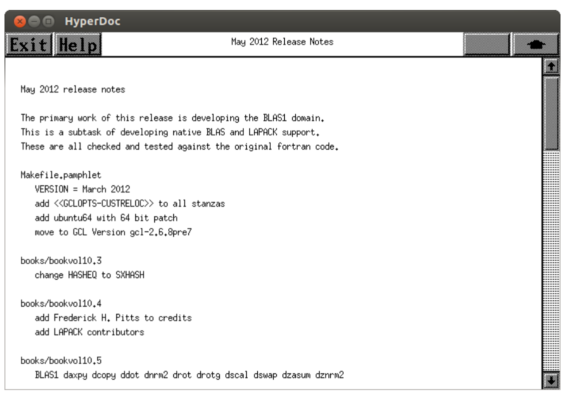
```
   en.wikipedia.org/wiki/Axiom_(computer_algebra_system) update date
   en.wikipedia.org/wiki/List_of_computer_algebra_systems update date

faq
   remove noweb
   fix mailing list links

readme
   add Edi Meier
   add Edi Meier
   add Jia Zhao Cong
   add Raoul Bourquin
   update credits

src/Makefile
   clean up src/algebra properly
   convert bookvol8 from noweb to chunks
   fix {OUT} bug
   handle special case files for bookvol11
   rename tangle.c to tanglec.c
   pamphlet extract src using chunk syntax
   pamphlet modified, noweb removed

src/algebra/Makefile
   add Comparable
   add FINITE-
   add LeftOreRing
   add MatrixManipulation
   add U16Matrix
   add U32Matrix
   add U32VectorPolynomialOperations
   add U8Matrix
   add fastxhtml extraction call
   add machinearithmetic.input
   compile FiniteFieldFactorization
   compile MAGCDOC
   fix failing test cases
   set up regression tests
   update graph info for OUT
   modified, noweb removed

src/algebra/libdb.text
   add Attributes as Categories

src/axiom-website/
   AxiomConference.pdf added
   axiom.sty
   CATS/
      charlwood.input added
      rich* added
```

```
      kamke*.txt added
   books/
      add 8.1, 10.5, bibliography
      add bookvol13
   community/
      fix mailing list links
   documentation.html
      add AxiomConference.pdf
      add Bottorff quote
      add Knuth quote, cleanup
      add Lamport quote
      add McNamara quote
      add Stroustrop quote
      add W.T. Gowers quote
   download.html
      add binaries
      add texlive-fonts-extra
   patches.html
      add patches
   releasenotes.html
       update for August 2014

src/booklets
   Makefile cleanup

src/clef
   Makefile rename tangle to tanglec
   edible.c modified, noweb removed

src/doc
   export endpaper as pdf
   remove axiom.bib
   rename tangle to tanglec
   use books/axiom.sty
   remove noweb
   booklet.c fix mailing list, remove noweb
   msgs/
      s2-us.msgs remove .as extension from compiler msg

src/etc
   Makefile remove noweb
   asq.c remove noweb

src/input/
   Makefile add input tests for
      charlwood, cherry, functioncode, machinarithmetic,
      pgcd, polygamma, rich*, romanpolynomials, rsa, segletes
      exptest, subset, tree, trigtests
   Makefile remove noweb
   add many tests
```

```
src/interp
   Makefile
      remove noweb
      remove autoload of ht-util
      remove bc-matrix
    bc-matrix.lisp removed
    br-con.lisp move to bookvol5
    debugsys cleanup
    g-timer.lisp RNUMP removed
    g-util.lisp functionp -> canFuncall?
    g-util.lisp remove functions
    ht-util.lisp deleted, merged with bookvol5
    i-coerce.lisp functionp -> canFuncall?
    i-funsel.lisp remove functions
    i-spec2.lisp. functionp -> canFuncall?
    interp-proclaims.lisp functionp -> canFuncall?
    regress.lisp improve diff output
    sys-pkg.lisp RNUMP removed
    temp.text deleted
    util.lisp functionp -> canFuncall?
    util.lisp remove autoload of ht-util
    util.lisp remove autoload trigger functions
    util.lisp remove bc-matrix autoload triggers
    util.lisp remove compression
    vmlisp.lisp RNUMP removed

src/scripts
   Makefile remove noweb
   boxhead removed
   boxtail removed
   boxup removed
   document -> books/extract
   showdvi removed
   tex/
      axiom.sty removed

src/share
   Makefile remove noweb
   algebra/compress removed
   algebra/browse, category, dependents, interp, operations, users
      add Attributes as Categories
      add LeftOreRing
      enumerate to FINITE
      no compression
      updated
   doc/
      hypertex/pages/util.ht deleted, unused

src/sman
```

```
   Makefile cleanup

zips/
   axiom.vim.tgz vim syntax file added
   gcl-cgywin.tgz added
   noweb-2.10a.tgz deleted, noweb removed
   noweb.* deleted, noweb removed
   all unused patches and versions of gcl deleted

\end{verbatim}
\endscroll
\autobuttons
\end{page}
```

May 2012 Release Notes



⇐ “What's New in Axiom” (releaseNotes) 1.1 on page 1

— releasenotes.ht —

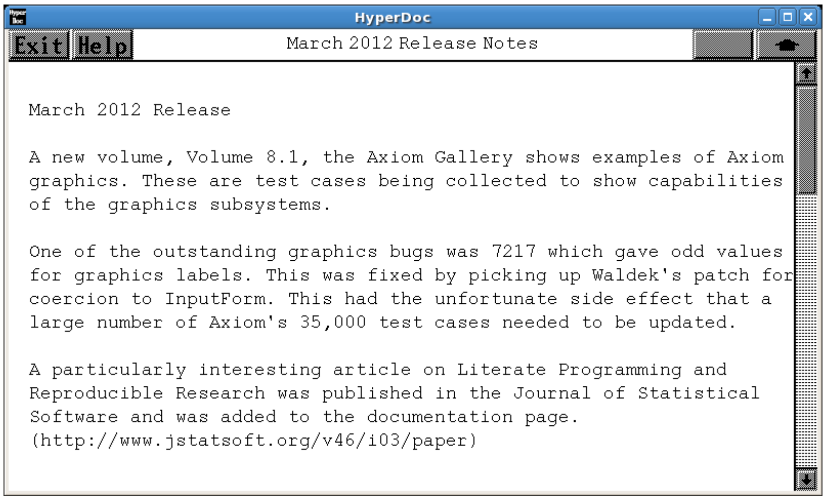
```
\begin{page}{may2012}{May 2012 Release Notes}
\beginscroll
\begin{verbatim}
May 2012 release notes

The primary work of this release is developing the BLAS1 domain.
This is a subtask of developing native BLAS and LAPACK support.
These are all checked and tested against the original fortran code.

Makefile.pamphlet
   VERSION = March 2012
   add <<GCLOPTS-CUSTRELOC>> to all stanzas
```

```
   add ubuntu64 with 64 bit patch
   move to GCL Version gcl-2.6.8pre7

books/bookvol10.3
   change HASHEQ to SXHASH

books/bookvol10.4
   add Frederick H. Pitts to credits
   add LAPACK contributors

books/bookvol10.5
   BLAS1 daxpy dcopy ddot dnrm2 drot drotg dscal dswap dzasum dznrm2
         icamax idamax isamax izamax zaxpy
   add LAPACK reference code
   add missing lapack routines

books/bookvol5
   add Frederick H. Pitts to credits
   add LAPACK contributors
   fix parsing for 64 bit systems
   reset si::*system-directory* to the null string

books/bookvol7.1
   update What's New Page

books/bookvolbib
   BLAS1 drotg references
   add paper references
   fix typo
   update references

books/bookvolbib.bib
   BLAS1 drotg references
   add LAPACK bibtex reference
   update references

books/ps/
   v71mar2012.eps added
   v71releasenotes.eps add release notes

faq
   add FAQ 53: Axiom won't build on Fedora
   fix FAQ 22: How do I check out the latest source?

lsp/Makefile.pamphlet
   move to GCL Version gcl-2.6.8pre7

readme
   Frederick H. Pitts <fred.pitts@comcast.net>
   add LAPACK contributors
```

```
src/algebra/Makefile
   fix MYUP, etc

src/axiom-website/download.html
   add debian binary
   add ubuntu
   update download list
   update download table

src/axiom-website
   releasenotes.html

src/input/Makefile
   add cohen.input
   add example of spad code
   add simplify.input

src/input
   clements.input add example of spad code
   cohen.input Joel Cohen algebra example
   simplify.input added from sci.math.symbolic

src/interp
   buildom.lisp: change HASHEQ to SXHASH
   sys-pkg.lisp: remove HASHEQ

zips/
   gcl-2.6.8pre7.h.linux.defs.patch port to pre7
   gcl-2.6.8pre7.o.read.d.patch port to pre7
   gcl-2.6.8pre7.tgz lisp fixes from Camm Maquire
   gcl-2.6.8pre7.unixport.init_gcl.lsp.in.patch port to pre7
   gcl-2.6.8pre7.unixport.makefile.patch port patch to pre7
\end{verbatim}
\endscroll
\autobuttons
\end{page}
```

———

March 2012 Release Notes



⇐ "What's New in Axiom" (releaseNotes) 1.1 on page 1

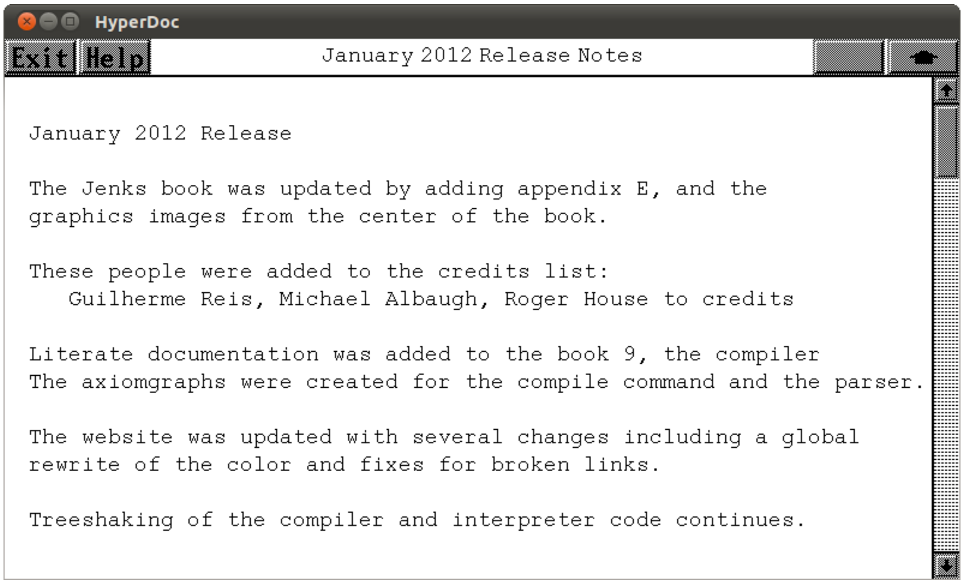
— **releasenotes.ht** —

```
\begin{page}{mar2012}{March 2012 Release Notes}
\beginscroll
\begin{verbatim}
March 2012 Release

A new volume, Volume 8.1, the Axiom Gallery shows examples of Axiom
graphics. These are test cases being collected to show capabilities
of the graphics subsystems.

One of the outstanding graphics bugs was 7217 which gave odd values
for graphics labels. This was fixed by picking up Waldek's patch for
coercion to InputForm. This had the unfortunate side effect that a
large number of Axiom's 35,000 test cases needed to be updated.

A particularly interesting article on Literate Programming and
Reproducible Research was published in the Journal of Statistical
Software and was added to the documentation page.
(http://www.jstatsoft.org/v46/i03/paper)

There are two new commands, )tangle and )regress.

A new function, getAncestors was added to the API domain.

The buglist was published and several bugs were added.

New references to Axiom were added to the bibliography, including
Dav12, Jen79, Dav80
```

```
Several people contributed to this release, including James Davenport,
Mark Clements, George Legendre, and Waldek Hebisch

bookvol5.pamphlet
   add )tangle and )regress commands

bookvol10.4.pamphlet
   add Mark Clements, George Legendre
   fix 7217 bugs

books/bookvol5
   add Mark Clements, George Legendre
   fix 7217 bugs

books/bookvol8.1
   A new volume, the Axiom Gallery, showing example graphics

books/bookvolbib
   add additional Axiom literature references

books/ps/
   add v81* for graphics gallery

readme
   add Mark Clements, George Legendre

src/algebra
   Makefile fix GUESS, clique1, and clique2 handling

src/axiom-website/
   add jstat article

src/input/
   add testpackage.input
   update pasta.input graphics test cases
   fix 7216 bugs

\end{verbatim}
\endscroll
\autobuttons
\end{page}
```

———-

January 2012 Release Notes



⇐ “What’s New in Axiom” (releaseNotes) 1.1 on page 1

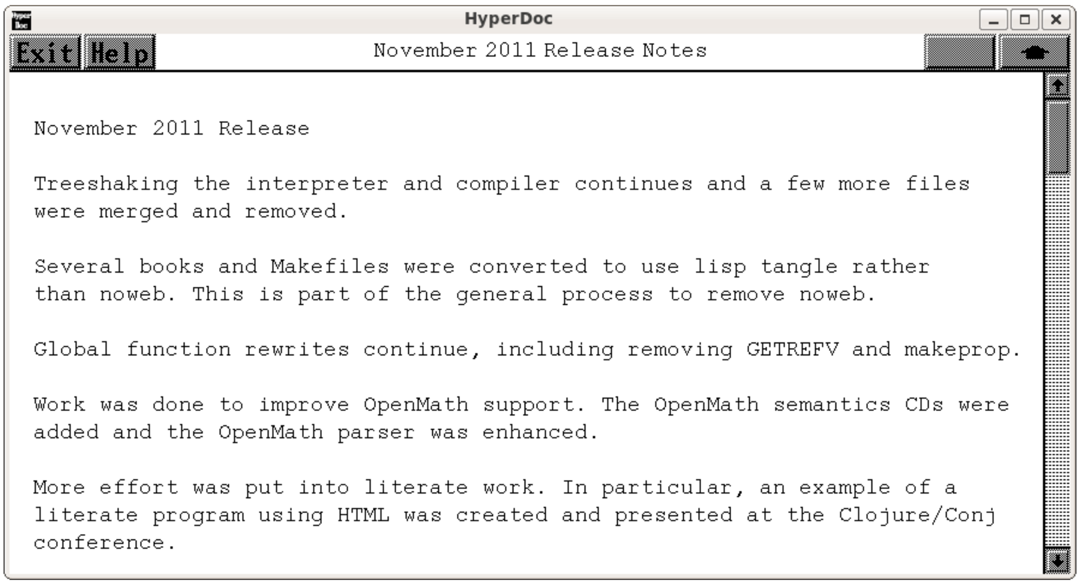
— **releasenotes.ht** —

```
\begin{page}{jan2012}{January 2012 Release Notes}
\beginscroll
\begin{verbatim}
January 2012 Release

The Jenks book was updated by adding appendix E, and the
graphics images from the center of the book.

These people were added to the credits list:
   Guilherme Reis, Michael Albaugh, Roger House to credits

Literate documentation was added to the book 9, the compiler
The axiomgraphs were created for the compile command and the parser.

The website was updated with several changes including a global
rewrite of the color and fixes for broken links.

Treeshaking of the compiler and interpreter code continues.

Work was done but not completed to move to GCL-2.6.8pre6

bookvol0.pamphlet
   add Appendix E
   add graphics images from middle of book
```

```
bookvol10.4.pamphlet
   add Guilherme Reis, Michael Albaugh, Roger House

books/bookvol5
   localize function names
   treeshake interpreter
   add Guilherme Reis, Michael Albaugh, Roger House

books/bookvol9
   add comment graphs
   code cleanup
   localize function names
   treeshake and document compiler

books/bookvolbib
   add additional Axiom literature references

books/ps/
   add v0hroot.eps, v0hyper.eps, v0page1.eps, v0page2.eps, v0page3.eps
   add v0page4.eps, v0page5.eps, v0page6.eps, v0page7.eps, v0page8.eps
   v9CommentRecording.eps comment recording chapter
   v9CommentSyntaxChecking.eps comment syntax chapter
   v9comdefine.eps document compiler
   v9compiler.eps document compiler

readme
   add Guilherme Reis, Michael Albaugh, Roger House

src/algebra
   Makefile fix autoload

src/axiom-website/
   global background color change to ECEA81
   axiom.png make background transparent
   books.html fix broken links
   litprog.html fix argument count
   litprog.html note HTML escape code flaw

  /axiomgraph/
     axiomgraph.tex literate form of js
     index.html add graphs

  /axiomgraph/js
     arbor.js add graphs
     axiomcode.js add graphs, parse tree, default compiler
     jquery-1.7.1.js add graphs

  /xiomgraph/maps/
     algebra.json add graphs
```

```
     comment.json update comment
     add compile.json for the compile command
     add compiler.json for the compiler
     add dstruct.json for data structures
     add funsel.json for function selection
     nine.json for global compiler
     parse.json for parser

  /axiomgraph/style/
     axiomgraph.css change background
     axiomicon.png add icon

  /hyperdoc
     change page background color

src/doc/
   axiom.sty fix comments

src/input/
   Makefile add i2e.input
   i2e.input demo InputForm to Expression(Integer)
   pasta.input graphics test cases

src/interp
   br-con.lisp localize function names, treeshake compiler
   cattable.lisp treeshake compiler
   ht-util.lisp localize function names
   i-funsel.lisp treeshake compiler
   parsing.lisp localize function names, treeshake compiler
   patches.lisp localize function names
   util.lisp treeshake compiler
   vmlisp.lisp localize function names, treeshake compiler

src/scripts/tex/
   axiom.sty fix comments


20111130 tpd src/axiom-website/patches.html 20111130.04.tpd.patch
20111130 tpd src/axiom-website/patches.html 20111130 update patch frontier
20111130 tpd src/axiom-website/releasenotes.html
20111130 tpd books/ps/v71releasenotes.eps add release notes
20111130 tpd books/ps/v71nov2011.eps added
20111130 tpd books/bookvol7.1 update What's New Page
20111130 tpd Makefile.pamphlet VERSION = November 2011
20111130 tpd Makefile VERSION = November 2011

\end{verbatim}
\endscroll
\autobuttons
\end{page}
```

November 2011 Release Notes



⇐ "What's New in Axiom" (releaseNotes) 1.1 on page 1

— **releasenotes.ht** —

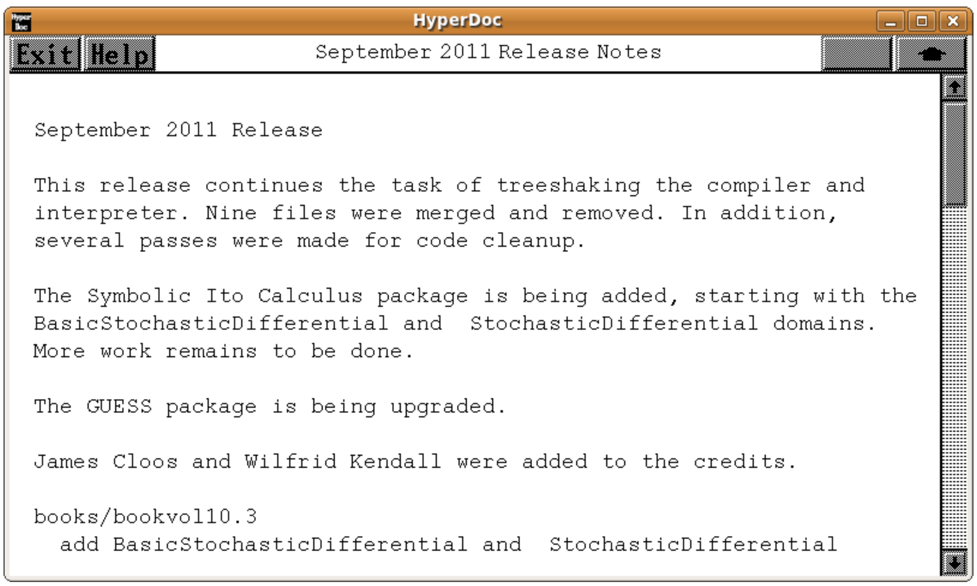
```
\begin{page}{nov2011}{November 2011 Release Notes}
\beginscroll
\begin{verbatim}
November 2011 Release

Treeshaking the interpreter and compiler continues and a few more files
were merged and removed.

Several books and Makefiles were converted to use lisp tangle rather
than noweb. This is part of the general process to remove noweb.

Global function rewrites continue, including removing GETREFV and makeprop.

Work was done to improve OpenMath support. The OpenMath semantics CDs were
added and the OpenMath parser was enhanced.

More effort was put into literate work. In particular, an example of a
literate program using HTML was created and presented at the Clojure/Conj
conference.
```

```
books/Makefile
   removed noweb, move to lisp tangle
   fix missing index files on PDfs

books/tangle.lisp
   fix .input file algebra extraction

books/bookvol10.2
   missing/unused function cleanup
   remove makeprop
   remove noweb, move to lisp tangle

books/bookvol10.3
   remove GETREFV
   remove makeprop
   remove noweb, move to lisp tangle

books/bookvol10.4
   remove noweb, move to lisp tangle

books/bookvol10.5
   remove noweb, move to lisp tangle

books/bookvol4
   missing/unused function cleanup
   remove makeprop

books/bookvol5
   OpenMath support
   add OpenMath stubs
   missing/unused function cleanup
   remove makeprop
   remove noweb, move to lisp tangle
   treeshake interpreter

books/bookvol7.1
   fix documentation
   update What's New Page

books/bookvol9
   missing/unused function cleanup
   remove makeprop
   treeshake and merge c-doc.lisp

books/bookvolbib
   Axiom OpenMath references
   add additional references
   add references
```

```
books/ps/
   v71releasenotes.eps add release notes
   v71sept2011.eps added

src/algebra/Makefile
   fix .input file algebra extraction
   remove noweb, move to lisp tangle

src/axiom-website/documentation.html
   add Alexander quote
   add Knuth quote
   add litprog.html
   add quote

src/axiom-website/download.html
   update to latest release

src/axiom-website/litprog.html
   added
   add quote
   literate program example update

src/interp/Makefile
   remove apply.lisp
   remove c-doc.lisp
   remove noweb, move to lisp tangle
   remove nruntime.lisp

src/interp
   apply.lisp removed, merged with bookvol9
   buildom.lisp remove GETREFV
   c-doc.lisp removed
   c-util.lisp missing/unused function cleanup
   category.lisp remove GETREFV
   functor.lisp missing/unused function cleanup, remove GETREFV
   g-util.lisp missing/unused function cleanup
   i-eval.lisp remove nruntime.lisp
   i-spec1.lisp treeshake interpreter
   i-spec2.lisp fix AN has sqrt: % -> %
   i-spec2.lisp remove makeprop
   nruncomp.lisp remove GETREFV, treeshake compiler
   nrungo.lisp remove nruntime.lisp, treeshake interpreter, remove GETREFV
   nruntime.lisp removed
   parsing.lisp remove makeprop
   sys-pkg.lisp missing/unused function cleanup, remove GETREFV, makeprop
   template.lisp remove GETREFV
   vmlisp.lisp missing/unused function cleanup, remove GETREFV, makeprop

zips
   add cds.tar.gz OpenMath support. add OpenMath CDs
```

```
\end{verbatim}
\endscroll
\autobuttons
\end{page}
```

September 2011 Release Notes



⇐ "What's New in Axiom" (releaseNotes) 1.1 on page 1

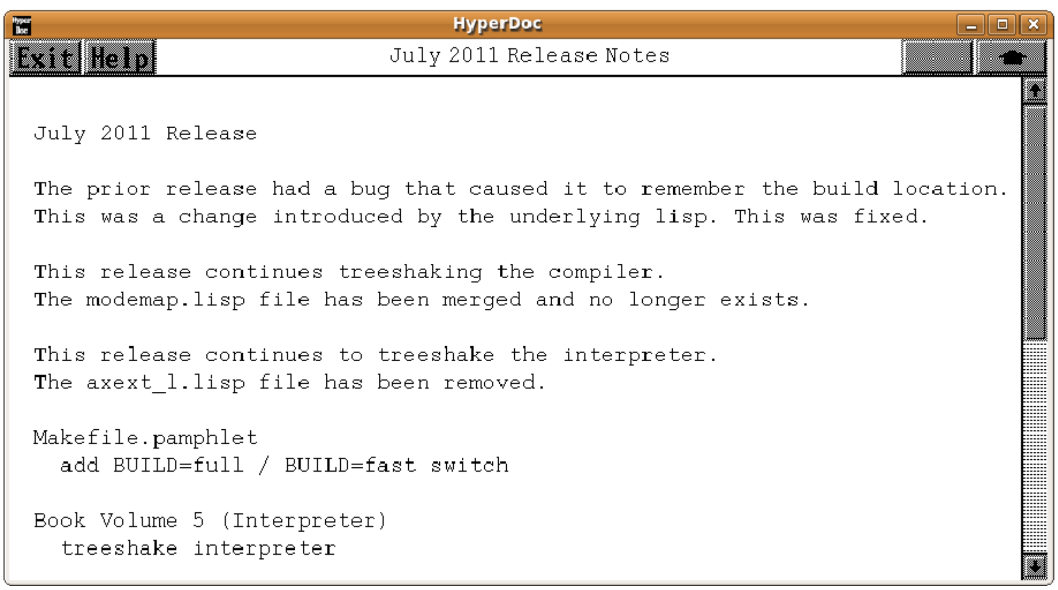
— **releasenotes.ht** —

```
\begin{page}{sept2011}{September 2011 Release Notes}
\beginscroll
\begin{verbatim}
September 2011 Release

This release continues the task of treeshaking the compiler and
interpreter. Nine files were merged and removed. In addition,
several passes were made for code cleanup.

The Symbolic Ito Calculus package is being added, starting with the
BasicStochasticDifferential and StochasticDifferential domains.
More work remains to be done.

The GUESS package is being upgraded.
```

```
James Cloos and Wilfrid Kendall were added to the credits.

books/bookvol10.3
  add BasicStochasticDifferential and  StochasticDifferential
  remove pairp
  start upgrade of GUESS package

books/bookvol10.4
  remove packageTran

books/bookvol10.5
  add StochasticDifferential

books/bookvol5
  add BasicStochasticDifferential domain
  merge nocompil.lisp, as.lisp, nci.lisp
  treeshake interpreter
  remove pairp, ncParseAndInterpretString, packageTran
  use qc(ad)r forms

books/bookvol7
  merge nocompil.lisp

books/bookvol7.1
  update What's New Page

books/bookvol9
  remove pairp, isPackageFunction
  treeshake compiler
  use qc(ad)r forms

books/bookvolbib
   add Kendall Ken99a, Ken99b literature references to Axiom

books/ps/
  v103basicstochasticdifferential.ps added
  v103stochasticdifferential.ps added
  v71july2011.eps
  v71releasenotes.eps add release notes

src/algebra/Makefile
  BasicStochasticDifferential domain
  add StochasticDifferential
  remove upper GUESS package code

src/axiom-website/download.html add Gentoo notes by James Cloos
src/axiom-website/download.html add ubuntu

src/axiom-website/releasenotes.html
```

```
src/doc/axiom.sty
  fix defplist

src/input/Makefile
  document finite field bug
  respect the BUILD=fast variable value

src/input/
  fix broken tests in axiom, cmds, setcmd, unittest1, unittest2
  ffieldbug.input: added

src/interp/Makefile
  remove as.lisp, compiler.lisp, database.lisp, define.lisp, foam_l
  remove g-opt.lisp, nci.lisp, package.lisp, ax.lisp

\end{verbatim}
\endscroll
\autobuttons
\end{page}
```

July 2011 Release Notes

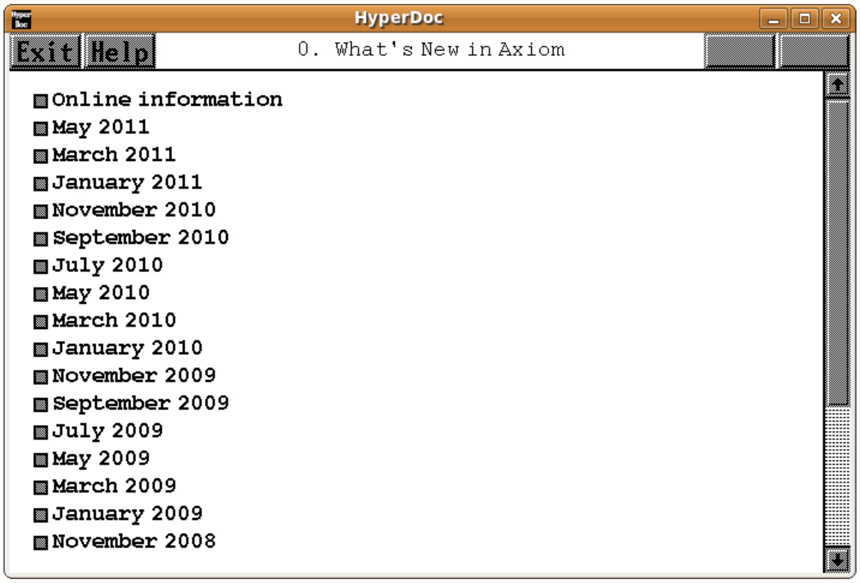


⇐ "What's New in Axiom" (releaseNotes) 1.1 on page 1

— **releasenotes.ht** —

```
\begin{page}{july2011}{July 2011 Release Notes}
\beginscroll
\begin{verbatim}
July 2011 Release

The prior release had a bug that caused it to remember the build location.
This was a change introduced by the underlying lisp. This was fixed.

This release continues treeshaking the compiler.
The modemap.lisp file has been merged and no longer exists.

This release continues to treeshake the interpreter.
The axext_l.lisp file has been removed.

Makefile.pamphlet
  add BUILD=full / BUILD=fast switch

Book Volume 5 (Interpreter)
  treeshake interpreter

Book Volume 9 (Compiler)
  treeshake compiler

Book Volume 10.3 (Domains)
  help and unit tests for RewriteRule

axiom.sty
  add /refsdollar and /defsdollar for better cross references

src/interp/Makefile
  add (si::reset-sys-paths) per Camm
  remove axext_l aldor support file
  remove modemap.lisp

\end{verbatim}
\endscroll
\autobuttons
\end{page}
```

May 2011 Release Notes



⇐ "What's New in Axiom" (releaseNotes) 1.1 on page 1

— **releasenotes.ht** —

```
\begin{page}{may2011}{May 2011 Release Notes}
\beginscroll
\begin{verbatim}
May 2011 Release

This release involved more work treeshaking of the compiler and
interpreter. Argument naming conventions were normalized to improve
the documentation and clarity.

Time was spent proofreading the books and reducing or eliminating
latex warnings. Missing index files were fixed and errors in the Jenks
book were fixed.

The website has been updated to make Arthur Ralf's book available
from every page. The website is being rewritten to be IPv6 enabled.

Doug Telford was added to the credits.

Book Volume 0 (Jenks book)
  proofread to chapter 1 (pdf page 86)
  replace "operator over" by "operate over"
  set textlength 400
  port changes to Arthur Ralfs HTML version
```

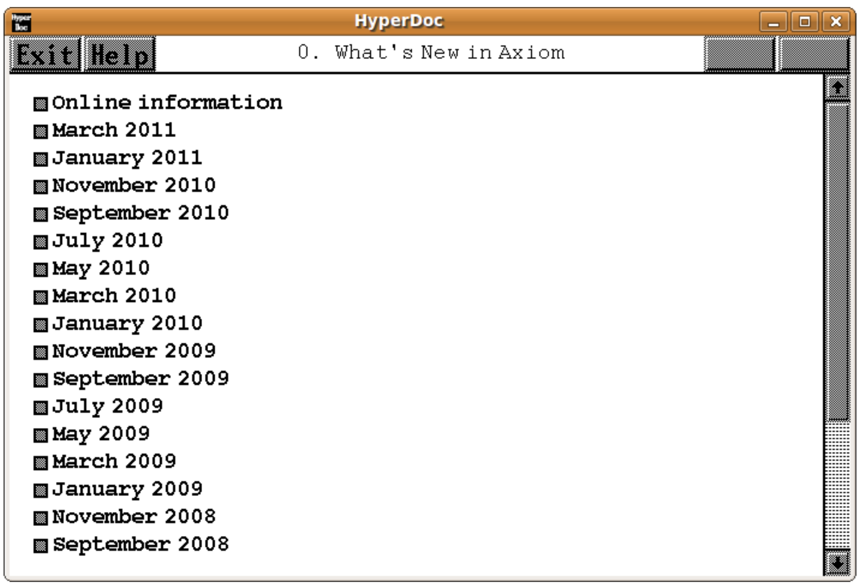
```
Book Volume 1 (Tutorial)
  proofread and copyedit
  set textlength 400
  update HTML image

Book Volume 2
  set textlength 400

Book Volume 3
  set textlength 400

Book Volume 4
  set textlength 400

Book Volume 5 (Interpreter)
  add Doug Telford to credits
  set textlength 400
  treeshake interpreter

Book Volume 6
  set textlength 400

Book Volume 7 (Hyperdoc)
  set textlength 400

Book Volume 7.1 (Hyperdoc pages)
  set textlength 400
  update What's New Page
  ps/v71mar2011.eps
  ps/v71releasenotes.eps add release notes

Book Volume 8
  set textlength 400

Book Volume 9 (Compiler)
  normalize argument names to top level functions
  treeshake compiler

Book Volume 10
  set textlength 400

Book Volume 10.1
  set textlength 400

Book Volume 10.2 (Categories)
  set textlength 400

Book Volume 10.3 (Domains)
  set textlength 400
```

```
Book Volume 10.4 (Packages)
  set textlength 400

Book Volume 10.5 (Numerics)
  set textlength 400

Book Volume 11
  set textlength 400

Book Volume 12
  set textlength 400

Biblography
  add several references
  set textlength 400

readme
  add Doug Telford to credits

website
  set up IPv6 address range
  download.html add binaries
  bugreport.html add bug reporting instructions

  The website now has an online version of the Jenks book which is a
  link to Arthur Ralf's HTML version. This has been updated with the
  proofreading changes for the latex version.

     *.html add online book link to Arthur Ralfs HTML version
     hyperdoc/axbook/ps/h-root.png port changes
     hyperdoc/axbook/section-0.1.xhtml, section-0.6.xhtml, section-0.7.xhtml

Makefiles
  books/Makefile fix missing index
  books/Makefile set textlength 400

Testing
  Makefile add davis.input
  Makefile add erf.input
  Makefile bug report regression test
  davis.input
  erf.input add examples of erf integration
  telford.input bug report regression test


src/interp/
  define.lisp treeshake compiler
  parsing.lisp treeshake compiler
  vmlisp.lisp treeshake compiler
```

```
\end{verbatim}
\endscroll
\autobuttons
\end{page}
```

———-

March 2011 Release Notes



⇐ "What's New in Axiom" (releaseNotes) 1.1 on page 1

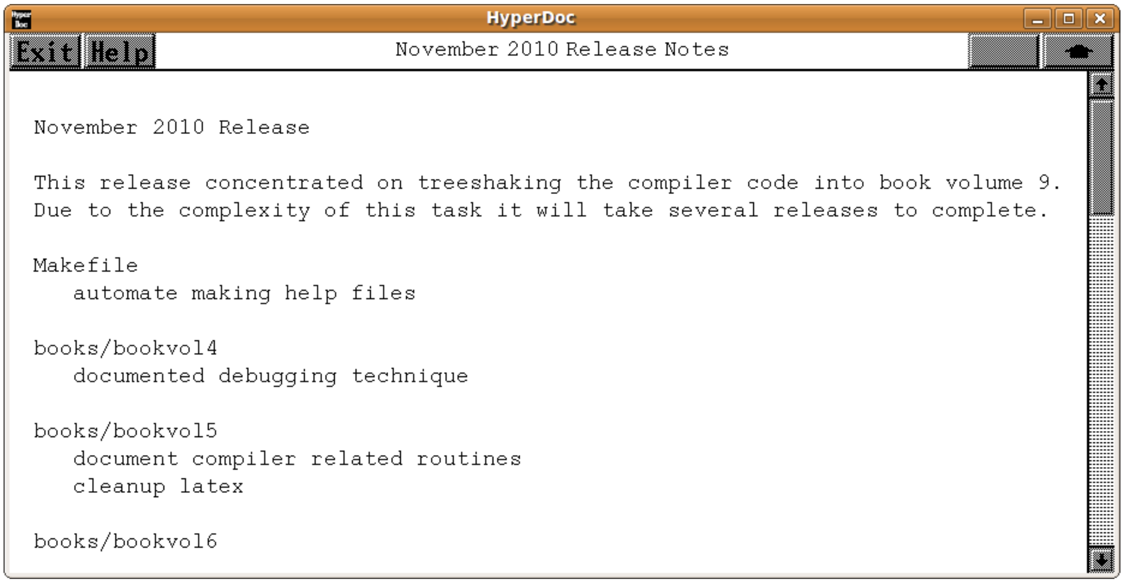
— **releasenotes.ht** —

```
\begin{page}{mar2011}{March 2011 Release Notes}
\beginscroll
\begin{verbatim}
March 2011 Release

This release involved more work treeshaking the compiler.

Book Volume 5 (Interpreter)
   add trapNumericErrors
   fixup seebook xrefs
   move numericFailure code for plot3d
   treeshake compiler
   treeshake compiler

Book Volume 9 (Compiler)
```

```
   add seebook changes
   fix special declaration for lines in preparse1
   fixup seebook xrefs
   treeshake compiler

Book Volume 10.3 (Domains)
   fix algebra ++ to --

Biblography
   add Davenport [Dav10], [DSTxx]
   add Fateman [Fat05]

tangle.lisp
   fix help file extraction

src/scripts/tex/axiom.sty
   add \seebook

src/interp/
 Makefile
    preload bookvol9
 g-util.lisp
    treeshake compiler
 parsing.lisp
    treeshake compiler
 util.lisp
    treeshake compiler
 vmlisp.lisp
    move numericFailure code for plot3d
    remove extra end{chunk}
    treeshake compiler


\end{verbatim}
\endscroll
\autobuttons
\end{page}
```

————-

January 2011 Release Notes



⇐ "What's New in Axiom" (releaseNotes) 1.1 on page 1

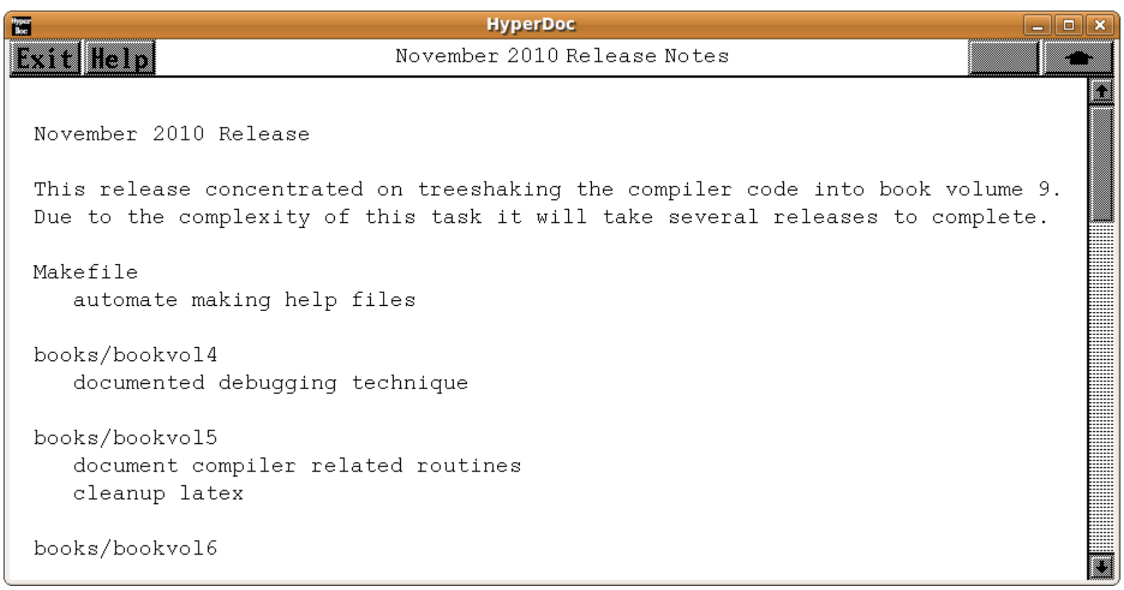
— releasenotes.ht —

```
\begin{page}{jan2011}{January 2011 Release Notes}
\beginscroll
\begin{verbatim}
January 2011 Release

This release involved more work treeshaking the compiler.
There is a move to using lisp tangle rather than noweb notangle.
Many of the latex error messages were fixed.

books/Makefile
   move bookvol9 to lisp tangle
   move to lisp tangle
   reduce the noise during make

books/bookvol5
   cleaning vmlisp
   convert to lisp tangle
   move to lisp tangle
   sort monitor results

books/bookvol6
   move to lisp tangle

books/bookvol7
```

```
   move to lisp tangle

books/bookvol7.1
   move to lisp tangle
   update What's New page

books/bookvol9
   cleaning vmlisp
   merge and remove newaux.lisp
   merge and remove postprop.lisp
   move to lisp tangle
   treeshake compiler

books/bookvol10.3
   cleaning vmlisp

books/bookvol10.4
   fixup IntersectionDivisorPackage chunk

books/bookvol12
   add discussion of GTFL

books/bookvolbib
   [Loe09], [Mar07], [Wei03], [Wei06], [Flo63], [Pra73]

books/ps/v71nov2010.eps
   update What's New page

books/ps/v71releasenotes.eps
   update What's New page

src/Makefile
   move to lisp tangle

src/axiom-website/download.html
   add ubuntu

src/axiom-website/releasenotes.html
   add release notes

src/input/Makefile
   clean up failing tests

src/input/
   *.input add \setlength{\textwidth}{400pt}
   fix overfull boxes
   clean up failing tests
   fix long lines
   series.input -- add series to polynomial example
   unit-i-funsel.input -- removed
```

```
src/interp
   Makefile convert bookvol5 to lisp tangle
   Makefile merge and remove newaux.lisp
   Makefile move to lisp tangle
   Makefile remove postprop.lisp
   src/interp/br-con.lisp cleaning vmlisp
   src/interp/compiler.lisp treeshake compiler
   src/interp/newaux.lisp remove newaux.lisp
   src/interp/postprop.lisp merged with bookvol9, removed.
   src/interp/regress.lisp handle "ok" on --S
   src/interp/vmlisp.lisp cleaning

\end{verbatim}
\endscroll
\autobuttons
\end{page}
```

November 2010 Release Notes



⇐ "What's New in Axiom" (releaseNotes) 1.1 on page 1

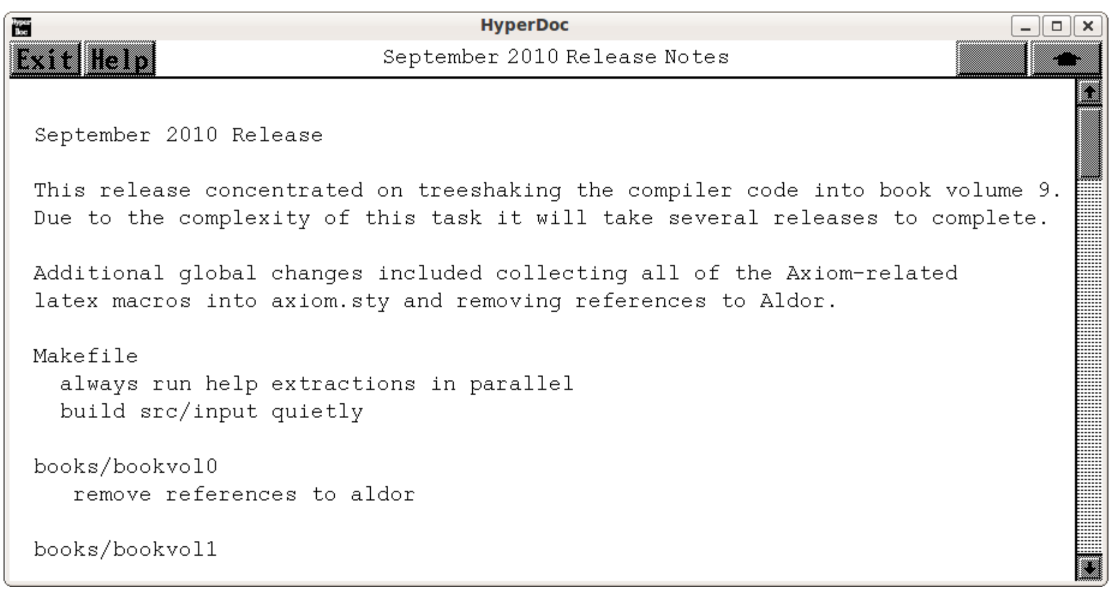
— **releasenotes.ht** —

```
\begin{page}{nov2010}{November 2010 Release Notes}
\beginscroll
```

```
\begin{verbatim}
November 2010 Release

This release concentrated on treeshaking the compiler code into book volume 9.
Due to the complexity of this task it will take several releases to complete.

Makefile
   automate making help files

books/bookvol4
   documented debugging technique

books/bookvol5
   document compiler related routines
   cleanup latex

books/bookvol6
   added section on research ideas

books/bookvol9
   treeshake compiler code
   document compiler code
   fix |special| bug
   merge and remove fnewmeta
   move meta code into bookvol9 from parsing.lisp

books/bookvolbib
   Chee Keng Yap [Yap00], Chudnovsky and Jenks [CJ86], Eric Weisstein [Wein],
   David and Gregory Chudnovsky [Chu89], Jenks, [Jen69], Kaufmann [KMJ00],
   Linger [LMW79], Wester [Wes99]

src/algebra/Makefile automate making input files

src/doc
   axiom.sty collect all script commands in one place
   axiom.sty consolidate latex macros

src/input
   setcmd.input clean up broken tests

src/interp
   Makefile merge and remove fnewmeta

   apply, br-con, define, info, category, modemap, postprop  fix |special| bug
   vmlisp   fix |special| bug

   compiler, define, fnewmeta, g-boot, interp-proclaims, iterator, modemap,
   parsing, postprop vmlisp treeshake compiler

   fnewmeta.lisp removed
```

```
   i-output.lisp remove droptrailingblanks

   parsing.lisp cleanup and reformat

   vmlisp.lisp comment out plist assignments
   vmlisp.lisp remove droptrailingblanks
   vmlisp.lisp rename some fnewmeta variables
   vmlisp.lisp rename variables to remove underscores

src/scripts/tex
   axiom.sty add defplist, usesstruct

src/doc
   axiom.sty add defplist, usesstruct

src/axiom-website/
   documentation.html add literate thinking quote

\end{verbatim}
\endscroll
\autobuttons
\end{page}
```

September 2010 Release Notes



⇐ “What’s New in Axiom” (releaseNotes) 1.1 on page 1

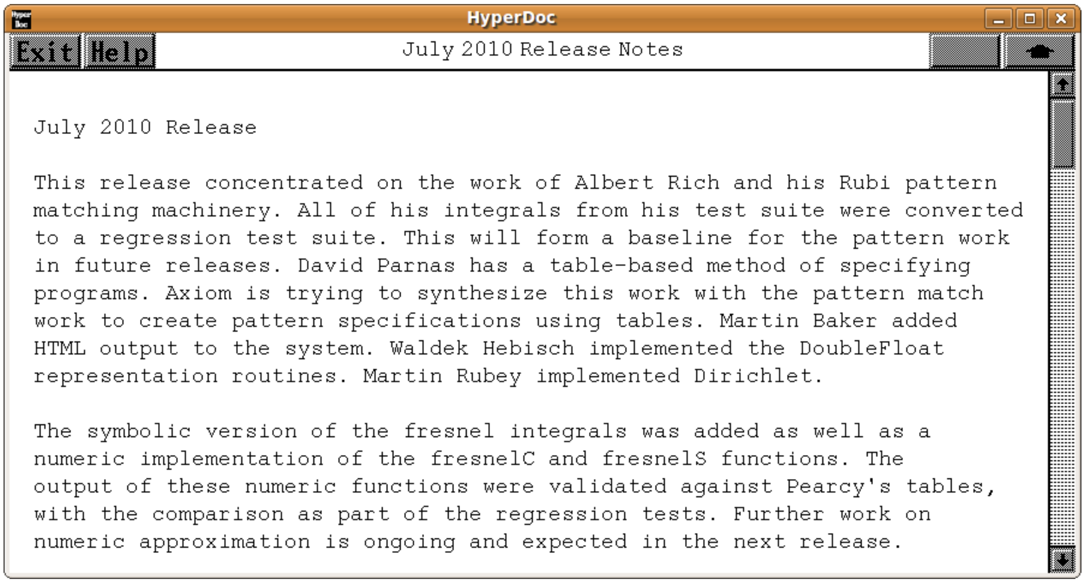
— **releasenotes.ht** —

```
\begin{page}{sept2010}{September 2010 Release Notes}
\beginscroll
\begin{verbatim}
September 2010 Release

This release concentrated on treeshaking the compiler code into book volume 9.
Due to the complexity of this task it will take several releases to complete.

Additional global changes included collecting all of the Axiom-related
latex macros into axiom.sty and removing references to Aldor.

Makefile
  always run help extractions in parallel
  build src/input quietly

books/bookvol0
   remove references to aldor

books/bookvol1
   remove references to aldor

books/bookvol5
   expose StreamTensor, U32Vector, U32VectorPolynomialOperations
   mark pure common lisp routines
   merge ptrop, varini
   move latex macros to axiom.sty
   remove POLYVEC
   remove compile, duplicated in vol9
   remove memq
   remove references to aldor
   treeshake
   remove $useNewParser

books/bookvol7.1
   move latex macros to axiom.sty
   remove references to aldor
   rewrite \pagehead to \pagetitle

books/bookvol9
   cross-reference functions and variables
   move latex macros to axiom.sty
   remove memq
   treeshake the compiler code
   remove $useNewParser

books/bookvol10
```

```
   move latex macros to axiom.sty
   move GOPTO from bookvol10.3

books/bookvol10.1
   move latex macros to axiom.sty

books/bookvol10.2
   move latex macros to axiom.sty

books/bookvol10.3
   add U32Vector, move GOPTO from bookvol10.4
   move latex macros to axiom.sty

books/bookvol10.4
   add StreamTensor, U32VectorPolynomialOperations
   fix ScriptTensor regression test
   move latex macros to axiom.sty
   remove POLYVEC
   update Chinese Remainder documentation

books/bookvol10.5
   move latex macros to axiom.sty

books/bookvolbib
   Parnas & Madey [PM95], Parnas & Jin [PJ10], GCL92, AS64, NIST10, RF94,
   Hamdy [Ham04], Steele [Ste90], Tim Lahey's Sage Integration Test Suite

books/ps/
   v103guessoptionfunctions0, v103u32vector, v104streamtensor,
   v104u32vectorpolynomialoperations, v104u32vectorpolynomialoperations

src/algebra
   Makefile help and test for StreamTensor
   Makefile help and test for U32Vector
   Makefile remove references to aldor
   Makefile test and help for POLYVEC
   Makefile remove POLYVEC
   Makefile add help and test for new algebra
   Makefile handle case-insensitive MAC filesystem
   Makefile remove new algebra scaffolding code

src/doc
   axiom.sty collect all script commands in one place
   axiom.sty consolidate latex macros

src/input
   Makefile add guess.input, manuel.input, risch.input
   guess.input test examples of the GUESS package
   kamke3.input clean up broken tests
   manuel.input add Manuel's integral to test suite
```

```
   richlog300-391.input clean up broken tests
   richtrig800-899.input clean up broken tests
   risch.input illustrate the Risch algorithm
   setcmd.input clean up broken tests
   test.input clean up broken tests

src/interp
   Makefile merge varini
   Makefile remove nspadaux, mark, pspad1, pspad2, ptrop, wi1, wi2
   *.lisp remove memq
   treeshake compiler -- br-con, cattable, compiler
   remove nspadaux.lisp, mark.lisp, pspad1.lisp, pspad2.lisp,
   remove ptrop.lisp, wi1.lisp, wi2.lisp
   add HTMLFormat code i-output.lisp, vmlisp.lisp

src/scripts/tex
   axiom.sty collect all script commands
   axiom.sty consolidate latex macros

src/axiom-website/
   documentation.html add Knuth quote per W. Sit
   download.html add debian, fedora, mandriva, opensuse, slackware, vector
   download.html update ubuntu yum advice



\end{verbatim}
\endscroll
\autobuttons
\end{page}
```

July 2010 Release Notes



⇐ “What’s New in Axiom” (releaseNotes) 1.1 on page 1

— releasenotes.ht —

```
\begin{page}{july2010}{July 2010 Release Notes}
\beginscroll
\begin{verbatim}
July 2010 Release

This release concentrated on the work of Albert Rich and his Rubi pattern
matching machinery. All of his integrals from his test suite were converted
to a regression test suite. This will form a baseline for the pattern work
in future releases. David Parnas has a table-based method of specifying
programs. Axiom is trying to synthesize this work with the pattern match
work to create pattern specifications using tables. Martin Baker added
HTML output to the system. Waldek Hebisch implemented the DoubleFloat
representation routines. Martin Rubey implemented Dirichlet.

The symbolic version of the fresnel integrals was added as well as a
numeric implementation of the fresnelC and fresnelS functions. The
output of these numeric functions were validated against Pearcy's tables,
with the comparison as part of the regression tests. Further work on
numeric approximation is ongoing and expected in the next release.

These people were added to the credits list:

  Albert D. Rich, David Parnas, Martin Baker
```

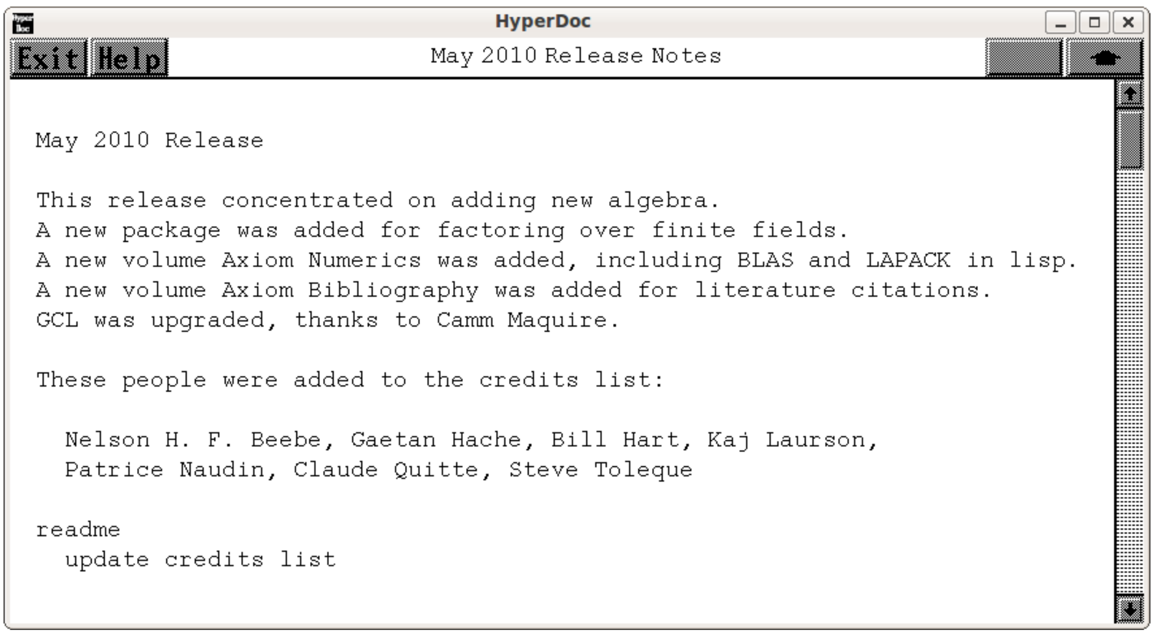
```
readme
  update credits list

Makefile
  add Mandriva
  fix environment variables
  segment test set choice from command line

books/axbook.tgz
  rewrite section 8.3.2 for current output

books/bookvol0
  rewrite section 8.3.2 for current output

books/bookvol10.2
  add fresnelS, fresnelC to LFCAT
  add test and help files for all categories

books/bookvol10.3
  add CDFMAT ComplexDoubleFloatMatrix
  add CDFVEC ComplexDoubleFloatVector
  add DIR Dirichlet
  add DFMAT DoubleFloatMatrix
  add DFVEC DoubleFloatVector
  add HTMLFORM HTMLFormat
  add fresnelS, fresnelC to EXPR

books/bookvol10.4
  add EXP3D Export3D
  add GDRAW  GnuDraw
  add fresnelC, fresnelS numeric computation to DFSFUN
  add fresnelS, fresnelC symbolics to LF, COMMONOP, LIMITPS

books/bookvol10.5
  start of numeric analysis of BLAS/LAPACK
  tighten array type declarations in BLAS
  use simple-array, not simple-vector type in BLAS

books/bookvol4
  document the Makefile build process

books/bookvol5
  change )version to include *build-version* info
  add macros to support algebra
  add Albert D. Rich, David Parnas, Martin Baker to credits

books/bookvol6
  fix deleted variable from axiom script
  remove AXIOMXLROOT reference
```

```
books/bookvol7.1
  document HTMLFormat

books/bookvol8
  add 3D graphic file format table

books/bookvolbib
  Adams [AL94], Brunelli [Bru09], Burcin [ES10], Dewar [Dew94], Golub [GL89],
  Higham [Hig02], Householder [Hou81], Jenks [JT18], [JT18a], [JT18b], [JWS87],
  Lecerf [Le96], Losch [Los60], Luke [Luk169], [Luk269], Pearcey [Pea56],
  Press [PTVF95], Taivalsaari [Tai96]

src/algebra/Makefile
  add help and test for new algebra
  handle case-insensitive MAC filesystem
  remove new algebra scaffolding code

src/doc/Makefile
  cleanly latex pure latex files

src/input/Makefile add TESTSET=notests
  add derivefail, exampleagcode, hyperell, paffexample,
  rule-based algebraic integration, rule-based rational integration,
  rule-based exponential integration, rule-based hyper integration,
  rule-based inverse hyperbolic integration, rule-based log integration,
  rule-based inverse trig integration, rule-based trig integration

zips
  remove aldor.20070901.tgz

src/input
  derivefail, examplegcode, hyperell, monitortest, paffexample,
  richalgebraic000-099, richalgebraic100-199, richalgebraic200-299,
  richalgebraic300-399, richalgebraic400-461, richerror000-078,
  richexponential, richhyper000-099, richhyper100-199, richhyper100-199,
  richhyper1000-1098, richhyper200-299, richhyper300-399,
  richhyper400-499, richhyper500-599, richhyper600-699, richhyper700-799,
  richhyper800-899, richhyper900-999, richintfunc000-032,
  richinvhyper000-099, richinvhyper100-199, richinvhyper200-296,
  richinvtrig000-092, richlog000-099, richlog100-199, richlog200-299,
  richlog300-391, richrational.input, richspecfunc000-022,
  richtrig000-099, richtrig100-199, richtrig200-299, richtrig300-399,
  richtrig400-499, richtrig500-599, richtrig600-699, richtrig700-799,
  richtrig800-899, richtrig900-920

src/interp
  i-output.lisp, vmlisp.lisp HTMLFormat support code

src/share/algebra/
  browse, category, compress, interp, operation, users daase updated
```

```
  remove libaldor.al

GCL was upgraded, thanks to Camm Maquire.
\end{verbatim}
\endscroll
\autobuttons
\end{page}
```

———-

May 2010 Release Notes



⇐ “What’s New in Axiom” (releaseNotes) 1.1 on page 1

— **releasenotes.ht** —

```
\begin{page}{may2010}{May 2010 Release Notes}
\beginscroll
\begin{verbatim}
May 2010 Release

This release concentrated on adding new algebra.
A new package was added for factoring over finite fields.
A new volume Axiom Numerics was added, including BLAS and LAPACK in lisp.
A new volume Axiom Bibliography was added for literature citations.
GCL was upgraded, thanks to Camm Maquire.
```

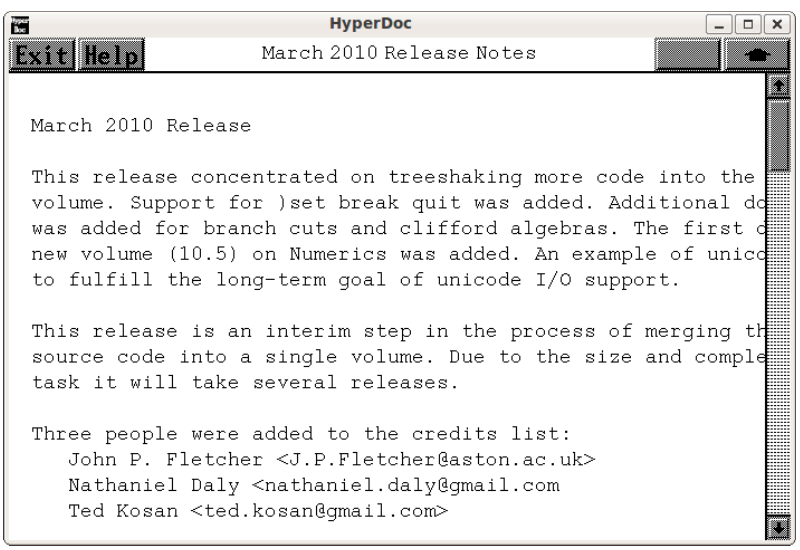
```
These people were added to the credits list:

  Nelson H. F. Beebe, Gaetan Hache, Bill Hart, Kaj Laurson,
  Patrice Naudin, Claude Quitte, Steve Toleque

readme
  update credits list

Makefile
  GCLVERSION gcl-2.6.8pre4 change
  clean books from src/algebra
  merge all fedoraN stanzas into fedora

books/Makefile
  add Axiom Bibliography book
  add bookvol10.5 add Axiom Algebra Numerics

books/bookvol4 Axiom Developers Guide
  document how to make graphs in algebra books
  document process to add new algebra
  document steps for adding algebra

books/bookvol5 Axiom Interpreter
  add Naudin, Laurson, Hache to credits
  change .spad.pamphlet to just .pamphlet
  Add new algebra to exposure list
  fix )describe to accept abbreviations

books/bookvol7.1 Axiom Hyperdoc Pages
  add mar2010 what's new page

books/bookvol10.2 Axiom Categories
  add AFSPCAT  AffineSpaceCategory
  add BLMETCT  BlowUpMethodCategory
  add DIVCAT   DivisorCategory
  add DSTRCAT  DesingTreeCategory
  add INFCLCT  InfinitlyClosePointCategory
  add LOCPOWC  LocalPowerSeriesCategory
  add PACEXTC  PseudoAlgebraicClosureOfAlgExtOfRationalNumberCategory
  add PACFFC   PseudoAlgebraicClosureOfFiniteFieldCategory
  add PACPERC  PseudoAlgebraicClosureOfPerfectFieldCategory
  add PACRATC  PseudoAlgebraicClosureOfRationalNumberCategory
  add PLACESC  PlacesCategory
  add PRSPCAT  ProjectiveSpaceCategory
  add SETCATD  SetCategoryWithDegree

books/bookvol10.3 Axiom Domains
  add AFFPL    AffinePlane
  add AFFPLPS  AffinePlaneOverPseudoAlgebraicClosureOfFiniteField
```

```
  add AFFSP    AffineSpace
  add BLHN     BlowUpWithHamburgerNoether
  add BLQT     BlowUpWithQuadTrans
  add DIV      Divisor
  add DSTREE   DesingTree
  add ICP      InfClsPt
  add INFCLSPS InfinitlyClosePointOverPseudoAlgebraicClosureOfFiniteField
  add INFCLSPT InfinitlyClosePoint
  add NSDPS    NeitherSparseOrDensePowerSeries
  add PACEXT   PseudoAlgebraicClosureOfAlgExtOfRationalNumber
  add PACOFF   PseudoAlgebraicClosureOfFiniteField
  add PACRAT   PseudoAlgebraicClosureOfRationalNumber
  add PLACES   Places
  add PLACESPS PlacesOverPseudoAlgebraicClosureOfFiniteField
  add PLCS     Plcs
  add PROJPL   ProjectivePlane
  add PROJPLPS ProjectivePlaneOverPseudoAlgebraicClosureOfFiniteField
  add PROJSP   ProjectiveSpace
  add UTSZ     UnivariateTaylorSeriesCZero
  document and test Interval

books/bookvol10.4 Axiom Packages
  add AFALGGRO AffineAlgebraicSetComputeWithGroebnerBasis
  add AFALGRES AffineAlgebraicSetComputeWithResultant
  add BLUPPACK BlowUpPackage
  add DTP      DesingTreePackage
  add FACTEXT  FactorisationOverPseudoAlgebraicClosureOfAlgExtOfRationalNumber
  add FACTRN   FactorisationOverPseudoAlgebraicClosureOfRationalNumber
  add FFFACTSE FiniteFieldFactorizationWithSizeParseBySideEffect
  add FFSQFR   FiniteFieldSquareFreeDecomposition
  add GPAFF    GeneralPackageForAlgebraicFunctionField
  add INTDIVP  IntersectionDivisorPackage
  add INTERGB  InterfaceGroebnerPackage
  add INTFRSP  InterpolateFormsPackage
  add LISYSER  LinearSystemFromPowerSeriesPackage
  add LOP      LinesOpPack
  add LPARSPT  LocalParametrizationOfSimplePointPackage
  add NPOLYGON NewtonPolygon
  add PAFF     PackageForAlgebraicFunctionField
  add PAFFFF   PackageForAlgebraicFunctionFieldOverFiniteField
  add PARAMP   ParametrizationPackage
  add PFORP    PackageForPoly
  add PLPKCRV  PolynomialPackageForCurve
  add PRJALGPK ProjectiveAlgebraicSetPackage
  add RFP      RootsFindingPackage

books/bookvol10.5 Axiom Numerics
  BLAS1 regress, help, and function documentation
  add BLAS1 dasum function
  add BLAS1 daxpy
```

```
  add BLAS1 dcopy

books/bookvolbib Axiom Bibliography
  Buh05 DLMF Mah05 Sei95 Seixx Sch92 SCC92 WJST90
  Du95, Ga95, Ha95, Ha96, HI96, HL95, LR88, St93
  add Assia Mahboubi [Mah05]
  add Soren L. Buhl [Buh05]
  add citation SDJ07
  rename and align biblio with bookvol10.1

faq
  FAQ 52: Who was User?

lsp/Makefile.pamphlet
  GCLVERSION gcl-2.6.8pre4

src/Makefile
  add Volume 10.5 Axiom Numerics
  change .spad.pamphlet to .pamphlet

src/algebra/Makefile
  help and test files for all new algebra
  remove unused .as.pamphlet files
    axtimer.as, ffrac.as, herm.as, interval.as, invnode.as,
    invrender.as, invtypes.as, invutils.as, iviews.as,
    mlift.spad.jhd, ndftip.as, nepip.as, noptip.as, nqip.as,
    nrc.as, nsfip.as removed

src/input/Makefile
  remove duplicate curl.input invocation
  add curry.input, davenport.input, liska.input, paff.input, zimmer.input
  fix biquat.input, chtheorem.input, chtheorem.input, cmds.input,
      complexfactor.input,dfloat.input, dftrig.input, dop.input,
      e1.input, ei.input, en.input, gamma.input, grpthry.input,
      gstbl.input, ico.input, numericgamma.input, test.input,
      unit-i-funsel.input, unittest1.input, unittest1.input,
      unittest2.input, unittest2.input

src/interp/Makefile
  add Volume 10.5 Axiom Numerics

src/share/algebra
  update browse.daase, category.daase, compress.daase, users.daase,
         dependents.daase, interp.daase, libdb.text, operation.daase

zips
  gcl-2.6.8pre4.tgz added, fix for ubuntu 9.10
  gcl-2.6.8pre4.h.linux.defs.patch added
  gcl-2.6.8pre4.o.read.d.patch added
  gcl-2.6.8pre4.unixport.init_gcl.lsp.in.patch
```

```
  gcl-2.6.8pre4.unixport.makefile.patch

src/axiom-website/documentation.html
  add Axiom Algebra Numerics
  literate programming quotes

src/axiom-website/download.html
  add march 2010 ubuntu
  update available binary list

\end{verbatim}
\endscroll
\autobuttons
\end{page}
```

March 2010 Release Notes



⇐ "What's New in Axiom" (releaseNotes) 1.1 on page 1

— releasenotes.ht —

```
\begin{page}{mar2010}{March 2010 Release Notes}
\beginscroll
\begin{verbatim}
March 2010 Release

This release concentrated on treeshaking more code into the Interpreter
```

volume. Support for)set break quit was added. Additional documentation was added for branch cuts and clifford algebras. The first draft of a new volume (10.5) on Numerics was added. An example of unicode was tested to fulfill the long-term goal of unicode I/O support.

This release is an interim step in the process of merging the interpreter source code into a single volume. Due to the size and complexity of the task it will take several releases.

Three people were added to the credits list:
   John P. Fletcher <J.P.Fletcher@aston.ac.uk>
   Nathaniel Daly <nathaniel.daly@gmail.com
   Ted Kosan <ted.kosan@gmail.com>

The Axiom website was completely rewritten based on a new style css provided by Nate. This change was also made to axiom-developer.com.

There is a new set option ")set break quit" per Ralf Hemmecke. This will cause the interpreter to quit on failure. Documentation was added to the books.

books/bookvol4 Axiom Developers Guide
   add)set break quit

books/bookvol5 Axiom Interpreter
   add)set break quit
   add support for IndexedBits
   begin documentation of macex
   merge and remove lisp files from src/interp
   remove non-common lisp macros
   rewrite to common lisp functions

books/bookvol7.1 Axiom Hyperdoc Pages
   add jan2010 what's new page
   books/ps/v71jan2010.eps added
   books/ps/v71releasenotes.eps updated for January 2010

books/bookvol8 Axiom Graphics
   redefine R1 in view3D for ARM processor

books/bookvol10 Axiom Algebra Implementation
   A new section on Elementary Functions branch cuts based on the
   Numerical Mathematics Consortium was added.

books/bookvol10.1 Axiom Algebra Theory
   A new chapter on the Clifford algebra was added.
   A quote on quaternions from Altman was added

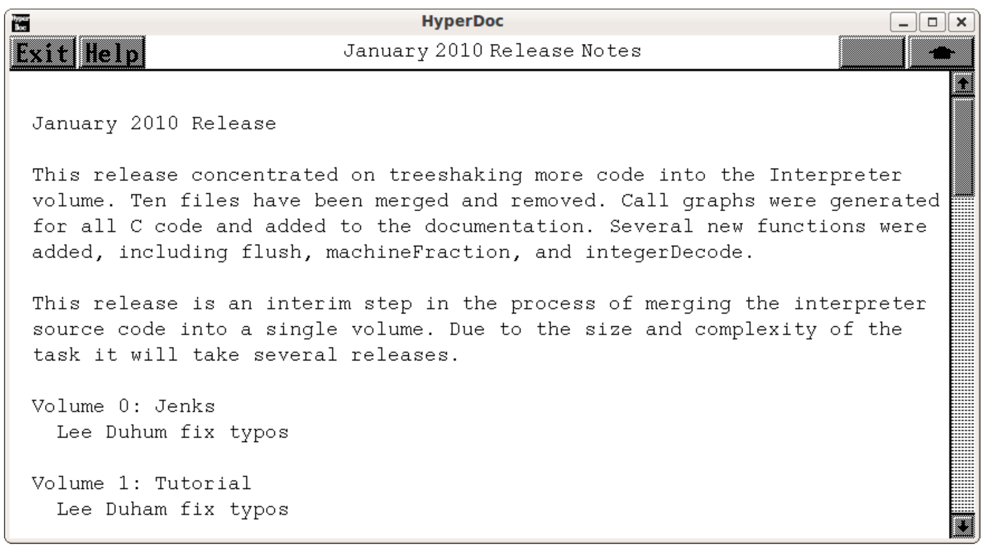
books/bookvol10.3 Axiom Domains
   defstream function and KAFILE test bug fixed

```
   fix IndexedBits range error
   remove non-common lisp macros
   rewrite to common lisp functions

books/bookvol10.4 Axiom Packages
   add Ted Kosan to credits
   fix broken credit test

books/bookvol10.5 Axiom Numerics
   first draft of numerics volume

faq
   FAQ 51: How can I do unicode in xterm?

src/algebra/Makefile
   unit test IndexedBits

src/input/Makefile
   add monitortest
   remove redundant kafile.input

src/input/
   cachedf.input     fix tests for )set break quit
   kafile.input      redundant with KAFILE test, removed
   monitortest.input unit test monitor code
   pmint.input       add comments
   pmint.input       update pmint with code
   setcmd.input      add )set break quit
   textfile.input    --I out failing test
   unittest2.input   add Nate Daly to credits
   wester.input      reformat into regression test file

src/interp
    remove compat.lisp, cparse.lisp, intint.lisp, macex.lisp,
           monitor.lisp, pf2sex.lisp, ptrees.lisp, serror.lisp

    remove MAKESTRING macro from all files

    remove non common lisp macros

    remove unused functions

    g-error.lisp add )set break quit
    htcheck.lisp add READLINE from compat.lisp
    nci.lisp pick up functions from intint
    patches.lisp move global variables to bookvol5
    posit.lisp move position functions to bookvol5
    util.lisp move global variables to bookvol5
    varini.lisp pick up functions from intint
    vmlisp.lisp add )set break quit
```

```
    vmlisp.lisp fix )set break resume bug
    vmlisp.lisp remove some define-functions

zips
   utf-8-demo.txt added to demo utf-8 I/O

\end{verbatim}
\endscroll
\autobuttons
\end{page}
```

———

January 2010 Release Notes



⇐ "What's New in Axiom" (releaseNotes) 1.1 on page 1

— **releasenotes.ht** —

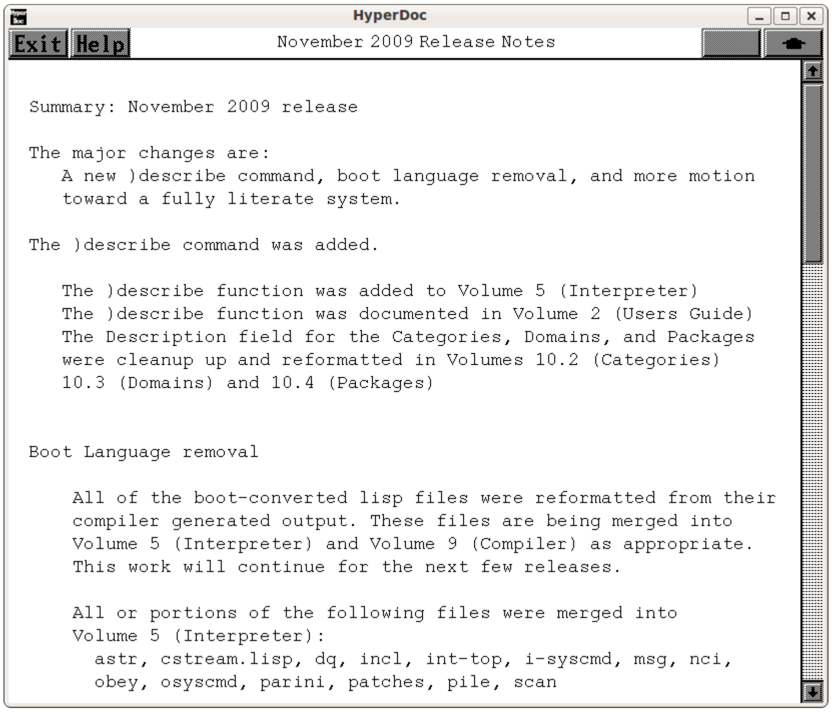
```
\begin{page}{jan2010}{January 2010 Release Notes}
\beginscroll
\begin{verbatim}
January 2010 Release

This release concentrated on treeshaking more code into the Interpreter
volume. Ten files have been merged and removed. Call graphs were generated
for all C code and added to the documentation. Several new functions were
added, including flush, machineFraction, and integerDecode.
```

```
This release is an interim step in the process of merging the interpreter
source code into a single volume. Due to the size and complexity of the
task it will take several releases.

Volume 0: Jenks
  Lee Duhum fix typos

Volume 1: Tutorial
  Lee Duham fix typos

Volume 5: Interpreter
   )describe no longer needs cat, dom, pkg arg
   add Lee Duham to credits list
   add banner and cleanup globals
   add DoubleFloat trig macros
   support DFLOAT machineFraction, integerDecode
   do not set si::*system-directory* in restart
   fix )display all output bug
   merge fname.lisp, alql.lisp, i-syscmd, i-toplev, pathname.lisp
   merge daase.lisp, exposed.lsp

Volume 7: Hyperdoc
   add call graph for ex2ht, htadd, hthits, hypertex, spadbuf

Volume 8: Graphics
   add call graph for view2d, view3d, viewalone,  viewman

Volume 10.2 Categories
  latex cleanup
  FileCategory add flush

Volume 10.3 Domains
  DoubleFloat add dfloat machineFraction, integerDecode
  DoubleFloat rewrite doublefloat to use typed macros
  File add flush

Volume 10.4 Packages
  document RepeatedSquaring
  fix API regression

lsp/Makefile add compiler::link per Camm Maguire

src/algebra
   remove exposed.lsp

src/axiom-website
   add fedora 10 nov2009 build
   add nov2009 builds
   move CVS instruction to git
   axbook add Stack and Queue
```

```
   axbook fix typos

src/input
   add ackerman to test function caching
   add dftrig to test DoubleFloat trig changes
   rewrite dfloat, dop, e1, ei, en, numericgamma using machineFraction
   r20bugs, setcmd, unittest1, unitest2 fix broken tests
   zimmbron fix typo in bibliography

src/interp
   remove alql, cformat, daase, fname, i-sysmcd, i-toplev, intfile,
          pathname, packtran

other src files
   add call graph information edible.c, asq.c
   src/doc/msgs/s2-us.msgs remove S2IZ0049D
   src/share/doc/msgs/s2-us.msgs removed

\end{verbatim}
\endscroll
\autobuttons
\end{page}
```

———-

November 2009 Release Notes



⇐ “What’s New in Axiom” (releaseNotes) 1.1 on page 1

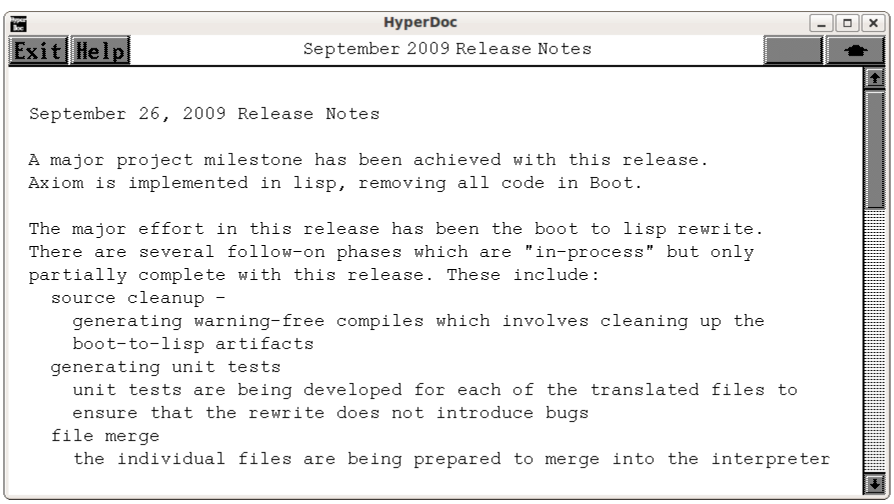
— releasenotes.ht —

```
\begin{page}{nov2009}{November 2009 Release Notes}
\beginscroll
\begin{verbatim}
Summary: November 2009 release

The major changes are:
   A new )describe command, boot language removal, and more motion
   toward a fully literate system.

The )describe command was added.

   The )describe function was added to Volume 5 (Interpreter)
   The )describe function was documented in Volume 2 (Users Guide)
   The Description field for the Categories, Domains, and Packages
   were cleanup up and reformatted in Volumes 10.2 (Categories)
   10.3 (Domains) and 10.4 (Packages)


Boot Language removal
```

```
    All of the boot-converted lisp files were reformatted from their
    compiler generated output. These files are being merged into
    Volume 5 (Interpreter) and Volume 9 (Compiler) as appropriate.
    This work will continue for the next few releases.

    All or portions of the following files were merged into
    Volume 5 (Interpreter):
      astr, cstream.lisp, dq, incl, int-top, i-syscmd, msg, nci,
      obey, osyscmd, parini, patches, pile, scan

    The src/boot subdirectory is gone, including the files:
      Makefile, boot-proclaims, boothdr, bootload, btincl2, btpile2,
      btscan2, ccl-bootsys, ccl-depsys.lsp, exports.lisp, npextras, ptyout,
      tyextra, typars, typrops, tytree1, sys-pkg, vmlisp, ptrees, wi2

    The bootsys image is no longer part of the build process


Patch ports from Fricas and Open-Axiom

    The Tuples patch was picked up and applied.

    The SXHASH function is the hash default in SetCategory

    The function ListOfTerms was renamed to listOfTerms


Input file changes

    New input files have been added to show how to compute
    results using Axiom or to create regression tests for fixes:
      complexfactor, rubey, zimmbron, branchcut, cachedf, finitegraph,
      newtonlisp, nonlinhomodiffeq, distexpr, numericgamma, donsimple
      solveperf, tuplebug, unit-macro, testprob, unittest2

    lexp was removed and moved to the LEXP algebra file

    dop and gstbl had minor fixes

New Help files and Function examples were added

    There are new help files:
      describe, AlgebraicallyClosedField, RationalFunctionSum,
      RadicalSolvePackage, PartialFractionPackage, Product,
      OrderedFreeMonoid

Website update:

  The developer.html page was rewritten. An old Scratchpad group
```

```
  photo was added to the site.

  Work continues on re-hosting the axiom-developer.org domain.

  We now own axiom-developer.com and axiom-developer.net which will
  be retargetted to the new host as soon as it is available

Interpreter changes:

    Axiom will sit in a single package in the near future.
    The VMLISP package was partially removed from the system.
    Work continues on this path.

    The util.ht file is created earlier in the parallel build
    so there are fewer compiler messages about documentation.

Research:

    A Cohen algebra domain is being developed to enable symbolic
    manipulation of expressions with explanations and controlled
    simplification.

\end{verbatim}
\endscroll
\autobuttons
\end{page}
```

September 2009 Release Notes

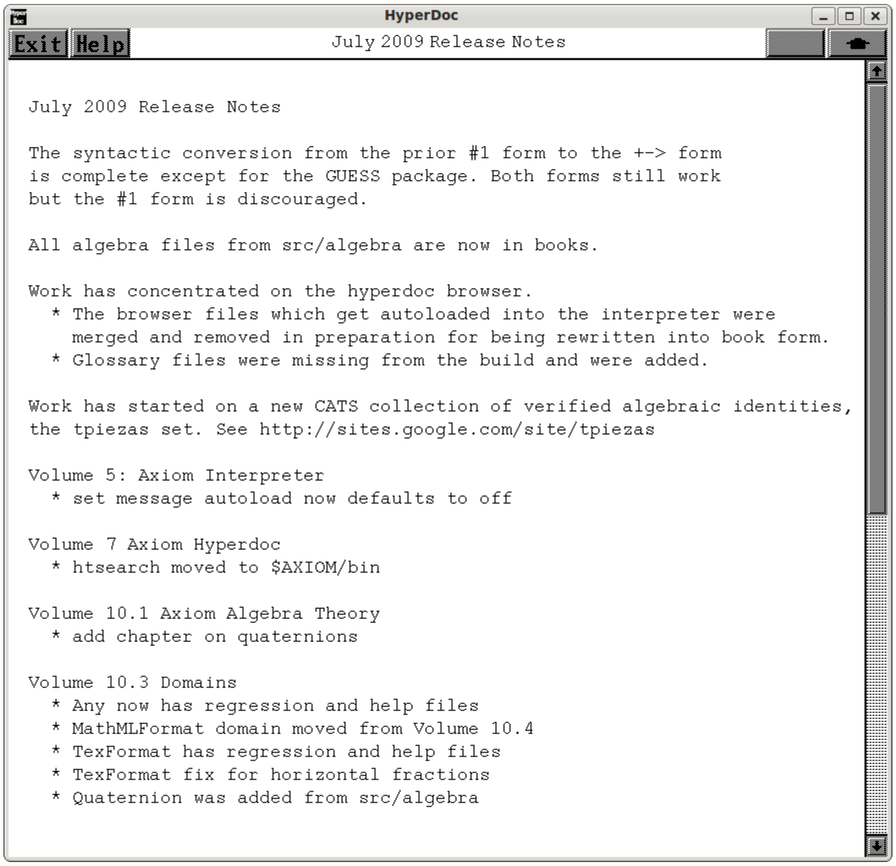


⇐ "What's New in Axiom" (releaseNotes) 1.1 on page 1
— **releasenotes.ht** —

```
\begin{page}{sept2009}{September 2009 Release Notes}
\beginscroll
\begin{verbatim}
September 26, 2009 Release Notes

A major project milestone has been achieved with this release.
Axiom is implemented in lisp, removing all code in Boot.

The major effort in this release has been the boot to lisp rewrite.
There are several follow-on phases which are "in-process" but only
partially complete with this release. These include:
  source cleanup -
    generating warning-free compiles which involves cleaning up the
    boot-to-lisp artifacts
  generating unit tests
    unit tests are being developed for each of the translated files to
    ensure that the rewrite does not introduce bugs
  file merge
    the individual files are being prepared to merge into the interpreter
    and compiler volumes.
  global variable handling
    the system uses many globals which are being either explained or
    rewritten to reduce coupling
  data structure extraction
    the various data structures used by the compiler and interpreter
    are being identified and explained
  API vs implementation
    the functions used externally vs the functions used internally are
    being identified
  code refactoring
    the generated lisp code is being rewritten into human form without
    changing the behavior

The target goal is to have the code cleanly merged into the interpreter
and compiler volumes in a logical way that form chapters related to function.

Additional changes in this release include:

  Add Steven Segletes to credits. He was the author of the paper that
  supplied the coefficients for computing E1. He has contributed a later,
  unpublished paper which has more accurate coefficients. This is in-plan
  to implement.

  Barry Trager contributed an example of computing Shannon Matrices.
  This has been added to the input/regression suite.
```

```
  The .dvi files for the interpreter are no longer being built. The
  individual pamphlets will shortly disappear.

  Debugsys is no longer built. I am the only user of it and I can build
  it when needed.

  Bootsys is no longer built. It is no longer needed.

  The Makefiles now use the := rather than = assignment limiting re-evals

  Parallel Make is now the default since the documentation build is now
  independent of the interpreter build

\end{verbatim}
\endscroll
\autobuttons
\end{page}
```

July 2009 Release Notes




```
July 2009 Release Notes

The syntactic conversion from the prior #1 form to the +-> form
is complete except for the GUESS package. Both forms still work
but the #1 form is discouraged.

All algebra files from src/algebra are now in books.

Work has concentrated on the hyperdoc browser.
  * The browser files which get autoloaded into the interpreter were
    merged and removed in preparation for being rewritten into book form.
  * Glossary files were missing from the build and were added.

Work has started on a new CATS collection of verified algebraic identities,
the tpiezas set. See http://sites.google.com/site/tpiezas

Volume 5: Axiom Interpreter
  * set message autoload now defaults to off

Volume 7 Axiom Hyperdoc
  * htsearch moved to $AXIOM/bin

Volume 10.1 Axiom Algebra Theory
  * add chapter on quaternions

Volume 10.3 Domains
  * Any now has regression and help files
  * MathMLFormat domain moved from Volume 10.4
  * TexFormat has regression and help files
  * TexFormat fix for horizontal fractions
  * Quaternion was added from src/algebra
```

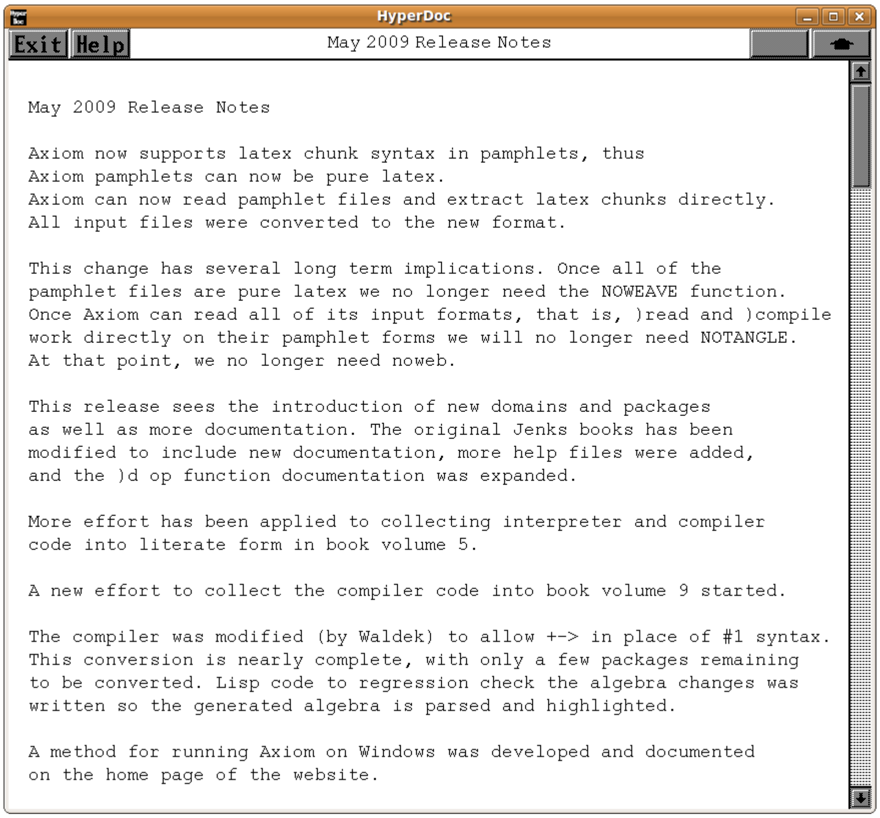
⇐ “What’s New in Axiom” (releaseNotes) 1.1 on page 1

— **releasenotes.ht** —

```
\begin{page}{july2009}{July 2009 Release Notes}
\beginscroll
\begin{verbatim}
July 2009 Release Notes

The syntactic conversion from the prior #1 form to the +-> form
is complete except for the GUESS package. Both forms still work
but the #1 form is discouraged.

All algebra files from src/algebra are now in books.

Work has concentrated on the hyperdoc browser.
  * The browser files which get autoloaded into the interpreter were
    merged and removed in preparation for being rewritten into book form.
```

```
  * Glossary files were missing from the build and were added.

Work has started on a new CATS collection of verified algebraic identities,
the tpiezas set. See http://sites.google.com/site/tpiezas

Volume 5: Axiom Interpreter
  * set message autoload now defaults to off

Volume 7 Axiom Hyperdoc
  * htsearch moved to $AXIOM/bin

Volume 10.1 Axiom Algebra Theory
  * add chapter on quaternions

Volume 10.3 Domains
  * Any now has regression and help files
  * MathMLFormat domain moved from Volume 10.4
  * TexFormat has regression and help files
  * TexFormat fix for horizontal fractions
  * Quaternion was added from src/algebra

Volume 10.4 Packages
   * IntegerNumberTheoryFunctions fix divisors regression
   * Waldek's QUATCT2 algebra was added, including a help file,
     regression tests and command examples.

Makefile
  * make help files in parallel
  * make xhtml pages in parallel
  * make books documentation in parallel

interpreter
  * src/interp/i-output.boot fix horizontal fractions

axiom-website
  * download.html add mandriva
  * download.html add may2009 binaries

input files
  * ffdemo.input fix steps 27, 57 due to divisors change
  * numbers.input fix random zero failure
  * spline.input explain how to compute 2D splines
  * tpiezas001.input CATS tests of algebraic identities
  * tpiezas002.input CATS tests of algebraic identities

\end{verbatim}
\endscroll
\autobuttons
\end{page}
```

May 2009 Release Notes



⇐ "What's New in Axiom" (releaseNotes) 1.1 on page 1

— releasenotes.ht —

```
\begin{page}{may2009}{May 2009 Release Notes}
\beginscroll
\begin{verbatim}
May 2009 Release Notes

Axiom now supports latex chunk syntax in pamphlets, thus
Axiom pamphlets can now be pure latex.
Axiom can now read pamphlet files and extract latex chunks directly.
All input files were converted to the new format.

This change has several long term implications. Once all of the
pamphlet files are pure latex we no longer need the NOWEAVE function.
Once Axiom can read all of its input formats, that is, )read and )compile
```

```
work directly on their pamphlet forms we will no longer need NOTANGLE.
At that point, we no longer need noweb.

This release sees the introduction of new domains and packages
as well as more documentation. The original Jenks books has been
modified to include new documentation, more help files were added,
and the )d op function documentation was expanded.

More effort has been applied to collecting interpreter and compiler
code into literate form in book volume 5.

A new effort to collect the compiler code into book volume 9 started.

The compiler was modified (by Waldek) to allow +-> in place of #1 syntax.
This conversion is nearly complete, with only a few packages remaining
to be converted. Lisp code to regression check the algebra changes was
written so the generated algebra is parsed and highlighted.

A method for running Axiom on Windows was developed and documented
on the home page of the website.

Unit testing of interpreter and compiler changes was added to the
regression testing. This will expand in future versions as changes
are made so ensure that things continue to function.

Volume 0: Axiom Jenks and Sutor
  Richard Jenks bio was added

  Section 9: Examples of Domains and Packages expanded
     ApplicationProgramInterface (API)
     Dequeue (DEQUEUE)
     EuclideanGroebnerBasisPackage
     GeneralDistributedMultivariatePolynomial
     GroebnerPackage
     HomogeneousDistributedMultivariatePolynomial
     NottinghamGroup
     Permutation
     RealSolvePackage
     TwoDimensionalViewport

  Cross references were fixed

  More spelling typo fixes

Volume 4: Axiom Developers Guide
  A section was added explaining how to find anonymous function signatures

Volume 5: Axiom Interpreter
  Michael Becker was added to the credit list
```

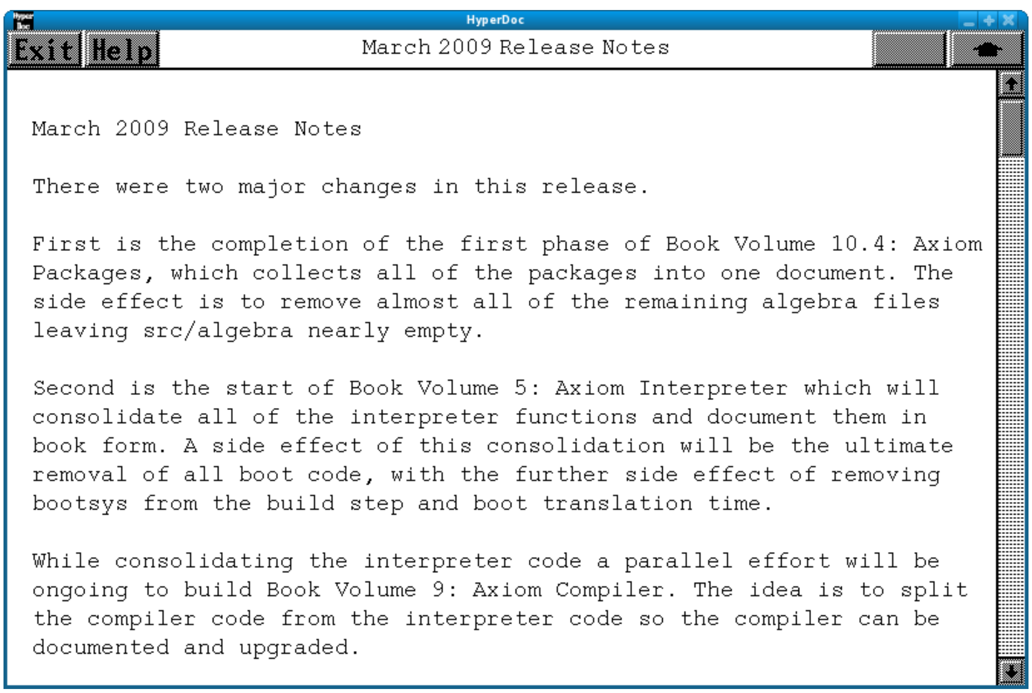
```
  )set debug is now a new top level command to collect all of
  the developer-level commands to enable/disable internal messages
  and tracing functions.

  \defunsec macro was added to handle section titles with docstrings

  More interpreter code was moved from other files into this book

  All regression test numbering was fixed to conform to the new,
  stronger regression test. Thus --S requires the "of NN" phrase
  or a FAILED message will be raised.

  The compiler root was moved to volume 9: Axiom Compiler

Volume 7: Axiom Hyperdoc
  Fixup verbatim breakage

Volume 7.1: Axiom Hyperdoc Pages
  Update What's New for March 2009

Volume 9: Axiom Compiler
  The compiler code is being collected from other parts of the
  system into this book. In particular,
    The root functions for the compiler were added
    The top level loop code of the compiler was added
    The next layer of compiler code was added

Volume 10.2: Axiom Algebra Categories
  There was a general change to allow +-> syntax in algebra and
  this was made to all existing categories

  MatrixCategory now has regression test cases, a help page,
  and )d op function examples

  Document binomial category

Volume 10.3: Axiom Algebra Domains
  There was a general change to allow +-> syntax in algebra and
  this was made to all existing domains

  SparseMultivariateTaylorSeries now has regression test cases, a help page,
  and )d op function examples

  Permutation now has regression test cases, a help page,
  and )d op function examples

  The regression tests now conform to the new, stronger
  regression test requirements

Volume 10.4: Axiom Algebra Packages
```

```
  There was a general change to allow +-> syntax in algebra and
  this was made to some existing packages. There is more to be done.

  Bezier package was added, including regression test cases, a help page,
  and )d op function examples

  CombiniatorialFunction now has regression test cases, a help page,
  and )d op function examples

  ElementaryFunction now has regression test cases, a help page,
  and )d op function examples

  IntegerCombinatoricFunctions now has regression test cases, a help page,
  and )d op function examples

  LazardSetSolvingPackage now has regression test cases, a help page,
  and )d op function examples

  The regression tests now conform to the new, stronger
  regression test requirements

Volume 12: Axiom Crystal
  Add Gelernter's observations on Layout and Linking

Bug fixes
   7191: set *system-directory* dynamically     (20090413.03.tpd.patch)
   7192: )edit now works because SPADEDIT added (20090414.02.tpd.patch)
   7197: fix hyperdoc/graphics failure          (20090530.01.tpd.patch)

Makefile changes
  Top Level Makfile
    The BOOKS environment shell variable was added to ENV
    Makefile.slackware chunk exists
    Makefile reports regression failures after builds

  lsp/Makefile
    Build tangle into the lisp image

  src/Makefile
    Copy bookvol9, the compiler,  to src/interp at build time

  src/algebra/Makefile
    Add Bezier package
    Add LazardSetSolvingPackage.help
    Add MatrixCategory.input, .help
    Add help, regress for ElementaryFunction
    Add input, help, examples for SparseMultivariateTaylorSeries
    Fix LazardSetSolvingPackage typo
    Move egrep to grep -E since egrep might not be installed
    Move help to bookvol5
```

```
  src/doc/Makefile
    Move help to bookvol5

  src/input/Makefile
    Add FRAC regression test
    Add unittest* files for regression testing non-algebra changes
    Change input files to latex tangle
    Fix regress format to conform to the new, improved regression testing

  src/interp/Makefile
    Build bookvol9, the compiler from book sources
    The old gclweb code was rewritten into tangle.lisp, gclweb was removed
    Move help to bookvol5
    More interpreter code was moved to bookvol5, the interpreter
    Debugsys has been updated to stop loading deleted files
    Several files were merged, rewritten, and removed:
      apply.boot, bootlex.lisp, comp.lisp, compiler.boot, cstream.boot,
      def.lisp, i-util.boot, incl.boot, int-top.boot, metalex.lisp,
      nci.lisp, parse.boot, parsing.lisp, postpar.boot, preparse.lisp,
      server.boot, setq.lisp, sockio.lisp, spad.lisp, spaderror.lisp,
      util.lisp, vmlisp.lisp

faq
  faq 50: Cannot find libXpm.a

readme
  Add Michael Becker to credit list

books/
  tangle.lisp lisp version of tangle command
  ps/v71mar2009.ps hypertex march 2009 release notes pic
  ps/v71releasenotes.ps hypertex what's new page pic


src/interp
  Rewrite apply.boot to apply.lisp
  Regression tests, regress.lisp, now checks for the "of NN" phrase

src/algebra
  Add Bezier package to exposed.lsp

src/axiom-website
  Add March 2009 column
  Add slackware column
  Axiom on Windows as html instructions
  Add March release notes
  Add binaries

src/doc/
```

```
  Add chunk environment to axiom.sty
  Old book sources were removed
  The spadhelp file was removed and added to the interpreter book

src/input/
  All of the .input files were modified to use latex tangle

src/scripts/tex/
  axiom.sty contains new latex macros

zips
  Old GCL versions were removed
    gcl-2.6.8pre.tgz removed
    gcl-2.6.7.tgz removed

\end{verbatim}
\endscroll
\autobuttons
\end{page}
```

March 2009 Release Notes



⇐ “What’s New in Axiom” (releaseNotes) 1.1 on page 1

— **releasenotes.ht** —

```
\begin{page}{mar2009}{March 2009 Release Notes}
\beginscroll
\begin{verbatim}
March 2009 Release Notes

There were two major changes in this release.

First is the completion of the first phase of Book Volume 10.4: Axiom
Packages, which collects all of the packages into one document. The
side effect is to remove almost all of the remaining algebra files
leaving src/algebra nearly empty.

Second is the start of Book Volume 5: Axiom Interpreter which will
consolidate all of the interpreter functions and document them in
book form. A side effect of this consolidation will be the ultimate
removal of all boot code, with the further side effect of removing
bootsys from the build step and boot translation time.

While consolidating the interpreter code a parallel effort will be
ongoing to build Book Volume 9: Axiom Compiler. The idea is to split
the compiler code from the interpreter code so the compiler can be
documented and upgraded.

Book Volume 0: Axiom Jenks book
  Add )include documentation

    The original book did not document the )include command. This is
    now documented.

  Add UnivariateSkewPolynomial

    This domain was added to the domain examples

Book Volume 4: Axiom Developers Guide
  Hyperdoc

    A Hyperdoc tutorial on making new pages was written

    A tutorial on spadhelp was added

Book Volume 5: Axiom Interpreter
  Command Handling

    The top level commands that start with open parenthesis are all
    kept in the $setOptions data structure. This structure was documented.
    Each command has a handler routine which was integrated with the
```

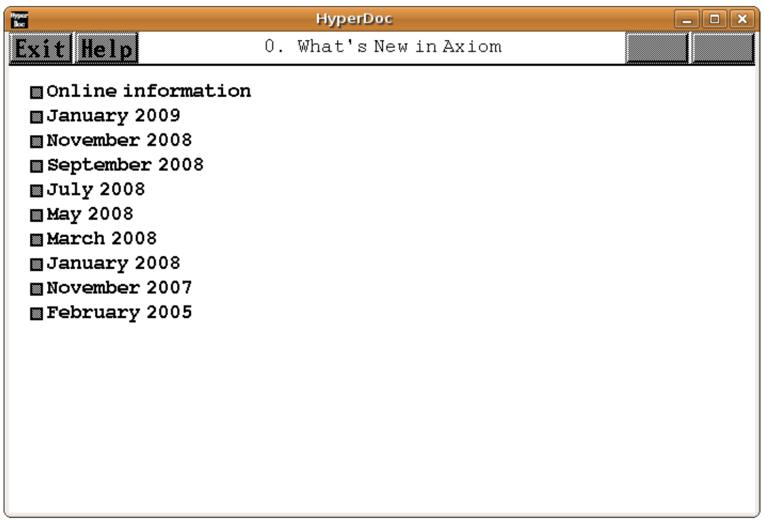
```
    structure in the proper place.

  New latex macros

    The \defun and \defmacro macros were added to ensure that
    there is a uniform handling of subsections and cross references
    for all functions and macros.

    The \defdollar macro was added to handle $foo variables in
    both index and cross reference.

    The \cmdhead macro was added to provide uniform handling of
    each of the top level commands in the $setOptions data structure

  Unit testing

    To ensure that changes did not break behavior two new regression
    test routines were added, unittest1 and setcmd. These test the
    top level command behavior.

  Trace functions added

    The trace.boot code was added.

Book Volume 7: Axiom Hyperdoc
  Documentation

    Add Scott Morrison's original hypertex plan notes

Book Volume 10.1: Axiom Algebra Theory
  Documentation

    Add tutorial chapter on Singular Value Decomposition

Book Volume 10.3 Axiom Domains
  Algebra fixes

    Johannes Grabmeier fixed the outputFixed behavior in Float.

    A regression test chunk was added to the Float domain.

    UnivariateSkewPolynomial was documented. A regression test,
    help file, and examples were added.

    Heap was documented. A regression test, help file, and
    examples were added.

    Dequeue was documented. A regression test, help file, and
    examples were added.
```

```
    Queue was documented. A regression test, help file, and
    examples were added.

    ArrayStack was documented. A regression test, help file, and
    examples were added.

    Stack was documented. A regression test, help file, and
    examples were added.

    NottinghamGroup was added, including regression and help files.
    Originally written by Martin Rubey.

  Regression tests

    The )sys rm now uses the -f flag to force deletion on all
    regression test files

Book Volume 10.4 Axiom Packages
  New Algebra

    All packages were moved into this volume. Almost all of the
    remaining algebra files were deleted.

    ApplicationProgramInterface (API) was added. It exposes the
    Axiom internal functions at the algebra level.

    UnivariateSkewPolynomial now has help, input, and examples

  Documentation

    The nag man pages were added to the related domain documentation

    Export lists were added for all packages.

  Regression tests

    The )sys rm now uses the -f flag to force deletion on all regression
    test files.

Boot to Lisp rewrite
  Deleted boot files

    setvars.boot is gone. The code has been rewritten and added to
    book volume 5. The code was rewritten and/or reformatted. All
    compile-time warnings were eliminated.

    setvart.boot is gone. The code has been rewritten and added to
    book volume 5. The code was rewritten and/or reformatted. All
    compile-time warnings were eliminated.
```

```
    trace.boot is gone. The code has been rewritten and added to
    book volume 5. The code was rewritten and/or reformatted. All
    compile-time warnings were eliminated.

Regression test
  cleanup
    )set break resume

     A subtle bug in regression testing caused by early exit of a
     test left some of the regression tests reporting success when
     there should have been failure. This was fixed by adding a
     )set break resume command to allow Axiom to continue the testing
     in the presence of an internal fault.

  new

     setcmd.input was added to regression test changes to the )set
     top level command

     unittest1.input was added to regresstion test other top level
     commands

     dop.input was added to unit test )d op example output

     frame.input was added to unit test frame handling

  removed

     bags.input was distributed among the algebra regression test
     files and bags.input was removed.

Bug fixes

  Bug 7183. The library handling code was repaired. Functions which do
  not exist were removed (e.g. input-libraries). The element values of
  the data structure are now pathnames.

  Bug 7182. )set mes auto did not indicate the current value.

  Bug 7180: )set compiler input add foo triggers a call to open-library
  which was a CCL function and no longer implemented. The elements of
  the data structure were changed.

  Bug 7179. Suprious sensitivity to )abb in documentation caused
  breakage because this was assumed to be an )abbreviation command.

  Bug 7178. Missing function inspect in ArrayStack

  Bug 7177. Missing function map! in ArrayStack
```

```
  Bug 7176. Missing function parts in ArrayStack

  Bug 7175. Missing function map in ArrayStack

  Bug 7174. Missing function map! in Stack

  Bug 7173. Missing function parts in Stack

  Bug 7172. map was missing from Stack(Integer)

  Bug 7170: )d op output failure due to X in algebra comments. The X
  symbol is now a marker for )d op examples and the use of X in comments
  is ambiguous.

  Bug 7141: )cd relative does not work

Interpreter (src/interp)
  Documentation

    document mmCost calculation

Help files
  Documentation

    The help file list is now dynamically generated from the algebra
    books.

readme
  Quote

    Add Doron Zeilberger quote about builders of CA systems.

Website
  Download

    Add fedora10 as a supported platform

    Add vector as a supported platform

Lisp
  New patch

    read-char-no-hang was changed by Camm to ignore newlines but
    Axiom's browser needs them to be recognized. A patch was added
    to back out Camm's change.

\end{verbatim}
\endscroll
\autobuttons
\end{page}
```

January 2009 Release Notes



⇐ “What’s New in Axiom” (releaseNotes) 1.1 on page 1

— **releasenotes.ht** —

```
\begin{page}{jan2009}{January 2009 Release Notes}
\beginscroll
\begin{verbatim}
January 2009 Release Notes

One major theme of this release was restructuring the system
to move all domains to literate form in book volume 10.3

A second major theme was the first Axiom youtube tutorial video.



Book Volume 0: Axiom Reference

  Remove obsolete references to SmallFloat

Book Volume 1: Axiom Tutorial

  Remove obsolete references to SmallFloat

Book Volume 4: Axiom Developers Guide
```

```
  Add section on generating graphviz graphs

Book Volume 7.1: Axiom Hyperdoc

  Give a complete path to htadd

Book Volume 10.3 Axiom Domains

  All domains have been removed from the algebra subdirectory
  and are now in literate form in book volume 10.3. Some
  domains include a help file, a regression test suite, and
  --X function examples for the )display command. Future work
  will extend this set.

  Each domain has a fully indexed list of functions exported.
  This allows hyperlinked access to functions.

  Each domain includes a graph segment that shows the list of
  categories, domains, and packages that provide immediate
  support in the prior build layer.

  Each domain includes a dotabb chunk which contains the
  source code necessary to recreate the graph.

  Domains which were associated with other domains in the same
  original spad file have hyperlinks to connect them.


Book Volume 11: Axiom Browser

  Lighten Bayou theme background image

Axiom Website (src/axiom-website)

  The axiom website is now under git control.
  The source tree is in src/axiom-website in the distribution.

  The build order graph exists. It contains a graph of all the
  constraints for each level. A line in the graph connects each
  category, domain, and package to every category, domain, and
  package in the prior layer that is required for the build.
  Multilayer constraints are not shown (although the data is
  in src/algebra/Makefile chunks) because the graph would be
  too complex.
    http://axiom-developer.org/axiom-website/hp.svg

  The Computer Algebra Test Suite (CATS) has a new test suite
  for ordinary differential equations (kamke*)
    http://axiom-developer.org/axiom-website/CATS
```

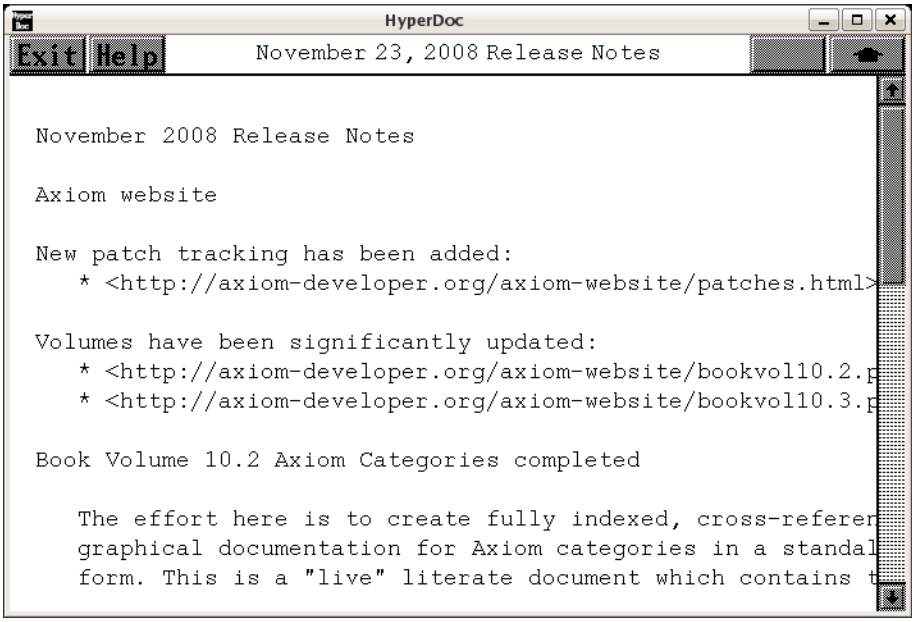
```
  The Ubuntu download now advises XTerm*metaSendsEscape: true
  due to misconfiguration of the meta key in emacs.
    http://axiom-developer.org/axiom-website/documentation.html

  A new binary for vector linux was added
    http://axiom-developer.org/axiom-website/download.html

  A new binary for doyen thumbdrive was added
  (work by Jose Alfredo Portes)
    http://axiom-developer.org/axiom-website/download.html

  The screenshots were recreated so the sources would be available.
  for matrixinmatrix.jpg, heatequation.jpg, axiomfirefox.jpg
    http://axiom-developer.org/axiom-website/screenshots.html

  The Axiom Information Sources video page was added, including
  a link to the youtube video, a local copy, and the supporting PDF.
    http://axiom-developer.org/axiom-website/videos.html

  A broken link to Book Volume 0: Axiom Reference was fixed.


Algebra (src/algebra)

  All domains were removed from spad files and moved to
  Book Volume 10.3 Axiom Domains

  The Makefile now contains the information necessary to
  construct the build order graph as literate chunks.

  The layers were restructured. Files were moved to ensure
  that each file was at its lowest possible layer.

  The Guess package was added. However, due to nonstandard
  package files the guess function does not yet work properly.
  These packages are being rewritten.

  The Guess package has circular references. A new clique
  mechanism was added to the Makefile to handle this.

  Automatic regression test files associated with the algebra
  sources were reviewed for conforming to regression standards.

  integer.spad was reverted to remove an algebra change that
  causes a regression (cherry picked from Fricas in 2007)

Interpreter (src/interp)

  Tim Lahey added to )credits
```

```
  Karl Hegbloom added to )credits

  htcheck.boot moves util.ht to the mnt/doc directory

  setq.lisp no longer had redundant release data information

  nrunfast.boot Float has exp : Float -> Float now works
  This change was cherry-picked from Fricas.


Testing (src/input)

  All regression test files were reviewed and cleaned to
  conform to a standard setup.

  A parallel build race condition was fixed in the Makefile

  xpbwpoly.input was removed as it is now associated with
  its source domain and automatically generated.

  hyperbolicrules.input was added

  Tim Lahey submitted a fix for bugs in schaum17.input

  A cherry-picked patch from Fricas in 2007 was removed
  because it causes an algebra regression. The regression
  test file was corrected.

  fixed.input was converted to a regression test file


Build process

  The lsp/Makefile was updated for GCL pre3 patches

  An ubuntu64 chunk was added as a system target

  src/Makefile outputs util.ht before compiles (bug 7146)

  books/Makefile uses bash SHELL to fix echo behavior.
  The standard sh echo behaves badly.

  books/Makefile now has amssymb in preamble for toc.tex


GCL (zips)

  Axiom now includes GCL pre3, the latest snapshot of GCL.
  This can be chosen as an option for building in the top
  level Makefile.pamphlet
```

```
  The lsp/Makefile was updated for GCL pre3 patches

faq

  FAQ 49: How do I get the lastest GCL?

readme

  Tim Lahey added
  Karl Hegbloom added

  The date information for releases was removed as it is
  available elsewhere

  A John Gorka quote was added:
  "What matters the most is what you do for free" -- John Gorka


Wikipedia

  http://en.wikipedia.org/wiki/Axiom_computer_algebra_system

  Add information about Documentation

  Add link to youtube video
    http://www.youtube.com/watch?v=CV8y3UrpadY

  New screenshots were added


Youtube

  Axiom Information Sources video was created and uploaded
    http://www.youtube.com/watch?v=CV8y3UrpadY
  To date there have been 498 views


Launchpad

  Gain administrative access to launchpad.net for Axiom (Page)
    https://launchpad.net/axiom

\end{verbatim}
\endscroll
\autobuttons
\end{page}
```

———-

November 23, 2008 Release Notes



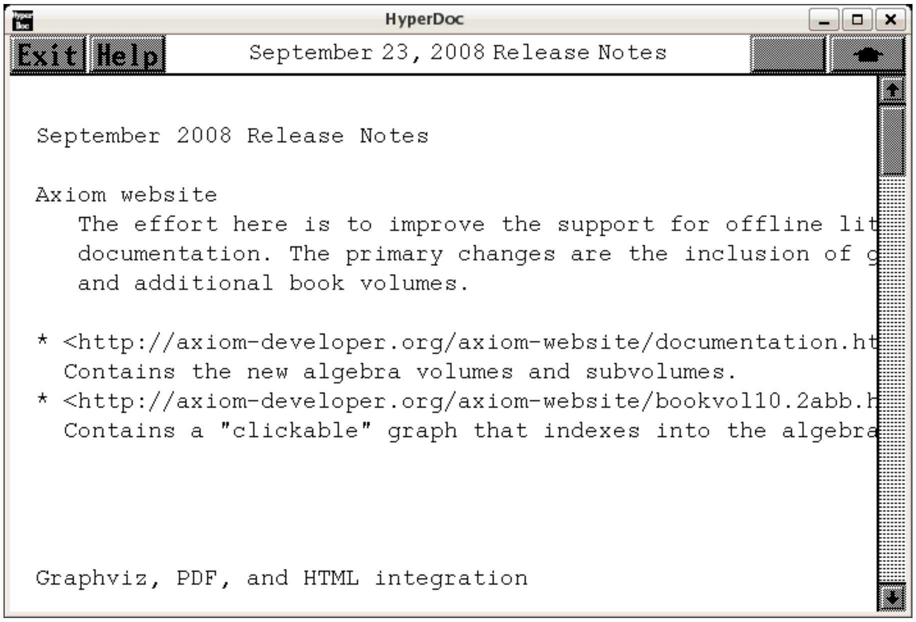
⇐ "What's New in Axiom" (releaseNotes) 1.1 on page 1

— releasenotes.ht —

```
\begin{page}{nov2008}{November 23, 2008 Release Notes}
\beginscroll
\begin{verbatim}
November 2008 Release Notes

Axiom website

New patch tracking has been added:
   * <http://axiom-developer.org/axiom-website/patches.html>

Volumes have been significantly updated:
   * <http://axiom-developer.org/axiom-website/bookvol10.2.pdf>
   * <http://axiom-developer.org/axiom-website/bookvol10.3.pdf>

Book Volume 10.2 Axiom Categories completed

   The effort here is to create fully indexed, cross-referenced,
   graphical documentation for Axiom categories in a standalone
   form. This is a "live" literate document which contains the
   actual source code used to build the system.

   * <http://axiom-developer.org/axiom-website/bookvol10.2.pdf>
```

```
Book Volume 10.3 Axiom Domains started

   This volume will contain the Axiom domains.

   * <http://axiom-developer.org/axiom-website/bookvol10.3.pdf>

Rosetta documentation

   Fix the Magnus URL

Input Files

   There is a new effort to automatically extract the algebra
   examples in order to regression test the user API to the
   algebra. In addition there is ongoing test work.

* New input files (Jakubi, Maltey, Rubey, Page, Daly}
     dhmatrix, reclos2
* Changed input files (Hebisch, Daly)
     sae, r20bugs

Build changes

* Testing is now run in parallel

\end{verbatim}
\endscroll
\autobuttons
\end{page}
```

September 23, 2008 Release Notes



⇐ “What’s New in Axiom” (releaseNotes) 1.1 on page 1

— releasenotes.ht —

```
\begin{page}{sept2008}{September 23, 2008 Release Notes}
\beginscroll
\begin{verbatim}
September 2008 Release Notes

Axiom website
   The effort here is to improve the support for offline literate
   documentation. The primary changes are the inclusion of graphs
   and additional book volumes.

* <http://axiom-developer.org/axiom-website/documentation.html>
  Contains the new algebra volumes and subvolumes.
* <http://axiom-developer.org/axiom-website/bookvol10.2abb.html>
  Contains a "clickable" graph that indexes into the algebra.




Graphviz, PDF, and HTML integration
   The effort here is to unify these three technologies in a way
   that simplifies the user interface and improves documentation
```

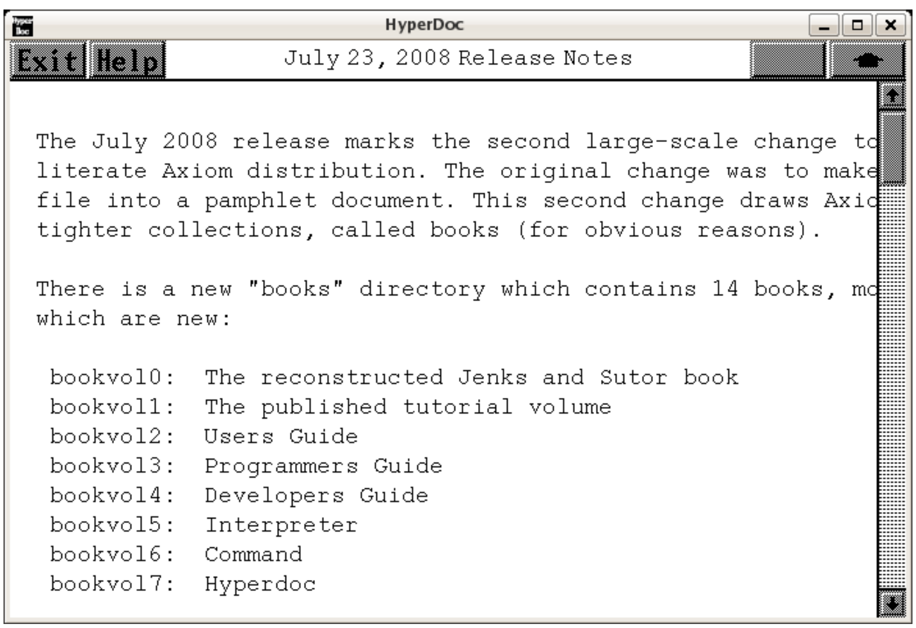
```
* Graphviz is used if available but not required
* Algebra graphs are automatically generated at build time
  from algebra source files
* Graphviz graphs now properly hyperlink into PDF files allowing
  any node in a graph to link to any document page




Book volume 0 (Jenks and Sutor)

* <http://axiom-developer.org/axiom-website/bookvol0.pdf>
* replace \over with \frac




Book volume 7.1 (Hyperdoc pages)
   The effort here is to create a literate document that contains
   all of the "live" pages used in hyperdoc. The PDF is being
   constructed so that a user can effectively "browse" the static
   hyperdoc pages, which are included, without a running Axiom.

* <http://axiom-developer.org/axiom-website/bookvol7.1.pdf>
* The source for all of the pages is now contained in this book.
* Hyperdoc now fetches the pages directly from the book.
* The hyper page directory and all files are gone.
* Some of the static page images are now inside the PDF
* Pages have href links allowing "in-pdf" navigation of pages





Book volume 10 (Algebra)
   The effort here is to create a way to describe and deeply
   document the algebra. This volume was split to better handle
   the structure of Axiom's information.

* Split into 5 volumes
   - 10   Implementation
   - 10.1 Theory
   - 10.2 Categories
   - 10.3 Domains
   - 10.4 Packages





Book volume 10.2 (Algebra Categories)
   The effort here is to create fully indexed, cross-referenced,
   graphical documentation for Axiom categories in a standalone
```

```
   form. This is a "live" literate document which contains the
   actual source code used to build the system.

* <http://axiom-developer.org/axiom-website/bookvol10.2.pdf>
* Contains 60 categories so far
* Has partial graphs for each category
* Has list of exported functions
* Has information about source of functions
* Has index cross reference by function and category
* Has PDF href links so that URLs work:
   <http://axiom-developer.org/axiom-website/bookvol10.2.pdf#nameddest=AGG>
* Has forward/backward links between categories
* Automatically generates "clickable" graphs:
   <http://axiom-developer.org/axiom-website/bookvol10.2abb.html>
* Graph clicking automatically opens to the proper source code



New algebra examples (Daly, Tsikas)
   The effort here is to create "real time" documentation that
   gives the end user an example of how to construct the proper
   arguments and call a function. This puts examples into the
   system so users don't need to consult other documents.

* )d op someop shows examples of function usage
* about 100 new function examples were added
* new comment syntax added to allow automatic API testing



Input Files
   There is a new effort to automatically extract the algebra
   examples in order to regression test the user API to the
   algebra. In addition there is ongoing test work.

* New input files (Hemmecke, Stumbo, Cyganski, Daly)
     bini, biquat, ifthenelse, liu, overload, sqrt3, typetower
* Changed input files (Hemmecke, Stumbo, Cyganski, Daly)
     bern, function, linalg, regset, test, tutchap2



Build changes

* graphics does not depend on compress, done at build time
* firefox html pages are now built before tests are run
```

```
Algebra changes

* FLAGG (FiniteLinearAggregate) -- removed a duplicate function



Interpreter changes (Page)

* add cost function to bottomUp output




\end{verbatim}
\endscroll
\autobuttons
\end{page}
```

———-

July 23, 2008 Release Notes



⇐ “What’s New in Axiom” (releaseNotes) 1.1 on page 1

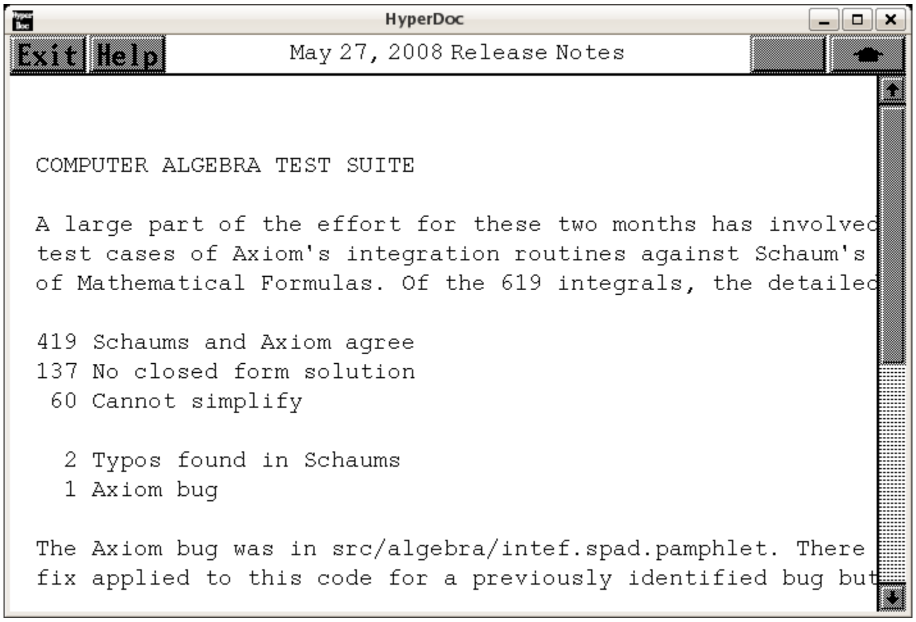
— **releasenotes.ht** —

```
\begin{page}{july2008}{July 23, 2008 Release Notes}
\beginscroll
\begin{verbatim}
The July 2008 release marks the second large-scale change toward a
literate Axiom distribution. The original change was to make every
file into a pamphlet document. This second change draws Axiom into
tighter collections, called books (for obvious reasons).

There is a new "books" directory which contains 14 books, most of
which are new:

 bookvol0:  The reconstructed Jenks and Sutor book
 bookvol1:  The published tutorial volume
 bookvol2:  Users Guide
 bookvol3:  Programmers Guide
 bookvol4:  Developers Guide
 bookvol5:  Interpreter
 bookvol6:  Command
 bookvol7:  Hyperdoc
 bookvol7.1 Hyperdoc Pages
 bookvol8:  Graphics
 bookvol9:  Compiler
 bookvol10: Algebra
 bookvol11: Browser
 bookvol12: Crystal

All of these books now exist. Portions of the current system have
been moved completely into book form (Graphics and Hyperdoc) so
the old files have been removed from the src tree. Both Graphics
and Hyperdoc are now built directly from their book. Work will
follow on other parts of the system.

These books are online at:
<http://axiom.axiom-developer.org/axiom-website/documentation.html>




There is an interesting side-effect of using literate technology.
Once you combine C and .h files into a single document so that each
file is a separate chunk it becomes obvious that there is no need for
local include files. The lines that read:
   #include "foo.h"
become
   begin{chunk}{foo.h}
and get expanded inline. Once you do this it also becomes obvious
that many include files get included multiple times (a clear waste
of disk I/O and preparser time). Further it becomes clear that there
```

```
is no need for creating tiny .o files since all of the source can
be combined into one C file using chunks.

These approaches were used to reduce the compile-time overhead
for both the graphics and the hyperdoc functions.




In addition to consolidating the source files for Graphics into
bookvol8 there were several other changes.

  * the functions for graphics, that is, viewman, view2d, view3d
    and viewalone are now built as single C files.
  * the graphics code was reorganized into chapters of related code
  * there is the beginnings of documentation
  * there is the beginnings of a test suite based on the
    CRC Handbook of Curves and Surfaces
  * the book is hyperlinked (Bill Page, Frederic Lehobey, Anatoly Raportirenko)
  * redundant code was eliminated
  * the code is fully indexed
  * the src/graph subdirectory is gone



In addition to consolidating the source files for Hyperdoc into
bookvol7 there were several other changes.

  * the functions for hyperdoc, that is, spadbuf, ex2ht, htadd, hthits
    and hypertex are now built as single C files
  * the hyperdoc code was reorganized into chapters of related code
  * there is the beginnngs of documentation
  * redundant code was eliminated
  * scroll wheel handling was added (Gregory Vanuxem)
  * the book is hyperlinked (Bill Page, Frederic Lehobey, Anatoly Raportirenko)
  * the code is fully indexed
  * )hd now works to start or restart Hyperdoc (Jose Alfredo Portes)
  * the src/hyper subdirectory is gone



In addition to consolidating the hyperdoc pages into bookvol7.1
there were several other changes:

  * the pages are hyperlinked
  * new latex macros were written to simplify page handling
  * images of hyperdoc pages were added
  * forward and backward hyperlinks between pages images works
    making it possible to "browse" the hyperdoc pages using only
    the single PDF file (Bill Page, Frederic Lehobey, Anatoly Raportirenko)
```

```
  * the pages are fully indexed
  * the src/hyper/pages subdirectory is gone



A combined table of contents PDF is now automatically generated
which covers all of the volumes. This makes it easier to find the
topic of interest.


In addition there were several other changes.

  * Some fixes were made for different platforms
      In particular, Fedora9 breaks the build and needs work

  * The configure script was replaced by instructions
      The standard (export AXIOM) works everywhere so the configure
      script is useless for guessing.

  * the FAQ was updated about git
      Since the May 2008 release the primary Gold source code platform
      is github. The FAQ was updated to reflect this.

  * the FAQ was updated about X11
      X11 has moved yet again so more notes are needed

  * move axbook to books
      The hyperlinked version of Jenks is now built from the books dir

  * add Ralf Hemmecke's documentation to ax.boot
  * add Monoid multiply to DirectProduct (Ralf Hemmecke)
  * add Monoid multiply regression test

\end{verbatim}
\endscroll
\autobuttons
\end{page}
```

May 27, 2008 Release Notes



⇐ "What's New in Axiom" (releaseNotes) 1.1 on page 1

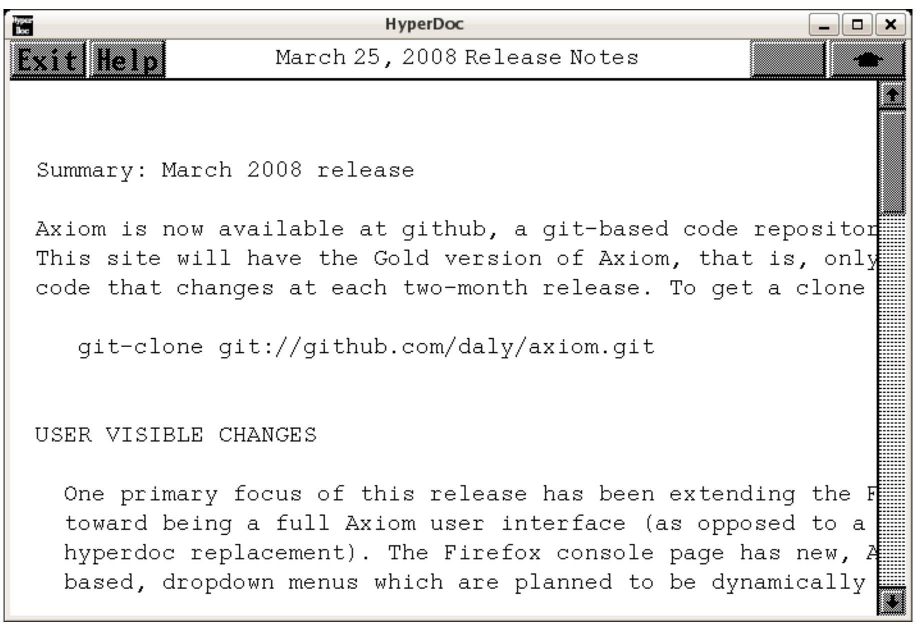
— **releasenotes.ht** —

```
\begin{page}{may2008}{May 27, 2008 Release Notes}
\beginscroll
\begin{verbatim}

COMPUTER ALGEBRA TEST SUITE

A large part of the effort for these two months has involved detailed
test cases of Axiom's integration routines against Schaum's Handbook
of Mathematical Formulas. Of the 619 integrals, the detailed results are:

419 Schaums and Axiom agree
137 No closed form solution
 60 Cannot simplify

  2 Typos found in Schaums
  1 Axiom bug

The Axiom bug was in src/algebra/intef.spad.pamphlet. There was a
fix applied to this code for a previously identified bug but the
previous fix was incorrect.

In addition, there were
```

```
 src/input/danzwill2.input.pamphlet added for the MIT Integration tests
 src/inputmapleok.input.pamphlet to fix typos
 src/input/kamke1.input.pamphlet had ode97 removed due to running time
 src/input/kamke2.input.pamphlet ode104, ode105 removed for running time

DOCUMENTATION
Max Tegmark's 'toe' diagram, src/doc/toe.gif

PORTING
        GCLOPTS-CUSTRELOC disable-locbfd for MACOSXPPC

\end{verbatim}
\endscroll
\autobuttons
\end{page}
```

———-

March 25, 2008 Release Notes



⇐ "What's New in Axiom" (releaseNotes) 1.1 on page 1

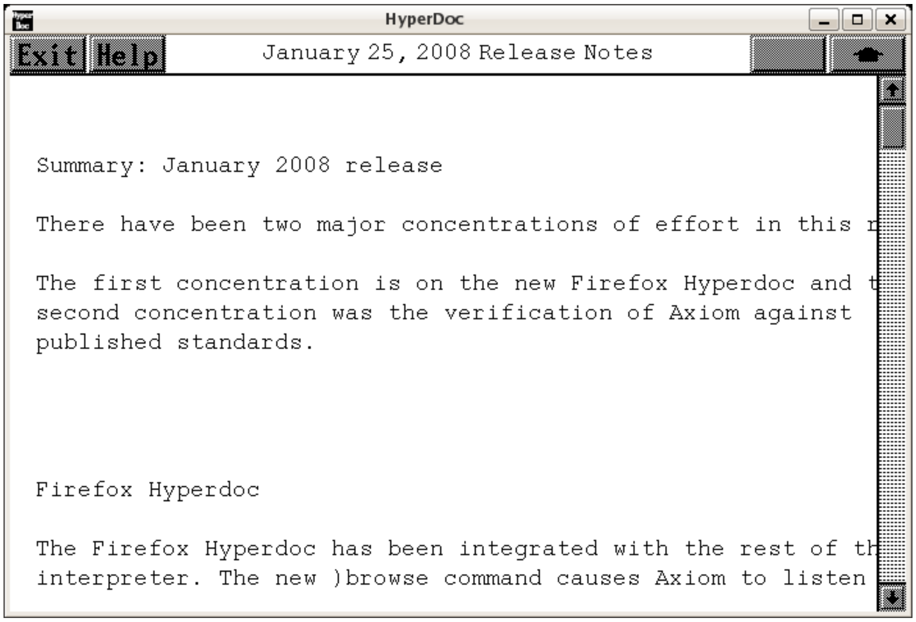
— **releasenotes.ht** —

```
\begin{page}{march2008}{March 25, 2008 Release Notes}
\beginscroll
```

```
\begin{verbatim}

Summary: March 2008 release

Axiom is now available at github, a git-based code repository.
This site will have the Gold version of Axiom, that is, only
code that changes at each two-month release. To get a clone type:

   git-clone git://github.com/daly/axiom.git


USER VISIBLE CHANGES

  One primary focus of this release has been extending the Firefox
  toward being a full Axiom user interface (as opposed to a simple
  hyperdoc replacement). The Firefox console page has new, AJAX
  based, dropdown menus which are planned to be dynamically updated
  to display available functions for the last computed type. This
  should make it much easier to find the applicable functions by
  category and type. They are currently static in this release.

* Firefox Pages

 o Dropdown menus were added to the Axiom console page
 o More hyperdoc pages were translated to Firefox/html
 o Bitmaps and graphics are now properly handled in pages
 o A minor mathml fix was applied (for invisible times)

* Refcard

 o An Axiom reference card of Axiom commands was created (src/doc/refcard)

* Examples

 o It is often difficult to figure the exact arguments required to call
   any given function in Axiom. The )display operation command used to
   only show the available modemaps. This command has now been changed.
    )display operation foo
   now shows examples of function calls for foo.

* Help

 o The plot routines have new help files and documentation

PORTING

 o Axiom was ported to MAC-OSX

 o The binary download page now has binaries for
   Ubuntu, OpenSUsE, Redhat9, Redhat72, Debian, MACOSX at
```

```
    <http://axiom.axiom-developer.org/axiom-website/download.html>

 o Binaries for the this release will be available shortly.


INTERNALS

* Compiler changes
 o  hashtables were used to speed up compiles

* Algebra changes
 o  There are new special functions, Ei,En,Ei1,Ei2,Ei3,Ei4,Ei5,Ei6
 o  The prime and BasicSieve functions are faster
 o  The Brent/Pollard algorithm was documented
 o  Bad gcd reductions are checked (heugcd regression test file added)
 o  The plot routines have new help files and documentation

* Makefile changes
 o  Bi-capital SVN copies are no longer made

* Interpreter changes
 o  Book Volume 5 has new documentation on the display function
 o  The display function code has been translated and moved to book volume 5
 o  PI has a higher internal precision
 o  Mappings are now properly hashed for Aldor

CATS (Computer Algebra Test Suite)

 o The differential equations regression tests are being checked against
   Mathematica, Maple, and Maxima. This has happended for the kamke2.input
   regression test file and will happen for the other regression tests.

o  Complex Gamma, logGamma, and log(Gamma) have additional tests and
   documentation.

\end{verbatim}
\endscroll
\autobuttons
\end{page}
```

January 25, 2008 Release Notes



⇐ "What's New in Axiom" (releaseNotes) 1.1 on page 1

— releasenotes.ht —

```
\begin{page}{january2008}{January 25, 2008 Release Notes}
\beginscroll
\begin{verbatim}

Summary: January 2008 release

There have been two major concentrations of effort in this release.

The first concentration is on the new Firefox Hyperdoc and the
second concentration was the verification of Axiom against
published standards.




Firefox Hyperdoc

The Firefox Hyperdoc has been integrated with the rest of the
interpreter. The new )browse command causes Axiom to listen and
serve hyperdoc pages on port 8085.

The interpreter was changed to add the )browse command. As a
```

```
side-effect new documentation was added to the interpreter
volume (bookvol5) to explain top-level command handling. In
addition, lisp and boot code was rewritten as part of the
literate change.

New sections were added to cover the beginning of the Computer
Algebra Test Suite (CATS) subsection which brings a focus on
compliance with published standards.

Arthur Ralf's mathml-enabled version of the Jenks book is fully
integrated into the Firefox Hyperdoc. Arthur also fixed some
rendering and ambiguity issues.

  axbook.tgz              fix the user/group settings
  axserver.spad           fix lastType output re: errors
  bookvol5                browse and top-level command handling
  bookvol11               add standards compliance for gamma
  gammacomplexinverse.png added
  gammacomplex.png        added
  gammareal3.png          added
  loggamma.png            added
  mathml.spad             fix ambiguity bug in mathml output
  mathml.spad             fix hex(10) mathml rendering
  mathml.spad             fix F,3 mathml rendering
  mathml.spad             remove code to eat %%
  psi.png                 added



Standards Verification

The Computer Algebra Test Suite (CATS) effort checks the results
that Axiom generates against published results. Axiom has an
extensive set of regression tests (the KAMKE suite) for ordinary
differential equations, for integration (the SCHAUM suite), and
for numeric special functions (the ABRAMOWITZ suite). In addition,
results have been checked against Mathematica, Maple, and Maxima.

  asinatan.input   regression for the functions asin and atan
  asinhatanh.input regression for the functions asinh and atanh
  besselk.input    regression for the function besselK
  e1.input         regression for the function E1
  en.input         regression for the function En
  exp.input        regression for the function exp
  gamma.input      regression for the function gamma
  log.input        regression for the functions log
  pfaffian.input   regression for the function pfaffian
  seccsc.input     regression for the functions sec and csc
  sincos.input     regression for the functions sin and cos
```

```
  sinhcosh.input   regression for the functions sinh and cosh
  tancot.input     regression for the functions tan and cot
  tanhcoth.input   regression for the functions tanh and coth
```

New Functions Added

Axiom is missing various special functions found in other computer algebra systems. This release adds two new ones, the Exponential Integral E1 and the higher order Exponential Integral En. These have been tested against the published results.

```
  special.spad E1 added
  special.spad En added
```

Bugs Fixed

There are 15 bug fixes in this release:

```
  bug 7015: fix hex(10) mathml rendering
  bug 7016: remove code to eat %%
  bug 7019: fix F,3 mathml rendering
  bug 7023: discardGraph free corrected
  bug 7042: ignore regression test gensym
  bug 7045: wrong Makefile for Xpm fix
  bug 7052: spurious remake of axbook
  bug 7054: /home/silver path in bookvol11
  bug 7057: ambiguity in mathml
  bug 7089/343: FreeAbelianGroup order
  bug 7090/355 handle besselK
  bug 7093: Function name fix
  bug 7100/149: numlock in hyperdoc
  bug 7101/204: MoreSystemCommand unnecessary loading
  bug 7102/412: Equality testing in TableAggregate
```

Regression test fixes

As changes happen in the system the regression tests are updated to reflect the new conditions. Changing the category of PositiveInteger caused (a + -bi) to print as (a - bi). New builds raised gensym faults which were fixed. And new builds change random numbers so the tests that depend on them are marked "ok" despite failures due to randomness.

The FreeAbelianGroup bug is tested.

```
  acplot.spad     fix output form of negative numbers
  calculus2.input fix function names
  classtalk.input ignore gensyms
  collect.input   fix function names
  dfloat.input    handle negative number output
  easter.input    fix function names
  elemnum.input   handle negative number output
  exlap.input     fix function names
  exsum.input     fix function names
  free.input      added to test bug
  grpthry.input   mark random generation failures ok
  grpthry.input   fix function names
  ico.input       mark random generation failures ok
  intg0.input     ignore gensyms
  is.input        type declare function
  kamke3.input    mark random generation failures ok
  knot2.input     fix function names
  lodo.spad       ignore regression test gensym
  mapleok.input   ignore gensyms
  mathml.input    handle new mathml sub/sup change
  ndftip.input    fix missing blank lines
  pmint.input     rewritten
  repa6.input     fix function names
  r20bugs.input   change spacing
  sf.spad         fix output form of negative numbers
  tbagg.input     regression for equality testing in TableAggregate
```

Algebra file changes

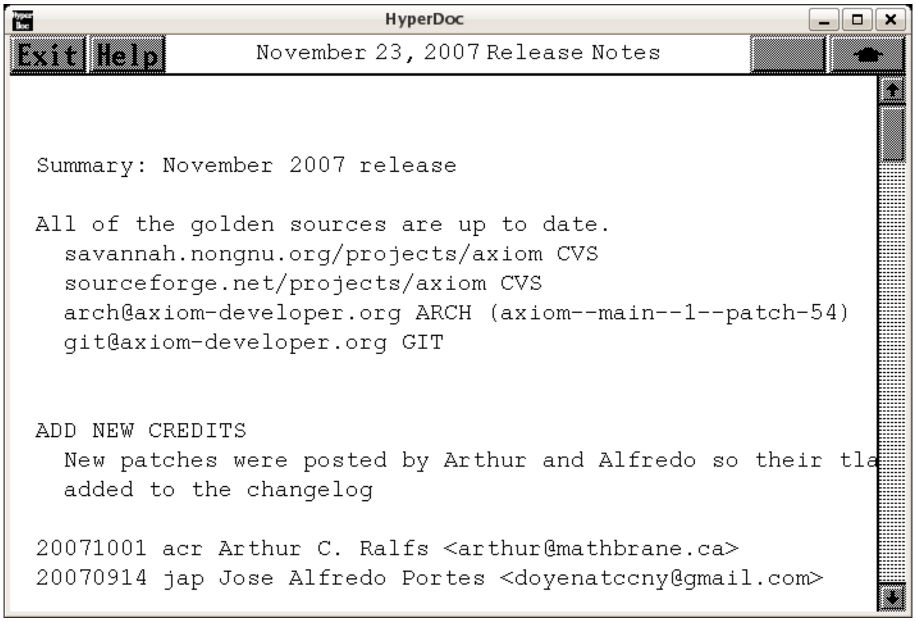
The fundamental change was a supposedly transparent move of
the category for PositiveInteger. This had the effect of changing
the output form and broke several regression tests. Some mathml
issues were fixed. The new functions E1 and En were added. The
FreeAbelianGroup bug was fixed.

```
  axserver.spad  fix lastType output re: errors
  acplot.spad    fix output form of negative numbers
  combfunc.spad  fix bold font handling
  integer.spad   category change for PositiveInteger
  sf.spad fix    output form of negative numbers
  sf.spad        handle besselK
  op.spad        handle besselK
  combfunc.spad  handle besselK
  free.spad      fix FreeAbelianGroup bug
  special.spad   add E1
```

```
  special.spad   add En
  mathml.spad    fix ambiguity bug in mathml output
  mathml.spad    fix hex(10) mathml rendering
  mathml.spad    fix F,3 mathml rendering
  mathml.spad    remove code to eat %%



Interpreter changes

The primary changes are the addition of bookvol11 for the Firefox
Hyperdoc and the literate documentation of the top level command
handling in bookvol5 (the interpreter) along with rewrites of the
lisp/boot code.

  bootfuns.lisp move $systemCommands to bookvol5
  bookvol5      browse and top-level command handling
  bookvol11     added
  http.lisp     mathObject2String for hex(10)
  incl.boot     move incBiteOff to bookvol5
  intint.lisp   move setCurrentLine to bookvol5
  int-top.boot  move ncloopCommand, etc. to bookvol5
  i-syscmd.boot move $SYSCOMMANDS to bookvol5
  Makefile      wrong Makefile for Xpm fix
  makegraph.c   discardGraph free corrected
  nci.lisp      move ncloopInclude to bookvol5
  setq.lisp     move command initialization to bookvol5



Documentation changes

  bookvol5      explain top level input handling (lisp/boot rewrite)
  combfunc.spad fix bold font handling
  axiom.sty     add binom



Patches released

  20071129.01.tpd.patch
  20071129.02.tpd.patch
  20071205.01.tpd.patch
  20071206.01.tpd.patch
  20071208.01.tpd.patch
  20071215.01.tpd.patch
  20071215.02.tpd.patch
  20071215.03.gxv.patch
  20071216.01.tpd.patch
  20071216.02.tpd.patch
```

```
  20071216.03.acr.patch
  20071217.01.acr.patch
  20071217.02.tpd.patch
  20071218.01.acr.patch
  20071225.01.sxw.patch
  20071228.01.tpd.patch
  20071229.01.jap.patch
  20071230.01.acr.patch
  20071230.02.tpd.patch
  20071230.03.tpd.patch
  20080102.01.tpd.patch
  20080103.01.tpd.patch
  20080104.01.tpd.patch
  20080104.02.tpd.patch
  20080106.01.tpd.patch
  20080107.01.tpd.patch
  20080107.02.tpd.patch
  20080107.03.tpd.patch
  20080116.01.tpd.patch
  20080119.01.tpd.patch
  20080119.02.tpd.patch
  20080119.03.tpd.patch
  20080120.01.gxv.patch
  20080120.02.tpd.patch
  20080120.03.tpd.patch

\end{verbatim}
\endscroll
\autobuttons
\end{page}
```

November 23, 2007 Release Notes



⇐ "What's New in Axiom" (releaseNotes) 1.1 on page 1

— **releasenotes.ht** —

```
\begin{page}{november2007}{November 23, 2007 Release Notes}
\beginscroll
\begin{verbatim}

Summary: November 2007 release

All of the golden sources are up to date.
  savannah.nongnu.org/projects/axiom CVS
  sourceforge.net/projects/axiom CVS
  arch@axiom-developer.org ARCH (axiom--main--1--patch-54)
  git@axiom-developer.org GIT


ADD NEW CREDITS
  New patches were posted by Arthur and Alfredo so their tlas were
  added to the changelog

20071001 acr Arthur C. Ralfs <arthur@mathbrane.ca>
20070914 jap Jose Alfredo Portes <doyenatccny@gmail.com>


PORT TO DIFFERENT SYSTEMS
```

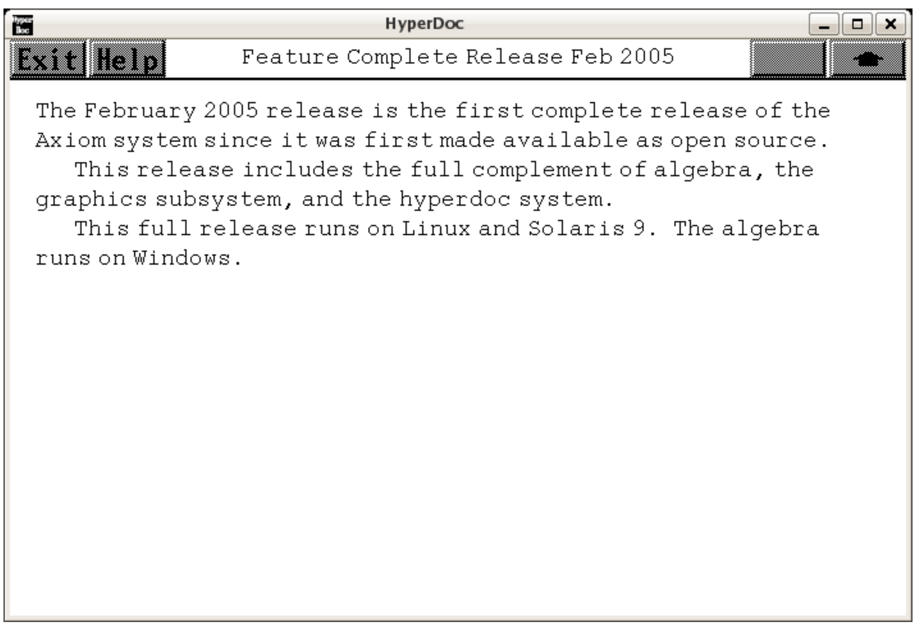
```
  As part of the new axiom website there is a binary release page.
  The stanzas for these supported systems were added.

20071119 tpd Makefile.pamphlet add fedora6,7,8 stanzas


NEW Axiom WEBSITE: http://axiom-developer.org STARTED.
  The new Axiom website (currently at axiom.axiom-developer.org)
  has been started. It will include the binary release page.


REMOVE OLD REGRESSION SYSTEM
  The previous regression test system was removed. A new combined
  regression test and help documentation system was built to replace
  this mechanism.

20070901 tpd src/input/Makefile remove ALGEBRA variable
20070901 tpd src/algebra/perm.spad remove TEST mechanism
20070901 tpd src/algebra/view2d.spad remove TEST mechanism
20070901 tpd src/algebra/fr.spad remove TEST mechanism


FIX BOOK DOCUMENTATION
  Minor typos have been discovered in the book during documentation.

20070905 tpd src/doc/book remove duplicate upperCase, lowerCase typo
20070903 tpd src/doc/bookvol4 fix typos
20070902 tpd src/doc/book MultiSet -> Multiset
20080829 tpd src/doc/book.pamphlet correct typo


FIX BUGS
  Various bugs have been found and fixed.

20071101 tpd src/interp/i-output.boot fix bugs 7010 (209), 7011
20070920 tpd src/input/Makefile add bug101.input regression test
20070920 tpd src/input/bug101.input test laplace(log(z),z,w) bug 101
20070920 wxh src/algebra/laplace.spad fix laplace(log(z),z,w) bug 101
20070916 tpd src/input/Makefile add bug103.input regression test
20070916 tpd src/input/bug103.input test solve(z=z,z) bug fix
20070916 tpd src/algebra/polycat.spad solve(z=z,z) bug fix
20070916 tpd src/algebra/catdef.spad add zero? to exquo
20070915 tpd merge bug100 branch
20070915 tpd src/input/Makefile add bug100.input regression test
20070915 tpd src/input/bug100.input test integrate((z^a+1)^b,z) infinite loop
20070915 wxh src/algebra/intef.spad fix integrate((z^a+1)^b,z) infinite loop
20070915 tpd src/algebra/carten minor edit for regression cleanup
20070914 wxh src/hyper/hyper fix bad bracing of )hd change
20070914 tpd src/algebra/fraction.spad remove double )spool command
20070914 tpd src/algebra/kl.spad remove double )spool command
```

```
20070914 tpd src/algebra/lindep.spad remove double )spool command
20070914 tpd src/algebra/radix.spad remove double )spool command


ENABLE DYNAMIC RESTART OF HYPERDOC
  Hyperdoc can now be started dynamically or restarted if killed.

20070914 jap adapt changes for )hd restart to Axiom sources
20070914 wxh src/sman/bookvol6 enable restart of hyperdoc with )hd
20070914 wxh src/include/sman.h1 enable restart of hyperdoc with )hd
20070914 wxh src/hyper/hyper enable restart of hyperdoc with )hd


SET UP THE NEW FIREFOX BASED HYPERDOC
  Hyperdoc is going away. A new version of hyperdoc is being built
  which uses html/javascript/mathml. These files change the interpreter
  and algebra to support the new hyperdoc machinery.

20071019 acr src/interp/http.lisp use new return values
20071019 acr src/algebra/axserver.spad use new return values
20071014 acr src/algebra/axserver.spad use getContentType(pathvar)
20071013 acr license/license.ralfs license rewrite
20071013 acr src/interp/http.lisp faster page service
20071013 acr src/algebra/axserver.spad faster page service
20071001 tpd src/algebra/exposed.lisp add (|AxiomServer| . AXSERV) to basic
20071001 tpd src/algebra/Makefile add axserver.spad
20071001 acr src/algebra/axserver.spad axserver socket connection code
20071001 tpd src/interp/Makefile add http.lisp
20071001 acr src/interp/http.lisp axserver socket connection code
20071001 acr license/license.ralfs added


REGRESSION TEST CALCULUS
  A new regression test suite for calculus is being built. The first
  of these files has been added to the system.

20070913 tpd src/input/Makefile schaum1.input added
20070913 tpd src/input/schaum1.input added


REGRESSION TEST ORDINARY DIFFERENTIAL EQUATIONS
  A regression test suite for ordinary differential equations was built.

20071005 tpd src/input/Makefile kamke7.input regression test added
20071005 tpd src/input/kamke7.input ODE regression test added
20071005 tpd src/input/Makefile kamke6.input regression test added
20071005 tpd src/input/kamke6.input ODE regression test added
20071005 tpd src/input/Makefile kamke5.input regression test added
20071005 tpd src/input/kamke5.input ODE regression test added
20071005 tpd src/input/Makefile kamke4.input regression test added
```

```
20071005 tpd src/input/kamke4.input ODE regression test added
20071005 tpd src/input/Makefile kamke3.input regression test added
20071005 tpd src/input/kamke3.input ODE regression test added
20071004 tpd src/input/Makefile kamke2.input regression test added
20071004 tpd src/input/kamke2.input ODE regression test added
20071004 tpd src/input/Makefile kamke1.input regression test added
20071004 tpd src/input/kamke1.input ODE regression test added
20071004 tpd src/input/Makefile kamke0.input regression test added
20071004 tpd src/input/kamke0.input ODE regression test added


REGRESSION TEST PFAFFIAN
  Martin added the pfaffian regression test. It was added and removed
  due to documentation license issues. New documentation is being written.

20070929 tpd src/input/Makefile pfaffian regression test removed
20070929 tpd src/input/pfaffian.input regression test removed
20070927 tpd src/input/Makefile pfaffian regression test added
20070927 mxr src/input/pfaffian.input regression test added


ADD PORTIONS OF THE GUESS PACKAGE
  The newton.spad file is actually part of the fffg.spad file so it
  was removed. The very top level spad functions in GUESS still do
  not work properly.

20070909 tpd src/algebra/newton.spad included in fffg.spad
20070909 tpd src/algebra/Makefile remove newton.spad (duplicate)


FIX BUILD PROCESS
  The build process was not properly suppressing output by default.

20070907 tpd src/algebra/acplot.spad fix PlaneAlgebraicCurvePlot.help NOISE
20070907 tpd src/algebra/Makefile make regression respect NOISE variable
20070907 tpd src/input/Makefile make regression respect NOISE variable


FIX INSTALL PROCESS
  The install process had a bug.

20070906 tpd Makefile int/sman/axiom command to target command for install
20070906 tpd src/sman/Makefile copy axiom command to int


ADD ALDOR RELEASE
  The aldor release has been added to zips. It will soon be part of
  the build mechanism but separately maintained like GCL.

20070901 tpd zips/aldor.20070901.tgz add pdf documentation
```

```
20070901 tpd zips/aldor.20070901.tgz added


ADD )HELP FACILITY
  The )help facility was recovered. The documentation is now integrated
  into the spad files and used both for help documentation and algebra
  regression testing.

\end{verbatim}
\endscroll
\autobuttons
\end{page}
```

Feature Complete Release Feb 2005



⇐ "What's New in Axiom" (releaseNotes) 1.1 on page 1

— **releasenotes.ht** —

```
\begin{page}{feb2005}{Feature Complete Release Feb 2005}
\beginscroll
The February 2005 release is the first complete release of the
Axiom system since it was first made available as open source.
```

```
This release includes the full complement of algebra, the graphics
subsystem, and the hyperdoc system.

This full release runs on Linux and Solaris 9.
The algebra runs on Windows.
\endscroll
\autobuttons
\end{page}
```

———

Chapter 2

Special hyperdoc pages

2.1 util.ht

This file contains macros for the Axiom HyperDoc hypertext facility. Most of the macros for the system are here though there may be some in individual .ht files that are of a local nature.

Names of software and facilities

— **util.ht** —

```
\newcommand{\Browse}{Browse}
\newcommand{\Language}{Axiom}
\newcommand{\SpadName}{\Language}
\newcommand{\LangName}{\Language}
\newcommand{\HyperName}{HyperDoc}
\newcommand{\Clef}{Clef}
\newcommand{\Lisp}{Common LISP}
\newcommand{\naglib}{NAG Foundation Library}
\newcommand{\GoBackToWork}
{\vspace{2}\newline
{Click on \  \UpButton{} \  to go back to what you were doing.}}
```

————-

Special hooks to Unix

All unix commands should be done as macros defined here so we don't have to go hunting when moving between Unix versions.

— util.ht —

```
\newcommand{\newspadclient}[1]
{xterm -n "#1" -e \$SPAD/bin/clef \$SPAD/bin/server/spadclient}
\newcommand{\searchwindow}[2]{\unixwindow{#1}{\$SPAD/lib/htsearch "#2"}}
\newcommand{\unixwindow}[2]{\unixlink{#1}{#2}}
\newcommand{\menuunixlink}[2]{\item\unixlink{\menuitemstyle{#1}}{#2}}
\newcommand{\menuunixcommand}[2]{\item\unixcommand{\menuitemstyle{#1}}{#2}}
\newcommand{\menuunixwindow}[2]{\item\unixwindow{\menuitemstyle{#1}}{#2}}
```

———-

HyperDoc menu macros

— util.ht —

```
% Example:
%
% \beginmenu
% \menulink{Thing One}{PageOne} la da di da da ...
% \menulink{Thin Two}{PageTwo}  do da day ...
% \item \ACmdMacro{\menuitemstyle{Thing Three}} la di da ...
% \endmenu

% The menu environment

\newcommand{\beginmenu}          {\beginitems[\MenuDotBitmap]}
\newcommand{\endmenu}            {\enditems}

% This is the usual format for a menu item.

\newcommand{\menuitemstyle}[1]   {{\MenuDotBitmap}#1}

% Often-used menu item forms

%   These two simply do links
\newcommand{\menudownlink}[2]    {\item\downlink{\menuitemstyle{#1}}{#2}}
\newcommand{\menulink}[2]        {\menudownlink{#1}{#2}}

%   This will cause lower level links to have a HOME button
\newcommand{\menumemolink}[2]    {\item\memolink{\menuitemstyle{#1}}{#2}}

%   This opens a new window for the linked page.
\newcommand{\menuwindowlink}[2]    {\item\windowlink{\menuitemstyle{#1}}{#2}}

%   These execute lisp commands in various flavors
\newcommand{\menulispcommand}[2]
```

```
{\item\lispcommand{\menuitemstyle{#1}}{#2}}
\newcommand{\menulispdownlink}[2]
{\item\lispdownlink{\menuitemstyle{#1}}{#2}}
\newcommand{\menulispmemolink}[2]
{\item\lispmemolink{\menuitemstyle{#1}}{#2}}
\newcommand{\menulispwindowlink}[2]
{\item\lispwindowlink{\menuitemstyle{#1}}{#2}}

%   This executes a unix command
\newcommand{\menuunixcmd}[2]{\item\unixcommand{\menuitemstyle{#1}}{#2}}
\newcommand{\searchresultentry}[3]
{\tab{3}\item#3\tab{8}\downlink{\menuitemstyle{#1}}{#2}\newline}
\newcommand{\newsearchresultentry}[3]
{\tab{3}\item#1\tab{8}\downlink{\menuitemstyle{#2}}{#3}\newline}
```

———

Bitmaps and bitmap manipulation macros

— **util.ht** —

```
\newcommand{\htbmdir}{\env{AXIOM}/doc/bitmaps}
\newcommand{\htbmfile}[1]{\htbmdir /#1.bitmap}
\newcommand{\htbitmap}[1]{\inputbitmap{\htbmfile{#1}}}
\newcommand{\ControlBitmap}[1]{\controlbitmap{\htbmfile{#1}}}

% next group of bitmaps frequently appear in the titlebar
\newcommand{\ContinueBitmap} {\ControlBitmap{continue}}
\newcommand{\DoItBitmap}     {\ControlBitmap{doit}}
\newcommand{\ExitBitmap}     {\ControlBitmap{exit3d}}
\newcommand{\HelpBitmap}     {\ControlBitmap{help3d}}
\newcommand{\ReturnBitmap}   {\ControlBitmap{home3d}}
\newcommand{\NoopBitmap}       {\ControlBitmap{noop3d}}
\newcommand{\UpBitmap}       {\ControlBitmap{up3d}}

\newcommand{\MenuDotBitmap}{\htbitmap{menudot}}

% Including control panel pixmaps for help pages:

\newcommand{\helpbit}[1]
{\centerline{\inputpixmap{\env{AXIOM}/doc/pixmaps/{#1}}}}
```

———

HyperDoc button objects

— util.ht —

```
\newcommand{\ContinueButton}[1]{\downlink{Click here}{#1} to continue.}
\newcommand{\ExitButton}[1]{\memolink{\ExitBitmap}{#1}}
\newcommand{\HelpButton}[1]{\memolink{\HelpBitmap}{#1}}
\newcommand{\StdHelpButton}{\HelpButton{ugHyperPage}}
\newcommand{\StdExitButton}{\ExitButton{ProtectedQuitPage}}
\newcommand{\UpButton}{\upbutton{\UpBitmap}{UpPage}}
\newcommand{\ReturnButton}{\returnbutton{\ReturnBitmap}{ReturnPage}}
\newcommand{\on}[1]{{\inputbox[1]{#1}{\htbmfile{pick}}
             {\htbmfile{unpick}}}}
\newcommand{\off}[1]{{\inputbox[0]{#1}{\htbmfile{pick}}
             {\htbmfile{unpick}}}}
```

———-

Standard HyperDoc button configurations

— util.ht —

```
\newcommand{\autobutt}[1]{\helppage{#1}}
\newcommand{\autobuttons}{}
\newcommand{\exitbuttons}{}

\newcommand{\autobuttLayout}[1]{\centerline{#1}}}
\newcommand{\autobuttMaker}[1]{\autobuttLayout{\HelpButton{#1}}}
\newcommand{\riddlebuttons}[1]{\autobuttLayout{\link{\HelpBitmap}{#1}}}

% Macro for downward compatibility (?).

\newcommand{\simplebox}[2]
{\inputbox[#1]{#2}{\htbitmap{xbox}}{\htbitmap{xopenbox}}}
```

———-

HyperDoc graphics macros

— util.ht —

```
% Including viewport bitmaps within \HyperName pages:
```

```
\newcommand{\viewport}[1]{\inputimage{{#1}.view/image}}
\newcommand{\axiomViewport}[1]
{\inputimage{\env{AXIOM}/doc/viewports/{#1}.view/image}}
\newcommand{\spadviewport}[1]{\axiomViewport{#1}}

% Creating a real live viewport:

\newcommand{\viewportbutton}[2]{\unixcommand{#1}{viewalone #2}}
\newcommand{\axiomViewportbutton}[2]
{\unixcommand{#1}{viewalone \$AXIOM/doc/viewports/{#2}}}
\newcommand{\spadviewportbutton}[2]{\axiomViewportbutton{#1}{#2}}

% Making active viewport buttons:

\newcommand{\viewportasbutton}[1]
{\unixcommand{\inputimage{{#1}.view/image}}{viewalone {#1}}}
\newcommand{\axiomViewportasbutton}[1]
{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/{#1}.view/image}}
{viewalone \$AXIOM/doc/viewports/{#1}}}
\newcommand{\spadviewportasbutton}[1]{\axiomViewportasbutton{#1}}
```

————-

TeX and LaTeX compatibility macros

— **util.ht** —

```
%% Begin macros that are needed because HD uses the wrong names

\newcommand{\center}[1]{\centerline{#1}}
\newcommand{\box}[1]{\fbox{#1}}

%% End   macros that are needed because HD uses the wrong names

\newcommand{\LARGE}{}
\newcommand{\LaTeX}{LaTeX}
\newcommand{\Large}{}
\newcommand{\TeX}{TeX}
\newcommand{\allowbreak}{}
\newcommand{\aleph}{\inputbitmap{\htbmdir{}/aleph.bitmap}}
\newcommand{\alpha}{\inputbitmap{\htbmdir{}/alpha.bitmap}}
\newcommand{\angle}{\inputbitmap{\htbmdir{}/angle.bitmap}}
\newcommand{\backslash}{\inputbitmap{\htbmdir{}/backslash.bitmap}}
\newcommand{\beta}{\inputbitmap{\htbmdir{}/beta.bitmap}}
\newcommand{\bigbreak}{\newline\newline}
\newcommand{\bot}{\inputbitmap{\htbmdir{}/bot.bitmap}}
```

```
\newcommand{\bullet}{\inputbitmap{\htbmdir{}/bullet.bitmap}}
\newcommand{\caption}[1]{\newline\centerline{#1}\newline}
\newcommand{\chi}{\inputbitmap{\htbmdir{}/chi.bitmap}}
\newcommand{\cite}[1]{bibliography entry for {\it #1}}
\newcommand{\cleardoublepage}{}
\newcommand{\coprod}{\inputbitmap{\htbmdir{}/coprod.bitmap}}
\newcommand{\del}{\inputbitmap{\htbmdir{}/del.bitmap}}
\newcommand{\delta}{\inputbitmap{\htbmdir{}/delta.bitmap}}
\newcommand{\Delta}{\inputbitmap{\htbmdir{}/delta-cap.bitmap}}
\newcommand{\div}{\inputbitmap{\htbmdir{}/div.bitmap}}
\newcommand{\dot}{\inputbitmap{\htbmdir{}/dot.bitmap}}
\newcommand{\ell}{\inputbitmap{\htbmdir{}/ell.bitmap}}
\newcommand{\emptyset}{\inputbitmap{\htbmdir{}/emptyset.bitmap}}
\newcommand{\epsilon}{\inputbitmap{\htbmdir{}/epsilon.bitmap}}
\newcommand{\epsffile}{}
\newcommand{\eta}{\inputbitmap{\htbmdir{}/eta.bitmap}}
\newcommand{\exists}{\inputbitmap{\htbmdir{}/exists.bitmap}}
\newcommand{\forall}{\inputbitmap{\htbmdir{}/forall.bitmap}}
\newcommand{\footnote}[1]{ {(#1)}}
\newcommand{\frenchspacing}{}
\newcommand{\gamma}{\inputbitmap{\htbmdir{}/gamma.bitmap}}
\newcommand{\Gamma}{\inputbitmap{\htbmdir{}/gamma-cap.bitmap}}
\newcommand{\hbar}{\inputbitmap{\htbmdir{}/hbar.bitmap}}
\newcommand{\hbox}[1]{{#1}}
\newcommand{\hfill}{}
\newcommand{\hfil}{}
\newcommand{\huge}{}
\newcommand{\Im}{\inputbitmap{\htbmdir{}/im-cap.bitmap}}
\newcommand{\imath}{\inputbitmap{\htbmdir{}/imath.bitmap}}
\newcommand{\infty}{\inputbitmap{\htbmdir{}/infty.bitmap}}
\newcommand{\int}{\inputbitmap{\htbmdir{}/int.bitmap}}
\newcommand{\iota}{\inputbitmap{\htbmdir{}/iota.bitmap}}
\newcommand{\index}[1]{}
\newcommand{\jmath}{\inputbitmap{\htbmdir{}/jmath.bitmap}}
\newcommand{\kappa}{\inputbitmap{\htbmdir{}/kappa.bitmap}}
\newcommand{\label}[1]{}
\newcommand{\lambda}{\inputbitmap{\htbmdir{}/lambda.bitmap}}
\newcommand{\Lambda}{\inputbitmap{\htbmdir{}/lambda-cap.bitmap}}
\newcommand{\large}{}
\newcommand{\ldots}{...}
\newcommand{\le}{<=}
\newcommand{\marginpar}[1]{}
\newcommand{\mu}{\inputbitmap{\htbmdir{}/mu.bitmap}}
\newcommand{\neg}{\inputbitmap{\htbmdir{}/neg.bitmap}}
\newcommand{\newpage}{}
\newcommand{\noindent}{\indent{0}}
\newcommand{\nonfrenchspacing}{}
\newcommand{\nabla}{\inputbitmap{\htbmdir{}/nabla.bitmap}}
\newcommand{\nu}{\inputbitmap{\htbmdir{}/nu.bitmap}}
\newcommand{\omega}{\inputbitmap{\htbmdir{}/omega.bitmap}}
```

```
\newcommand{\Omega}{\inputbitmap{\htbmdir{}/omega-cap.bitmap}}
\newcommand{\pageref}[1]{???}
\newcommand{\parallel}{\inputbitmap{\htbmdir{}/parallel.bitmap}}
\newcommand{\partial}{\inputbitmap{\htbmdir{}/partial.bitmap}}
\newcommand{\phi}{\inputbitmap{\htbmdir{}/phi.bitmap}}
\newcommand{\Phi}{\inputbitmap{\htbmdir{}/phi-cap.bitmap}}
\newcommand{\pi}{\inputbitmap{\htbmdir{}/pi.bitmap}}
\newcommand{\Pi}{\inputbitmap{\htbmdir{}/pi-cap.bitmap}}
\newcommand{\prime}{\inputbitmap{\htbmdir{}/prime.bitmap}}
\newcommand{\prod}{\inputbitmap{\htbmdir{}/prod.bitmap}}
\newcommand{\protect}{}
\newcommand{\psi}{\inputbitmap{\htbmdir{}/psi.bitmap}}
\newcommand{\Psi}{\inputbitmap{\htbmdir{}/psi-cap.bitmap}}
\newcommand{\quad}{\inputbitmap{\htbmdir{}/quad.bitmap}}
\newcommand{\Re}{\inputbitmap{\htbmdir{}/re-cap.bitmap}}
\newcommand{\rho}{\inputbitmap{\htbmdir{}/rho.bitmap}}
\newcommand{\sc}{\rm}
\newcommand{\sf}{\bf}
\newcommand{\sigma}{\inputbitmap{\htbmdir{}/sigma.bitmap}}
\newcommand{\Sigma}{\inputbitmap{\htbmdir{}/sigma-cap.bitmap}}
\newcommand{\small}{}
\newcommand{\sum}{\inputbitmap{\htbmdir{}/sum.bitmap}}
\newcommand{\surd}{\inputbitmap{\htbmdir{}/surd.bitmap}}
\newcommand{\tau}{\inputbitmap{\htbmdir{}/tau.bitmap}}
\newcommand{\theta}{\inputbitmap{\htbmdir{}/theta.bitmap}}
\newcommand{\Theta}{\inputbitmap{\htbmdir{}/theta-cap.bitmap}}
\newcommand{\times}{\inputbitmap{\htbmdir{}/times.bitmap}}
\newcommand{\top}{\inputbitmap{\htbmdir{}/top.bitmap}}
\newcommand{\triangle}{\inputbitmap{\htbmdir{}/triangle.bitmap}}
\newcommand{\upsilon}{\inputbitmap{\htbmdir{}/upsilon.bitmap}}
\newcommand{\Upsilon}{\inputbitmap{\htbmdir{}/upsilon-cap.bitmap}}
\newcommand{\vbox}[1]{{#1}}
\newcommand{\wp}{\inputbitmap{\htbmdir{}/wp.bitmap}}
\newcommand{\xi}{\inputbitmap{\htbmdir{}/xi.bitmap}}
\newcommand{\Xi}{\inputbitmap{\htbmdir{}/xi-cap.bitmap}}
\newcommand{\zeta}{\inputbitmap{\htbmdir{}/zeta.bitmap}}
\newcommand{\bs}{\\}
```

——–

Book and .ht page macros

— util.ht —

```
\newcommand{\beginImportant}{\horizontalline}
\newcommand{\endImportant}{\horizontalline}
%
```

```
% following handles things like "i-th" but uses "-th"
\newcommand{\eth}[1]{{#1}-th}}
%
\newcommand{\texnewline}{}
\newcommand{\texbreak}{}
\newcommand{\Gallery}{{Axiom Images}}
\newcommand{\exptypeindex}[1]{}
\newcommand{\gotoevenpage}{}
\newcommand{\ignore}[1]{}
\newcommand{\ind}{\newline\tab{3}}
\newcommand{\labelSpace}[1]{}
\newcommand{\mathOrSpad}[1]{{\spad{#1}}}
\newcommand{\menuspadref}[2]{\menudownlink{#1}{#2Page}}
\newcommand{\menuxmpref}[1]{\menudownlink{`#1'}{#1XmpPage}}
% comment and then \spadcommand or spadsrc
\newcommand{\noOutputXtc}[2]{\xtc{#1}{#2}}
\newcommand{\not=}{\inputbitmap{\htbmdir{}/not=.bitmap}}
\newcommand{\notequal}{\inputbitmap{\htbmdir{}/notequal.bitmap}}
% comment and then \spadcommand or spadsrc
\newcommand{\nullXtc}[2]{\xtc{#1}{#2}}
\newcommand{\nullspadcommand}[1]{\spadcommand}
% Use this instead of \par for now.
\newcommand{\pp}{\newline}
% comment and then \spadcommand or spadsrc
\newcommand{\psXtc}[3]{\xtc{#1}{#2}}
\newcommand{\ref}[1]{(see the graph)}
\newcommand{\showBlurb}[1]
{Issue the system command \spadcmd{)show #1} to display the full list
of operations defined by \spadtype{#1}.}
\newcommand{\smath}[1]{\mathOrSpad{#1}}
\newcommand{\spadFileExt}{.spad}
\newcommand{\spadkey}[1]{}
\newcommand{\spadref} [1]{{\it #1}}
\newcommand{\spadsig}[2]{\spadtype{#1} {\tt ->} \spadtype{#2}}
\newcommand{\axiomSig}[2]{\axiomType{#1} {\tt ->} \axiomType{#2}}
\newcommand{\subscriptIt}[2]{{\it {#1}\_{#2}}}
\newcommand{\subscriptText}[2]{{\it {#1}\_{\it #2}}}
\newcommand{\subsubsection}[1]{\newline\indent{0}{\bf #1}\newline\newline}
\newcommand{\syscmdindex}[1]{}              % system command index macro
\newcommand{\threedim}{three-dimensional}
\newcommand{\twodim}{two-dimensional}
\newcommand{\unind}{\newline}
\newcommand{\void}{the unique value of \spadtype{Void}}
\newcommand{\xdefault}[1]{The default value is {\tt "#1"}.}
\newcommand{\xmpLine}[2]{{\tt #1}\newline}   % have to improve someday
\newcommand{\xmpref}[1]{\downlink{`#1'}{#1XmpPage}}
% comment and then \spadcommand or spadsrc
\newcommand{\xtc}[2]{#1 #2}

% glossary terms
```

```
\newcommand{\spadgloss}[1]{\lispdownlink{#1}{(|htGloss| '|#1|)}}
\newcommand{\spadglossSee}[2]{\lispdownlink{#1}{(|htGloss| '|#2|)}}

% use this for syntax punctuation: \axiomSyntax{::}
\newcommand{\axiomSyntax}[1]{``{\tt #1}''}
\newcommand{\spadSyntax}[1]{\axiomSyntax{#1}}

% constructors
\newcommand{\axiomType}[1]{\lispdownlink{#1}{(|spadType| '|#1|)}}
\newcommand{\spadtype}[1]{\axiomType{#1}}
% things that browse can't handle
\newcommand{\nonLibAxiomType}[1]{{\it #1}}
\newcommand{\pspadtype}[1]{\nonLibAxiomType{#1}}

\newcommand{\axiom}   [1]{{\tt #1}} % note font
\newcommand{\spad}    [1]{\axiom{#1}}
% exists in ++ comments; to be removed
\newcommand{\spadvar} [1]{\axiom{#1}}
\newcommand{\s}       [1]{\axiom{#1}}

\newcommand{\httex}[2]{#1}
\newcommand{\texht}[2]{#2}

% Function names:
%
% The X versions below are used because Axiom function names that end
% in ``!'' cause problems for makeindex for had-copy.
%
% Example: \spadfunFromX{reverse}{List} prints as   reverse!
%
% In the "From" versions, the first arg is function name,
% second is constructor where exported.
%
% Use the "op" flavors of "-", "+", "*" etc, otherwise the "fun" flavors

\newcommand{\userfun} [1]{{\bf #1}} % example, non-library function names

\newcommand{\fakeAxiomFun}[1]{{\bf #1}} % not really a library function
\newcommand{\pspadfun} [1]{\fakeAxiomFun{#1}}

\newcommand{\axiomFun} [1]{\lispdownlink{#1}{(|oPage| '|#1|)}}
\newcommand{\spadfun} [1]{\axiomFun{#1}}
\newcommand{\axiomFunX}[1]{\axiomFun{#1!}}
\newcommand{\spadfunX}[1]{\axiomFun{#1!}}

\newcommand{\axiomFunFrom}[2]{\lispdownlink{#1}{(|oPageFrom| '|#1| '|#2|)}}
\newcommand{\spadfunFrom}[2]{\axiomFunFrom{#1}{#2}}
\newcommand{\axiomFunFromX}[2]{\axiomFunFrom{#1!}{#2}}
\newcommand{\spadfunFromX}[2]{\axiomFunFrom{#1!}{#2}}
```

```
\newcommand{\axiomOp} [1]{\lispdownlink{#1}{(|oPage| '|#1|)}}
\newcommand{\spadop}  [1]{\axiomOp{#1}}
\newcommand{\axiomOpX}[1]{\axiomOp{#1!}}

\newcommand{\axiomOpFrom}[2]{\lispdownlink{#1}{(|oPageFrom| '|#1| '|#2|)}}
\newcommand{\spadopFrom} [2]{\axiomOpFrom{#1}{#2}}
\newcommand{\axiomOpFromX}[2] {\axiomOpFrom{#1!}{#2}}
\newcommand{\spadopFromX}[2] {\axiomOpFrom{#1!}{#2}}

\newcommand{\spadsyscom}[1]{{\tt #1}}
\newcommand{\spadcmd}[1]{\spadsyscom{#1}}
\newcommand{\spadsys}[1]{\spadsyscom{#1}}


% Following macros should be phased out in favor of ones above:

\newcommand{\gloss}[1]{\lispdownlink{#1}{(|htGloss| '|#1|)}}
\newcommand{\spadglos}[1]{\lispdownlink{#1}{(|htGloss| '|#1|)}}
\newcommand{\glossSee}[2]{\lispdownlink{#1}{(|htGloss| '|#2|)}}
```

———-

Browse macros

— util.ht —

```
\newcommand{\undocumented}[0]{is not documented yet}
\newcommand{\aliascon}[2]{\lispdownlink{#1}{(|conPage| '|#2|)}}
\newcommand{\aliasdom}[2]{\lispdownlink{#1}{(|conPage| '|#2|)}}
\newcommand{\andexample}[1]{\indent{5}\spadcommand{#1}\indent{0}\newline}
\newcommand{\blankline}{\vspace{1}\newline }
\newcommand{\con}[1]{\lispdownlink{#1}{(|conPage| '|#1|)}}

\newcommand{\conf}[2]{\lispdownlink{#1}{(|conPage| '{#2})}}
% generalizes "con" to allow arbitrary title and form

\newcommand{\ops}[3]{\lisplink{#1}{(|conOpPage| #2 '{#3})}}
% does lisplink to operation page of a constructor or form
% #1 is constructor name or form, without fences, e.g. "Matrix(Integer)"
% #2 is page number, extracted from $curPage (see fromHeading/dbOpsForm)
% #3 is constructor name or form, with fences, e.g. "(|Matrix| (|Integer|))"

\newcommand{\dlink}[2]{\downlink{#2}{#1}}
\newcommand{\dom}[1]{\lispdownlink{#1}{(|conPage| '|#1|)}}
\newcommand{\example}[1]
{\newline\indent{5}\spadcommand{#1}\indent{0}\newline}
\newcommand{\lisp}[2]{\lispdownlink{#2}{#1}}
```

```
\newcommand{\spadatt} [1]{{\it #1}}
\newcommand{\indented}[2]
{\indentrel{#1}\newline #2\indentrel{-#1}\newline}
\newcommand{\keyword}[1]{\lispdownlink{#1}{(|htsn| '|#1|)}}
\newcommand{\op}[1]{\lispdownlink{#1}{(|htsn| '|#1|)}}
\newcommand{\spadignore}[1]{#1}
```

———-

Support for output and graph paste-ins

— **util.ht** —

```
\newcommand{\axiomcommand}[1]{\spadcommand{#1}}
\newcommand{\axiomgraph}[1]{\spadgraph{#1}}

\newcommand{\pastecommand}[1]{\spadpaste{#1}}
\newcommand{\pastegraph}[1]{\graphpaste{#1}}

\newcommand{\showpaste}{\htbitmap{sdown3d}}
\newcommand{\hidepaste}{\htbitmap{sup3d}}
\newcommand{\spadpaste}[1]{
  \newline
  \begin{paste}{\pagename Empty \examplenumber}
{\pagename Patch \examplenumber}
  \pastebutton{\pagename Empty \examplenumber}{\showpaste}
  \tab{5}\spadcommand{#1}
  \end{paste}
}

\newcommand{\graphpaste}[1]{
  \newline
  \begin{paste}{\pagename Empty \examplenumber}
{\pagename Patch \examplenumber}
  \pastebutton{\pagename Empty \examplenumber}{\showpaste}
  \tab{5}\spadgraph{#1}
  \end{paste}
}
```

———-

Hook for including a local menu item on the rootpage

— **util.ht** —

```
\newcommand{\localinfo}{}
```

———-

Not Connected to Axiom

⇐ "Standard Pages" (HTXLinkPage2) 21.22 on page 2672

— util.ht —

```
\begin{page}{SpadNotConnectedPage}{Not Connected to Axiom}
\beginscroll
Hyperdoc isn't connected to Axiom, therefore cannot execute
the button you pressed.
%
\GoBackToWork{}
\endscroll
\end{page}
```

———-

Do You Really Want to Exit?

— util.ht —

```
\begin{page}{ProtectedQuitPage}{Do You Really Want to Exit?}
\beginscroll
{Click again on \  \ExitButton{QuitPage} \  to terminate Hyperdoc.}
\vspace{1}\newline
\centerline{OR}
\GoBackToWork{}
\endscroll
\autobuttons
\end{page}
```

———-

Missing Page

— util.ht —

```
\begin{page}{UnknownPage}{Missing Page}
\beginscroll
\pp
The page you requested was not found in the Hyperdoc database.
\GoBackToWork{}
\endscroll
\end{page}
```

———-

Something is Wrong

— util.ht —

```
\begin{page}{ErrorPage}{Something is Wrong}
\beginscroll
{For some reason the page you tried to link to cannot be formatted.}
\GoBackToWork{}
\endscroll
\autobuttons
\end{page}
```

———-

Sorry!

— util.ht —

```
\begin{page}{Unlinked}{Sorry!}
\beginscroll
{This link is not implemented yet.}
\GoBackToWork{}
\endscroll
\autobuttons
\end{page}
```

———-

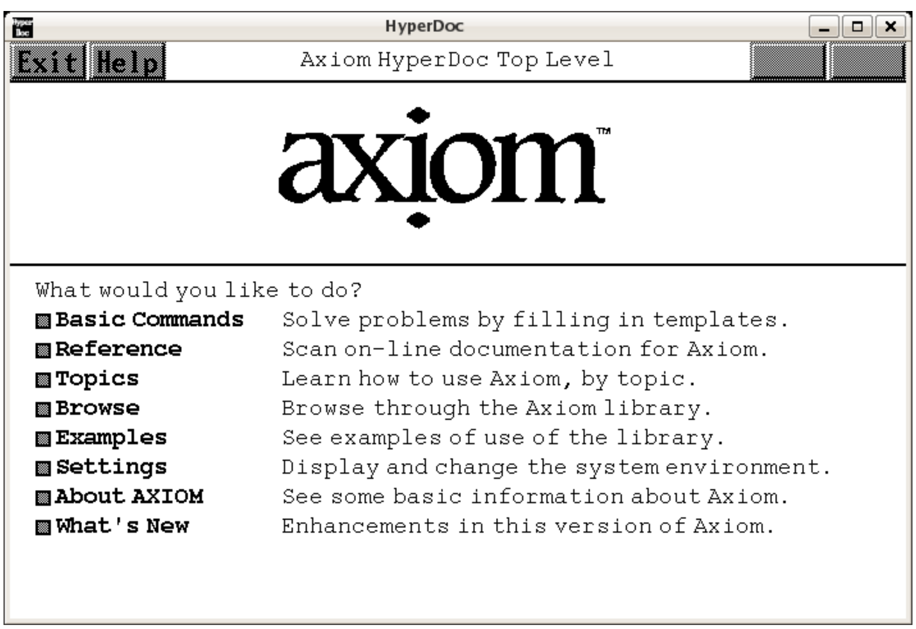
Chapter 3

Hyperdoc pages

The hyperdoc pages can be viewed as a forest of rooted pages. The main routine in hypertex will look for a page called “RootPage”.
⇐ “Root Page” (RootPage) 3.1 on page 117

3.1 rootpage.ht

Axiom HyperDoc Top Level



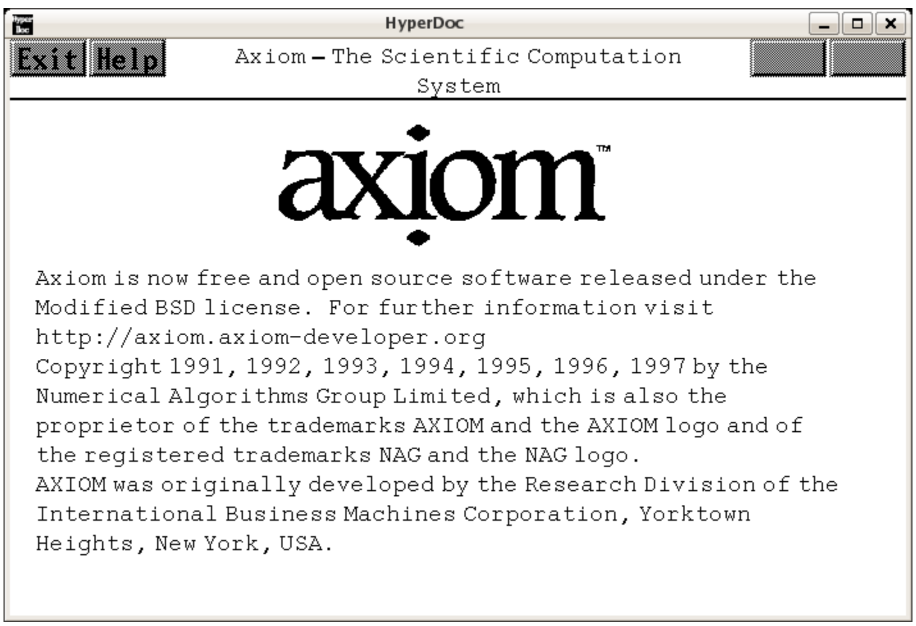
⇒ "Basic Commands" (BasicCommand) 3.6 on page 155
⇒ "Reference" (TopReferencePage) 3.1 on page 123
⇒ "Topics" (TopicPage) 3.108 on page 1313
⇒ "Browse" (Man0Page) 3.71 on page 962
⇒ "Examples" (TopExamplePage) 3.1 on page 121
⇒ "Settings" (TopSettingsPage) 3.1 on page 120
⇒ "About Axiom" (RootPageLogo) 3.1 on page 119
⇒ "What's New" (releaseNotes) 1.1 on page 1

— **rootpage.ht** —

```
\begin{page}{RootPage}{Axiom HyperDoc Top Level}
\beginscroll
\centerline{\inputbitmap{\htbmdir{}/axiom1.bitmap}}
\horizontalline
\newline\tab{0}
What would you like to do?

\beginmenu

\menuwindowlink{Basic Commands}{BasicCommand}
\tab{16}Solve problems by filling in templates.

\menuwindowlink{Reference}{TopReferencePage}
\tab{16}Scan on-line documentation for Axiom.

\menuwindowlink{Topics}{TopicPage}
\tab{16}Learn how to use Axiom, by topic.

\menuwindowlink{Browse}{Man0Page}
\tab{16}Browse through the Axiom library.

\menuwindowlink{Examples}{TopExamplePage}
\tab{16}See examples of use of the library.

\menuwindowlink{Settings}{TopSettingsPage}
\tab{16}Display and change the system environment.

%\menuwindowlink{NAG Link}{htxl}
%\tab{16} Link to NAG Numerical Library.

%\menuwindowlink{\inputbitmap{\htbmdir{}/anna.xbm.tiny}}{UXANNA}
%\tab{16} The Axiom/NAG Numerical Analyst Expert System

\menuwindowlink{About Axiom}{RootPageLogo}
\tab{16}See some basic information about Axiom.

%\menuwindowlink{What's New}{ugWhatsNewTwoTwoPage}
\menuwindowlink{What's New}{releaseNotes}
\tab{16}Enhancements in this version of Axiom.
```

```
%menulispwindowlink is unsafe:TTT
%\menulispwindowlink{Settings}{(|htSystemVariables|)}
%\tab{16}Change an Axiom system variable.
%\menulispwindowlink{User Variables}{(|htUserVariables|)}
%\tab{16}Review user variables and \lispwindowlink{history}{(|htHistory|)}


%\localinfo
\endmenu
\endscroll
\autobutt{ugHyperPage}
\end{page}
```

Axiom – The Scientific Computation System



⇐ "Root Page" (RootPage) 3.1 on page 117

— **rootpage.ht** —

```
\begin{page}{RootPageLogo}{Axiom -- The Scientific Computation System}
\beginscroll
\centerline{\inputbitmap{\htbmdir{}/axiom1.bitmap}}
Axiom is now free and open source software released under the Modified
```

```
BSD license. For further information visit
\newline
http://axiom.axiom-developer.org
\newline
Copyright 1991, 1992, 1993, 1994, 1995, 1996, 1997 by
the Numerical Algorithms Group Limited, which is also the
proprietor of the trademarks Axiom and the Axiom logo and
of the registered trademarks NAG and the NAG logo.
\newline
Axiom was originally developed by the Research Division of
the International Business Machines Corporation, Yorktown
Heights, New York, USA.
\endscroll
\end{page}
```

System Commands

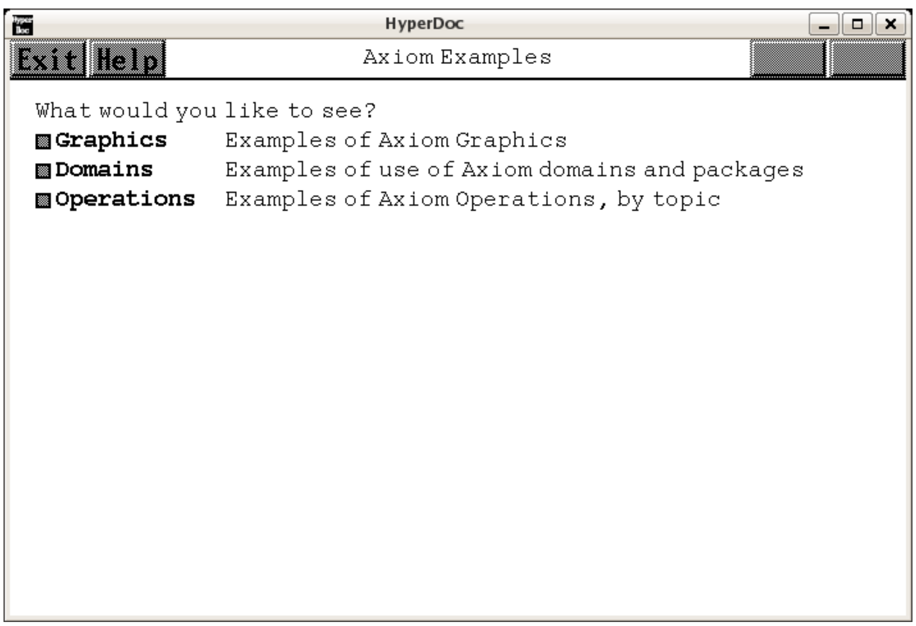
⇐ “Root Page” (RootPage) 3.1 on page 117
⇒ “Commands” (ugSysCmdPage) 19 on page 2538
The “Settings” link is implemented in lisp.

— **rootpage.ht** —

```
\begin{page}{TopSettingsPage}{System Commands}
```

```
\beginscroll
System commands are used to perform Axiom environment
management and change Axiom system variables.
\beginmenu
\menumemolink{Commands}{ugSysCmdPage}
\tab{12}System commands that control your environment.

\menulispmemolink{Settings}{(|htSystemVariables|)}
\tab{12}Change an Axiom variable.

\endmenu
\endscroll
\end{page}
```

Axiom Examples



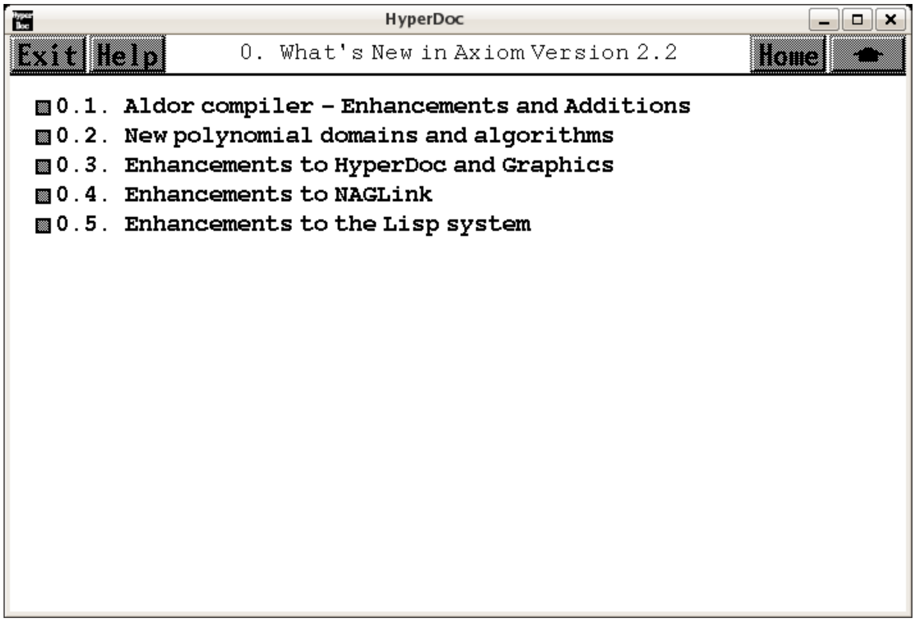
⇐ "Root Page" (RootPage) 3.1 on page 117
⇒ "Graphics" (GraphicsExamplePage) 3.50 on page 602
⇒ "Domains" (ExamplesExposedPage) 3.117 on page 1369
⇒ "Operations" (ExampleCoverPage) 3.20 on page 304

— **rootpage.ht** —

```
\begin{page}{TopExamplePage}{Axiom Examples}
```

```
\beginscroll
What would you like to see?
\beginmenu
\item\menudownlink{Graphics}{GraphicsExamplePage}
\tab{12}Examples of Axiom Graphics
\item\menudownlink{Domains}{ExamplesExposedPage}
\tab{12}Examples of use of Axiom domains and packages
\item\menudownlink{Operations}{ExampleCoverPage}
\tab{12}Examples of Axiom Operations, by topic
\endmenu
\endscroll
\autobuttons
\end{page}

\beginmenu
\item\menudownlink{Graphics}{GraphicsExamplePage}
\tab{12}Examples of Axiom Graphics
\item\menudownlink{Domains}{ExamplesExposedPage}
\tab{12}Examples of use of Axiom domains and packages
\item\menudownlink{Operations}{ExampleCoverPage}
\tab{12}Examples of Axiom Operations, by topic
\endmenu
\endscroll
\autobuttons
\end{page}
```

Axiom Reference



⇐ "Root Page" (RootPage) 3.1 on page 117
⇒ "this font" (YouTriedIt) 3.78 on page 1018
⇒ "What's New" (ugWhatsNewTwoTwoPage) 5 on page 1473
⇒ "Axiom Book" (UsersGuidePage) 4 on page 1470
⇒ "NAG Library" (FoundationLibraryDocPage) 3.1 on page 125
⇒ "Topics" (TopicPage) 3.108 on page 1313
⇒ "Language" (ugLangPage) ?? on page ??
⇒ "Examples" (ExamplesExposedPage) 3.117 on page 1369
⇒ "Commands" (ugSysCmdPage) 19 on page 2538
⇒ "Glossary" (GlossaryPage) 3.49 on page 579
⇒ "HyperDoc" (HTXTopPage) 21.28 on page 2699
⇒ "Search" (RefSearchPage) 3.71 on page 951

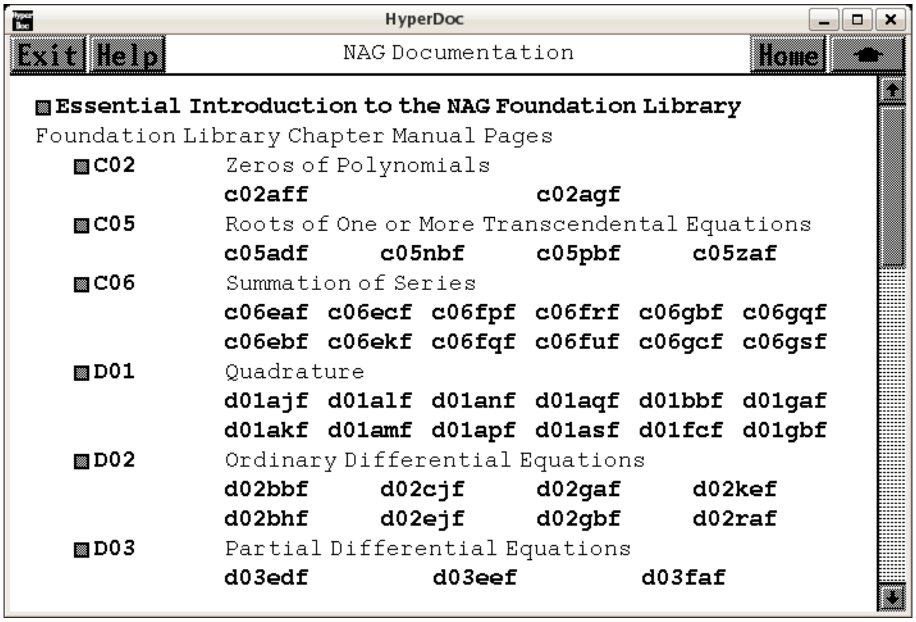
— **rootpage.ht** —

```
\begin{page}{TopReferencePage}{Axiom Reference}
\newline
\pp
To select an item, move the cursor with the mouse to
any word in \downlink{this font}{YouTriedIt} and click the left mouse button.
\beginscroll
\beginmenu
\menumemolink{What's New}{ugWhatsNewTwoTwoPage}
\tab{12}A synopsis of what's new in this release.
```

```
\menumemolink{Axiom Book}{UsersGuidePage}
\tab{12}The on-line version of the Jenks/Sutor book.

%\menumemolink{\asharp{} Guide}{AsUsersGuidePage}
%\tab{12}The on-line \asharp{} Users Guide.

\menumemolink{NAG Library}{FoundationLibraryDocPage}
\tab{12}The on-line \naglib{} documentation.

\menumemolink{Topics}{TopicPage}
\tab{12}Learn how to use Axiom, by topic.

\menumemolink{Language}{ugLangPage}
\tab{12}Introduction to the Axiom language.

\menumemolink{Examples}{ExamplesExposedPage}
\tab{12}Examples for exposed domains and packages

\menumemolink{Commands}{ugSysCmdPage}
\tab{12}System commands that control your workspace.

%\menumemolink{Operations}{NoPageYet} \tab{12}
%A guide to useful operations

%\menulispcommand{System Variables}{(|htsv|)}\tab{12}
%\tab{16}View and change a system-defined variable

\menumemolink{Glossary}{GlossaryPage}\tab{12}
A glossary of Axiom terms.

\menumemolink{Hyperdoc}{HTXTopPage}
\tab{12} How to write your own Hyperdoc pages.

\menumemolink{Search}{RefSearchPage}
\tab{12} Reference pages for occurrences of a string.
\endmenu
\endscroll
\autobuttons
\end{page}
```

———-

NAG Documentation



⇐ "Reference" (TopReferencePage) 3.1 on page 123

— **rootpage.ht** —

```
\begin{page}{FoundationLibraryDocPage}{NAG Documentation}
\begin{scroll}

\beginitems
\item\downlink{\menuitemstyle
{Essential Introduction to the NAG Foundation Library}}{manpageXXintro}
\item Foundation Library Chapter Manual Pages
\beginitems
\item\downlink{\menuitemstyle{C02}}
{manpageXXc02}\tab{8} Zeros of Polynomials
\indentrel{8}\newline
\table{
{\downlink{c02aff}{manpageXXc02aff}}
{\downlink{c02agf}{manpageXXc02agf}}
}\indentrel{-8}
\item\downlink{\menuitemstyle{C05}}
{manpageXXc05}\tab{8} Roots of One or More Transcendental Equations
\indentrel{8}\newline
\table{
{\downlink{c05adf}{manpageXXc05adf}}
{\downlink{c05nbf}{manpageXXc05nbf}}
{\downlink{c05pbf}{manpageXXc05pbf}}
```

```
{\downlink{c05zaf}{manpageXXc05zaf}}
}\indentrel{-8}
\item\downlink{\menuitemstyle{C06}}
{manpageXXc06}\tab{8} Summation of Series
\indentrel{8}\newline
\table{
{\downlink{c06eaf}{manpageXXc06eaf}}
{\downlink{c06ebf}{manpageXXc06ebf}}
{\downlink{c06ecf}{manpageXXc06ecf}}
{\downlink{c06ekf}{manpageXXc06ekf}}
{\downlink{c06fpf}{manpageXXc06fpf}}
{\downlink{c06fqf}{manpageXXc06fqf}}
{\downlink{c06frf}{manpageXXc06frf}}
{\downlink{c06fuf}{manpageXXc06fuf}}
{\downlink{c06gbf}{manpageXXc06gbf}}
{\downlink{c06gcf}{manpageXXc06gcf}}
{\downlink{c06gqf}{manpageXXc06gqf}}
{\downlink{c06gsf}{manpageXXc06gsf}}
}\indentrel{-8}
\item\downlink{\menuitemstyle{D01}}{manpageXXd01}\tab{8} Quadrature
\indentrel{8}\newline
\table{
{\downlink{d01ajf}{manpageXXd01ajf}}
{\downlink{d01akf}{manpageXXd01akf}}
{\downlink{d01alf}{manpageXXd01alf}}
{\downlink{d01amf}{manpageXXd01amf}}
{\downlink{d01anf}{manpageXXd01anf}}
{\downlink{d01apf}{manpageXXd01apf}}
{\downlink{d01aqf}{manpageXXd01aqf}}
{\downlink{d01asf}{manpageXXd01asf}}
{\downlink{d01bbf}{manpageXXd01bbf}}
{\downlink{d01fcf}{manpageXXd01fcf}}
{\downlink{d01gaf}{manpageXXd01gaf}}
{\downlink{d01gbf}{manpageXXd01gbf}}
}\indentrel{-8}
\item\downlink{\menuitemstyle{D02}}
{manpageXXd02}\tab{8} Ordinary Differential Equations
\indentrel{8}\newline
\table{
{\downlink{d02bbf}{manpageXXd02bbf}}
{\downlink{d02bhf}{manpageXXd02bhf}}
{\downlink{d02cjf}{manpageXXd02cjf}}
{\downlink{d02ejf}{manpageXXd02ejf}}
{\downlink{d02gaf}{manpageXXd02gaf}}
{\downlink{d02gbf}{manpageXXd02gbf}}
{\downlink{d02kef}{manpageXXd02kef}}
{\downlink{d02raf}{manpageXXd02raf}}
}\indentrel{-8}
\item\downlink{\menuitemstyle{D03}}
{manpageXXd03}\tab{8} Partial Differential Equations
```

```
\indentrel{8}\newline
\table{
{\downlink{d03edf}{manpageXXd03edf}}
{\downlink{d03eef}{manpageXXd03eef}}
{\downlink{d03faf}{manpageXXd03faf}}
}\indentrel{-8}
\item\downlink{\menuitemstyle{E01}}{manpageXXe01}\tab{8} Interpolation
\indentrel{8}\newline
\table{
{\downlink{e01baf}{manpageXXe01baf}}
{\downlink{e01bef}{manpageXXe01bef}}
{\downlink{e01bff}{manpageXXe01bff}}
{\downlink{e01bgf}{manpageXXe01bgf}}
{\downlink{e01bhf}{manpageXXe01bhf}}
{\downlink{e01daf}{manpageXXe01daf}}
{\downlink{e01saf}{manpageXXe01saf}}
{\downlink{e01sbf}{manpageXXe01sbf}}
{\downlink{e01sef}{manpageXXe01sef}}
{\downlink{e01sff}{manpageXXe01sff}}
}\indentrel{-8}
\item\downlink{\menuitemstyle{E02}}
{manpageXXe02}\tab{8} Curve and Surface Fitting
\indentrel{8}\newline
\table{
{\downlink{e02adf}{manpageXXe02adf}}
{\downlink{e02aef}{manpageXXe02aef}}
{\downlink{e02agf}{manpageXXe02agf}}
{\downlink{e02ahf}{manpageXXe02ahf}}
{\downlink{e02ajf}{manpageXXe02ajf}}
{\downlink{e02akf}{manpageXXe02akf}}
{\downlink{e02baf}{manpageXXe02baf}}
{\downlink{e02bbf}{manpageXXe02bbf}}
{\downlink{e02bcf}{manpageXXe02bcf}}
{\downlink{e02bdf}{manpageXXe02bdf}}
{\downlink{e02bef}{manpageXXe02bef}}
{\downlink{e02daf}{manpageXXe02daf}}
{\downlink{e02dcf}{manpageXXe02dcf}}
{\downlink{e02ddf}{manpageXXe02ddf}}
{\downlink{e02def}{manpageXXe02def}}
{\downlink{e02dff}{manpageXXe02dff}}
{\downlink{e02gaf}{manpageXXe02gaf}}
{\downlink{e02zaf}{manpageXXe02zaf}}
}\indentrel{-8}
\item\downlink{\menuitemstyle{E04}}
{manpageXXe04}\tab{8} Minimizing or Maximizing a Function
\indentrel{8}\newline
\table{
{\downlink{e04dgf}{manpageXXe04dgf}}
{\downlink{e04djf}{manpageXXe04djf}}
{\downlink{e04dkf}{manpageXXe04dkf}}
```

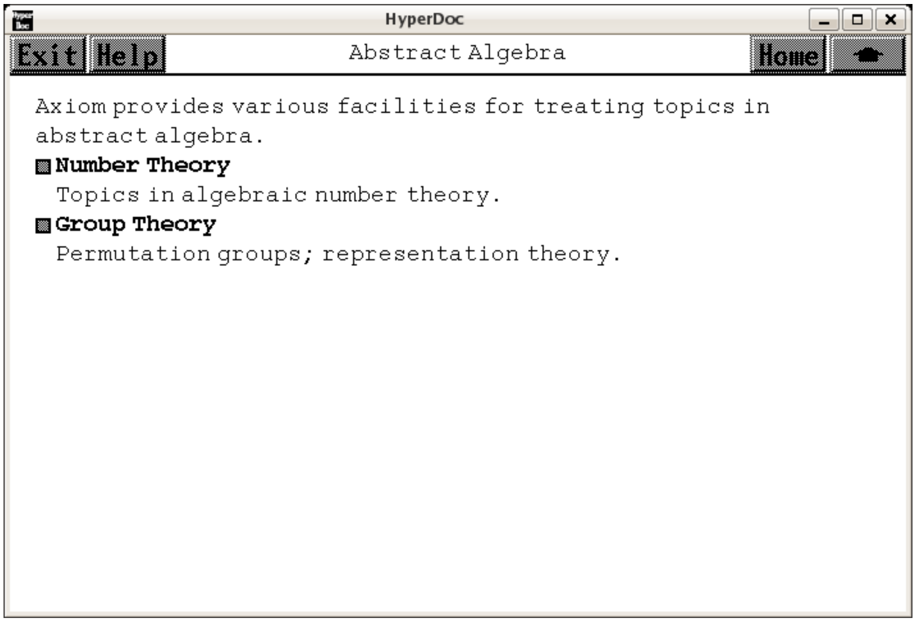
```
{\downlink{e04fdf}{manpageXXe04fdf}}
{\downlink{e04gcf}{manpageXXe04gcf}}
{\downlink{e04jaf}{manpageXXe04jaf}}
{\downlink{e04mbf}{manpageXXe04mbf}}
{\downlink{e04naf}{manpageXXe04naf}}
{\downlink{e04ucf}{manpageXXe04ucf}}
{\downlink{e04udf}{manpageXXe04udf}}
{\downlink{e04uef}{manpageXXe04uef}}
{\downlink{e04ycf}{manpageXXe04ycf}}
}\indentrel{-8}
\item\downlink{\menuitemstyle{F}}{manpageXXf}\tab{8} Linear Algebra
\item\downlink{\menuitemstyle{F01}}
{manpageXXf01}\tab{8} Matrix Operations, Including Inversion
\indentrel{8}\newline
\table{
{\downlink{f01brf}{manpageXXf01brf}}
{\downlink{f01bsf}{manpageXXf01bsf}}
{\downlink{f01maf}{manpageXXf01maf}}
{\downlink{f01mcf}{manpageXXf01mcf}}
{\downlink{f01qcf}{manpageXXf01qcf}}
{\downlink{f01qdf}{manpageXXf01qdf}}
{\downlink{f01qef}{manpageXXf01qef}}
{\downlink{f01rcf}{manpageXXf01rcf}}
{\downlink{f01rdf}{manpageXXf01rdf}}
{\downlink{f01ref}{manpageXXf01ref}}
}\indentrel{-8}
\item\downlink{\menuitemstyle{F02}}
{manpageXXf02}\tab{8} Eigenvalues and Eigenvectors
\indentrel{8}\newline
\table{
{\downlink{f02aaf}{manpageXXf02aaf}}
{\downlink{f02abf}{manpageXXf02abf}}
{\downlink{f02adf}{manpageXXf02adf}}
{\downlink{f02aef}{manpageXXf02aef}}
{\downlink{f02aff}{manpageXXf02aff}}
{\downlink{f02agf}{manpageXXf02agf}}
{\downlink{f02ajf}{manpageXXf02ajf}}
{\downlink{f02akf}{manpageXXf02akf}}
{\downlink{f02awf}{manpageXXf02awf}}
{\downlink{f02axf}{manpageXXf02axf}}
{\downlink{f02bbf}{manpageXXf02bbf}}
{\downlink{f02bjf}{manpageXXf02bjf}}
{\downlink{f02fjf}{manpageXXf02fjf}}
{\downlink{f02wef}{manpageXXf02wef}}
{\downlink{f02xef}{manpageXXf02xef}}
}\indentrel{-8}
\item\downlink{\menuitemstyle{F04}}
{manpageXXf04}\tab{8} Simultaneous Linear Equations
\indentrel{8}\newline
\table{
```

```
{\downlink{f04adf}{manpageXXf04adf}}
{\downlink{f04arf}{manpageXXf04arf}}
{\downlink{f04asf}{manpageXXf04asf}}
{\downlink{f04atf}{manpageXXf04atf}}
{\downlink{f04axf}{manpageXXf04axf}}
{\downlink{f04faf}{manpageXXf04faf}}
{\downlink{f04jgf}{manpageXXf04jgf}}
{\downlink{f04maf}{manpageXXf04maf}}
{\downlink{f04mbf}{manpageXXf04mbf}}
{\downlink{f04mcf}{manpageXXf04mcf}}
{\downlink{f04qaf}{manpageXXf04qaf}}
}\indentrel{-8}
\item\downlink{\menuitemstyle{F07}}
{manpageXXf07}\tab{8} Linear Equations (LAPACK)
\indentrel{8}\newline
\table{
{\downlink{f07adf}{manpageXXf07adf}}
{\downlink{f07aef}{manpageXXf07aef}}
{\downlink{f07fdf}{manpageXXf07fdf}}
{\downlink{f07fef}{manpageXXf07fef}}
}\indentrel{-8}
\item\downlink{\menuitemstyle{S}}
{manpageXXs}\tab{8} Approximations of Special Functions
\indentrel{8}\newline
\table{
{\downlink{s01eaf}{manpageXXs01eaf}}
{\downlink{s13aaf}{manpageXXs13aaf}}
{\downlink{s13acf}{manpageXXs13acf}}
{\downlink{s13adf}{manpageXXs13adf}}
{\downlink{s14aaf}{manpageXXs14aaf}}
{\downlink{s14abf}{manpageXXs14abf}}
{\downlink{s14baf}{manpageXXs14baf}}
{\downlink{s15adf}{manpageXXs15adf}}
{\downlink{s15aef}{manpageXXs15aef}}
{\downlink{s17acf}{manpageXXs17acf}}
{\downlink{s17adf}{manpageXXs17adf}}
{\downlink{s17aef}{manpageXXs17aef}}
{\downlink{s17aff}{manpageXXs17aff}}
{\downlink{s17agf}{manpageXXs17agf}}
{\downlink{s17ahf}{manpageXXs17ahf}}
{\downlink{s17ajf}{manpageXXs17ajf}}
{\downlink{s17akf}{manpageXXs17akf}}
{\downlink{s17dcf}{manpageXXs17dcf}}
{\downlink{s17def}{manpageXXs17def}}
{\downlink{s17dgf}{manpageXXs17dgf}}
{\downlink{s17dhf}{manpageXXs17dhf}}
{\downlink{s17dlf}{manpageXXs17dlf}}
{\downlink{s18acf}{manpageXXs18acf}}
{\downlink{s18adf}{manpageXXs18adf}}
{\downlink{s18aef}{manpageXXs18aef}}
```

```
{\downlink{s18aff}{manpageXXs18aff}}
{\downlink{s18dcf}{manpageXXs18dcf}}
{\downlink{s18def}{manpageXXs18def}}
{\downlink{s19aaf}{manpageXXs19aaf}}
{\downlink{s19abf}{manpageXXs19abf}}
{\downlink{s19acf}{manpageXXs19acf}}
{\downlink{s19adf}{manpageXXs19adf}}
{\downlink{s20acf}{manpageXXs20acf}}
{\downlink{s20adf}{manpageXXs20adf}}
{\downlink{s21baf}{manpageXXs21baf}}
{\downlink{s21bbf}{manpageXXs21bbf}}
{\downlink{s21bcf}{manpageXXs21bcf}}
{\downlink{s21bdf}{manpageXXs21bdf}}
}\indentrel{-8}
\enditems
\item\downlink{\menuitemstyle
{Introduction to NAG On-Line Documentation}}{manpageXXonline}
\item\downlink{\menuitemstyle{Keywords in Context}}{manpageXXkwic}
\item\downlink{\menuitemstyle{List of all \naglib{} Routines}}
{manpageXXsummary}
\item\downlink{\menuitemstyle
{Converting from the Workstation Library}}{manpageXXconvert}
\enditems
\end{scroll}
\end{page}
```

3.2 algebra.ht

Abstract Algebra



⇐ "Topics" (TopicPage) 3.108 on page 1313
⇒ "Number Theory" (NumberTheoryPage) 3.2 on page 132
⇒ "Group Theory" (GroupTheoryPage) 3.51 on page 664

— **algebra.ht** —

```
\begin{page}{AlgebraPage}{Abstract Algebra}
\beginscroll
Axiom provides various facilities for treating topics in abstract
algebra.
\beginmenu
\menulink{Number Theory}{NumberTheoryPage} \newline
Topics in algebraic number theory.
%\menulink{Algebraic Geometry}{AlgebraicGeometryPage} \newline
%Computational algebraic geometry: Groebner bases, integral bases,
%divisors on curves.
\menulink{Group Theory}{GroupTheoryPage} \newline
Permutation groups; representation theory.
\endmenu
\endscroll
\autobuttons
\end{page}
```

———-

Number Theory

⇒ "Galois Groups" (ugProblemGaloisPage) 12 on page 2306
⇒ "Number Theory Functions" (IntNumberTheoryFnsXmpPage) 3.56 on page 722

— **algebra.ht** —

```
\begin{page}{NumberTheoryPage}{Number Theory}
\beginscroll
Here are some sample computations using Axiom's algebraic number
facilities.
\beginmenu
\menulink{Galois Groups}{ugProblemGaloisPage} \newline
Computation of Galois groups using factorizations over number fields.
\menulink{Number Theory Functions}
{IntNumberTheoryFnsXmpPage}\newline
Some functions of interest to number theorists.
\endmenu
\endscroll
\autobuttons
\end{page}
```

———-

3.3 alist.ht

AssociationList

⇒ "Table" (TableXmpPage) 3.106 on page 1298
⇒ "List" (ListXmpPage) 3.64 on page 866

— **alist.ht** —

```
\begin{page}{AssociationListXmpPage}{AssociationList}
\beginscroll

The \spadtype{AssociationList} constructor provides a general
structure for associative storage.  This type provides association
lists in which data objects can be saved according to keys of any
type.  For a given association list, specific types must be chosen for
the keys and entries.  You can think of the representation of an
association list as a list of records with key and entry fields.
```

```
Association lists are a form of table and so most of the operations
available for \spadtype{Table} are also available for
\spadtype{AssociationList}.  They can also be viewed as lists and can
be manipulated accordingly.

\xtc{
This is a \pspadtype{Record} type with age and gender fields.
}{
\spadpaste{Data := Record(monthsOld : Integer, gender : String) \bound{Data}}
}
\xtc{
In this expression, \spad{al} is declared to be an association
list whose keys are strings and whose entries are the above records.
}{
\spadpaste{al : AssociationList(String,Data) \free{Data}\bound{al}}
}
\xtc{
The \spadfunFrom{table}{AssociationList} operation is used to create
an empty association list.
}{
\spadpaste{al := table() \free{al}\bound{al1}}
}
\xtc{
You can use assignment syntax to add things to the association list.
}{
\spadpaste{al."bob" := [407,"male"]\$Data \free{al1}\bound{al2}}
}
\xtc{
}{
\spadpaste{al."judith" := [366,"female"]\$Data \free{al2}\bound{al3}}
}
\xtc{
}{
\spadpaste{al."katie" := [24,"female"]\$Data \free{al3}\bound{al4}}
}
\xtc{
Perhaps we should have included a species field.
}{
\spadpaste{al."smokie" := [200,"female"]\$Data \free{al4}\bound{al5}}
}
\xtc{
Now look at what is in the association list.
Note that the last-added (key, entry) pair is at the beginning of the list.
}{
\spadpaste{al \free{al5}}
}
\xtc{
You can reset the entry for an existing key.
}{
\spadpaste{al."katie" := [23,"female"]\$Data \free{al5}\bound{al6}}
```

```
}
\xtc{
Use \spadfunFromX{delete}{AssociationList} to destructively remove
an element of the association list.
Use \spadfunFrom{delete}{AssociationList} to return a copy of the
association list with the element deleted.
The second argument is the index of the element to delete.
}{
\spadpaste{delete!(al,1) \free{al6}\bound{al7}}
}

For more information about tables,
see \downlink{`Table'}{TableXmpPage}\ignore{Table}.
For more information about lists,
see \downlink{`List'}{ListXmpPage}\ignore{List}.
\showBlurb{AssociationList}
\endscroll
\autobuttons
\end{page}

\begin{patch}{AssociationListXmpPagePatch1}
\begin{paste}{AssociationListXmpPageFull1}{AssociationListXmpPageEmpty1}
\pastebutton{AssociationListXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{
Data := Record(monthsOld : Integer, gender : String)\bound{Data }}
\indentrel{3}\begin{verbatim}
   (1)  Record(monthsOld: Integer,gender: String)
                                           Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{AssociationListXmpPageEmpty1}
\begin{paste}{AssociationListXmpPageEmpty1}{AssociationListXmpPagePatch1}
\pastebutton{AssociationListXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{
Data := Record(monthsOld : Integer, gender : String)\bound{Data }}
\end{paste}\end{patch}

\begin{patch}{AssociationListXmpPagePatch2}
\begin{paste}{AssociationListXmpPageFull2}{AssociationListXmpPageEmpty2}
\pastebutton{AssociationListXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{
al : AssociationList(String,Data)\free{Data }\bound{al }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{AssociationListXmpPageEmpty2}
\begin{paste}{AssociationListXmpPageEmpty2}{AssociationListXmpPagePatch2}
```

```
\pastebutton{AssociationListXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{
al : AssociationList(String,Data)\free{Data }\bound{al }}
\end{paste}\end{patch}

\begin{patch}{AssociationListXmpPagePatch3}
\begin{paste}{AssociationListXmpPageFull3}{AssociationListXmpPageEmpty3}
\pastebutton{AssociationListXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{al := table()\free{al }\bound{al1 }}
\indentrel{3}\begin{verbatim}
   (3)  table()
Type: AssociationList(String,Record(monthsOld: Integer,gender: String))
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{AssociationListXmpPageEmpty3}
\begin{paste}{AssociationListXmpPageEmpty3}{AssociationListXmpPagePatch3}
\pastebutton{AssociationListXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{al := table()\free{al }\bound{al1 }}
\end{paste}\end{patch}

\begin{patch}{AssociationListXmpPagePatch4}
\begin{paste}{AssociationListXmpPageFull4}{AssociationListXmpPageEmpty4}
\pastebutton{AssociationListXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{al."bob" := [407,"male"]$Data\free{al1 }\bound{al2 }}
\indentrel{3}\begin{verbatim}
   (4)  [monthsOld= 407,gender= "male"]
        Type: Record(monthsOld: Integer,gender: String)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{AssociationListXmpPageEmpty4}
\begin{paste}{AssociationListXmpPageEmpty4}{AssociationListXmpPagePatch4}
\pastebutton{AssociationListXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{al."bob" := [407,"male"]$Data\free{al1 }\bound{al2 }}
\end{paste}\end{patch}

\begin{patch}{AssociationListXmpPagePatch5}
\begin{paste}{AssociationListXmpPageFull5}{AssociationListXmpPageEmpty5}
\pastebutton{AssociationListXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{
al."judith" := [366,"female"]$Data\free{al2 }\bound{al3 }}
\indentrel{3}\begin{verbatim}
   (5)  [monthsOld= 366,gender= "female"]
        Type: Record(monthsOld: Integer,gender: String)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{AssociationListXmpPageEmpty5}
\begin{paste}{AssociationListXmpPageEmpty5}{AssociationListXmpPagePatch5}
```

```
\pastebutton{AssociationListXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{
al."judith" := [366,"female"]$Data\free{al2 }\bound{al3 }}
\end{paste}\end{patch}

\begin{patch}{AssociationListXmpPagePatch6}
\begin{paste}{AssociationListXmpPageFull6}{AssociationListXmpPageEmpty6}
\pastebutton{AssociationListXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{
al."katie" := [24,"female"]$Data\free{al3 }\bound{al4 }}
\indentrel{3}\begin{verbatim}
   (6)  [monthsOld= 24,gender= "female"]
        Type: Record(monthsOld: Integer,gender: String)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{AssociationListXmpPageEmpty6}
\begin{paste}{AssociationListXmpPageEmpty6}{AssociationListXmpPagePatch6}
\pastebutton{AssociationListXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{
al."katie" := [24,"female"]$Data\free{al3 }\bound{al4 }}
\end{paste}\end{patch}

\begin{patch}{AssociationListXmpPagePatch7}
\begin{paste}{AssociationListXmpPageFull7}{AssociationListXmpPageEmpty7}
\pastebutton{AssociationListXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{
al."smokie" := [200,"female"]$Data\free{al4 }\bound{al5 }}
\indentrel{3}\begin{verbatim}
   (7)  [monthsOld= 200,gender= "female"]
        Type: Record(monthsOld: Integer,gender: String)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{AssociationListXmpPageEmpty7}
\begin{paste}{AssociationListXmpPageEmpty7}{AssociationListXmpPagePatch7}
\pastebutton{AssociationListXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{
al."smokie" := [200,"female"]$Data\free{al4 }\bound{al5 }}
\end{paste}\end{patch}

\begin{patch}{AssociationListXmpPagePatch8}
\begin{paste}{AssociationListXmpPageFull8}{AssociationListXmpPageEmpty8}
\pastebutton{AssociationListXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{al\free{al5 }}
\indentrel{3}\begin{verbatim}
   (8)
   table
      "smokie"= [monthsOld= 200,gender= "female"]
  ,
```

```
      "katie"= [monthsOld= 24,gender= "female"]
  ,
      "judith"= [monthsOld= 366,gender= "female"]
  ,
      "bob"= [monthsOld= 407,gender= "male"]
Type: AssociationList(String,Record(monthsOld: Integer,gender: String))
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{AssociationListXmpPageEmpty8}
\begin{paste}{AssociationListXmpPageEmpty8}{AssociationListXmpPagePatch8}
\pastebutton{AssociationListXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{al\free{al5 }}
\end{paste}\end{patch}

\begin{patch}{AssociationListXmpPagePatch9}
\begin{paste}{AssociationListXmpPageFull9}{AssociationListXmpPageEmpty9}
\pastebutton{AssociationListXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{
al."katie" := [23,"female"]$Data\free{al5 }\bound{al6 }}
\indentrel{3}\begin{verbatim}
   (9)  [monthsOld= 23,gender= "female"]
        Type: Record(monthsOld: Integer,gender: String)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{AssociationListXmpPageEmpty9}
\begin{paste}{AssociationListXmpPageEmpty9}{AssociationListXmpPagePatch9}
\pastebutton{AssociationListXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{
al."katie" := [23,"female"]$Data\free{al5 }\bound{al6 }}
\end{paste}\end{patch}

\begin{patch}{AssociationListXmpPagePatch10}
\begin{paste}{AssociationListXmpPageFull10}{AssociationListXmpPageEmpty10}
\pastebutton{AssociationListXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{delete!(al,1)\free{al6 }\bound{al7 }}
\indentrel{3}\begin{verbatim}
   (10)
   table
      "katie"= [monthsOld= 23,gender= "female"]
  ,
      "judith"= [monthsOld= 366,gender= "female"]
  ,
      "bob"= [monthsOld= 407,gender= "male"]
Type: AssociationList(String,Record(monthsOld: Integer,gender: String))
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{AssociationListXmpPageEmpty10}
```

```
\begin{paste}
{AssociationListXmpPageEmpty10}{AssociationListXmpPagePatch10}
\pastebutton{AssociationListXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{delete!(al,1)\free{al6 }\bound{al7 }}
\end{paste}\end{patch}
```

———-

3.4 array1.ht

OneDimensionalArray

⇒ "Vector" (VectorXmpPage) 3.114 on page 1351
⇒ "FlexibleArray" (FlexibleArrayXmpPage) 3.39 on page 472

— **array1.ht** —

```
\begin{page}{OneDimensionalArrayXmpPage}{OneDimensionalArray}
\beginscroll
The \spadtype{OneDimensionalArray} domain is used for storing data in
a one-dimensional indexed data structure.  Such an array is a
homogeneous data structure in that all the entries of the array must
belong to the same Axiom domain.  Each array has a fixed length
specified by the user and arrays are not extensible.  The indexing of
one-dimensional arrays is one-based.  This means that the ``first''
element of an array is given the index \spad{1}.
See also \downlink{`Vector'}{VectorXmpPage}\ignore{Vector} and
\downlink{`FlexibleArray'}{FlexibleArrayXmpPage}\ignore{FlexibleArray}.
\xtc{
To create a one-dimensional array, apply the
operation \spadfun{oneDimensionalArray} to a list.
}{
\spadpaste{oneDimensionalArray [i**2 for i in 1..10]}
}
\xtc{
Another approach is to first create \spad{a}, a one-dimensional
array of 10 \spad{0}'s.
\spadtype{OneDimensionalArray} has the convenient
abbreviation \spadtype{ARRAY1}.
}{
\spadpaste{a : ARRAY1 INT := new(10,0)\bound{a}}
}
\xtc{
Set each \spad{i}th element to i, then display the result.
}{
```

```
\spadpaste{for i in 1..10 repeat a.i := i; a\bound{a1}\free{a}}
}
\xtc{
Square each element by mapping the function
\texht{$i \mapsto i^2$}{i +-> i**2} onto each element.
}{
\spadpaste{map!(i +-> i ** 2,a); a\bound{a3}\free{a2}}
}
\xtc{
Reverse the elements in place.
}{
\spadpaste{reverse! a\bound{a4}\free{a3}}
}
\xtc{
Swap the \spad{4}th and \spad{5}th element.
}{
\spadpaste{swap!(a,4,5); a\bound{a5}\free{a4}}
}
\xtc{
Sort the elements in place.
}{
\spadpaste{sort! a \bound{a6}\free{a5}}
}
\xtc{
Create a new one-dimensional array \spad{b} containing the
last 5 elements of \spad{a}.
}{
\spadpaste{b := a(6..10)\bound{b}\free{a6}}
}
\xtc{
Replace the first 5 elements of \spad{a} with those of \spad{b}.
}{
\spadpaste{copyInto!(a,b,1)\free{b}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{OneDimensionalArrayXmpPagePatch1}
\begin{paste}{OneDimensionalArrayXmpPageFull1}
{OneDimensionalArrayXmpPageEmpty1}
\pastebutton{OneDimensionalArrayXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{oneDimensionalArray [i**2 for i in 1..10]}
\indentrel{3}\begin{verbatim}
   (1)  [1,4,9,16,25,36,49,64,81,100]
              Type: OneDimensionalArray PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{OneDimensionalArrayXmpPageEmpty1}
\begin{paste}{OneDimensionalArrayXmpPageEmpty1}
{OneDimensionalArrayXmpPagePatch1}
\pastebutton{OneDimensionalArrayXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{oneDimensionalArray [i**2 for i in 1..10]}
\end{paste}\end{patch}

\begin{patch}{OneDimensionalArrayXmpPagePatch2}
\begin{paste}{OneDimensionalArrayXmpPageFull2}
{OneDimensionalArrayXmpPageEmpty2}
\pastebutton{OneDimensionalArrayXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{a : ARRAY1 INT := new(10,0)\bound{a }}
\indentrel{3}\begin{verbatim}
   (2)  [0,0,0,0,0,0,0,0,0,0]
                      Type: OneDimensionalArray Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OneDimensionalArrayXmpPageEmpty2}
\begin{paste}{OneDimensionalArrayXmpPageEmpty2}
{OneDimensionalArrayXmpPagePatch2}
\pastebutton{OneDimensionalArrayXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{a : ARRAY1 INT := new(10,0)\bound{a }}
\end{paste}\end{patch}

\begin{patch}{OneDimensionalArrayXmpPagePatch3}
\begin{paste}{OneDimensionalArrayXmpPageFull3}
{OneDimensionalArrayXmpPageEmpty3}
\pastebutton{OneDimensionalArrayXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{
for i in 1..10 repeat a.i := i; a\bound{a1 }\free{a }}
\indentrel{3}\begin{verbatim}
   (3)  [1,2,3,4,5,6,7,8,9,10]
                      Type: OneDimensionalArray Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OneDimensionalArrayXmpPageEmpty3}
\begin{paste}{OneDimensionalArrayXmpPageEmpty3}
{OneDimensionalArrayXmpPagePatch3}
\pastebutton{OneDimensionalArrayXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{
for i in 1..10 repeat a.i := i; a\bound{a1 }\free{a }}
\end{paste}\end{patch}

\begin{patch}{OneDimensionalArrayXmpPagePatch4}
\begin{paste}{OneDimensionalArrayXmpPageFull4}
{OneDimensionalArrayXmpPageEmpty4}
\pastebutton{OneDimensionalArrayXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{map!(i +-> i ** 2,a); a\bound{a3 }\free{a2 }}
```

```
\indentrel{3}\begin{verbatim}
   (4)  [1,4,9,16,25,36,49,64,81,100]
                     Type: OneDimensionalArray Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OneDimensionalArrayXmpPageEmpty4}
\begin{paste}{OneDimensionalArrayXmpPageEmpty4}
{OneDimensionalArrayXmpPagePatch4}
\pastebutton{OneDimensionalArrayXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{map!(i +-> i ** 2,a); a\bound{a3 }\free{a2 }}
\end{paste}\end{patch}

\begin{patch}{OneDimensionalArrayXmpPagePatch5}
\begin{paste}{OneDimensionalArrayXmpPageFull5}
{OneDimensionalArrayXmpPageEmpty5}
\pastebutton{OneDimensionalArrayXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{reverse! a\bound{a4 }\free{a3 }}
\indentrel{3}\begin{verbatim}
   (5)  [100,81,64,49,36,25,16,9,4,1]
                     Type: OneDimensionalArray Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OneDimensionalArrayXmpPageEmpty5}
\begin{paste}{OneDimensionalArrayXmpPageEmpty5}
{OneDimensionalArrayXmpPagePatch5}
\pastebutton{OneDimensionalArrayXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{reverse! a\bound{a4 }\free{a3 }}
\end{paste}\end{patch}

\begin{patch}{OneDimensionalArrayXmpPagePatch6}
\begin{paste}{OneDimensionalArrayXmpPageFull6}
{OneDimensionalArrayXmpPageEmpty6}
\pastebutton{OneDimensionalArrayXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{swap!(a,4,5); a\bound{a5 }\free{a4 }}
\indentrel{3}\begin{verbatim}
   (6)  [100,81,64,36,49,25,16,9,4,1]
                     Type: OneDimensionalArray Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OneDimensionalArrayXmpPageEmpty6}
\begin{paste}{OneDimensionalArrayXmpPageEmpty6}
{OneDimensionalArrayXmpPagePatch6}
\pastebutton{OneDimensionalArrayXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{swap!(a,4,5); a\bound{a5 }\free{a4 }}
\end{paste}\end{patch}

\begin{patch}{OneDimensionalArrayXmpPagePatch7}
```

```
\begin{paste}{OneDimensionalArrayXmpPageFull7}
{OneDimensionalArrayXmpPageEmpty7}
\pastebutton{OneDimensionalArrayXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{sort! a\bound{a6 }\free{a5 }}
\indentrel{3}\begin{verbatim}
   (7)  [1,4,9,16,25,36,49,64,81,100]
                      Type: OneDimensionalArray Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OneDimensionalArrayXmpPageEmpty7}
\begin{paste}{OneDimensionalArrayXmpPageEmpty7}
{OneDimensionalArrayXmpPagePatch7}
\pastebutton{OneDimensionalArrayXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{sort! a\bound{a6 }\free{a5 }}
\end{paste}\end{patch}

\begin{patch}{OneDimensionalArrayXmpPagePatch8}
\begin{paste}{OneDimensionalArrayXmpPageFull8}
{OneDimensionalArrayXmpPageEmpty8}
\pastebutton{OneDimensionalArrayXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{b := a(6..10)\bound{b }\free{a6 }}
\indentrel{3}\begin{verbatim}
   (8)  [36,49,64,81,100]
                      Type: OneDimensionalArray Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OneDimensionalArrayXmpPageEmpty8}
\begin{paste}{OneDimensionalArrayXmpPageEmpty8}
{OneDimensionalArrayXmpPagePatch8}
\pastebutton{OneDimensionalArrayXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{b := a(6..10)\bound{b }\free{a6 }}
\end{paste}\end{patch}

\begin{patch}{OneDimensionalArrayXmpPagePatch9}
\begin{paste}{OneDimensionalArrayXmpPageFull9}
{OneDimensionalArrayXmpPageEmpty9}
\pastebutton{OneDimensionalArrayXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{copyInto!(a,b,1)\free{b }}
\indentrel{3}\begin{verbatim}
   (9)  [36,49,64,81,100,36,49,64,81,100]
                      Type: OneDimensionalArray Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OneDimensionalArrayXmpPageEmpty9}
\begin{paste}{OneDimensionalArrayXmpPageEmpty9}
{OneDimensionalArrayXmpPagePatch9}
\pastebutton{OneDimensionalArrayXmpPageEmpty9}{\showpaste}
```

```
\tab{5}\spadcommand{copyInto!(a,b,1)\free{b }}
\end{paste}\end{patch}
```

———-

3.5 array2.ht

TwoDimensionalArray

⇒ “The Any Domain” (ugTypesAnyNonePage) 7 on page 1668
⇒ “Matrix” (MatrixXmpPage) 3.75 on page 984
⇒ “OneDimensionalArray” (OneDimensionalArrayXmpPage) 3.4 on page 138

— **array2.ht** —

```
\begin{page}{TwoDimensionalArrayXmpPage}{TwoDimensionalArray}
\beginscroll

The \spadtype{TwoDimensionalArray} domain is used for storing data in
a \twodim{} data structure indexed by row and by column.  Such an
array is a homogeneous data structure in that all the entries of the
array must belong to the same Axiom domain (although see
\downlink{``The Any Domain''}{ugTypesAnyNonePage} in Section 2.6
\ignore{ugTypesAnyNone}).  Each array has a fixed
number of rows and columns specified by the user and arrays are not
extensible.  In Axiom, the indexing of two-dimensional arrays is
one-based.  This means that both the ``first'' row of an array and the
``first'' column of an array are given the index \spad{1}.  Thus, the
entry in the upper left corner of an array is in position
\spad{(1,1)}.


The operation \spadfunFrom{new}{TwoDimensionalArray} creates an array
with a specified number of rows and columns and fills the components
of that array with a specified entry.  The arguments of this operation
specify the number of rows, the number of columns, and the entry.
\xtc{
This creates a five-by-four array of integers, all of whose entries are
zero.
}{
\spadpaste{arr : ARRAY2 INT := new(5,4,0) \bound{arr}}
}
The entries of this array can be set to other integers using
the operation \spadfunFrom{setelt}{TwoDimensionalArray}.

\xtc{
```

```
Issue this to set the element in the upper left corner of this array to
\spad{17}.
}{
\spadpaste{setelt(arr,1,1,17) \free{arr}\bound{arr1}}
}
\xtc{
Now the first element of the array is \spad{17.}
}{
\spadpaste{arr \free{arr1}}
}
\xtc{
Likewise, elements of an array are extracted using the operation
\spadfunFrom{elt}{TwoDimensionalArray}.
}{
\spadpaste{elt(arr,1,1) \free{arr1}}
}
\xtc{
Another way to use these two operations is as follows.
This sets the element in position \spad{(3,2)} of the array to \spad{15}.
}{
\spadpaste{arr(3,2) := 15 \free{arr1}\bound{arr2}}
}
\xtc{
This extracts the element in position \spad{(3,2)} of the array.
}{
\spadpaste{arr(3,2) \free{arr2}}
}
The operations \spadfunFrom{elt}{TwoDimensionalArray} and
\spadfunFrom{setelt}{TwoDimensionalArray} come equipped with an error
check which verifies that the indices are in the proper ranges.
For example, the above array has five rows and four columns, so if you ask
for the entry in position \spad{(6,2)} with \spad{arr(6,2)} Axiom
displays an error message.
If there is no need for an error check, you can call the operations
\spadfunFrom{qelt}{TwoDimensionalArray} and
\spadfunFromX{qsetelt}{TwoDimensionalArray} which provide the same
functionality but without the error check.
Typically, these operations are called in well-tested programs.

\xtc{
The operations \spadfunFrom{row}{TwoDimensionalArray} and
\spadfunFrom{column}{TwoDimensionalArray} extract rows and columns,
respectively, and return objects of \spadtype{OneDimensionalArray} with
the same underlying element type.
}{
\spadpaste{row(arr,1) \free{arr2}}
}
\xtc{
}{
\spadpaste{column(arr,1) \free{arr2}}
```

```
}

\xtc{
You can determine the dimensions of an array by calling the
operations \spadfunFrom{nrows}{TwoDimensionalArray} and
\spadfunFrom{ncols}{TwoDimensionalArray},
which return the number of rows and columns, respectively.
}{
\spadpaste{nrows(arr) \free{arr2}}
}
\xtc{
}{
\spadpaste{ncols(arr) \free{arr2}}
}
\xtc{
To apply an operation to every element of an array, use
\spadfunFrom{map}{TwoDimensionalArray}.
This creates a new array.
This expression negates every element.
}{
\spadpaste{map(-,arr) \free{arr2}}
}
\xtc{
This creates an array where all the elements are doubled.
}{
\spadpaste{map((x +-> x + x),arr) \free{arr2}}
}
\xtc{
To change the array destructively, use
\spadfunFromX{map}{TwoDimensionalArray} instead of
\spadfunFrom{map}{TwoDimensionalArray}.
If you need to make a copy of any array, use
\spadfunFrom{copy}{TwoDimensionalArray}.
}{
\spadpaste{arrc := copy(arr) \bound{arrc}\free{arr2}}
}
\xtc{
}{
\spadpaste{map!(-,arrc) \free{arrc}}
}
\xtc{
}{
\spadpaste{arrc \free{arrc}}
}
\xtc{
}{
\spadpaste{arr  \free{arr2}}
}

\xtc{
```

```
Use \spadfunFrom{member?}{TwoDimensionalArray} to see if a given element
is in an array.
}{
\spadpaste{member?(17,arr) \free{arr2}}
}
\xtc{
}{
\spadpaste{member?(10317,arr) \free{arr2}}
}
\xtc{
To see how many times an element appears in an array, use
\spadfunFrom{count}{TwoDimensionalArray}.
}{
\spadpaste{count(17,arr) \free{arr2}}
}
\xtc{
}{
\spadpaste{count(0,arr) \free{arr2}}
}

For more information about the operations available for
\spadtype{TwoDimensionalArray}, issue \spadcmd{)show
TwoDimensionalArray}.
For information on related topics, see
\downlink{`Matrix'}{MatrixXmpPage}\ignore{Matrix} and
\downlink{`OneDimensionalArray'}{OneDimensionalArrayXmpPage}
\ignore{OneDimensionalArray}.
\endscroll
\autobuttons
\end{page}

\begin{patch}{TwoDimensionalArrayXmpPagePatch1}
\begin{paste}{TwoDimensionalArrayXmpPageFull1}
{TwoDimensionalArrayXmpPageEmpty1}
\pastebutton{TwoDimensionalArrayXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{arr : ARRAY2 INT := new(5,4,0)\bound{arr }}
\indentrel{3}\begin{verbatim}
        0  0  0  0

        0  0  0  0

   (1)  0  0  0  0

        0  0  0  0

        0  0  0  0
                      Type: TwoDimensionalArray Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{TwoDimensionalArrayXmpPageEmpty1}
\begin{paste}{TwoDimensionalArrayXmpPageEmpty1}
{TwoDimensionalArrayXmpPagePatch1}
\pastebutton{TwoDimensionalArrayXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{arr : ARRAY2 INT := new(5,4,0)\bound{arr }}
\end{paste}\end{patch}

\begin{patch}{TwoDimensionalArrayXmpPagePatch2}
\begin{paste}{TwoDimensionalArrayXmpPageFull2}
{TwoDimensionalArrayXmpPageEmpty2}
\pastebutton{TwoDimensionalArrayXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{setelt(arr,1,1,17)\free{arr }\bound{arr1 }}
\indentrel{3}\begin{verbatim}
   (2)  17
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TwoDimensionalArrayXmpPageEmpty2}
\begin{paste}{TwoDimensionalArrayXmpPageEmpty2}
{TwoDimensionalArrayXmpPagePatch2}
\pastebutton{TwoDimensionalArrayXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{setelt(arr,1,1,17)\free{arr }\bound{arr1 }}
\end{paste}\end{patch}

\begin{patch}{TwoDimensionalArrayXmpPagePatch3}
\begin{paste}{TwoDimensionalArrayXmpPageFull3}
{TwoDimensionalArrayXmpPageEmpty3}
\pastebutton{TwoDimensionalArrayXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{arr\free{arr1 }}
\indentrel{3}\begin{verbatim}
        17  0  0  0

        0   0  0  0

   (3)  0   0  0  0

        0   0  0  0

        0   0  0  0
                      Type: TwoDimensionalArray Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TwoDimensionalArrayXmpPageEmpty3}
\begin{paste}{TwoDimensionalArrayXmpPageEmpty3}
{TwoDimensionalArrayXmpPagePatch3}
\pastebutton{TwoDimensionalArrayXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{arr\free{arr1 }}
\end{paste}\end{patch}
```

```
\begin{patch}{TwoDimensionalArrayXmpPagePatch4}
\begin{paste}{TwoDimensionalArrayXmpPageFull4}
{TwoDimensionalArrayXmpPageEmpty4}
\pastebutton{TwoDimensionalArrayXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{elt(arr,1,1)\free{arr1 }}
\indentrel{3}\begin{verbatim}
   (4)  17
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TwoDimensionalArrayXmpPageEmpty4}
\begin{paste}{TwoDimensionalArrayXmpPageEmpty4}
{TwoDimensionalArrayXmpPagePatch4}
\pastebutton{TwoDimensionalArrayXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{elt(arr,1,1)\free{arr1 }}
\end{paste}\end{patch}

\begin{patch}{TwoDimensionalArrayXmpPagePatch5}
\begin{paste}{TwoDimensionalArrayXmpPageFull5}
{TwoDimensionalArrayXmpPageEmpty5}
\pastebutton{TwoDimensionalArrayXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{arr(3,2) := 15\free{arr1 }\bound{arr2 }}
\indentrel{3}\begin{verbatim}
   (5)  15
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TwoDimensionalArrayXmpPageEmpty5}
\begin{paste}{TwoDimensionalArrayXmpPageEmpty5}
{TwoDimensionalArrayXmpPagePatch5}
\pastebutton{TwoDimensionalArrayXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{arr(3,2) := 15\free{arr1 }\bound{arr2 }}
\end{paste}\end{patch}

\begin{patch}{TwoDimensionalArrayXmpPagePatch6}
\begin{paste}{TwoDimensionalArrayXmpPageFull6}
{TwoDimensionalArrayXmpPageEmpty6}
\pastebutton{TwoDimensionalArrayXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{arr(3,2)\free{arr2 }}
\indentrel{3}\begin{verbatim}
   (6)  15
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TwoDimensionalArrayXmpPageEmpty6}
\begin{paste}{TwoDimensionalArrayXmpPageEmpty6}
```

```
{TwoDimensionalArrayXmpPagePatch6}
\pastebutton{TwoDimensionalArrayXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{arr(3,2)\free{arr2 }}
\end{paste}\end{patch}

\begin{patch}{TwoDimensionalArrayXmpPagePatch7}
\begin{paste}{TwoDimensionalArrayXmpPageFull7}
{TwoDimensionalArrayXmpPageEmpty7}
\pastebutton{TwoDimensionalArrayXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{row(arr,1)\free{arr2 }}
\indentrel{3}\begin{verbatim}
   (7)  [17,0,0,0]
                      Type: OneDimensionalArray Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TwoDimensionalArrayXmpPageEmpty7}
\begin{paste}{TwoDimensionalArrayXmpPageEmpty7}
{TwoDimensionalArrayXmpPagePatch7}
\pastebutton{TwoDimensionalArrayXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{row(arr,1)\free{arr2 }}
\end{paste}\end{patch}

\begin{patch}{TwoDimensionalArrayXmpPagePatch8}
\begin{paste}{TwoDimensionalArrayXmpPageFull8}
{TwoDimensionalArrayXmpPageEmpty8}
\pastebutton{TwoDimensionalArrayXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{column(arr,1)\free{arr2 }}
\indentrel{3}\begin{verbatim}
   (8)  [17,0,0,0,0]
                      Type: OneDimensionalArray Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TwoDimensionalArrayXmpPageEmpty8}
\begin{paste}{TwoDimensionalArrayXmpPageEmpty8}
{TwoDimensionalArrayXmpPagePatch8}
\pastebutton{TwoDimensionalArrayXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{column(arr,1)\free{arr2 }}
\end{paste}\end{patch}

\begin{patch}{TwoDimensionalArrayXmpPagePatch9}
\begin{paste}{TwoDimensionalArrayXmpPageFull9}
{TwoDimensionalArrayXmpPageEmpty9}
\pastebutton{TwoDimensionalArrayXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{nrows(arr)\free{arr2 }}
\indentrel{3}\begin{verbatim}
   (9)  5
                                  Type: PositiveInteger
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TwoDimensionalArrayXmpPageEmpty9}
\begin{paste}{TwoDimensionalArrayXmpPageEmpty9}
{TwoDimensionalArrayXmpPagePatch9}
\pastebutton{TwoDimensionalArrayXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{nrows(arr)\free{arr2 }}
\end{paste}\end{patch}

\begin{patch}{TwoDimensionalArrayXmpPagePatch10}
\begin{paste}{TwoDimensionalArrayXmpPageFull10}
{TwoDimensionalArrayXmpPageEmpty10}
\pastebutton{TwoDimensionalArrayXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{ncols(arr)\free{arr2 }}
\indentrel{3}\begin{verbatim}
   (10)  4
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TwoDimensionalArrayXmpPageEmpty10}
\begin{paste}{TwoDimensionalArrayXmpPageEmpty10}
{TwoDimensionalArrayXmpPagePatch10}
\pastebutton{TwoDimensionalArrayXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{ncols(arr)\free{arr2 }}
\end{paste}\end{patch}

\begin{patch}{TwoDimensionalArrayXmpPagePatch11}
\begin{paste}{TwoDimensionalArrayXmpPageFull11}
{TwoDimensionalArrayXmpPageEmpty11}
\pastebutton{TwoDimensionalArrayXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{map(-,arr)\free{arr2 }}
\indentrel{3}\begin{verbatim}
         - 17   0    0  0

          0     0    0  0

   (11)   0    - 15  0  0

          0     0    0  0

          0     0    0  0
                      Type: TwoDimensionalArray Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TwoDimensionalArrayXmpPageEmpty11}
\begin{paste}{TwoDimensionalArrayXmpPageEmpty11}
{TwoDimensionalArrayXmpPagePatch11}
\pastebutton{TwoDimensionalArrayXmpPageEmpty11}{\showpaste}
```

```
\tab{5}\spadcommand{map(-,arr)\free{arr2 }}
\end{paste}\end{patch}

\begin{patch}{TwoDimensionalArrayXmpPagePatch12}
\begin{paste}{TwoDimensionalArrayXmpPageFull12}
{TwoDimensionalArrayXmpPageEmpty12}
\pastebutton{TwoDimensionalArrayXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{map((x +-> x + x),arr)\free{arr2 }}
\indentrel{3}\begin{verbatim}
         34  0   0  0

         0   0   0  0

   (12)  0   30  0  0

         0   0   0  0

         0   0   0  0
                      Type: TwoDimensionalArray Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TwoDimensionalArrayXmpPageEmpty12}
\begin{paste}{TwoDimensionalArrayXmpPageEmpty12}
{TwoDimensionalArrayXmpPagePatch12}
\pastebutton{TwoDimensionalArrayXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{map((x +-> x + x),arr)\free{arr2 }}
\end{paste}\end{patch}

\begin{patch}{TwoDimensionalArrayXmpPagePatch13}
\begin{paste}{TwoDimensionalArrayXmpPageFull13}
{TwoDimensionalArrayXmpPageEmpty13}
\pastebutton{TwoDimensionalArrayXmpPageFull13}{\hidepaste}
\tab{5}\spadcommand{arrc := copy(arr)\bound{arrc }\free{arr2 }}
\indentrel{3}\begin{verbatim}
         17  0   0  0

         0   0   0  0

   (13)  0   15  0  0

         0   0   0  0

         0   0   0  0
                      Type: TwoDimensionalArray Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TwoDimensionalArrayXmpPageEmpty13}
\begin{paste}{TwoDimensionalArrayXmpPageEmpty13}
```

```
{TwoDimensionalArrayXmpPagePatch13}
\pastebutton{TwoDimensionalArrayXmpPageEmpty13}{\showpaste}
\tab{5}\spadcommand{arrc := copy(arr)\bound{arrc }\free{arr2 }}
\end{paste}\end{patch}

\begin{patch}{TwoDimensionalArrayXmpPagePatch14}
\begin{paste}{TwoDimensionalArrayXmpPageFull14}
{TwoDimensionalArrayXmpPageEmpty14}
\pastebutton{TwoDimensionalArrayXmpPageFull14}{\hidepaste}
\tab{5}\spadcommand{map!(-,arrc)\free{arrc }}
\indentrel{3}\begin{verbatim}
         - 17   0    0  0

          0     0    0  0

   (14)   0    - 15  0  0

          0     0    0  0

          0     0    0  0
                      Type: TwoDimensionalArray Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TwoDimensionalArrayXmpPageEmpty14}
\begin{paste}{TwoDimensionalArrayXmpPageEmpty14}
{TwoDimensionalArrayXmpPagePatch14}
\pastebutton{TwoDimensionalArrayXmpPageEmpty14}{\showpaste}
\tab{5}\spadcommand{map!(-,arrc)\free{arrc }}
\end{paste}\end{patch}

\begin{patch}{TwoDimensionalArrayXmpPagePatch15}
\begin{paste}{TwoDimensionalArrayXmpPageFull15}
{TwoDimensionalArrayXmpPageEmpty15}
\pastebutton{TwoDimensionalArrayXmpPageFull15}{\hidepaste}
\tab{5}\spadcommand{arrc\free{arrc }}
\indentrel{3}\begin{verbatim}
         - 17   0    0  0

          0     0    0  0

   (15)   0    - 15  0  0

          0     0    0  0

          0     0    0  0
                      Type: TwoDimensionalArray Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

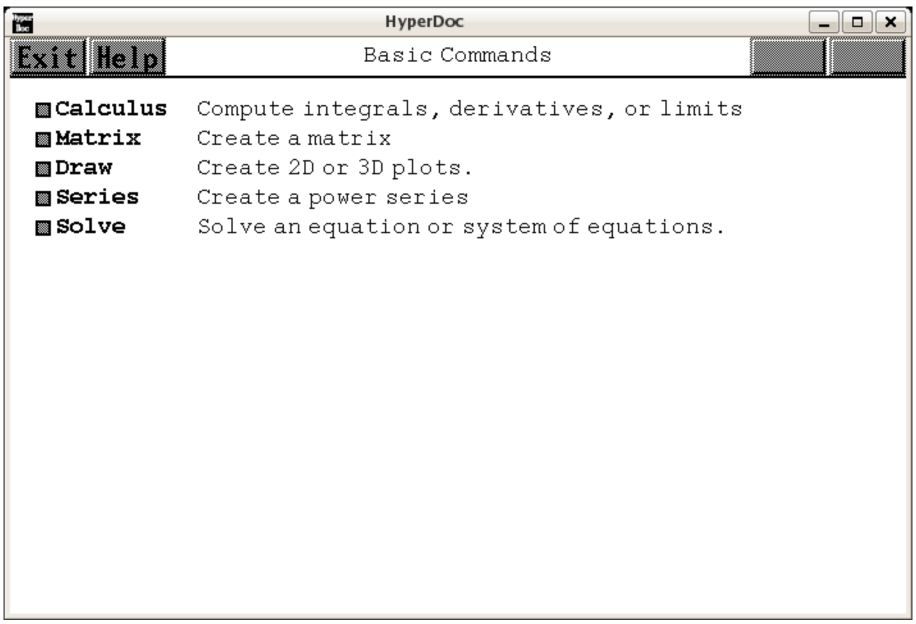
```
\begin{patch}{TwoDimensionalArrayXmpPageEmpty15}
\begin{paste}{TwoDimensionalArrayXmpPageEmpty15}
{TwoDimensionalArrayXmpPagePatch15}
\pastebutton{TwoDimensionalArrayXmpPageEmpty15}{\showpaste}
\tab{5}\spadcommand{arrc\free{arrc }}
\end{paste}\end{patch}

\begin{patch}{TwoDimensionalArrayXmpPagePatch16}
\begin{paste}{TwoDimensionalArrayXmpPageFull16}
{TwoDimensionalArrayXmpPageEmpty16}
\pastebutton{TwoDimensionalArrayXmpPageFull16}{\hidepaste}
\tab{5}\spadcommand{arr\free{arr2 }}
\indentrel{3}\begin{verbatim}
         17  0   0  0

         0   0   0  0

   (16)  0   15  0  0

         0   0   0  0

         0   0   0  0
                      Type: TwoDimensionalArray Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TwoDimensionalArrayXmpPageEmpty16}
\begin{paste}{TwoDimensionalArrayXmpPageEmpty16}
{TwoDimensionalArrayXmpPagePatch16}
\pastebutton{TwoDimensionalArrayXmpPageEmpty16}{\showpaste}
\tab{5}\spadcommand{arr\free{arr2 }}
\end{paste}\end{patch}

\begin{patch}{TwoDimensionalArrayXmpPagePatch17}
\begin{paste}{TwoDimensionalArrayXmpPageFull17}
{TwoDimensionalArrayXmpPageEmpty17}
\pastebutton{TwoDimensionalArrayXmpPageFull17}{\hidepaste}
\tab{5}\spadcommand{member?(17,arr)\free{arr2 }}
\indentrel{3}\begin{verbatim}
   (17)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TwoDimensionalArrayXmpPageEmpty17}
\begin{paste}{TwoDimensionalArrayXmpPageEmpty17}
{TwoDimensionalArrayXmpPagePatch17}
\pastebutton{TwoDimensionalArrayXmpPageEmpty17}{\showpaste}
\tab{5}\spadcommand{member?(17,arr)\free{arr2 }}
\end{paste}\end{patch}
```

```
\begin{patch}{TwoDimensionalArrayXmpPagePatch18}
\begin{paste}{TwoDimensionalArrayXmpPageFull18}
{TwoDimensionalArrayXmpPageEmpty18}
\pastebutton{TwoDimensionalArrayXmpPageFull18}{\hidepaste}
\tab{5}\spadcommand{member?(10317,arr)\free{arr2 }}
\indentrel{3}\begin{verbatim}
   (18)  false
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TwoDimensionalArrayXmpPageEmpty18}
\begin{paste}{TwoDimensionalArrayXmpPageEmpty18}
{TwoDimensionalArrayXmpPagePatch18}
\pastebutton{TwoDimensionalArrayXmpPageEmpty18}{\showpaste}
\tab{5}\spadcommand{member?(10317,arr)\free{arr2 }}
\end{paste}\end{patch}

\begin{patch}{TwoDimensionalArrayXmpPagePatch19}
\begin{paste}{TwoDimensionalArrayXmpPageFull19}
{TwoDimensionalArrayXmpPageEmpty19}
\pastebutton{TwoDimensionalArrayXmpPageFull19}{\hidepaste}
\tab{5}\spadcommand{count(17,arr)\free{arr2 }}
\indentrel{3}\begin{verbatim}
   (19)  1
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TwoDimensionalArrayXmpPageEmpty19}
\begin{paste}{TwoDimensionalArrayXmpPageEmpty19}
{TwoDimensionalArrayXmpPagePatch19}
\pastebutton{TwoDimensionalArrayXmpPageEmpty19}{\showpaste}
\tab{5}\spadcommand{count(17,arr)\free{arr2 }}
\end{paste}\end{patch}

\begin{patch}{TwoDimensionalArrayXmpPagePatch20}
\begin{paste}{TwoDimensionalArrayXmpPageFull20}
{TwoDimensionalArrayXmpPageEmpty20}
\pastebutton{TwoDimensionalArrayXmpPageFull20}{\hidepaste}
\tab{5}\spadcommand{count(0,arr)\free{arr2 }}
\indentrel{3}\begin{verbatim}
   (20)  18
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TwoDimensionalArrayXmpPageEmpty20}
\begin{paste}{TwoDimensionalArrayXmpPageEmpty20}
```

```
{TwoDimensionalArrayXmpPagePatch20}
\pastebutton{TwoDimensionalArrayXmpPageEmpty20}{\showpaste}
\tab{5}\spadcommand{count(0,arr)\free{arr2 }}
\end{paste}\end{patch}
```

3.6 basic.ht

Basic Commands



⇐ “Root Page” (RootPage) 3.1 on page 117
⇒ “Calculus” (Calculus) 3.6 on page 156
⇒ “Matrix” (LispFunctions) 3.71 on page 952
⇒ “Draw” (LispFunctions) 3.71 on page 952
⇒ “Series” (LispFunctions) 3.71 on page 952
⇒ “Solve” (LispFunctions) 3.71 on page 952

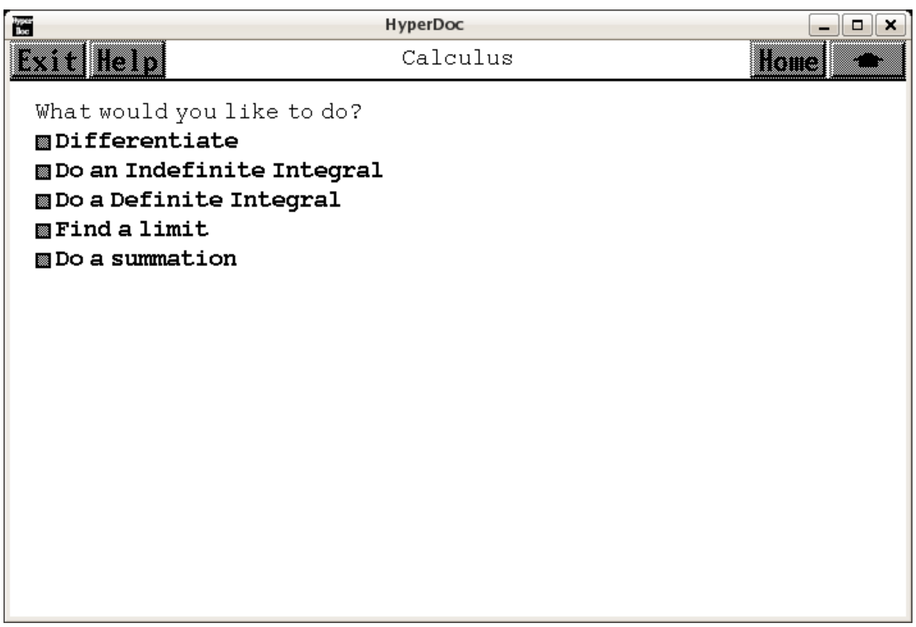
— **basic.ht** —

```
\begin{page}{BasicCommand}{Basic Commands}
\beginscroll
\beginmenu
\menumemolink{Calculus}{Calculus}\tab{10}
```

```
  Compute integrals, derivatives, or limits
\menulispmemolink{Matrix}{(|bcMatrix|)}\tab{10}
  Create a matrix
%\menulispmemolink{Operations}{(|bcExpand|)}
%  Expand, factor, simplify, substitute, etc.
\menulispmemolink{Draw}{(|bcDraw|)}\tab{10}
  Create 2D or 3D plots.
\menulispmemolink{Series}{(|bcSeries|)}\tab{10}
  Create a power series
\menulispmemolink{Solve}{(|bcSolve|)}\tab{10}
  Solve an equation or system of equations.
\endmenu
\endscroll
\autobuttons
\end{page}
```

———

Calculus



⇐ “Basic Commands” (BasicCommand) 3.6 on page 155
⇒ “Differentiate” (LispFunctions) 3.71 on page 952
⇒ “Do an Indefinite Integral” (LispFunctions) 3.71 on page 952
⇒ “Do a Definite Integral” (LispFunctions) 3.71 on page 952
⇒ “Find a Limit” (LispFunctions) 3.71 on page 952

⇒ "Do a summation" (LispFunctions) 3.71 on page 952

— basic.ht —

```
\begin{page}{Calculus}{Calculus}
\beginscroll
What would you like to do?
\beginmenu
\menulispdownlink{Differentiate}{(|bcDifferentiate|)}\space{}
\menulispdownlink{Do an Indefinite Integral}
{(|bcIndefiniteIntegrate|)}\space{}
\menulispdownlink{Do a Definite Integral}{(|bcDefiniteIntegrate|)}\space{}
\menulispdownlink{Find a limit}{(|bcLimit|)}\space{}
\menulispdownlink{Do a summation}{(|bcSum|)}\space{}
%\menulispdownlink{Compute a product}{(|bcProduct|)}\space{}
\endmenu
\endscroll
\autobuttons
\end{page}
```

3.7 bbtree.ht

BalancedBinaryTree

— bbtree.ht —

```
\begin{page}{BalancedBinaryTreeXmpPage}{BalancedBinaryTree}
\beginscroll
\spadtype{BalancedBinaryTrees(S)} is the domain
of balanced binary trees with elements of type \spad{S} at the nodes.
A binary tree is either \spadfun{empty} or else
consists of a \spadfun{node} having a \spadfun{value}
and two branches, each branch a binary tree.
A balanced binary tree is one that is balanced with respect its leaves.
One with \texht{$2^k$}{2**k} leaves is
perfectly ``balanced'': the tree has minimum depth, and
the \spadfun{left} and \spadfun{right}
branch of every interior node is identical in shape.

Balanced binary trees are useful in algebraic computation for
so-called ``divide-and-conquer'' algorithms.  Conceptually, the data
for a problem is initially placed at the root of the tree.  The
original data is then split into two subproblems, one for each
subtree.  And so on.  Eventually, the problem is solved at the leaves
```

```
of the tree.  A solution to the original problem is obtained by some
mechanism that can reassemble the pieces.  In fact, an implementation
of the Chinese Remainder Algorithm using balanced binary trees was
first proposed by David Y. Y.  Yun at the IBM T. J.  Watson Research
Center in Yorktown Heights, New York, in 1978.  It served as the
prototype for polymorphic algorithms in Axiom.

In what follows, rather than perform a series of computations with
a single expression, the expression is reduced modulo a number of
integer primes, a computation is done with modular arithmetic for
each prime, and the Chinese Remainder Algorithm is used to obtain
the answer to the original problem.
We illustrate this principle with the computation of
\texht{$12^2 = 144$}{12 ** 2 = 144}.

\xtc{
A list of moduli.
}{
\spadpaste{lm := [3,5,7,11]\bound{lm}}
}
\xtc{
The expression \spad{modTree(n, lm)}
creates a balanced binary tree with leaf values
\spad{n mod m} for each modulus \spad{m} in \spad{lm}.
}{
\spadpaste{modTree(12,lm)\free{lm}}
}
\xtc{
Operation \spad{modTree} does this using
operations on balanced binary trees.
We trace its steps.
Create a balanced binary tree \spad{t} of zeros with four leaves.
}{
\spadpaste{t := balancedBinaryTree(\#lm, 0)\bound{t}\free{lm}}
}
\xtc{
The leaves of the tree are set to the individual moduli.
}{
\spadpaste{setleaves!(t,lm)\bound{t1}\free{t}}
}
\xtc{
Use \spadfunX{mapUp} to do
a bottom-up traversal of \spad{t}, setting each interior node
to the product of the values at the nodes of its children.
}{
\spadpaste{mapUp!(t,_*)\bound{t2}\free{t1}}
}
\xtc{
The value at the node of every subtree is the product of the moduli
of the leaves of the subtree.
```

```
}{
\spadpaste{t \bound{t3}\free{t2}}
}
\xtc{
Operation \spadfunX{mapDown}\spad{(t,a,fn)} replaces the value
\spad{v} at each node of \spad{t} by \spad{fn(a,v)}.
}{
\spadpaste{mapDown!(t,12,_rem)\bound{t4}\free{t3}}
}
\xtc{
The operation \spadfun{leaves} returns the leaves of the resulting
tree.
In this case, it returns the list of \spad{12 mod m} for each
modulus \spad{m}.
}{
\spadpaste{leaves \%\bound{t5}\free{t4}}
}
\xtc{
Compute the square of the images of \spad{12} modulo each \spad{m}.
}{
\spadpaste{squares := [x**2 rem m for x in \% for m in lm]\bound{t6}\free{t5}}
}
\xtc{
Call the Chinese Remainder Algorithm to get the
answer for \texht{$12^2$}{12**2}.
}{
\spadpaste{chineseRemainder(\%,lm)\free{t6}}
}
\endscroll
\autobuttons
\end{page}

\begin{patch}{BalancedBinaryTreeXmpPagePatch1}
\begin{paste}{BalancedBinaryTreeXmpPageFull1}
{BalancedBinaryTreeXmpPageEmpty1}
\pastebutton{BalancedBinaryTreeXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{lm := [3,5,7,11]\bound{lm }}
\indentrel{3}\begin{verbatim}
   (1)  [3,5,7,11]
                             Type: List PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BalancedBinaryTreeXmpPageEmpty1}
\begin{paste}{BalancedBinaryTreeXmpPageEmpty1}
{BalancedBinaryTreeXmpPagePatch1}
\pastebutton{BalancedBinaryTreeXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{lm := [3,5,7,11]\bound{lm }}
\end{paste}\end{patch}
```

```
\begin{patch}{BalancedBinaryTreeXmpPagePatch2}
\begin{paste}{BalancedBinaryTreeXmpPageFull2}
{BalancedBinaryTreeXmpPageEmpty2}
\pastebutton{BalancedBinaryTreeXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{modTree(12,lm)\free{lm }}
\indentrel{3}\begin{verbatim}
   (2)  [0,2,5,1]
                                    Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BalancedBinaryTreeXmpPageEmpty2}
\begin{paste}{BalancedBinaryTreeXmpPageEmpty2}
{BalancedBinaryTreeXmpPagePatch2}
\pastebutton{BalancedBinaryTreeXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{modTree(12,lm)\free{lm }}
\end{paste}\end{patch}

\begin{patch}{BalancedBinaryTreeXmpPagePatch3}
\begin{paste}{BalancedBinaryTreeXmpPageFull3}
{BalancedBinaryTreeXmpPageEmpty3}
\pastebutton{BalancedBinaryTreeXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{
t := balancedBinaryTree(\#lm, 0)\bound{t }\free{lm }}
\indentrel{3}\begin{verbatim}
   (3)  [[0,0,0],0,[0,0,0]]
            Type: BalancedBinaryTree NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BalancedBinaryTreeXmpPageEmpty3}
\begin{paste}{BalancedBinaryTreeXmpPageEmpty3}
{BalancedBinaryTreeXmpPagePatch3}
\pastebutton{BalancedBinaryTreeXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{t := balancedBinaryTree(\#lm, 0)\bound{t }\free{lm }}
\end{paste}\end{patch}

\begin{patch}{BalancedBinaryTreeXmpPagePatch4}
\begin{paste}{BalancedBinaryTreeXmpPageFull4}
{BalancedBinaryTreeXmpPageEmpty4}
\pastebutton{BalancedBinaryTreeXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{setleaves!(t,lm)\bound{t1 }\free{t }}
\indentrel{3}\begin{verbatim}
   (4)  [[3,0,5],0,[7,0,11]]
            Type: BalancedBinaryTree NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BalancedBinaryTreeXmpPageEmpty4}
\begin{paste}{BalancedBinaryTreeXmpPageEmpty4}
```

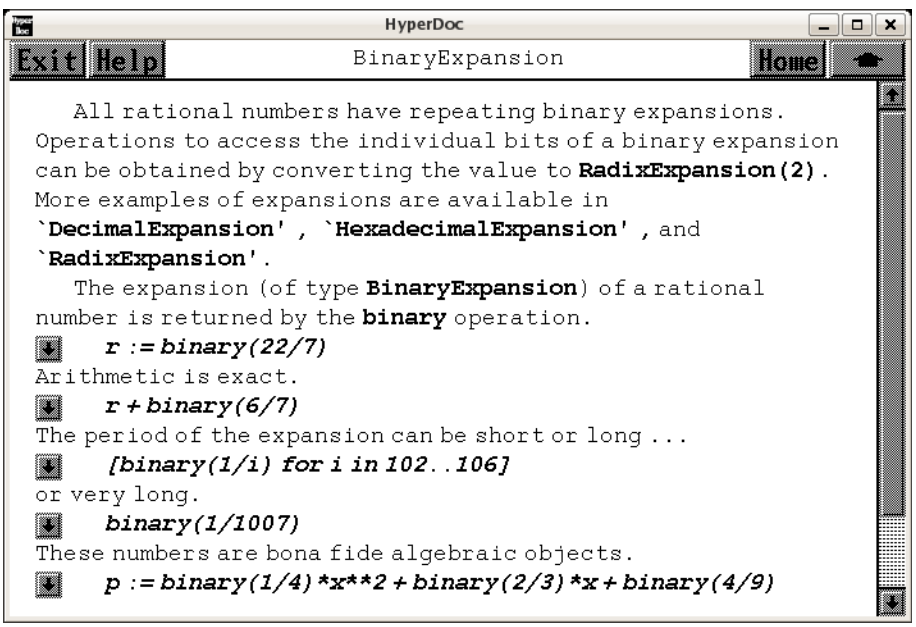
```
{BalancedBinaryTreeXmpPagePatch4}
\pastebutton{BalancedBinaryTreeXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{setleaves!(t,lm)\bound{t1 }\free{t }}
\end{paste}\end{patch}

\begin{patch}{BalancedBinaryTreeXmpPagePatch5}
\begin{paste}{BalancedBinaryTreeXmpPageFull5}
{BalancedBinaryTreeXmpPageEmpty5}
\pastebutton{BalancedBinaryTreeXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{mapUp!(t,_*)\bound{t2 }\free{t1 }}
\indentrel{3}\begin{verbatim}
   (5)  1155
                                 Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BalancedBinaryTreeXmpPageEmpty5}
\begin{paste}{BalancedBinaryTreeXmpPageEmpty5}
{BalancedBinaryTreeXmpPagePatch5}
\pastebutton{BalancedBinaryTreeXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{mapUp!(t,_*)\bound{t2 }\free{t1 }}
\end{paste}\end{patch}

\begin{patch}{BalancedBinaryTreeXmpPagePatch6}
\begin{paste}{BalancedBinaryTreeXmpPageFull6}
{BalancedBinaryTreeXmpPageEmpty6}
\pastebutton{BalancedBinaryTreeXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{t\bound{t3 }\free{t2 }}
\indentrel{3}\begin{verbatim}
   (6)  [[3,15,5],1155,[7,77,11]]
            Type: BalancedBinaryTree NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BalancedBinaryTreeXmpPageEmpty6}
\begin{paste}{BalancedBinaryTreeXmpPageEmpty6}
{BalancedBinaryTreeXmpPagePatch6}
\pastebutton{BalancedBinaryTreeXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{t\bound{t3 }\free{t2 }}
\end{paste}\end{patch}

\begin{patch}{BalancedBinaryTreeXmpPagePatch7}
\begin{paste}{BalancedBinaryTreeXmpPageFull7}
{BalancedBinaryTreeXmpPageEmpty7}
\pastebutton{BalancedBinaryTreeXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{mapDown!(t,12,_rem)\bound{t4 }\free{t3 }}
\indentrel{3}\begin{verbatim}
   (7)  [[0,12,2],12,[5,12,1]]
            Type: BalancedBinaryTree NonNegativeInteger
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BalancedBinaryTreeXmpPageEmpty7}
\begin{paste}{BalancedBinaryTreeXmpPageEmpty7}
{BalancedBinaryTreeXmpPagePatch7}
\pastebutton{BalancedBinaryTreeXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{mapDown!(t,12,_rem)\bound{t4 }\free{t3 }}
\end{paste}\end{patch}

\begin{patch}{BalancedBinaryTreeXmpPagePatch8}
\begin{paste}{BalancedBinaryTreeXmpPageFull8}
{BalancedBinaryTreeXmpPageEmpty8}
\pastebutton{BalancedBinaryTreeXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{leaves \%\bound{t5 }\free{t4 }}
\indentrel{3}\begin{verbatim}
   (8)  [0,2,5,1]
                          Type: List NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BalancedBinaryTreeXmpPageEmpty8}
\begin{paste}{BalancedBinaryTreeXmpPageEmpty8}
{BalancedBinaryTreeXmpPagePatch8}
\pastebutton{BalancedBinaryTreeXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{leaves \%\bound{t5 }\free{t4 }}
\end{paste}\end{patch}

\begin{patch}{BalancedBinaryTreeXmpPagePatch9}
\begin{paste}{BalancedBinaryTreeXmpPageFull9}
{BalancedBinaryTreeXmpPageEmpty9}
\pastebutton{BalancedBinaryTreeXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{
squares := [x**2 rem m for x in \% for m in lm]\bound{t6 }\free{t5 }}
\indentrel{3}\begin{verbatim}
   (9)  [0,4,4,1]
                          Type: List NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BalancedBinaryTreeXmpPageEmpty9}
\begin{paste}{BalancedBinaryTreeXmpPageEmpty9}
{BalancedBinaryTreeXmpPagePatch9}
\pastebutton{BalancedBinaryTreeXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{
squares := [x**2 rem m for x in \% for m in lm]\bound{t6 }\free{t5 }}
\end{paste}\end{patch}

\begin{patch}{BalancedBinaryTreeXmpPagePatch10}
\begin{paste}{BalancedBinaryTreeXmpPageFull10}
{BalancedBinaryTreeXmpPageEmpty10}
```

```
\pastebutton{BalancedBinaryTreeXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{chineseRemainder(\%,lm)\free{t6 }}
\indentrel{3}\begin{verbatim}
   (10)  144
                                    Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BalancedBinaryTreeXmpPageEmpty10}
\begin{paste}{BalancedBinaryTreeXmpPageEmpty10}
{BalancedBinaryTreeXmpPagePatch10}
\pastebutton{BalancedBinaryTreeXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{chineseRemainder(\%,lm)\free{t6 }}
\end{paste}\end{patch}
```

3.8 binary.ht

BinaryExpansion



⇐ "Axiom Number Types" (NumberPage) 3.80 on page 1021
⇐ "Integer" (IntegerXmpPage) 3.55 on page 696
⇒ "DecimalExpansion" (DecimalExpansionXmpPage) 3.21 on page 348

⇒ “RadixExpansion” (RadixExpansionXmpPage) 3.90 on page 1140
⇒ “HexadecimalExpansion” (HexExpansionXmpPage) 3.54 on page 692

— binary.ht —

```
\begin{page}{BinaryExpansionXmpPage}{BinaryExpansion}
\beginscroll
All rational numbers have repeating binary expansions.
Operations to access the individual bits of a binary expansion can
be obtained by converting the value to \spadtype{RadixExpansion(2)}.
More examples of expansions are available in
\downlink{‘DecimalExpansion’}{DecimalExpansionXmpPage}
\ignore{DecimalExpansion},
\downlink{‘HexadecimalExpansion’}{HexExpansionXmpPage}
\ignore{HexadecimalExpansion}, and
\downlink{‘RadixExpansion’}{RadixExpansionXmpPage}\ignore{RadixExpansion}.

\xtc{
The expansion (of type \spadtype{BinaryExpansion}) of a rational number
is returned by the \spadfunFrom{binary}{BinaryExpansion} operation.
}{
\spadpaste{r := binary(22/7) \bound{r}}
}
\xtc{
Arithmetic is exact.
}{
\spadpaste{r + binary(6/7) \free{r}}
}
\xtc{
The period of the expansion can be short or long \ldots
}{
\spadpaste{[binary(1/i) for i in 102..106] }
}
\xtc{
or very long.
}{
\spadpaste{binary(1/1007) }
}
\xtc{
These numbers are bona fide algebraic objects.
}{
\spadpaste{p := binary(1/4)*x**2 + binary(2/3)*x + binary(4/9)  \bound{p}}
}
\xtc{
}{
\spadpaste{q := D(p, x) \free{p}\bound{q}}
}
\xtc{
}{
\spadpaste{g := gcd(p, q)            \free{p q}\bound{g}}
}
```

```
\endscroll
\autobuttons
\end{page}

\begin{patch}{BinaryExpansionXmpPagePatch1}
\begin{paste}{BinaryExpansionXmpPageFull1}{BinaryExpansionXmpPageEmpty1}
\pastebutton{BinaryExpansionXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{r := binary(22/7)\bound{r }}
\indentrel{3}\begin{verbatim}
          ---
   (1)  11.001
                                    Type: BinaryExpansion
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BinaryExpansionXmpPageEmpty1}
\begin{paste}{BinaryExpansionXmpPageEmpty1}{BinaryExpansionXmpPagePatch1}
\pastebutton{BinaryExpansionXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{r := binary(22/7)\bound{r }}
\end{paste}\end{patch}

\begin{patch}{BinaryExpansionXmpPagePatch2}
\begin{paste}{BinaryExpansionXmpPageFull2}{BinaryExpansionXmpPageEmpty2}
\pastebutton{BinaryExpansionXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{r + binary(6/7)\free{r }}
\indentrel{3}\begin{verbatim}
   (2)  100
                                    Type: BinaryExpansion
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BinaryExpansionXmpPageEmpty2}
\begin{paste}{BinaryExpansionXmpPageEmpty2}{BinaryExpansionXmpPagePatch2}
\pastebutton{BinaryExpansionXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{r + binary(6/7)\free{r }}
\end{paste}\end{patch}

\begin{patch}{BinaryExpansionXmpPagePatch3}
\begin{paste}{BinaryExpansionXmpPageFull3}{BinaryExpansionXmpPageEmpty3}
\pastebutton{BinaryExpansionXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{[binary(1/i) for i in 102..106]}
\indentrel{3}\begin{verbatim}
   (3)
         --------
   [0.000000101,
       ---------------------------------------------------
    0.000000100111111000100010110010111100111001001010100 1,
          ------------    ------------
    0.000000100111011, 0.000000100111,
```

```
     0.0
       OVERBAR
          00000100110101001000011100111110110010101101111
            00011
     ]
                             Type: List BinaryExpansion
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BinaryExpansionXmpPageEmpty3}
\begin{paste}{BinaryExpansionXmpPageEmpty3}{BinaryExpansionXmpPagePatch3}
\pastebutton{BinaryExpansionXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{[binary(1/i) for i in 102..106]}
\end{paste}\end{patch}

\begin{patch}{BinaryExpansionXmpPagePatch4}
\begin{paste}{BinaryExpansionXmpPageFull4}{BinaryExpansionXmpPageEmpty4}
\pastebutton{BinaryExpansionXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{binary(1/1007)}
\indentrel{3}\begin{verbatim}
   (4)
   0.
     OVERBAR
        00000000010000010001010010010111100000111111100001
          011111100101100011111010001001110010011001110001
          100100101010111101101001100000000110000110011111
          011100011010001011110100100011110110000101011110
          111001110101011100110010100101110000000111000111
          110010000000100100100110111001010100111010001101
          110110101110001001000001100101101100000001011001
          011111000101000001010101011010110000011011011110
          100101011111111010111010100110010000101001101100
          0100110001000100001000011000111010011110001
                                  Type: BinaryExpansion
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BinaryExpansionXmpPageEmpty4}
\begin{paste}{BinaryExpansionXmpPageEmpty4}{BinaryExpansionXmpPagePatch4}
\pastebutton{BinaryExpansionXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{binary(1/1007)}
\end{paste}\end{patch}

\begin{patch}{BinaryExpansionXmpPagePatch5}
\begin{paste}{BinaryExpansionXmpPageFull5}{BinaryExpansionXmpPageEmpty5}
\pastebutton{BinaryExpansionXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{
p := binary(1/4)*x**2 + binary(2/3)*x + binary(4/9)\bound{p }}
\indentrel{3}\begin{verbatim}
             2     --      ------
```

```
   (5)  0.01x  + 0.10x + 0.011100
                       Type: Polynomial BinaryExpansion
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BinaryExpansionXmpPageEmpty5}
\begin{paste}{BinaryExpansionXmpPageEmpty5}{BinaryExpansionXmpPagePatch5}
\pastebutton{BinaryExpansionXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{
p := binary(1/4)*x**2 + binary(2/3)*x + binary(4/9)\bound{p }}
\end{paste}\end{patch}

\begin{patch}{BinaryExpansionXmpPagePatch6}
\begin{paste}{BinaryExpansionXmpPageFull6}{BinaryExpansionXmpPageEmpty6}
\pastebutton{BinaryExpansionXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{q := D(p, x)\free{p }\bound{q }}
\indentrel{3}\begin{verbatim}
                  --
   (6)  0.1x + 0.10
                       Type: Polynomial BinaryExpansion
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BinaryExpansionXmpPageEmpty6}
\begin{paste}{BinaryExpansionXmpPageEmpty6}{BinaryExpansionXmpPagePatch6}
\pastebutton{BinaryExpansionXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{q := D(p, x)\free{p }\bound{q }}
\end{paste}\end{patch}

\begin{patch}{BinaryExpansionXmpPagePatch7}
\begin{paste}{BinaryExpansionXmpPageFull7}{BinaryExpansionXmpPageEmpty7}
\pastebutton{BinaryExpansionXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{g := gcd(p, q)\free{p q }\bound{g }}
\indentrel{3}\begin{verbatim}
               --
   (7)  x + 1.01
                       Type: Polynomial BinaryExpansion
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BinaryExpansionXmpPageEmpty7}
\begin{paste}{BinaryExpansionXmpPageEmpty7}{BinaryExpansionXmpPagePatch7}
\pastebutton{BinaryExpansionXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{g := gcd(p, q)\free{p q }\bound{g }}
\end{paste}\end{patch}
```

3.9 bmcat.ht

Bit Map Catalog

— bmcat.ht —

```
\begin{page}{BitMaps}{Bit Map Catalog}
\beginscroll
\space{} {\inputbitmap{\htbmdir{}/xfbox.bitmap}}      xfbox    \space{4}
\space{} {\inputbitmap{\htbmdir{}/xfcirc.bitmap}}     xfcirc   \space{4}
\space{} {\inputbitmap{\htbmdir{}/xnobox.bitmap}}     Xnobox   \space{2}
\space{} {\inputbitmap{\htbmdir{}/xnocirc.bitmap}}    xnocirc  \space{2}
\newline
\space{} {\inputbitmap{\htbmdir{}/xfullfbox.bitmap}}  xfullfbox
\space{} {\inputbitmap{\htbmdir{}/xfullfcirc.bitmap}} xfullfcirc
\space{} {\inputbitmap{\htbmdir{}/xfullbox.bitmap}}   xfullbox
\space{} {\inputbitmap{\htbmdir{}/xfullcirc.bitmap}}  xfullcirc
\newline
\space{} {\inputbitmap{\htbmdir{}/xgreyfbox.bitmap}}  xgreyfbox
\space{} {\inputbitmap{\htbmdir{}/xgreyfcirc.bitmap}} xgreyfcirc
\space{} {\inputbitmap{\htbmdir{}/xgreybox.bitmap}}   xgreybox
\space{} {\inputbitmap{\htbmdir{}/xgreycirc.bitmap}}  xgreycirc
\newline
\space{} {\inputbitmap{\htbmdir{}/xopenfbox.bitmap}}  xopenfbox
\space{} {\inputbitmap{\htbmdir{}/xopenfcirc.bitmap}} xopenfcirc
\space{} {\inputbitmap{\htbmdir{}/xopenbox.bitmap}}   xopenbox
\space{} {\inputbitmap{\htbmdir{}/xopencirc.bitmap}}  xopencirc
\newline
\space{} {\inputbitmap{\htbmdir{}/xtickfbox.bitmap}}  xtickfbox
\space{} {\inputbitmap{\htbmdir{}/xtickfcirc.bitmap}} xtickfcirc
\space{} {\inputbitmap{\htbmdir{}/xtickbox.bitmap}}   xtickbox
\space{} {\inputbitmap{\htbmdir{}/xtickcirc.bitmap}}  xtickcirc
\newline
\space{} {\inputbitmap{\htbmdir{}/xxfbox.bitmap}}     xxfbox    \space{3}
\space{} {\inputbitmap{\htbmdir{}/xxfcirc.bitmap}}    xxfcirc   \space{3}
\space{} {\inputbitmap{\htbmdir{}/xxbox.bitmap}}      xxbox     \space{3}
\space{} {\inputbitmap{\htbmdir{}/xxcirc.bitmap}}     xxcirc    \space{3}
\horizontalline
\space{} {\inputbitmap{\htbmdir{}/xnoface.bitmap}}    xnoface
\space{} {\inputbitmap{\htbmdir{}/xhappy.bitmap}}     xhappy
\space{} {\inputbitmap{\htbmdir{}/xsad.bitmap}}       xsad
\space{} {\inputbitmap{\htbmdir{}/xdesp.bitmap}}      xdesp
\space{} {\inputbitmap{\htbmdir{}/xperv.bitmap}}      xperv
\horizontalline
\space{} {\inputbitmap{\htbmdir{}/exit.bitmap}} exit
\space{} {\inputbitmap{\htbmdir{}/help3.bitmap}} help3
\newline
\space{} {\inputbitmap{\htbmdir{}/down3.bitmap}} down3
```

```
\space{} {\inputbitmap{\htbmdir{}/up3.bitmap}} up3
\space{} {\inputbitmap{\htbmdir{}/return3.bitmap}} return3
\newline
\space{} {\inputbitmap{\htbmdir{}/tear.bitmap}} tear
\space{} {\inputbitmap{\htbmdir{}/eqpage.bitmap}} eqpage
\horizontalline
\space{} {\inputbitmap{\htbmdir{}/sup.bm}} sup.bm
\space{} {\inputbitmap{\htbmdir{}/sup.bitmap}} sup
\space{} {\inputbitmap{\htbmdir{}/sdown.bm}} sdown.bm
\newline
\space{} {\inputbitmap{\htbmdir{}/ht_icon}} ht_icon
\space{} {\inputbitmap{\htbmdir{}/smile.bitmap}} smile
\newline
\space{} {\inputbitmap{\htbmdir{}/drown.bm}} drown.bm
\space{} {\inputbitmap{\htbmdir{}/help2.bitmap}} help2
\space{} {\inputbitmap{\htbmdir{}/return.bitmap}} return
\space{} {\inputbitmap{\htbmdir{}/back.bitmap}} back
\endscroll
\autobuttons
\end{page}
```

3.10 bop.ht

BasicOperator

— bop.ht —

```
\begin{page}{BasicOperatorXmpPage}{BasicOperator}
\beginscroll

A basic operator is an object that can be symbolically
applied to a list of arguments from a set, the result being
a kernel over that set or an expression.
In addition to this section, please see
\downlink{‘Expression’}{ExpressionXmpPage}\ignore{Expression}
and \downlink{‘Kernel’}{KernelXmpPage}\ignore{Kernel}
for additional information and examples.

You create an object of type \axiomType{BasicOperator} by using the
\axiomFunFrom{operator}{BasicOperator} operation.
This first form of this operation has one argument and it
must be a symbol.
The symbol should be quoted in case the
name has been used as an identifier to which a value
```

```
has been assigned.

A frequent application of \axiomType{BasicOperator} is the
creation of an operator to represent the unknown function when
solving a differential equation.
\xtc{
Let \axiom{y} be the unknown function in terms of \axiom{x}.
}{
\spadpaste{y := operator 'y \bound{y}}
}
%
\xtc{
This is how you enter
the equation \axiom{y'' + y' + y = 0}.
}{
\spadpaste{deq := D(y x, x, 2) + D(y x, x) + y x = 0\bound{e1}\free{y}}
}
%
\xtc{
To solve the above equation, enter this.
}{
\spadpaste{solve(deq, y, x) \free{e1}\free{y}}
}
See \downlink{``Solution of Differential Equations''}{ugProblemDEQPage}
in Section 8.10\ignore{ugProblemDEQ}
for this kind of use of \axiomType{BasicOperator}.

Use the single argument form of
\axiomFunFrom{operator}{BasicOperator} (as above) when you intend
to use the operator to create functional expressions with an
arbitrary number of arguments
\xtc{
{\it Nary} means an arbitrary number of arguments can be used
in the functional expressions.
}{
\spadpaste{nary? y \free{y}}
}
\xtc{
}{
\spadpaste{unary? y \free{y}}
}
Use the two-argument form when you want to restrict the number of
arguments in the functional expressions created with the operator.
\xtc{
This operator can only be used to create functional expressions
with one argument.
}{
\spadpaste{opOne := operator('opOne, 1) \bound{opOne}}
}
\xtc{
```

```
}{
\spadpaste{nary? opOne \free{opOne}}
}
\xtc{
}{
\spadpaste{unary? opOne \free{opOne}}
}
\xtc{
Use \axiomFunFrom{arity}{BasicOperator} to learn the number of
arguments that can be used.
It returns {\tt "false"} if the operator is nary.
}{
\spadpaste{arity opOne \free{opOne}}
}
\xtc{
Use \axiomFunFrom{name}{BasicOperator} to learn the name of an
operator.
}{
\spadpaste{name opOne \free{opOne}}
}
\xtc{
Use \axiomFunFrom{is?}{BasicOperator} to learn if an operator has a
particular name.
}{
\spadpaste{is?(opOne, 'z2) \free{opOne}}
}
\xtc{
You can also use a string as the name to be tested against.
}{
\spadpaste{is?(opOne, "opOne") \free{opOne}}
}

You can attached named properties to an operator.
These are rarely used at the top-level of the Axiom
interactive environment but are used with Axiom
library source code.
\xtc{
By default, an operator has no properties.
}{
\spadpaste{properties y \free{y}}
}
The interface for setting and getting properties is somewhat awkward
because the property values are stored as values of type
\axiomType{None}.
\xtc{
Attach a property by using \axiomFunFrom{setProperty}{BasicOperator}.
}{
\spadpaste{setProperty(y, "use", "unknown function" :: None )
\free{y}\bound{spy}}
}
```

```
\xtc{
}{
\spadpaste{properties y \free{y spy}}
}
\xtc{
We {\it know} the property value has type \axiomType{String}.
}{
\spadpaste{property(y, "use") :: None pretend String \free{y spy}}
}
\xtc{
Use \axiomFunFrom{deleteProperty!}{BasicOperator} to destructively
remove a property.
}{
\spadpaste{deleteProperty!(y, "use") \free{y spy}\bound{dpy}}
}
\xtc{
}{
\spadpaste{properties y \free{dpy}}
}
\endscroll
\autobuttons
\end{page}

\begin{patch}{BasicOperatorXmpPagePatch1}
\begin{paste}{BasicOperatorXmpPageFull1}{BasicOperatorXmpPageEmpty1}
\pastebutton{BasicOperatorXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{y := operator 'y\bound{y }}
\indentrel{3}\begin{verbatim}
   (1)  y
                                    Type: BasicOperator
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BasicOperatorXmpPageEmpty1}
\begin{paste}{BasicOperatorXmpPageEmpty1}{BasicOperatorXmpPagePatch1}
\pastebutton{BasicOperatorXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{y := operator 'y\bound{y }}
\end{paste}\end{patch}

\begin{patch}{BasicOperatorXmpPagePatch2}
\begin{paste}{BasicOperatorXmpPageFull2}{BasicOperatorXmpPageEmpty2}
\pastebutton{BasicOperatorXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{
deq := D(y x, x, 2) + D(y x, x) + y x = 0\bound{e1 }\free{y }}
\indentrel{3}\begin{verbatim}
         ,,        ,
   (2)  y  (x) + y (x) + y(x)= 0

                      Type: Equation Expression Integer
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BasicOperatorXmpPageEmpty2}
\begin{paste}{BasicOperatorXmpPageEmpty2}{BasicOperatorXmpPagePatch2}
\pastebutton{BasicOperatorXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{
deq := D(y x, x, 2) + D(y x, x) + y x = 0\bound{e1 }\free{y }}
\end{paste}\end{patch}

\begin{patch}{BasicOperatorXmpPagePatch3}
\begin{paste}{BasicOperatorXmpPageFull3}{BasicOperatorXmpPageEmpty3}
\pastebutton{BasicOperatorXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{solve(deq, y, x)\free{e1 }\free{y }}
\indentrel{3}\begin{verbatim}
   (3)
   [particular= 0,
                            x     x
                          -    -
                x\3      2     2    x\3
    basis= [cos()%e   ,%e   sin()]]
                  2                   2
Type: Union(Record(particular: Expression Integer,
                   basis: List Expression Integer),...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BasicOperatorXmpPageEmpty3}
\begin{paste}{BasicOperatorXmpPageEmpty3}{BasicOperatorXmpPagePatch3}
\pastebutton{BasicOperatorXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{solve(deq, y, x)\free{e1 }\free{y }}
\end{paste}\end{patch}

\begin{patch}{BasicOperatorXmpPagePatch4}
\begin{paste}{BasicOperatorXmpPageFull4}{BasicOperatorXmpPageEmpty4}
\pastebutton{BasicOperatorXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{nary? y\free{y }}
\indentrel{3}\begin{verbatim}
   (4)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BasicOperatorXmpPageEmpty4}
\begin{paste}{BasicOperatorXmpPageEmpty4}{BasicOperatorXmpPagePatch4}
\pastebutton{BasicOperatorXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{nary? y\free{y }}
\end{paste}\end{patch}

\begin{patch}{BasicOperatorXmpPagePatch5}
\begin{paste}{BasicOperatorXmpPageFull5}{BasicOperatorXmpPageEmpty5}
```

```
\pastebutton{BasicOperatorXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{unary? y\free{y }}
\indentrel{3}\begin{verbatim}
   (5)  false
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BasicOperatorXmpPageEmpty5}
\begin{paste}{BasicOperatorXmpPageEmpty5}{BasicOperatorXmpPagePatch5}
\pastebutton{BasicOperatorXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{unary? y\free{y }}
\end{paste}\end{patch}

\begin{patch}{BasicOperatorXmpPagePatch6}
\begin{paste}{BasicOperatorXmpPageFull6}{BasicOperatorXmpPageEmpty6}
\pastebutton{BasicOperatorXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{opOne := operator('opOne, 1)\bound{opOne }}
\indentrel{3}\begin{verbatim}
   (6)  opOne
                                    Type: BasicOperator
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BasicOperatorXmpPageEmpty6}
\begin{paste}{BasicOperatorXmpPageEmpty6}{BasicOperatorXmpPagePatch6}
\pastebutton{BasicOperatorXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{opOne := operator('opOne, 1)\bound{opOne }}
\end{paste}\end{patch}

\begin{patch}{BasicOperatorXmpPagePatch7}
\begin{paste}{BasicOperatorXmpPageFull7}{BasicOperatorXmpPageEmpty7}
\pastebutton{BasicOperatorXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{nary? opOne\free{opOne }}
\indentrel{3}\begin{verbatim}
   (7)  false
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BasicOperatorXmpPageEmpty7}
\begin{paste}{BasicOperatorXmpPageEmpty7}{BasicOperatorXmpPagePatch7}
\pastebutton{BasicOperatorXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{nary? opOne\free{opOne }}
\end{paste}\end{patch}

\begin{patch}{BasicOperatorXmpPagePatch8}
\begin{paste}{BasicOperatorXmpPageFull8}{BasicOperatorXmpPageEmpty8}
\pastebutton{BasicOperatorXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{unary? opOne\free{opOne }}
```

```
\indentrel{3}\begin{verbatim}
   (8)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BasicOperatorXmpPageEmpty8}
\begin{paste}{BasicOperatorXmpPageEmpty8}{BasicOperatorXmpPagePatch8}
\pastebutton{BasicOperatorXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{unary? opOne\free{opOne }}
\end{paste}\end{patch}

\begin{patch}{BasicOperatorXmpPagePatch9}
\begin{paste}{BasicOperatorXmpPageFull9}{BasicOperatorXmpPageEmpty9}
\pastebutton{BasicOperatorXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{arity opOne\free{opOne }}
\indentrel{3}\begin{verbatim}
   (9)  1
                    Type: Union(NonNegativeInteger,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BasicOperatorXmpPageEmpty9}
\begin{paste}{BasicOperatorXmpPageEmpty9}{BasicOperatorXmpPagePatch9}
\pastebutton{BasicOperatorXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{arity opOne\free{opOne }}
\end{paste}\end{patch}

\begin{patch}{BasicOperatorXmpPagePatch10}
\begin{paste}{BasicOperatorXmpPageFull10}{BasicOperatorXmpPageEmpty10}
\pastebutton{BasicOperatorXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{name opOne\free{opOne }}
\indentrel{3}\begin{verbatim}
   (10)  opOne
                                           Type: Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BasicOperatorXmpPageEmpty10}
\begin{paste}{BasicOperatorXmpPageEmpty10}{BasicOperatorXmpPagePatch10}
\pastebutton{BasicOperatorXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{name opOne\free{opOne }}
\end{paste}\end{patch}

\begin{patch}{BasicOperatorXmpPagePatch11}
\begin{paste}{BasicOperatorXmpPageFull11}{BasicOperatorXmpPageEmpty11}
\pastebutton{BasicOperatorXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{is?(opOne, 'z2)\free{opOne }}
\indentrel{3}\begin{verbatim}
   (11)  false
```

```
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BasicOperatorXmpPageEmpty11}
\begin{paste}{BasicOperatorXmpPageEmpty11}{BasicOperatorXmpPagePatch11}
\pastebutton{BasicOperatorXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{is?(opOne, 'z2)\free{opOne }}
\end{paste}\end{patch}

\begin{patch}{BasicOperatorXmpPagePatch12}
\begin{paste}{BasicOperatorXmpPageFull12}{BasicOperatorXmpPageEmpty12}
\pastebutton{BasicOperatorXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{is?(opOne, "opOne")\free{opOne }}
\indentrel{3}\begin{verbatim}
   (12)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BasicOperatorXmpPageEmpty12}
\begin{paste}{BasicOperatorXmpPageEmpty12}{BasicOperatorXmpPagePatch12}
\pastebutton{BasicOperatorXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{is?(opOne, "opOne")\free{opOne }}
\end{paste}\end{patch}

\begin{patch}{BasicOperatorXmpPagePatch13}
\begin{paste}{BasicOperatorXmpPageFull13}{BasicOperatorXmpPageEmpty13}
\pastebutton{BasicOperatorXmpPageFull13}{\hidepaste}
\tab{5}\spadcommand{properties y\free{y }}
\indentrel{3}\begin{verbatim}
   (13)  table()
                     Type: AssociationList(String,None)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BasicOperatorXmpPageEmpty13}
\begin{paste}{BasicOperatorXmpPageEmpty13}{BasicOperatorXmpPagePatch13}
\pastebutton{BasicOperatorXmpPageEmpty13}{\showpaste}
\tab{5}\spadcommand{properties y\free{y }}
\end{paste}\end{patch}

\begin{patch}{BasicOperatorXmpPagePatch14}
\begin{paste}{BasicOperatorXmpPageFull14}{BasicOperatorXmpPageEmpty14}
\pastebutton{BasicOperatorXmpPageFull14}{\hidepaste}
\tab{5}\spadcommand{
setProperty(y, "use", "unknown function" :: None )\free{y }\bound{spy }}
\indentrel{3}\begin{verbatim}
   (14)  y
                                    Type: BasicOperator
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BasicOperatorXmpPageEmpty14}
\begin{paste}{BasicOperatorXmpPageEmpty14}{BasicOperatorXmpPagePatch14}
\pastebutton{BasicOperatorXmpPageEmpty14}{\showpaste}
\tab{5}\spadcommand{
setProperty(y, "use", "unknown function" :: None )\free{y }\bound{spy }}
\end{paste}\end{patch}

\begin{patch}{BasicOperatorXmpPagePatch15}
\begin{paste}{BasicOperatorXmpPageFull15}{BasicOperatorXmpPageEmpty15}
\pastebutton{BasicOperatorXmpPageFull15}{\hidepaste}
\tab{5}\spadcommand{properties y\free{y spy }}
\indentrel{3}\begin{verbatim}
   (15)  table("use"= NONE)
                     Type: AssociationList(String,None)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BasicOperatorXmpPageEmpty15}
\begin{paste}{BasicOperatorXmpPageEmpty15}{BasicOperatorXmpPagePatch15}
\pastebutton{BasicOperatorXmpPageEmpty15}{\showpaste}
\tab{5}\spadcommand{properties y\free{y spy }}
\end{paste}\end{patch}

\begin{patch}{BasicOperatorXmpPagePatch16}
\begin{paste}{BasicOperatorXmpPageFull16}{BasicOperatorXmpPageEmpty16}
\pastebutton{BasicOperatorXmpPageFull16}{\hidepaste}
\tab{5}\spadcommand{property(y, "use") :: None pretend String\free{y spy }}
\indentrel{3}\begin{verbatim}
   (16)  "unknown function"
                                           Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BasicOperatorXmpPageEmpty16}
\begin{paste}{BasicOperatorXmpPageEmpty16}{BasicOperatorXmpPagePatch16}
\pastebutton{BasicOperatorXmpPageEmpty16}{\showpaste}
\tab{5}\spadcommand{property(y, "use") :: None pretend String\free{y spy }}
\end{paste}\end{patch}

\begin{patch}{BasicOperatorXmpPagePatch17}
\begin{paste}{BasicOperatorXmpPageFull17}{BasicOperatorXmpPageEmpty17}
\pastebutton{BasicOperatorXmpPageFull17}{\hidepaste}
\tab{5}\spadcommand{deleteProperty!(y, "use")\free{y spy }\bound{dpy }}
\indentrel{3}\begin{verbatim}
   (17)  y
                                    Type: BasicOperator
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BasicOperatorXmpPageEmpty17}
\begin{paste}{BasicOperatorXmpPageEmpty17}{BasicOperatorXmpPagePatch17}
\pastebutton{BasicOperatorXmpPageEmpty17}{\showpaste}
\tab{5}\spadcommand{deleteProperty!(y, "use")\free{y spy }\bound{dpy }}
\end{paste}\end{patch}

\begin{patch}{BasicOperatorXmpPagePatch18}
\begin{paste}{BasicOperatorXmpPageFull18}{BasicOperatorXmpPageEmpty18}
\pastebutton{BasicOperatorXmpPageFull18}{\hidepaste}
\tab{5}\spadcommand{properties y\free{dpy }}
\indentrel{3}\begin{verbatim}
   (18)  table()
                    Type: AssociationList(String,None)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BasicOperatorXmpPageEmpty18}
\begin{paste}{BasicOperatorXmpPageEmpty18}{BasicOperatorXmpPagePatch18}
\pastebutton{BasicOperatorXmpPageEmpty18}{\showpaste}
\tab{5}\spadcommand{properties y\free{dpy }}
\end{paste}\end{patch}
```

———-

3.11 bstree.ht

BinarySearchTree

— bstree.ht —

```
\begin{page}{BinarySearchTreeXmpPage}{BinarySearchTree}
\beginscroll

\spadtype{BinarySearchTree(R)} is the domain of binary trees with
elements of type \spad{R}, ordered across the nodes of the tree.  A
non-empty binary search tree has a value of type \spad{R}, and
\spadfun{right} and \spadfun{left} binary search subtrees.  If a
subtree is empty, it is displayed as a period (‘‘.’’).

\xtc{
Define a list of values to be placed across the tree.
The resulting tree has \spad{8} at the root;
all other elements are in the left subtree.
```

```
}{
\spadpaste{lv := [8,3,5,4,6,2,1,5,7]\bound{lv}}
}
\xtc{
A convenient way to create a binary search tree is to
apply the operation \spadfun{binarySearchTree} to a list of elements.
}{
\spadpaste{t := binarySearchTree lv\free{lv}\bound{t}}
}
\xtc{
Another approach is to first create an empty binary search tree of integers.
}{
\spadpaste{emptybst := empty()\$BSTREE(INT)\bound{e}}
}
\xtc{
Insert the value \spad{8}.
This establishes \spad{8} as the root of the binary search tree.
Values inserted later that are less than \spad{8} get stored in
the \spadfun{left} subtree, others in the \spadfun{right} subtree.
}{
\spadpaste{t1 := insert!(8,emptybst)\free{e}\bound{t1}}
}
\xtc{
Insert the value \spad{3}. This number becomes the root of the
\spadfun{left} subtree of \spad{t1}.
For optimal retrieval, it is thus important to insert the
middle elements first.
}{
\spadpaste{insert!(3,t1)\free{t1}}
}
\xtc{
We go back to the original tree \spad{t}.
The leaves of the binary search tree are those which have empty
\spadfun{left} and \spadfun{right} subtrees.
}{
\spadpaste{leaves t\free{t}}
}
\xtc{
The operation
\spadfun{split}\spad{(k,t)} returns a \spadgloss{record} containing
the two subtrees: one with all elements ``less'' than \spad{k},
another with elements ``greater'' than \spad{k}.
}{
\spadpaste{split(3,t)\free{t}}
}
\xtc{
Define \userfun{insertRoot} to insert new elements by
creating a new node.
}{
\spadpaste{insertRoot: (INT,BSTREE INT) -> BSTREE INT\bound{x}}
```

```
}
\xtc{
The new node puts the inserted value between its
``less'' tree and ``greater'' tree.
}{
\begin{spadsrc}[\bound{x1}\free{x}]
insertRoot(x, t) ==
    a := split(x, t)
    node(a.less, x, a.greater)
\end{spadsrc}
}
\xtc{
Function \userfun{buildFromRoot} builds
a binary search tree from a list of elements \spad{ls}
and the empty tree \spad{emptybst}.
}{
\spadpaste{buildFromRoot ls == reduce(insertRoot,ls,emptybst)
\bound{x2}\free{x1 e}}
}
\xtc{
Apply this to the reverse of the list \spad{lv}.
}{
\spadpaste{rt := buildFromRoot reverse lv\bound{rt}\free{lv x2}}
}
\xtc{
Have Axiom check that these are equal.
}{
\spadpaste{(t = rt)@Boolean\free{rt t}}
}
\endscroll
\autobuttons
\end{page}

\begin{patch}{BinarySearchTreeXmpPagePatch1}
\begin{paste}{BinarySearchTreeXmpPageFull1}{BinarySearchTreeXmpPageEmpty1}
\pastebutton{BinarySearchTreeXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{lv := [8,3,5,4,6,2,1,5,7]\bound{lv }}
\indentrel{3}\begin{verbatim}
   (1)  [8,3,5,4,6,2,1,5,7]
                             Type: List PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BinarySearchTreeXmpPageEmpty1}
\begin{paste}{BinarySearchTreeXmpPageEmpty1}{BinarySearchTreeXmpPagePatch1}
\pastebutton{BinarySearchTreeXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{lv := [8,3,5,4,6,2,1,5,7]\bound{lv }}
\end{paste}\end{patch}

\begin{patch}{BinarySearchTreeXmpPagePatch2}
```

```
\begin{paste}{BinarySearchTreeXmpPageFull2}{BinarySearchTreeXmpPageEmpty2}
\pastebutton{BinarySearchTreeXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{t := binarySearchTree lv\free{lv }\bound{t }}
\indentrel{3}\begin{verbatim}
   (2)  [[[1,2,.],3,[4,5,[5,6,7]]],8,.]
                 Type: BinarySearchTree PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BinarySearchTreeXmpPageEmpty2}
\begin{paste}{BinarySearchTreeXmpPageEmpty2}{BinarySearchTreeXmpPagePatch2}
\pastebutton{BinarySearchTreeXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{t := binarySearchTree lv\free{lv }\bound{t }}
\end{paste}\end{patch}

\begin{patch}{BinarySearchTreeXmpPagePatch3}
\begin{paste}{BinarySearchTreeXmpPageFull3}{BinarySearchTreeXmpPageEmpty3}
\pastebutton{BinarySearchTreeXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{emptybst := empty()$BSTREE(INT)\bound{e }}
\indentrel{3}\begin{verbatim}
   (3)  []
                         Type: BinarySearchTree Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BinarySearchTreeXmpPageEmpty3}
\begin{paste}{BinarySearchTreeXmpPageEmpty3}{BinarySearchTreeXmpPagePatch3}
\pastebutton{BinarySearchTreeXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{emptybst := empty()$BSTREE(INT)\bound{e }}
\end{paste}\end{patch}

\begin{patch}{BinarySearchTreeXmpPagePatch4}
\begin{paste}{BinarySearchTreeXmpPageFull4}{BinarySearchTreeXmpPageEmpty4}
\pastebutton{BinarySearchTreeXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{t1 := insert!(8,emptybst)\free{e }\bound{t1 }}
\indentrel{3}\begin{verbatim}
   (4)  8
                         Type: BinarySearchTree Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BinarySearchTreeXmpPageEmpty4}
\begin{paste}{BinarySearchTreeXmpPageEmpty4}{BinarySearchTreeXmpPagePatch4}
\pastebutton{BinarySearchTreeXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{t1 := insert!(8,emptybst)\free{e }\bound{t1 }}
\end{paste}\end{patch}

\begin{patch}{BinarySearchTreeXmpPagePatch5}
\begin{paste}{BinarySearchTreeXmpPageFull5}{BinarySearchTreeXmpPageEmpty5}
\pastebutton{BinarySearchTreeXmpPageFull5}{\hidepaste}
```

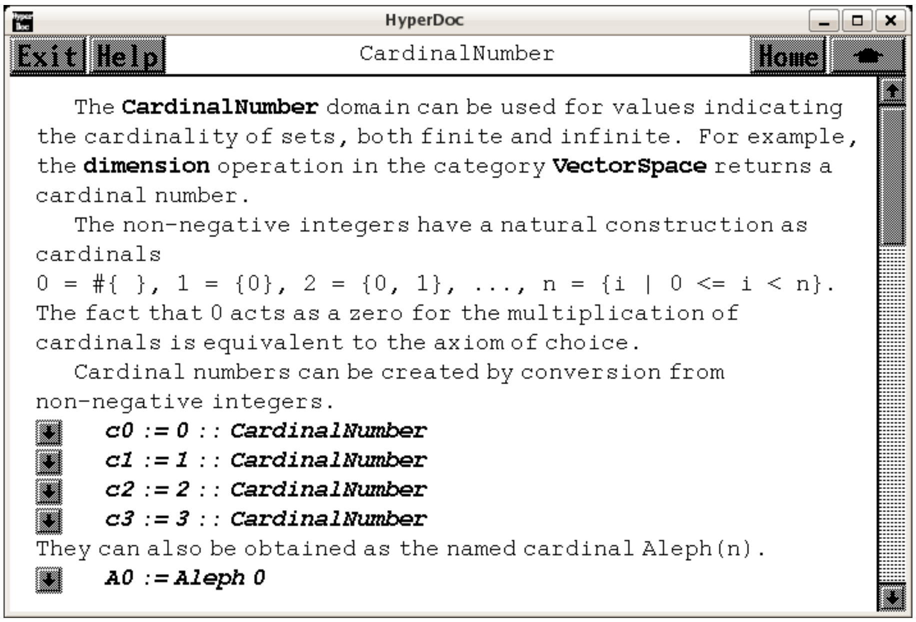
```
\tab{5}\spadcommand{insert!(3,t1)\free{t1 }}
\indentrel{3}\begin{verbatim}
   (5)  [3,8,.]
                         Type: BinarySearchTree Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BinarySearchTreeXmpPageEmpty5}
\begin{paste}{BinarySearchTreeXmpPageEmpty5}{BinarySearchTreeXmpPagePatch5}
\pastebutton{BinarySearchTreeXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{insert!(3,t1)\free{t1 }}
\end{paste}\end{patch}

\begin{patch}{BinarySearchTreeXmpPagePatch6}
\begin{paste}{BinarySearchTreeXmpPageFull6}{BinarySearchTreeXmpPageEmpty6}
\pastebutton{BinarySearchTreeXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{leaves t\free{t }}
\indentrel{3}\begin{verbatim}
   (6)  [1,4,5,7]
                             Type: List PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BinarySearchTreeXmpPageEmpty6}
\begin{paste}{BinarySearchTreeXmpPageEmpty6}{BinarySearchTreeXmpPagePatch6}
\pastebutton{BinarySearchTreeXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{leaves t\free{t }}
\end{paste}\end{patch}

\begin{patch}{BinarySearchTreeXmpPagePatch7}
\begin{paste}{BinarySearchTreeXmpPageFull7}{BinarySearchTreeXmpPageEmpty7}
\pastebutton{BinarySearchTreeXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{split(3,t)\free{t }}
\indentrel{3}\begin{verbatim}
   (7)
   [less= [1,2,.],greater= [[.,3,[4,5,[5,6,7]]],8,.]]
Type: Record(less: BinarySearchTree PositiveInteger,
             greater: BinarySearchTree PositiveInteger)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BinarySearchTreeXmpPageEmpty7}
\begin{paste}{BinarySearchTreeXmpPageEmpty7}{BinarySearchTreeXmpPagePatch7}
\pastebutton{BinarySearchTreeXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{split(3,t)\free{t }}
\end{paste}\end{patch}

\begin{patch}{BinarySearchTreeXmpPagePatch8}
\begin{paste}{BinarySearchTreeXmpPageFull8}{BinarySearchTreeXmpPageEmpty8}
\pastebutton{BinarySearchTreeXmpPageFull8}{\hidepaste}
```

```
\tab{5}\spadcommand{insertRoot: (INT,BSTREE INT) -> BSTREE INT\bound{x }}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BinarySearchTreeXmpPageEmpty8}
\begin{paste}{BinarySearchTreeXmpPageEmpty8}{BinarySearchTreeXmpPagePatch8}
\pastebutton{BinarySearchTreeXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{insertRoot: (INT,BSTREE INT) -> BSTREE INT\bound{x }}
\end{paste}\end{patch}

\begin{patch}{BinarySearchTreeXmpPagePatch9}
\begin{paste}{BinarySearchTreeXmpPageFull9}{BinarySearchTreeXmpPageEmpty9}
\pastebutton{BinarySearchTreeXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{insertRoot(x, t) ==
    a := split(x, t)
    node(a.less, x, a.greater)
\bound{x1 }\free{x }}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BinarySearchTreeXmpPageEmpty9}
\begin{paste}{BinarySearchTreeXmpPageEmpty9}{BinarySearchTreeXmpPagePatch9}
\pastebutton{BinarySearchTreeXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{insertRoot(x, t) ==
    a := split(x, t)
    node(a.less, x, a.greater)
\bound{x1 }\free{x }}
\end{paste}\end{patch}

\begin{patch}{BinarySearchTreeXmpPagePatch10}
\begin{paste}{BinarySearchTreeXmpPageFull10}{BinarySearchTreeXmpPageEmpty10}
\pastebutton{BinarySearchTreeXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{
buildFromRoot ls == reduce(insertRoot,ls,emptybst)\bound{x2 }\free{x1 e }}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BinarySearchTreeXmpPageEmpty10}
\begin{paste}{BinarySearchTreeXmpPageEmpty10}
{BinarySearchTreeXmpPagePatch10}
\pastebutton{BinarySearchTreeXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{
buildFromRoot ls == reduce(insertRoot,ls,emptybst)\bound{x2 }\free{x1 e }}
\end{paste}\end{patch}
```

```
\begin{patch}{BinarySearchTreeXmpPagePatch11}
\begin{paste}{BinarySearchTreeXmpPageFull11}
{BinarySearchTreeXmpPageEmpty11}
\pastebutton{BinarySearchTreeXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{rt := buildFromRoot reverse lv\bound{rt }\free{lv x2 }}
\indentrel{3}\begin{verbatim}
   (11)  [[[1,2,.],3,[4,5,[5,6,7]]],8,.]
                        Type: BinarySearchTree Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BinarySearchTreeXmpPageEmpty11}
\begin{paste}{BinarySearchTreeXmpPageEmpty11}
{BinarySearchTreeXmpPagePatch11}
\pastebutton{BinarySearchTreeXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{rt := buildFromRoot reverse lv\bound{rt }\free{lv x2 }}
\end{paste}\end{patch}

\begin{patch}{BinarySearchTreeXmpPagePatch12}
\begin{paste}{BinarySearchTreeXmpPageFull12}
{BinarySearchTreeXmpPageEmpty12}
\pastebutton{BinarySearchTreeXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{(t = rt)@Boolean\free{rt t }}
\indentrel{3}\begin{verbatim}
   (12)  true
                                         Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{BinarySearchTreeXmpPageEmpty12}
\begin{paste}{BinarySearchTreeXmpPageEmpty12}
{BinarySearchTreeXmpPagePatch12}
\pastebutton{BinarySearchTreeXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{(t = rt)@Boolean\free{rt t }}
\end{paste}\end{patch}
```

3.12 card.ht

CardinalNumber



⇐ "Axiom Number Types" (NumberPage) 3.80 on page 1021

— **card.ht** —

```
\begin{page}{CardinalNumberXmpPage}{CardinalNumber}
\beginscroll
The \spadtype{CardinalNumber} domain can be used for values indicating
the cardinality of sets, both finite and infinite.
For example, the \spadfunFrom{dimension}{VectorSpace} operation in the
category \spadtype{VectorSpace} returns a cardinal number.

The non-negative integers have a natural construction as cardinals
\begin{verbatim}
0 = #{ }, 1 = {0}, 2 = {0, 1}, ..., n = {i | 0 <= i < n}.
\end{verbatim}
The fact that \spad{0} acts as a zero for the multiplication of
cardinals is equivalent to the axiom of choice.

\xtc{
Cardinal numbers can be created by conversion from non-negative integers.
}{
\spadpaste{c0 := 0 :: CardinalNumber \bound{c0}}
}
\xtc{
```

```
}{
\spadpaste{c1 := 1 :: CardinalNumber \bound{c1}}
}
\xtc{
}{
\spadpaste{c2 := 2 :: CardinalNumber \bound{c2}}
}
\xtc{
}{
\spadpaste{c3 := 3 :: CardinalNumber \bound{c3}}
}
\xtc{
They can also be obtained as the named cardinal \spad{Aleph(n)}.
}{
\spadpaste{A0 := Aleph 0 \bound{A0}}
}
\xtc{
}{
\spadpaste{A1 := Aleph 1 \bound{A1}}
}
\xtc{
The \spadfunFrom{finite?}{CardinalNumber} operation tests whether a
value is a finite cardinal, that is, a non-negative integer.
}{
\spadpaste{finite? c2 \free{c2}}
}
\xtc{
}{
\spadpaste{finite? A0 \free{A0}}
}
\xtc{
Similarly, the \spadfunFrom{countable?}{CardinalNumber}
operation determines whether a value is
a countable cardinal, that is, finite or \spad{Aleph(0)}.
}{
\spadpaste{countable? c2 \free{c2}}
}
\xtc{
}{
\spadpaste{countable? A0 \free{A0}}
}
\xtc{
}{
\spadpaste{countable? A1 \free{A1}}
}
Arithmetic operations are defined on cardinal numbers as follows:
If \spad{x = \#X}  and  \spad{y = \#Y} then

\indent{0}
\spad{x+y  = \#(X+Y)}
```

```
\tab{20} cardinality of the disjoint union \newline
\spad{x-y  = \#(X-Y)}
\tab{20} cardinality of the relative complement \newline
\spad{x*y  = \#(X*Y)}
\tab{20} cardinality of the Cartesian product \newline
\spad{x**y = \#(X**Y)}
\tab{20} cardinality of the set of maps from \spad{Y} to \spad{X} \newline

\xtc{
Here are some arithmetic examples.
}{
\spadpaste{[c2 + c2, c2 + A1] \free{c2 A1}}
}
\xtc{
}{
\spadpaste{[c0*c2, c1*c2, c2*c2, c0*A1, c1*A1, c2*A1, A0*A1]
\free{c0,c1,c2,A0,A1}}
}
\xtc{
}{
\spadpaste{[c2**c0, c2**c1, c2**c2, A1**c0, A1**c1, A1**c2]
\free{c0,c1,c2,A1}}
}
\xtc{
Subtraction is a partial operation: it is not defined
when subtracting a larger cardinal from a smaller one, nor
when subtracting two equal infinite cardinals.
}{
\spadpaste{[c2-c1, c2-c2, c2-c3, A1-c2, A1-A0, A1-A1]
\free{c1,c2,c3,A0,A1}}
}
The generalized continuum hypothesis asserts that
\begin{verbatim}
2**Aleph i = Aleph(i+1)
\end{verbatim}
and is independent of the axioms of set theory.\footnote{Goedel,
{\it The consistency of the continuum hypothesis,}
Ann. Math. Studies, Princeton Univ. Press, 1940.}
\xtc{
The \spadtype{CardinalNumber} domain provides an operation to assert
whether the hypothesis is to be assumed.
}{
\spadpaste{generalizedContinuumHypothesisAssumed true \bound{GCH}}
}
\xtc{
When the generalized continuum hypothesis
is assumed, exponentiation to a transfinite power is allowed.
}{
\spadpaste{[c0**A0, c1**A0, c2**A0, A0**A0, A0**A1, A1**A0, A1**A1]
\free{c0,c1,c2,A0,A1,GCH}}
```

```
}

Three commonly encountered cardinal numbers are

\indent{0}
\spad{a = \#}{\bf Z} \tab{20}       countable infinity \newline
\spad{c = \#}{\bf R} \tab{20}      the continuum       \newline
\spad{f = \#\{g| g: [0,1] -> {\bf R}\}}                \newline

\xtc{
In this domain, these values are obtained under the
generalized continuum hypothesis in this way.
}{
\spadpaste{a := Aleph 0 \free{GCH}\bound{a}}
}
\xtc{
}{
\spadpaste{c := 2**a    \free{a}  \bound{c}}
}
\xtc{
}{
\spadpaste{f := 2**c    \free{c}  \bound{f}}
}
\endscroll
\autobuttons
\end{page}

\begin{patch}{CardinalNumberXmpPagePatch1}
\begin{paste}{CardinalNumberXmpPageFull1}{CardinalNumberXmpPageEmpty1}
\pastebutton{CardinalNumberXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{c0 := 0 :: CardinalNumber\bound{c0 }}
\indentrel{3}\begin{verbatim}
   (1)  0
                                   Type: CardinalNumber
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CardinalNumberXmpPageEmpty1}
\begin{paste}{CardinalNumberXmpPageEmpty1}{CardinalNumberXmpPagePatch1}
\pastebutton{CardinalNumberXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{c0 := 0 :: CardinalNumber\bound{c0 }}
\end{paste}\end{patch}

\begin{patch}{CardinalNumberXmpPagePatch2}
\begin{paste}{CardinalNumberXmpPageFull2}{CardinalNumberXmpPageEmpty2}
\pastebutton{CardinalNumberXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{c1 := 1 :: CardinalNumber\bound{c1 }}
\indentrel{3}\begin{verbatim}
   (2)  1
                                   Type: CardinalNumber
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CardinalNumberXmpPageEmpty2}
\begin{paste}{CardinalNumberXmpPageEmpty2}{CardinalNumberXmpPagePatch2}
\pastebutton{CardinalNumberXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{c1 := 1 :: CardinalNumber\bound{c1 }}
\end{paste}\end{patch}

\begin{patch}{CardinalNumberXmpPagePatch3}
\begin{paste}{CardinalNumberXmpPageFull3}{CardinalNumberXmpPageEmpty3}
\pastebutton{CardinalNumberXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{c2 := 2 :: CardinalNumber\bound{c2 }}
\indentrel{3}\begin{verbatim}
   (3)  2
                                   Type: CardinalNumber
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CardinalNumberXmpPageEmpty3}
\begin{paste}{CardinalNumberXmpPageEmpty3}{CardinalNumberXmpPagePatch3}
\pastebutton{CardinalNumberXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{c2 := 2 :: CardinalNumber\bound{c2 }}
\end{paste}\end{patch}

\begin{patch}{CardinalNumberXmpPagePatch4}
\begin{paste}{CardinalNumberXmpPageFull4}{CardinalNumberXmpPageEmpty4}
\pastebutton{CardinalNumberXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{c3 := 3 :: CardinalNumber\bound{c3 }}
\indentrel{3}\begin{verbatim}
   (4)  3
                                   Type: CardinalNumber
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CardinalNumberXmpPageEmpty4}
\begin{paste}{CardinalNumberXmpPageEmpty4}{CardinalNumberXmpPagePatch4}
\pastebutton{CardinalNumberXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{c3 := 3 :: CardinalNumber\bound{c3 }}
\end{paste}\end{patch}

\begin{patch}{CardinalNumberXmpPagePatch5}
\begin{paste}{CardinalNumberXmpPageFull5}{CardinalNumberXmpPageEmpty5}
\pastebutton{CardinalNumberXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{A0 := Aleph 0\bound{A0 }}
\indentrel{3}\begin{verbatim}
   (5)  Aleph(0)
                                   Type: CardinalNumber
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{CardinalNumberXmpPageEmpty5}
\begin{paste}{CardinalNumberXmpPageEmpty5}{CardinalNumberXmpPagePatch5}
\pastebutton{CardinalNumberXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{A0 := Aleph 0\bound{A0 }}
\end{paste}\end{patch}

\begin{patch}{CardinalNumberXmpPagePatch6}
\begin{paste}{CardinalNumberXmpPageFull6}{CardinalNumberXmpPageEmpty6}
\pastebutton{CardinalNumberXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{A1 := Aleph 1\bound{A1 }}
\indentrel{3}\begin{verbatim}
   (6)  Aleph(1)
                                  Type: CardinalNumber
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CardinalNumberXmpPageEmpty6}
\begin{paste}{CardinalNumberXmpPageEmpty6}{CardinalNumberXmpPagePatch6}
\pastebutton{CardinalNumberXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{A1 := Aleph 1\bound{A1 }}
\end{paste}\end{patch}

\begin{patch}{CardinalNumberXmpPagePatch7}
\begin{paste}{CardinalNumberXmpPageFull7}{CardinalNumberXmpPageEmpty7}
\pastebutton{CardinalNumberXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{finite? c2\free{c2 }}
\indentrel{3}\begin{verbatim}
   (7)  true
                                         Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CardinalNumberXmpPageEmpty7}
\begin{paste}{CardinalNumberXmpPageEmpty7}{CardinalNumberXmpPagePatch7}
\pastebutton{CardinalNumberXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{finite? c2\free{c2 }}
\end{paste}\end{patch}

\begin{patch}{CardinalNumberXmpPagePatch8}
\begin{paste}{CardinalNumberXmpPageFull8}{CardinalNumberXmpPageEmpty8}
\pastebutton{CardinalNumberXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{finite? A0\free{A0 }}
\indentrel{3}\begin{verbatim}
   (8)  false
                                         Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CardinalNumberXmpPageEmpty8}
```

```
\begin{paste}{CardinalNumberXmpPageEmpty8}{CardinalNumberXmpPagePatch8}
\pastebutton{CardinalNumberXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{finite? A0\free{A0 }}
\end{paste}\end{patch}

\begin{patch}{CardinalNumberXmpPagePatch9}
\begin{paste}{CardinalNumberXmpPageFull9}{CardinalNumberXmpPageEmpty9}
\pastebutton{CardinalNumberXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{countable? c2\free{c2 }}
\indentrel{3}\begin{verbatim}
   (9)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CardinalNumberXmpPageEmpty9}
\begin{paste}{CardinalNumberXmpPageEmpty9}{CardinalNumberXmpPagePatch9}
\pastebutton{CardinalNumberXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{countable? c2\free{c2 }}
\end{paste}\end{patch}

\begin{patch}{CardinalNumberXmpPagePatch10}
\begin{paste}{CardinalNumberXmpPageFull10}{CardinalNumberXmpPageEmpty10}
\pastebutton{CardinalNumberXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{countable? A0\free{A0 }}
\indentrel{3}\begin{verbatim}
   (10)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CardinalNumberXmpPageEmpty10}
\begin{paste}{CardinalNumberXmpPageEmpty10}{CardinalNumberXmpPagePatch10}
\pastebutton{CardinalNumberXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{countable? A0\free{A0 }}
\end{paste}\end{patch}

\begin{patch}{CardinalNumberXmpPagePatch11}
\begin{paste}{CardinalNumberXmpPageFull11}{CardinalNumberXmpPageEmpty11}
\pastebutton{CardinalNumberXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{countable? A1\free{A1 }}
\indentrel{3}\begin{verbatim}
   (11)  false
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CardinalNumberXmpPageEmpty11}
\begin{paste}{CardinalNumberXmpPageEmpty11}{CardinalNumberXmpPagePatch11}
\pastebutton{CardinalNumberXmpPageEmpty11}{\showpaste}
```

```
\tab{5}\spadcommand{countable? A1\free{A1 }}
\end{paste}\end{patch}

\begin{patch}{CardinalNumberXmpPagePatch12}
\begin{paste}{CardinalNumberXmpPageFull12}{CardinalNumberXmpPageEmpty12}
\pastebutton{CardinalNumberXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{[c2 + c2, c2 + A1]\free{c2 A1 }}
\indentrel{3}\begin{verbatim}
   (12)  [4,Aleph(1)]
                              Type: List CardinalNumber
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CardinalNumberXmpPageEmpty12}
\begin{paste}{CardinalNumberXmpPageEmpty12}{CardinalNumberXmpPagePatch12}
\pastebutton{CardinalNumberXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{[c2 + c2, c2 + A1]\free{c2 A1 }}
\end{paste}\end{patch}

\begin{patch}{CardinalNumberXmpPagePatch13}
\begin{paste}{CardinalNumberXmpPageFull13}{CardinalNumberXmpPageEmpty13}
\pastebutton{CardinalNumberXmpPageFull13}{\hidepaste}
\tab{5}\spadcommand{
[c0*c2, c1*c2, c2*c2, c0*A1, c1*A1, c2*A1, A0*A1]\free{c0 c1 c2 A0 A1 }}
\indentrel{3}\begin{verbatim}
   (13)  [0,2,4,0,Aleph(1),Aleph(1),Aleph(1)]
                              Type: List CardinalNumber
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CardinalNumberXmpPageEmpty13}
\begin{paste}{CardinalNumberXmpPageEmpty13}{CardinalNumberXmpPagePatch13}
\pastebutton{CardinalNumberXmpPageEmpty13}{\showpaste}
\tab{5}\spadcommand{
[c0*c2, c1*c2, c2*c2, c0*A1, c1*A1, c2*A1, A0*A1]\free{c0 c1 c2 A0 A1 }}
\end{paste}\end{patch}

\begin{patch}{CardinalNumberXmpPagePatch14}
\begin{paste}{CardinalNumberXmpPageFull14}{CardinalNumberXmpPageEmpty14}
\pastebutton{CardinalNumberXmpPageFull14}{\hidepaste}
\tab{5}\spadcommand{
[c2**c0, c2**c1, c2**c2, A1**c0, A1**c1, A1**c2]\free{c0 c1 c2 A1 }}
\indentrel{3}\begin{verbatim}
   (14)  [1,2,4,1,Aleph(1),Aleph(1)]
                              Type: List CardinalNumber
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CardinalNumberXmpPageEmpty14}
\begin{paste}{CardinalNumberXmpPageEmpty14}{CardinalNumberXmpPagePatch14}
```

```
\pastebutton{CardinalNumberXmpPageEmpty14}{\showpaste}
\tab{5}\spadcommand{
[c2**c0, c2**c1, c2**c2, A1**c0, A1**c1, A1**c2]\free{c0 c1 c2 A1 }}
\end{paste}\end{patch}

\begin{patch}{CardinalNumberXmpPagePatch15}
\begin{paste}{CardinalNumberXmpPageFull15}{CardinalNumberXmpPageEmpty15}
\pastebutton{CardinalNumberXmpPageFull15}{\hidepaste}
\tab{5}\spadcommand{
[c2-c1, c2-c2, c2-c3, A1-c2, A1-A0, A1-A1]\free{c1 c2 c3 A0 A1 }}
\indentrel{3}\begin{verbatim}
   (15)  [1,0,"failed",Aleph(1),Aleph(1),"failed"]
              Type: List Union(CardinalNumber,"failed")
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CardinalNumberXmpPageEmpty15}
\begin{paste}{CardinalNumberXmpPageEmpty15}{CardinalNumberXmpPagePatch15}
\pastebutton{CardinalNumberXmpPageEmpty15}{\showpaste}
\tab{5}\spadcommand{
[c2-c1, c2-c2, c2-c3, A1-c2, A1-A0, A1-A1]\free{c1 c2 c3 A0 A1 }}
\end{paste}\end{patch}

\begin{patch}{CardinalNumberXmpPagePatch16}
\begin{paste}{CardinalNumberXmpPageFull16}{CardinalNumberXmpPageEmpty16}
\pastebutton{CardinalNumberXmpPageFull16}{\hidepaste}
\tab{5}\spadcommand{generalizedContinuumHypothesisAssumed true\bound{GCH }}
\indentrel{3}\begin{verbatim}
   (16)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CardinalNumberXmpPageEmpty16}
\begin{paste}{CardinalNumberXmpPageEmpty16}{CardinalNumberXmpPagePatch16}
\pastebutton{CardinalNumberXmpPageEmpty16}{\showpaste}
\tab{5}\spadcommand{generalizedContinuumHypothesisAssumed true\bound{GCH }}
\end{paste}\end{patch}

\begin{patch}{CardinalNumberXmpPagePatch17}
\begin{paste}{CardinalNumberXmpPageFull17}{CardinalNumberXmpPageEmpty17}
\pastebutton{CardinalNumberXmpPageFull17}{\hidepaste}
\tab{5}\spadcommand{
[c0**A0, c1**A0, c2**A0, A0**A0, A0**A1, A1**A0, A1**A1]
\free{c0 c1 c2 A0 A1 GCH }}
\indentrel{3}\begin{verbatim}
   (17)
   [0,1,Aleph(1),Aleph(1),Aleph(2),Aleph(1),Aleph(2)]
                              Type: List CardinalNumber
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CardinalNumberXmpPageEmpty17}
\begin{paste}{CardinalNumberXmpPageEmpty17}{CardinalNumberXmpPagePatch17}
\pastebutton{CardinalNumberXmpPageEmpty17}{\showpaste}
\tab{5}\spadcommand{
[c0**A0, c1**A0, c2**A0, A0**A0, A0**A1, A1**A0, A1**A1]
\free{c0 c1 c2 A0 A1 GCH }}
\end{paste}\end{patch}

\begin{patch}{CardinalNumberXmpPagePatch18}
\begin{paste}{CardinalNumberXmpPageFull18}{CardinalNumberXmpPageEmpty18}
\pastebutton{CardinalNumberXmpPageFull18}{\hidepaste}
\tab{5}\spadcommand{a := Aleph 0\free{GCH }\bound{a }}
\indentrel{3}\begin{verbatim}
   (18)  Aleph(0)
                                   Type: CardinalNumber
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CardinalNumberXmpPageEmpty18}
\begin{paste}{CardinalNumberXmpPageEmpty18}{CardinalNumberXmpPagePatch18}
\pastebutton{CardinalNumberXmpPageEmpty18}{\showpaste}
\tab{5}\spadcommand{a := Aleph 0\free{GCH }\bound{a }}
\end{paste}\end{patch}

\begin{patch}{CardinalNumberXmpPagePatch19}
\begin{paste}{CardinalNumberXmpPageFull19}{CardinalNumberXmpPageEmpty19}
\pastebutton{CardinalNumberXmpPageFull19}{\hidepaste}
\tab{5}\spadcommand{c := 2**a\free{a }\bound{c }}
\indentrel{3}\begin{verbatim}
   (19)  Aleph(1)
                                   Type: CardinalNumber
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CardinalNumberXmpPageEmpty19}
\begin{paste}{CardinalNumberXmpPageEmpty19}{CardinalNumberXmpPagePatch19}
\pastebutton{CardinalNumberXmpPageEmpty19}{\showpaste}
\tab{5}\spadcommand{c := 2**a\free{a }\bound{c }}
\end{paste}\end{patch}

\begin{patch}{CardinalNumberXmpPagePatch20}
\begin{paste}{CardinalNumberXmpPageFull20}{CardinalNumberXmpPageEmpty20}
\pastebutton{CardinalNumberXmpPageFull20}{\hidepaste}
\tab{5}\spadcommand{f := 2**c\free{c }\bound{f }}
\indentrel{3}\begin{verbatim}
   (20)  Aleph(2)
                                   Type: CardinalNumber
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CardinalNumberXmpPageEmpty20}
\begin{paste}{CardinalNumberXmpPageEmpty20}{CardinalNumberXmpPagePatch20}
\pastebutton{CardinalNumberXmpPageEmpty20}{\showpaste}
\tab{5}\spadcommand{f := 2**c\free{c }\bound{f }}
\end{paste}\end{patch}
```

——–

3.13 carten.ht

CartesianTensor

— carten.ht —

```
\begin{page}{CartesianTensorXmpPage}{CartesianTensor}
\beginscroll

\spadtype{CartesianTensor(i0,dim,R)}
provides Cartesian tensors with
components belonging to a commutative ring \spad{R}.
Tensors can be described as a generalization of vectors and matrices.
This gives a concise {\it tensor algebra} for multilinear objects
supported by the \spadtype{CartesianTensor} domain.
You can form the inner or outer product of any two tensors and you can add
or subtract tensors with the same number of components.
Additionally, various forms of traces and transpositions are useful.

The \spadtype{CartesianTensor} constructor allows you to specify the
minimum index for subscripting.
In what follows we discuss in detail how to manipulate tensors.

\xtc{
Here we construct the domain of Cartesian tensors of dimension 2 over the
integers, with indices starting at 1.
}{
\spadpaste{CT := CARTEN(i0 := 1, 2, Integer) \bound{CT i0}}
}

\subsubsection{Forming tensors}
\labelSpace{2pc}
\xtc{
Scalars can be converted to tensors of rank zero.
}{
\spadpaste{t0: CT := 8 \free{CT}\bound{t0}}
```

```
}
\xtc{
}{
\spadpaste{rank t0 \free{t0}}
}
\xtc{
Vectors (mathematical direct products, rather than one dimensional array
structures) can be converted to tensors of rank one.
}{
\spadpaste{v: DirectProduct(2, Integer) := directProduct [3,4] \bound{v}}
}
\xtc{
}{
\spadpaste{Tv: CT := v \free{v CT}\bound{Tv}}
}
\xtc{
Matrices can be converted to tensors of rank two.
}{
\spadpaste{m: SquareMatrix(2, Integer) := matrix [[1,2],[4,5]] \bound{m}}
}
\xtc{
}{
\spadpaste{Tm: CT := m \bound{Tm}\free{CT m}}
}
\xtc{
}{
\spadpaste{n: SquareMatrix(2, Integer) := matrix [[2,3],[0,1]] \bound{n}}
}
\xtc{
}{
\spadpaste{Tn: CT := n \bound{Tn}\free{CT n}}
}
\xtc{
In general, a tensor of rank \spad{k} can be formed by making a
list of rank \spad{k-1} tensors or, alternatively, a \spad{k}-deep nested
list of lists.
}{
\spadpaste{t1: CT := [2, 3] \free{CT}\bound{t1}}
}
\xtc{
}{
\spadpaste{rank t1 \free{t1}}
}
\xtc{
}{
\spadpaste{t2: CT := [t1, t1] \free{CT t1}\bound{t2}}
}
\xtc{
}{
\spadpaste{t3: CT := [t2, t2] \free{CT t2}\bound{t3}}
```

```
}
\xtc{
}{
\spadpaste{tt: CT := [t3, t3]; tt := [tt, tt] \free{CT t3}\bound{tt}}
}
\xtc{
}{
\spadpaste{rank tt \free{tt}}
}

\subsubsection{Multiplication}
%\head{subsection}{Multiplication}{ugxCartenMult}

Given two tensors of rank \spad{k1} and \spad{k2}, the outer
{CartesianTensor}
\spadfunFrom{product}{CartesianTensor} forms a new tensor of rank
\spad{k1+k2}.
\xtc{
Here
\texht{$T_{mn}(i,j,k,l) = T_m(i,j) \  T_n(k,l).$}{%
\spad{Tmn(i,j,k,l) = Tm(i,j)*Tn(k,l).}}
}{
\spadpaste{Tmn := product(Tm, Tn)\free{Tm Tn}\bound{Tmn}}
}

The inner product (\spadfunFrom{contract}{CartesianTensor}) forms a tensor
{CartesianTensor}
of rank \spad{k1+k2-2}.
This product generalizes the vector dot product and matrix-vector product
by summing component products along two indices.
\xtc{
Here we sum along the second index of \texht{$T_m$}{\spad{Tm}} and the
first index of \texht{$T_v$}{\spad{Tv}}.
Here
\texht{$T_{mv} = \sum_{j=1}^{\hbox{\tiny\rm dim}} T_m(i,j) \  T_v(j)$}{%
\spad{Tmv(i) = sum Tm(i,j) * Tv(j) for j in 1..dim.}}
}{
\spadpaste{Tmv := contract(Tm,2,Tv,1)\free{Tm Tv}\bound{Tmv}}
}

The multiplication operator \spadopFrom{*}{CartesianTensor} is scalar
multiplication or an inner product depending on the ranks of the
arguments.
\xtc{
If either argument is rank zero it is treated as scalar multiplication.
Otherwise, \spad{a*b} is the inner product summing the last index of
\spad{a} with the first index of \spad{b}.
}{
\spadpaste{Tm*Tv \free{Tm Tv}}
}
```

```
\xtc{
This definition is consistent with the inner product on matrices
and vectors.
}{
\spadpaste{Tmv = m * v \free{Tmv m v}}
}

\subsubsection{Selecting Components}
%\head{subsection}{Selecting Components}{ugxCartenSelect}

\labelSpace{2pc}
\xtc{
For tensors of low rank (that is, four or less), components can be selected
by applying the tensor to its indices.
}{
\spadpaste{t0()      \free{t0}}
}
\xtc{
}{
\spadpaste{t1(1+1)   \free{t1}}
}
\xtc{
}{
\spadpaste{t2(2,1)   \free{t2}}
}
\xtc{
}{
\spadpaste{t3(2,1,2) \free{t3}}
}
\xtc{
}{
\spadpaste{Tmn(2,1,2,1) \free{Tmn}}
}
\xtc{
A general indexing mechanism is provided for a list of indices.
}{
\spadpaste{t0[]        \free{t0}}
}
\xtc{
}{
\spadpaste{t1[2]        \free{t1}}
}
\xtc{
}{
\spadpaste{t2[2,1]      \free{t2}}
}
\xtc{
The general mechanism works for tensors of arbitrary rank, but is
somewhat less efficient since the intermediate index list must be created.
}{
```

```
\spadpaste{t3[2,1,2]    \free{t3}}
}
\xtc{
}{
\spadpaste{Tmn[2,1,2,1] \free{Tmn}}
}

\subsubsection{Contraction}
%\head{subsection}{Contraction}{ugxCartenContract}

A ``contraction'' between two tensors is an inner product, as we have
seen above.
You can also contract a pair of indices of a single tensor.
This corresponds to a ``trace'' in linear algebra.
The expression \spad{contract(t,k1,k2)} forms a new tensor by summing the
diagonal given by indices in position \spad{k1} and \spad{k2}.
\xtc{
This is the tensor given by
\texht{$xT_{mn} = \sum_{k=1}^{\hbox{\tiny\rm dim}} T_{mn}(k,k,i,j).$}{%
\spad{cTmn(i,j) = sum Tmn(k,k,i,j) for k in 1..dim.}}
}{
\spadpaste{cTmn := contract(Tmn,1,2) \free{Tmn}\bound{cTmn}}
}
\xtc{
Since \spad{Tmn} is the outer product of matrix \spad{m} and matrix \spad{n},
the above is equivalent to this.
}{
\spadpaste{trace(m) * n \free{m n}}
}
\xtc{
In this and the next few examples,
we show all possible contractions of \spad{Tmn} and their matrix
algebra equivalents.
}{
\spadpaste{contract(Tmn,1,2) = trace(m) * n      \free{Tmn m n}}
}
\xtc{
}{
\spadpaste{contract(Tmn,1,3) = transpose(m) * n  \free{Tmn m n}}
}
\xtc{
}{
\spadpaste{contract(Tmn,1,4) = transpose(m) * transpose(n) \free{Tmn m n}}
}
\xtc{
}{
\spadpaste{contract(Tmn,2,3) = m * n             \free{Tmn m n}}
}
\xtc{
}{
```

```
\spadpaste{contract(Tmn,2,4) = m * transpose(n)  \free{Tmn m n}}
}
\xtc{
}{
\spadpaste{contract(Tmn,3,4) = trace(n) * m      \free{Tmn m n}}
}

\subsubsection{Transpositions}
%\head{subsection}{Transpositions}{ugxCartenTranspos}

You can exchange any desired pair of indices using the
\spadfunFrom{transpose}{CartesianTensor} operation.

\labelSpace{1pc}
\xtc{
Here the indices in positions one and three are exchanged, that is,
\texht{$tT_{mn}(i,j,k,l) = T_{mn}(k,j,i,l).$}{%
\spad{tTmn(i,j,k,l) = Tmn(k,j,i,l).}}
}{
\spadpaste{tTmn := transpose(Tmn,1,3) \free{tTmn}}
}
\xtc{
If no indices are specified, the first and last index are exchanged.
}{
\spadpaste{transpose Tmn \free{Tmn}}
}
\xtc{
This is consistent with the matrix transpose.
}{
\spadpaste{transpose Tm = transpose m \free{Tm m}}
}

If a more complicated reordering of the indices is required, then the
\spadfunFrom{reindex}{CartesianTensor} operation can be used.
This operation allows the indices to be arbitrarily permuted.
\xtc{
This defines
\texht{$rT_{mn}(i,j,k,l) = \allowbreak T_{mn}(i,l,j,k).$}{%
\spad{rTmn(i,j,k,l) = Tmn(i,l,j,k).}}
}{
\spadpaste{rTmn := reindex(Tmn, [1,4,2,3]) \free{Tmn}\bound{rTmn}}
}

\subsubsection{Arithmetic}
%\head{subsection}{Arithmetic}{ugxCartenArith}

\labelSpace{2pc}
\xtc{
Tensors of equal rank can be added or subtracted so arithmetic
expressions can be used to produce new tensors.
```

```
}{
\spadpaste{tt := transpose(Tm)*Tn - Tn*transpose(Tm) \free{Tm Tn}\bound{tt2}}
}
\xtc{
}{
\spadpaste{Tv*(tt+Tn) \free{tt2 Tv Tn}}
}
\xtc{
}{
\spadpaste{reindex(product(Tn,Tn),[4,3,2,1])+3*Tn*product(Tm,Tm) \free{Tm Tn}}
}

\subsubsection{Specific Tensors}
%\head{subsection}{Specific Tensors}{ugxCartenSpecific}

Two specific tensors have properties which depend only on the
dimension.

\labelSpace{1pc}
\xtc{
The Kronecker delta satisfies
\texht{}{%
\begin{verbatim}
             +-
             |   1  if i  = j
delta(i,j) = |
             |   0  if i ^= j
             +-
\end{verbatim}
}%
}{
\spadpaste{delta:  CT := kroneckerDelta()   \free{CT}\bound{delta}}
}
\xtc{
This can be used to reindex via contraction.
}{
\spadpaste{contract(Tmn, 2, delta, 1) = reindex(Tmn, [1,3,4,2])
\free{Tmn delta}}
}

\xtc{
The Levi Civita symbol determines the sign of a permutation of indices.
}{
\spadpaste{epsilon:CT := leviCivitaSymbol() \free{CT}\bound{epsilon}}
}
Here we have:
\texht{}{%
\begin{verbatim}
epsilon(i1,...,idim)
     = +1  if i1,...,idim is an even permutation of i0,...,i0+dim-1
```

```
     = -1  if i1,...,idim is an  odd permutation of i0,...,i0+dim-1
     =  0  if i1,...,idim is not   a permutation of i0,...,i0+dim-1
\end{verbatim}
}
\xtc{
This property can be used to form determinants.
}{
\spadpaste{contract(epsilon*Tm*epsilon, 1,2) = 2 * determinant m \free{epsilon Tm m}}
}

\subsubsection{Properties of the CartesianTensor domain}
%\head{subsection}{Properties of the CartesianTensor domain}{ugxCartenProp}

\spadtype{GradedModule(R,E)} denotes ``\spad{E}-graded \spad{R}-module'',
that is, a collection of \spad{R}-modules indexed by an abelian monoid
\spad{E.}
An element \spad{g} of \spad{G[s]} for some specific \spad{s}
in \spad{E} is said to be an element of \spad{G} with
\spadfunFrom{degree}{GradedModule} \spad{s}.
Sums are defined in each module \spad{G[s]} so two elements of
\spad{G} can be added if they have the same degree.
Morphisms can be defined and composed by degree to give the
mathematical category of graded modules.

\spadtype{GradedAlgebra(R,E)} denotes ``\spad{E}-graded \spad{R}-algebra.''
{CartesianTensor}
A graded algebra is a graded module together with a degree preserving
\spad{R}-bilinear map, called the \spadfunFrom{product}{GradedAlgebra}.

\begin{verbatim}
degree(product(a,b))= degree(a) + degree(b)

product(r*a,b) = product(a,r*b) = r*product(a,b)
product(a1+a2,b) = product(a1,b) + product(a2,b)
product(a,b1+b2) = product(a,b1) + product(a,b2)
product(a,product(b,c)) = product(product(a,b),c)
\end{verbatim}

The domain \spadtype{CartesianTensor(i0, dim, R)} belongs
to the category \spadtype{GradedAlgebra(R, NonNegativeInteger)}.
The non-negative integer \spadfunFrom{degree}{GradedAlgebra}
is the tensor rank
and the graded algebra \spadfunFrom{product}{GradedAlgebra}
is the tensor outer product.
The graded module addition captures the notion that
only tensors of equal rank can be added.

If \spad{V} is a vector space of dimension \spad{dim} over \spad{R},
then the tensor module \spad{T[k](V)} is defined as
\begin{verbatim}
```

```
T[0](V) = R
T[k](V) = T[k-1](V) * V
\end{verbatim}
where \spadop{*} denotes the \spad{R}-module tensor
\spadfunFrom{product}{GradedAlgebra}.
\spadtype{CartesianTensor(i0,dim,R)} is the graded algebra in which
the degree \spad{k} module is \spad{T[k](V)}.

\subsubsection{Tensor Calculus}
%\head{subsection}{Tensor Calculus}{ugxCartenCalculus}

It should be noted here that often tensors are used in the context of
tensor-valued manifold maps.  This leads to the notion of covariant
and contravariant bases with tensor component functions transforming
in specific ways under a change of coordinates on the manifold.  This
is no more directly supported by the \spadtype{CartesianTensor} domain
than it is by the \spadtype{Vector} domain.  However, it is possible
to have the components implicitly represent component maps by choosing
a polynomial or expression type for the components.  In this case, it
is up to the user to satisfy any constraints which arise on the basis
of this interpretation.
\endscroll
\autobuttons
\end{page}

\begin{patch}{CartesianTensorXmpPagePatch1}
\begin{paste}{CartesianTensorXmpPageFull1}{CartesianTensorXmpPageEmpty1}
\pastebutton{CartesianTensorXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{CT := CARTEN(i0 := 1, 2, Integer)\bound{CT i0 }}
\indentrel{3}\begin{verbatim}
   (1)  CartesianTensor(1,2,Integer)
                                          Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPageEmpty1}
\begin{paste}{CartesianTensorXmpPageEmpty1}{CartesianTensorXmpPagePatch1}
\pastebutton{CartesianTensorXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{CT := CARTEN(i0 := 1, 2, Integer)\bound{CT i0 }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch2}
\begin{paste}{CartesianTensorXmpPageFull2}{CartesianTensorXmpPageEmpty2}
\pastebutton{CartesianTensorXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{t0: CT := 8\free{CT }\bound{t0 }}
\indentrel{3}\begin{verbatim}
   (2)  8
                     Type: CartesianTensor(1,2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{CartesianTensorXmpPageEmpty2}
\begin{paste}{CartesianTensorXmpPageEmpty2}{CartesianTensorXmpPagePatch2}
\pastebutton{CartesianTensorXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{t0: CT := 8\free{CT }\bound{t0 }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch3}
\begin{paste}{CartesianTensorXmpPageFull3}{CartesianTensorXmpPageEmpty3}
\pastebutton{CartesianTensorXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{rank t0\free{t0 }}
\indentrel{3}\begin{verbatim}
   (3)  0
                              Type: NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPageEmpty3}
\begin{paste}{CartesianTensorXmpPageEmpty3}{CartesianTensorXmpPagePatch3}
\pastebutton{CartesianTensorXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{rank t0\free{t0 }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch4}
\begin{paste}{CartesianTensorXmpPageFull4}{CartesianTensorXmpPageEmpty4}
\pastebutton{CartesianTensorXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{
v: DirectProduct(2, Integer) := directProduct [3,4]\bound{v }}
\indentrel{3}\begin{verbatim}
   (4)  [3,4]
                        Type: DirectProduct(2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPageEmpty4}
\begin{paste}{CartesianTensorXmpPageEmpty4}{CartesianTensorXmpPagePatch4}
\pastebutton{CartesianTensorXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{
v: DirectProduct(2, Integer) := directProduct [3,4]\bound{v }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch5}
\begin{paste}{CartesianTensorXmpPageFull5}{CartesianTensorXmpPageEmpty5}
\pastebutton{CartesianTensorXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{Tv: CT := v\free{v CT }\bound{Tv }}
\indentrel{3}\begin{verbatim}
   (5)  [3,4]
                    Type: CartesianTensor(1,2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{CartesianTensorXmpPageEmpty5}
\begin{paste}{CartesianTensorXmpPageEmpty5}{CartesianTensorXmpPagePatch5}
\pastebutton{CartesianTensorXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{Tv: CT := v\free{v CT }\bound{Tv }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch6}
\begin{paste}{CartesianTensorXmpPageFull6}{CartesianTensorXmpPageEmpty6}
\pastebutton{CartesianTensorXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{
m: SquareMatrix(2, Integer) := matrix [[1,2],[4,5]]\bound{m }}
\indentrel{3}\begin{verbatim}
        1  2
   (6)
        4  5
                          Type: SquareMatrix(2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPageEmpty6}
\begin{paste}{CartesianTensorXmpPageEmpty6}{CartesianTensorXmpPagePatch6}
\pastebutton{CartesianTensorXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{
m: SquareMatrix(2, Integer) := matrix [[1,2],[4,5]]\bound{m }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch7}
\begin{paste}{CartesianTensorXmpPageFull7}{CartesianTensorXmpPageEmpty7}
\pastebutton{CartesianTensorXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{Tm: CT := m\bound{Tm }\free{CT m }}
\indentrel{3}\begin{verbatim}
        1  2
   (7)
        4  5
                     Type: CartesianTensor(1,2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPageEmpty7}
\begin{paste}{CartesianTensorXmpPageEmpty7}{CartesianTensorXmpPagePatch7}
\pastebutton{CartesianTensorXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{Tm: CT := m\bound{Tm }\free{CT m }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch8}
\begin{paste}{CartesianTensorXmpPageFull8}{CartesianTensorXmpPageEmpty8}
\pastebutton{CartesianTensorXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{
n: SquareMatrix(2, Integer) := matrix [[2,3],[0,1]]\bound{n }}
```

```
\indentrel{3}\begin{verbatim}
        2  3
   (8)
        0  1
                          Type: SquareMatrix(2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPageEmpty8}
\begin{paste}{CartesianTensorXmpPageEmpty8}{CartesianTensorXmpPagePatch8}
\pastebutton{CartesianTensorXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{
n: SquareMatrix(2, Integer) := matrix [[2,3],[0,1]]\bound{n }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch9}
\begin{paste}{CartesianTensorXmpPageFull9}{CartesianTensorXmpPageEmpty9}
\pastebutton{CartesianTensorXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{Tn: CT := n\bound{Tn }\free{CT n }}
\indentrel{3}\begin{verbatim}
        2  3
   (9)
        0  1
                     Type: CartesianTensor(1,2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPageEmpty9}
\begin{paste}{CartesianTensorXmpPageEmpty9}{CartesianTensorXmpPagePatch9}
\pastebutton{CartesianTensorXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{Tn: CT := n\bound{Tn }\free{CT n }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch10}
\begin{paste}{CartesianTensorXmpPageFull10}{CartesianTensorXmpPageEmpty10}
\pastebutton{CartesianTensorXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{t1: CT := [2, 3]\free{CT }\bound{t1 }}
\indentrel{3}\begin{verbatim}
   (10)  [2,3]
                     Type: CartesianTensor(1,2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPageEmpty10}
\begin{paste}{CartesianTensorXmpPageEmpty10}{CartesianTensorXmpPagePatch10}
\pastebutton{CartesianTensorXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{t1: CT := [2, 3]\free{CT }\bound{t1 }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch11}
```

```
\begin{paste}{CartesianTensorXmpPageFull11}{CartesianTensorXmpPageEmpty11}
\pastebutton{CartesianTensorXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{rank t1\free{t1 }}
\indentrel{3}\begin{verbatim}
   (11)  1
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPageEmpty11}
\begin{paste}{CartesianTensorXmpPageEmpty11}{CartesianTensorXmpPagePatch11}
\pastebutton{CartesianTensorXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{rank t1\free{t1 }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch12}
\begin{paste}{CartesianTensorXmpPageFull12}{CartesianTensorXmpPageEmpty12}
\pastebutton{CartesianTensorXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{t2: CT := [t1, t1]\free{CT t1 }\bound{t2 }}
\indentrel{3}\begin{verbatim}
         2  3
   (12)
         2  3
                     Type: CartesianTensor(1,2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPageEmpty12}
\begin{paste}{CartesianTensorXmpPageEmpty12}{CartesianTensorXmpPagePatch12}
\pastebutton{CartesianTensorXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{t2: CT := [t1, t1]\free{CT t1 }\bound{t2 }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch13}
\begin{paste}{CartesianTensorXmpPageFull13}{CartesianTensorXmpPageEmpty13}
\pastebutton{CartesianTensorXmpPageFull13}{\hidepaste}
\tab{5}\spadcommand{t3: CT := [t2, t2]\free{CT t2 }\bound{t3 }}
\indentrel{3}\begin{verbatim}
          2  3 2  3
   (13)  [    ,    ]
          2  3 2  3
                     Type: CartesianTensor(1,2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPageEmpty13}
\begin{paste}{CartesianTensorXmpPageEmpty13}{CartesianTensorXmpPagePatch13}
\pastebutton{CartesianTensorXmpPageEmpty13}{\showpaste}
\tab{5}\spadcommand{t3: CT := [t2, t2]\free{CT t2 }\bound{t3 }}
\end{paste}\end{patch}
```

```
\begin{patch}{CartesianTensorXmpPagePatch14}
\begin{paste}{CartesianTensorXmpPageFull14}{CartesianTensorXmpPageEmpty14}
\pastebutton{CartesianTensorXmpPageFull14}{\hidepaste}
\tab{5}\spadcommand{
tt: CT := [t3, t3]; tt := [tt, tt]\free{CT t3 }\bound{tt }}
\indentrel{3}\begin{verbatim}
          2  3  2  3 2  3  2  3

          2  3  2  3 2  3  2  3
   (14)  [              ,              ]
          2  3  2  3 2  3  2  3

          2  3  2  3 2  3  2  3
                     Type: CartesianTensor(1,2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPageEmpty14}
\begin{paste}{CartesianTensorXmpPageEmpty14}{CartesianTensorXmpPagePatch14}
\pastebutton{CartesianTensorXmpPageEmpty14}{\showpaste}
\tab{5}\spadcommand{
tt: CT := [t3, t3]; tt := [tt, tt]\free{CT t3 }\bound{tt }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch15}
\begin{paste}{CartesianTensorXmpPageFull15}{CartesianTensorXmpPageEmpty15}
\pastebutton{CartesianTensorXmpPageFull15}{\hidepaste}
\tab{5}\spadcommand{rank tt\free{tt }}
\indentrel{3}\begin{verbatim}
   (15)  5
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPageEmpty15}
\begin{paste}{CartesianTensorXmpPageEmpty15}{CartesianTensorXmpPagePatch15}
\pastebutton{CartesianTensorXmpPageEmpty15}{\showpaste}
\tab{5}\spadcommand{rank tt\free{tt }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch16}
\begin{paste}{CartesianTensorXmpPageFull16}{CartesianTensorXmpPageEmpty16}
\pastebutton{CartesianTensorXmpPageFull16}{\hidepaste}
\tab{5}\spadcommand{Tmn := product(Tm, Tn)\free{Tm Tn }\bound{Tmn }}
\indentrel{3}\begin{verbatim}
         2  3    4  6

         0  1    0  2
   (16)
```

```
        8  12  10  15

        0  4   0   5
                    Type: CartesianTensor(1,2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPageEmpty16}
\begin{paste}{CartesianTensorXmpPageEmpty16}{CartesianTensorXmpPagePatch16}
\pastebutton{CartesianTensorXmpPageEmpty16}{\showpaste}
\tab{5}\spadcommand{Tmn := product(Tm, Tn)\free{Tm Tn }\bound{Tmn }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch17}
\begin{paste}{CartesianTensorXmpPageFull17}{CartesianTensorXmpPageEmpty17}
\pastebutton{CartesianTensorXmpPageFull17}{\hidepaste}
\tab{5}\spadcommand{Tmv := contract(Tm,2,Tv,1)\free{Tm Tv }\bound{Tmv }}
\indentrel{3}\begin{verbatim}
   (17)  [11,32]
                    Type: CartesianTensor(1,2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPageEmpty17}
\begin{paste}{CartesianTensorXmpPageEmpty17}{CartesianTensorXmpPagePatch17}
\pastebutton{CartesianTensorXmpPageEmpty17}{\showpaste}
\tab{5}\spadcommand{Tmv := contract(Tm,2,Tv,1)\free{Tm Tv }\bound{Tmv }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch18}
\begin{paste}{CartesianTensorXmpPageFull18}{CartesianTensorXmpPageEmpty18}
\pastebutton{CartesianTensorXmpPageFull18}{\hidepaste}
\tab{5}\spadcommand{Tm*Tv\free{Tm Tv }}
\indentrel{3}\begin{verbatim}
   (18)  [11,32]
                    Type: CartesianTensor(1,2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPageEmpty18}
\begin{paste}{CartesianTensorXmpPageEmpty18}{CartesianTensorXmpPagePatch18}
\pastebutton{CartesianTensorXmpPageEmpty18}{\showpaste}
\tab{5}\spadcommand{Tm*Tv\free{Tm Tv }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch19}
\begin{paste}{CartesianTensorXmpPageFull19}{CartesianTensorXmpPageEmpty19}
\pastebutton{CartesianTensorXmpPageFull19}{\hidepaste}
\tab{5}\spadcommand{Tmv = m * v\free{Tmv m v }}
\indentrel{3}\begin{verbatim}
```

```
   (19)  [11,32]= [11,32]
            Type: Equation CartesianTensor(1,2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPageEmpty19}
\begin{paste}{CartesianTensorXmpPageEmpty19}{CartesianTensorXmpPagePatch19}
\pastebutton{CartesianTensorXmpPageEmpty19}{\showpaste}
\tab{5}\spadcommand{Tmv = m * v\free{Tmv m v }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch20}
\begin{paste}{CartesianTensorXmpPageFull20}{CartesianTensorXmpPageEmpty20}
\pastebutton{CartesianTensorXmpPageFull20}{\hidepaste}
\tab{5}\spadcommand{t0()\free{t0 }}
\indentrel{3}\begin{verbatim}
   (20)  8
                                    Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPageEmpty20}
\begin{paste}{CartesianTensorXmpPageEmpty20}{CartesianTensorXmpPagePatch20}
\pastebutton{CartesianTensorXmpPageEmpty20}{\showpaste}
\tab{5}\spadcommand{t0()\free{t0 }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch21}
\begin{paste}{CartesianTensorXmpPageFull21}{CartesianTensorXmpPageEmpty21}
\pastebutton{CartesianTensorXmpPageFull21}{\hidepaste}
\tab{5}\spadcommand{t1(1+1)\free{t1 }}
\indentrel{3}\begin{verbatim}
   (21)  3
                                    Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPageEmpty21}
\begin{paste}{CartesianTensorXmpPageEmpty21}{CartesianTensorXmpPagePatch21}
\pastebutton{CartesianTensorXmpPageEmpty21}{\showpaste}
\tab{5}\spadcommand{t1(1+1)\free{t1 }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch22}
\begin{paste}{CartesianTensorXmpPageFull22}{CartesianTensorXmpPageEmpty22}
\pastebutton{CartesianTensorXmpPageFull22}{\hidepaste}
\tab{5}\spadcommand{t2(2,1)\free{t2 }}
\indentrel{3}\begin{verbatim}
   (22)  2
                                    Type: PositiveInteger
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPageEmpty22}
\begin{paste}{CartesianTensorXmpPageEmpty22}{CartesianTensorXmpPagePatch22}
\pastebutton{CartesianTensorXmpPageEmpty22}{\showpaste}
\tab{5}\spadcommand{t2(2,1)\free{t2 }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch23}
\begin{paste}{CartesianTensorXmpPageFull23}{CartesianTensorXmpPageEmpty23}
\pastebutton{CartesianTensorXmpPageFull23}{\hidepaste}
\tab{5}\spadcommand{t3(2,1,2)\free{t3 }}
\indentrel{3}\begin{verbatim}
   (23)  3
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPageEmpty23}
\begin{paste}{CartesianTensorXmpPageEmpty23}{CartesianTensorXmpPagePatch23}
\pastebutton{CartesianTensorXmpPageEmpty23}{\showpaste}
\tab{5}\spadcommand{t3(2,1,2)\free{t3 }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch24}
\begin{paste}{CartesianTensorXmpPageFull24}{CartesianTensorXmpPageEmpty24}
\pastebutton{CartesianTensorXmpPageFull24}{\hidepaste}
\tab{5}\spadcommand{Tmn(2,1,2,1)\free{Tmn }}
\indentrel{3}\begin{verbatim}
   (24)  0
                               Type: NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPageEmpty24}
\begin{paste}{CartesianTensorXmpPageEmpty24}{CartesianTensorXmpPagePatch24}
\pastebutton{CartesianTensorXmpPageEmpty24}{\showpaste}
\tab{5}\spadcommand{Tmn(2,1,2,1)\free{Tmn }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch25}
\begin{paste}{CartesianTensorXmpPageFull25}{CartesianTensorXmpPageEmpty25}
\pastebutton{CartesianTensorXmpPageFull25}{\hidepaste}
\tab{5}\spadcommand{t0[]\free{t0 }}
\indentrel{3}\begin{verbatim}
   (25)  8
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{CartesianTensorXmpPageEmpty25}
\begin{paste}{CartesianTensorXmpPageEmpty25}{CartesianTensorXmpPagePatch25}
\pastebutton{CartesianTensorXmpPageEmpty25}{\showpaste}
\tab{5}\spadcommand{t0[]\free{t0 }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch26}
\begin{paste}{CartesianTensorXmpPageFull26}{CartesianTensorXmpPageEmpty26}
\pastebutton{CartesianTensorXmpPageFull26}{\hidepaste}
\tab{5}\spadcommand{t1[2]\free{t1 }}
\indentrel{3}\begin{verbatim}
   (26)  3
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPageEmpty26}
\begin{paste}{CartesianTensorXmpPageEmpty26}{CartesianTensorXmpPagePatch26}
\pastebutton{CartesianTensorXmpPageEmpty26}{\showpaste}
\tab{5}\spadcommand{t1[2]\free{t1 }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch27}
\begin{paste}{CartesianTensorXmpPageFull27}{CartesianTensorXmpPageEmpty27}
\pastebutton{CartesianTensorXmpPageFull27}{\hidepaste}
\tab{5}\spadcommand{t2[2,1]\free{t2 }}
\indentrel{3}\begin{verbatim}
   (27)  2
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPageEmpty27}
\begin{paste}{CartesianTensorXmpPageEmpty27}{CartesianTensorXmpPagePatch27}
\pastebutton{CartesianTensorXmpPageEmpty27}{\showpaste}
\tab{5}\spadcommand{t2[2,1]\free{t2 }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch28}
\begin{paste}{CartesianTensorXmpPageFull28}{CartesianTensorXmpPageEmpty28}
\pastebutton{CartesianTensorXmpPageFull28}{\hidepaste}
\tab{5}\spadcommand{t3[2,1,2]\free{t3 }}
\indentrel{3}\begin{verbatim}
   (28)  3
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPageEmpty28}
```

```
\begin{paste}{CartesianTensorXmpPageEmpty28}{CartesianTensorXmpPagePatch28}
\pastebutton{CartesianTensorXmpPageEmpty28}{\showpaste}
\tab{5}\spadcommand{t3[2,1,2]\free{t3 }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch29}
\begin{paste}{CartesianTensorXmpPageFull29}{CartesianTensorXmpPageEmpty29}
\pastebutton{CartesianTensorXmpPageFull29}{\hidepaste}
\tab{5}\spadcommand{Tmn[2,1,2,1]\free{Tmn }}
\indentrel{3}\begin{verbatim}
   (29)  0
                               Type: NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPageEmpty29}
\begin{paste}{CartesianTensorXmpPageEmpty29}{CartesianTensorXmpPagePatch29}
\pastebutton{CartesianTensorXmpPageEmpty29}{\showpaste}
\tab{5}\spadcommand{Tmn[2,1,2,1]\free{Tmn }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch30}
\begin{paste}{CartesianTensorXmpPageFull30}{CartesianTensorXmpPageEmpty30}
\pastebutton{CartesianTensorXmpPageFull30}{\hidepaste}
\tab{5}\spadcommand{cTmn := contract(Tmn,1,2)\free{Tmn }\bound{cTmn }}
\indentrel{3}\begin{verbatim}
         12  18
   (30)
         0   6
                     Type: CartesianTensor(1,2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPageEmpty30}
\begin{paste}{CartesianTensorXmpPageEmpty30}{CartesianTensorXmpPagePatch30}
\pastebutton{CartesianTensorXmpPageEmpty30}{\showpaste}
\tab{5}\spadcommand{cTmn := contract(Tmn,1,2)\free{Tmn }\bound{cTmn }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch31}
\begin{paste}{CartesianTensorXmpPageFull31}{CartesianTensorXmpPageEmpty31}
\pastebutton{CartesianTensorXmpPageFull31}{\hidepaste}
\tab{5}\spadcommand{trace(m) * n\free{m n }}
\indentrel{3}\begin{verbatim}
         12  18
   (31)
         0   6
                          Type: SquareMatrix(2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{CartesianTensorXmpPageEmpty31}
\begin{paste}{CartesianTensorXmpPageEmpty31}{CartesianTensorXmpPagePatch31}
\pastebutton{CartesianTensorXmpPageEmpty31}{\showpaste}
\tab{5}\spadcommand{trace(m) * n\free{m n }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch32}
\begin{paste}{CartesianTensorXmpPageFull32}{CartesianTensorXmpPageEmpty32}
\pastebutton{CartesianTensorXmpPageFull32}{\hidepaste}
\tab{5}\spadcommand{contract(Tmn,1,2) = trace(m) * n\free{Tmn m n }}
\indentrel{3}\begin{verbatim}
         12  18  12  18
   (32)        =
         0   6   0   6
            Type: Equation CartesianTensor(1,2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPageEmpty32}
\begin{paste}{CartesianTensorXmpPageEmpty32}{CartesianTensorXmpPagePatch32}
\pastebutton{CartesianTensorXmpPageEmpty32}{\showpaste}
\tab{5}\spadcommand{contract(Tmn,1,2) = trace(m) * n\free{Tmn m n }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch33}
\begin{paste}{CartesianTensorXmpPageFull33}{CartesianTensorXmpPageEmpty33}
\pastebutton{CartesianTensorXmpPageFull33}{\hidepaste}
\tab{5}\spadcommand{contract(Tmn,1,3) = transpose(m) * n\free{Tmn m n }}
\indentrel{3}\begin{verbatim}
         2  7   2  7
   (33)       =
         4  11  4  11
            Type: Equation CartesianTensor(1,2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPageEmpty33}
\begin{paste}{CartesianTensorXmpPageEmpty33}{CartesianTensorXmpPagePatch33}
\pastebutton{CartesianTensorXmpPageEmpty33}{\showpaste}
\tab{5}\spadcommand{contract(Tmn,1,3) = transpose(m) * n\free{Tmn m n }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch34}
\begin{paste}{CartesianTensorXmpPageFull34}{CartesianTensorXmpPageEmpty34}
\pastebutton{CartesianTensorXmpPageFull34}{\hidepaste}
\tab{5}\spadcommand{
contract(Tmn,1,4) = transpose(m) * transpose(n)\free{Tmn m n }}
\indentrel{3}\begin{verbatim}
         14  4  14  4
```

```
   (34)        =
         19  5  19  5
            Type: Equation CartesianTensor(1,2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPageEmpty34}
\begin{paste}{CartesianTensorXmpPageEmpty34}{CartesianTensorXmpPagePatch34}
\pastebutton{CartesianTensorXmpPageEmpty34}{\showpaste}
\tab{5}\spadcommand{
contract(Tmn,1,4) = transpose(m) * transpose(n)\free{Tmn m n }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch35}
\begin{paste}{CartesianTensorXmpPageFull35}{CartesianTensorXmpPageEmpty35}
\pastebutton{CartesianTensorXmpPageFull35}{\hidepaste}
\tab{5}\spadcommand{contract(Tmn,2,3) = m * n\free{Tmn m n }}
\indentrel{3}\begin{verbatim}
         2  5   2  5
   (35)        =
         8  17  8  17
            Type: Equation CartesianTensor(1,2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPageEmpty35}
\begin{paste}{CartesianTensorXmpPageEmpty35}{CartesianTensorXmpPagePatch35}
\pastebutton{CartesianTensorXmpPageEmpty35}{\showpaste}
\tab{5}\spadcommand{contract(Tmn,2,3) = m * n\free{Tmn m n }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch36}
\begin{paste}{CartesianTensorXmpPageFull36}{CartesianTensorXmpPageEmpty36}
\pastebutton{CartesianTensorXmpPageFull36}{\hidepaste}
\tab{5}\spadcommand{contract(Tmn,2,4) = m * transpose(n)\free{Tmn m n }}
\indentrel{3}\begin{verbatim}
         8   2  8   2
   (36)        =
         23  5  23  5
            Type: Equation CartesianTensor(1,2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPageEmpty36}
\begin{paste}{CartesianTensorXmpPageEmpty36}{CartesianTensorXmpPagePatch36}
\pastebutton{CartesianTensorXmpPageEmpty36}{\showpaste}
\tab{5}\spadcommand{contract(Tmn,2,4) = m * transpose(n)\free{Tmn m n }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch37}
```

```
\begin{paste}{CartesianTensorXmpPageFull37}{CartesianTensorXmpPageEmpty37}
\pastebutton{CartesianTensorXmpPageFull37}{\hidepaste}
\tab{5}\spadcommand{contract(Tmn,3,4) = trace(n) * m\free{Tmn m n }}
\indentrel{3}\begin{verbatim}
         3   6   3   6
   (37)        =
         12  15  12  15
            Type: Equation CartesianTensor(1,2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPageEmpty37}
\begin{paste}{CartesianTensorXmpPageEmpty37}{CartesianTensorXmpPagePatch37}
\pastebutton{CartesianTensorXmpPageEmpty37}{\showpaste}
\tab{5}\spadcommand{contract(Tmn,3,4) = trace(n) * m\free{Tmn m n }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch38}
\begin{paste}{CartesianTensorXmpPageFull38}{CartesianTensorXmpPageEmpty38}
\pastebutton{CartesianTensorXmpPageFull38}{\hidepaste}
\tab{5}\spadcommand{tTmn := transpose(Tmn,1,3)\free{tTmn }}
\indentrel{3}\begin{verbatim}
         2  3   4   6

         8  12  10  15
   (38)
         0  1    0  2

         0  4    0  5
                     Type: CartesianTensor(1,2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPageEmpty38}
\begin{paste}{CartesianTensorXmpPageEmpty38}{CartesianTensorXmpPagePatch38}
\pastebutton{CartesianTensorXmpPageEmpty38}{\showpaste}
\tab{5}\spadcommand{tTmn := transpose(Tmn,1,3)\free{tTmn }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch39}
\begin{paste}{CartesianTensorXmpPageFull39}{CartesianTensorXmpPageEmpty39}
\pastebutton{CartesianTensorXmpPageFull39}{\hidepaste}
\tab{5}\spadcommand{transpose Tmn\free{Tmn }}
\indentrel{3}\begin{verbatim}
         2  8   4  10

         0  0   0  0
   (39)
         3  12  6  15
```

```
        1  4   2  5
                    Type: CartesianTensor(1,2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPageEmpty39}
\begin{paste}{CartesianTensorXmpPageEmpty39}{CartesianTensorXmpPagePatch39}
\pastebutton{CartesianTensorXmpPageEmpty39}{\showpaste}
\tab{5}\spadcommand{transpose Tmn\free{Tmn }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch40}
\begin{paste}{CartesianTensorXmpPageFull40}{CartesianTensorXmpPageEmpty40}
\pastebutton{CartesianTensorXmpPageFull40}{\hidepaste}
\tab{5}\spadcommand{transpose Tm = transpose m\free{Tm m }}
\indentrel{3}\begin{verbatim}
        1  4  1  4
   (40)      =
        2  5  2  5
           Type: Equation CartesianTensor(1,2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPageEmpty40}
\begin{paste}{CartesianTensorXmpPageEmpty40}{CartesianTensorXmpPagePatch40}
\pastebutton{CartesianTensorXmpPageEmpty40}{\showpaste}
\tab{5}\spadcommand{transpose Tm = transpose m\free{Tm m }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch41}
\begin{paste}{CartesianTensorXmpPageFull41}{CartesianTensorXmpPageEmpty41}
\pastebutton{CartesianTensorXmpPageFull41}{\hidepaste}
\tab{5}\spadcommand{rTmn := reindex(Tmn, [1,4,2,3])\free{Tmn }\bound{rTmn }}
\indentrel{3}\begin{verbatim}
        2  0   3  1

        4  0   6  2
   (41)
        8   0  12  4

        10  0  15  5
                    Type: CartesianTensor(1,2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPageEmpty41}
\begin{paste}{CartesianTensorXmpPageEmpty41}{CartesianTensorXmpPagePatch41}
\pastebutton{CartesianTensorXmpPageEmpty41}{\showpaste}
\tab{5}\spadcommand{rTmn := reindex(Tmn, [1,4,2,3])\free{Tmn }\bound{rTmn }}
\end{paste}\end{patch}
```

```
\begin{patch}{CartesianTensorXmpPagePatch42}
\begin{paste}{CartesianTensorXmpPageFull42}{CartesianTensorXmpPageEmpty42}
\pastebutton{CartesianTensorXmpPageFull42}{\hidepaste}
\tab{5}\spadcommand{
tt := transpose(Tm)*Tn - Tn*transpose(Tm)\free{Tm Tn }\bound{tt2 }}
\indentrel{3}\begin{verbatim}
         - 6  - 16
   (42)
          2    6
                     Type: CartesianTensor(1,2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPageEmpty42}
\begin{paste}{CartesianTensorXmpPageEmpty42}{CartesianTensorXmpPagePatch42}
\pastebutton{CartesianTensorXmpPageEmpty42}{\showpaste}
\tab{5}\spadcommand{
tt := transpose(Tm)*Tn - Tn*transpose(Tm)\free{Tm Tn }\bound{tt2 }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch43}
\begin{paste}{CartesianTensorXmpPageFull43}{CartesianTensorXmpPageEmpty43}
\pastebutton{CartesianTensorXmpPageFull43}{\hidepaste}
\tab{5}\spadcommand{Tv*(tt+Tn)\free{tt2 Tv Tn }}
\indentrel{3}\begin{verbatim}
   (43)  [- 4,- 11]
                     Type: CartesianTensor(1,2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPageEmpty43}
\begin{paste}{CartesianTensorXmpPageEmpty43}{CartesianTensorXmpPagePatch43}
\pastebutton{CartesianTensorXmpPageEmpty43}{\showpaste}
\tab{5}\spadcommand{Tv*(tt+Tn)\free{tt2 Tv Tn }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch44}
\begin{paste}{CartesianTensorXmpPageFull44}{CartesianTensorXmpPageEmpty44}
\pastebutton{CartesianTensorXmpPageFull44}{\hidepaste}
\tab{5}\spadcommand{
reindex(product(Tn,Tn),[4,3,2,1])+3*Tn*product(Tm,Tm)\free{Tm Tn }}
\indentrel{3}\begin{verbatim}
         46   84   57   114

         174  212  228  285
   (44)
          18  24    17  30

          57  63    63  76
```

```
                    Type: CartesianTensor(1,2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPageEmpty44}
\begin{paste}{CartesianTensorXmpPageEmpty44}{CartesianTensorXmpPagePatch44}
\pastebutton{CartesianTensorXmpPageEmpty44}{\showpaste}
\tab{5}\spadcommand{
reindex(product(Tn,Tn),[4,3,2,1])+3*Tn*product(Tm,Tm)\free{Tm Tn }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch45}
\begin{paste}{CartesianTensorXmpPageFull45}{CartesianTensorXmpPageEmpty45}
\pastebutton{CartesianTensorXmpPageFull45}{\hidepaste}
\tab{5}\spadcommand{delta: CT := kroneckerDelta()\free{CT }\bound{delta }}
\indentrel{3}\begin{verbatim}
         1  0
   (45)
         0  1
                    Type: CartesianTensor(1,2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPageEmpty45}
\begin{paste}{CartesianTensorXmpPageEmpty45}{CartesianTensorXmpPagePatch45}
\pastebutton{CartesianTensorXmpPageEmpty45}{\showpaste}
\tab{5}\spadcommand{delta: CT := kroneckerDelta()\free{CT }\bound{delta }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch46}
\begin{paste}{CartesianTensorXmpPageFull46}{CartesianTensorXmpPageEmpty46}
\pastebutton{CartesianTensorXmpPageFull46}{\hidepaste}
\tab{5}\spadcommand{
contract(Tmn, 2, delta, 1) = reindex(Tmn, [1,3,4,2])\free{Tmn delta }}
\indentrel{3}\begin{verbatim}
          2  4   0  0   2  4   0  0

          3  6   1  2   3  6   1  2
   (46)                  =
         8   10  0  0  8   10  0  0

         12  15  4  5  12  15  4  5
            Type: Equation CartesianTensor(1,2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPageEmpty46}
\begin{paste}{CartesianTensorXmpPageEmpty46}{CartesianTensorXmpPagePatch46}
\pastebutton{CartesianTensorXmpPageEmpty46}{\showpaste}
\tab{5}\spadcommand{
```

```
contract(Tmn, 2, delta, 1) = reindex(Tmn, [1,3,4,2])\free{Tmn delta }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch47}
\begin{paste}{CartesianTensorXmpPageFull47}{CartesianTensorXmpPageEmpty47}
\pastebutton{CartesianTensorXmpPageFull47}{\hidepaste}
\tab{5}\spadcommand{
epsilon:CT := leviCivitaSymbol()\free{CT }\bound{epsilon }}
\indentrel{3}\begin{verbatim}
          0   1
   (47)
         - 1  0
                     Type: CartesianTensor(1,2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPageEmpty47}
\begin{paste}{CartesianTensorXmpPageEmpty47}
{CartesianTensorXmpPagePatch47}
\pastebutton{CartesianTensorXmpPageEmpty47}{\showpaste}
\tab{5}\spadcommand{
epsilon:CT := leviCivitaSymbol()\free{CT }\bound{epsilon }}
\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPagePatch48}
\begin{paste}{CartesianTensorXmpPageFull48}{CartesianTensorXmpPageEmpty48}
\pastebutton{CartesianTensorXmpPageFull48}{\hidepaste}
\tab{5}\spadcommand{
contract(epsilon*Tm*epsilon, 1,2) = 2 * determinant m\free{epsilon Tm m }}
\indentrel{3}\begin{verbatim}
   (48)  - 6= - 6
            Type: Equation CartesianTensor(1,2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CartesianTensorXmpPageEmpty48}
\begin{paste}{CartesianTensorXmpPageEmpty48}{CartesianTensorXmpPagePatch48}
\pastebutton{CartesianTensorXmpPageEmpty48}{\showpaste}
\tab{5}\spadcommand{
contract(epsilon*Tm*epsilon, 1,2) = 2 * determinant m\free{epsilon Tm m }}
\end{paste}\end{patch}
```

3.14 cclass.ht

CharacterClass

— cclass.ht —

```
\begin{page}{CharacterClassXmpPage}{CharacterClass}
\beginscroll

The \spadtype{CharacterClass} domain allows classes of characters to be
defined and manipulated efficiently.

\xtc{
Character classes can be created by giving either a string or a list
of characters.
}{
\spadpaste{cl1 :=
charClass [char "a", char "e", char "i", char "o", char "u", char "y"]
\bound{cl1}}
}
\xtc{
}{
\spadpaste{cl2 := charClass "bcdfghjklmnpqrstvwxyz" \bound{cl2}}
}
\xtc{
A number of character classes are predefined for convenience.
}{
\spadpaste{digit()}
}
\xtc{
}{
\spadpaste{hexDigit()}
}
\xtc{
}{
\spadpaste{upperCase()}
}
\xtc{
}{
\spadpaste{lowerCase()}
}
\xtc{
}{
\spadpaste{alphabetic()}
}
\xtc{
}{
\spadpaste{alphanumeric()}
```

```
}
\xtc{
You can quickly test whether a character belongs to a class.
}{
\spadpaste{member?(char "a", cl1) \free{cl1}}
}
\xtc{
}{
\spadpaste{member?(char "a", cl2) \free{cl2}}
}
\xtc{
Classes have the usual set operations because
the \spadtype{CharacterClass} domain belongs to the category
\spadtype{FiniteSetAggregate(Character)}.
}{
\spadpaste{intersect(cl1, cl2)  \free{cl1 cl2}}
}
\xtc{
}{
\spadpaste{union(cl1,cl2)       \free{cl1 cl2}}
}
\xtc{
}{
\spadpaste{difference(cl1,cl2)  \free{cl1 cl2}}
}
\xtc{
}{
\spadpaste{intersect(complement(cl1),cl2)  \free{cl1 cl2}}
}
\xtc{
You can modify character classes by adding or removing characters.
}{
\spadpaste{insert!(char "a", cl2) \free{cl2}\bound{cl22}}
}
\xtc{
}{
\spadpaste{remove!(char "b", cl2) \free{cl22}\bound{cl23}}
}

For more information on related topics, see
\downlink{`Character'}{CharacterXmpPage}\ignore{Character} and
\downlink{`String'}{StringXmpPage}\ignore{String}.
%
\showBlurb{CharacterClass}
\endscroll
\autobuttons
\end{page}

\begin{patch}{CharacterClassXmpPagePatch1}
\begin{paste}{CharacterClassXmpPageFull1}{CharacterClassXmpPageEmpty1}
```

```
\pastebutton{CharacterClassXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{
cl1:=charClass[char "a", char "e", char "i", char "o", char "u", char "y"]
\bound{cl1 }}
\indentrel{3}\begin{verbatim}
   (1)  "aeiouy"
                                   Type: CharacterClass
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CharacterClassXmpPageEmpty1}
\begin{paste}{CharacterClassXmpPageEmpty1}{CharacterClassXmpPagePatch1}
\pastebutton{CharacterClassXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{
cl1:=charClass [char "a", char "e", char "i", char "o", char "u", char "y"]
\bound{cl1 }}
\end{paste}\end{patch}

\begin{patch}{CharacterClassXmpPagePatch2}
\begin{paste}{CharacterClassXmpPageFull2}{CharacterClassXmpPageEmpty2}
\pastebutton{CharacterClassXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{cl2 := charClass "bcdfghjklmnpqrstvwxyz"\bound{cl2 }}
\indentrel{3}\begin{verbatim}
   (2)  "bcdfghjklmnpqrstvwxyz"
                                   Type: CharacterClass
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CharacterClassXmpPageEmpty2}
\begin{paste}{CharacterClassXmpPageEmpty2}{CharacterClassXmpPagePatch2}
\pastebutton{CharacterClassXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{cl2 := charClass "bcdfghjklmnpqrstvwxyz"\bound{cl2 }}
\end{paste}\end{patch}

\begin{patch}{CharacterClassXmpPagePatch3}
\begin{paste}{CharacterClassXmpPageFull3}{CharacterClassXmpPageEmpty3}
\pastebutton{CharacterClassXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{digit()}
\indentrel{3}\begin{verbatim}
   (3)  "0123456789"
                                   Type: CharacterClass
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CharacterClassXmpPageEmpty3}
\begin{paste}{CharacterClassXmpPageEmpty3}{CharacterClassXmpPagePatch3}
\pastebutton{CharacterClassXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{digit()}
\end{paste}\end{patch}
```

```
\begin{patch}{CharacterClassXmpPagePatch4}
\begin{paste}{CharacterClassXmpPageFull4}{CharacterClassXmpPageEmpty4}
\pastebutton{CharacterClassXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{hexDigit()}
\indentrel{3}\begin{verbatim}
   (4)  "0123456789ABCDEFabcdef"
                                  Type: CharacterClass
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CharacterClassXmpPageEmpty4}
\begin{paste}{CharacterClassXmpPageEmpty4}{CharacterClassXmpPagePatch4}
\pastebutton{CharacterClassXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{hexDigit()}
\end{paste}\end{patch}

\begin{patch}{CharacterClassXmpPagePatch5}
\begin{paste}{CharacterClassXmpPageFull5}{CharacterClassXmpPageEmpty5}
\pastebutton{CharacterClassXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{upperCase()}
\indentrel{3}\begin{verbatim}
   (5)  "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
                                  Type: CharacterClass
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CharacterClassXmpPageEmpty5}
\begin{paste}{CharacterClassXmpPageEmpty5}{CharacterClassXmpPagePatch5}
\pastebutton{CharacterClassXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{upperCase()}
\end{paste}\end{patch}

\begin{patch}{CharacterClassXmpPagePatch6}
\begin{paste}{CharacterClassXmpPageFull6}{CharacterClassXmpPageEmpty6}
\pastebutton{CharacterClassXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{lowerCase()}
\indentrel{3}\begin{verbatim}
   (6)  "abcdefghijklmnopqrstuvwxyz"
                                  Type: CharacterClass
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CharacterClassXmpPageEmpty6}
\begin{paste}{CharacterClassXmpPageEmpty6}{CharacterClassXmpPagePatch6}
\pastebutton{CharacterClassXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{lowerCase()}
\end{paste}\end{patch}

\begin{patch}{CharacterClassXmpPagePatch7}
\begin{paste}{CharacterClassXmpPageFull7}{CharacterClassXmpPageEmpty7}
```

```
\pastebutton{CharacterClassXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{alphabetic()}
\indentrel{3}\begin{verbatim}
   (7)
   "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"
                                   Type: CharacterClass
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CharacterClassXmpPageEmpty7}
\begin{paste}{CharacterClassXmpPageEmpty7}{CharacterClassXmpPagePatch7}
\pastebutton{CharacterClassXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{alphabetic()}
\end{paste}\end{patch}

\begin{patch}{CharacterClassXmpPagePatch8}
\begin{paste}{CharacterClassXmpPageFull8}{CharacterClassXmpPageEmpty8}
\pastebutton{CharacterClassXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{alphanumeric()}
\indentrel{3}\begin{verbatim}
   (8)
  "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqr
  stuvwxyz"
                                   Type: CharacterClass
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CharacterClassXmpPageEmpty8}
\begin{paste}{CharacterClassXmpPageEmpty8}{CharacterClassXmpPagePatch8}
\pastebutton{CharacterClassXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{alphanumeric()}
\end{paste}\end{patch}

\begin{patch}{CharacterClassXmpPagePatch9}
\begin{paste}{CharacterClassXmpPageFull9}{CharacterClassXmpPageEmpty9}
\pastebutton{CharacterClassXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{member?(char "a", cl1)\free{cl1 }}
\indentrel{3}\begin{verbatim}
   (9)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CharacterClassXmpPageEmpty9}
\begin{paste}{CharacterClassXmpPageEmpty9}{CharacterClassXmpPagePatch9}
\pastebutton{CharacterClassXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{member?(char "a", cl1)\free{cl1 }}
\end{paste}\end{patch}

\begin{patch}{CharacterClassXmpPagePatch10}
```

```
\begin{paste}{CharacterClassXmpPageFull10}{CharacterClassXmpPageEmpty10}
\pastebutton{CharacterClassXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{member?(char "a", cl2)\free{cl2 }}
\indentrel{3}\begin{verbatim}
   (10)  false
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CharacterClassXmpPageEmpty10}
\begin{paste}{CharacterClassXmpPageEmpty10}{CharacterClassXmpPagePatch10}
\pastebutton{CharacterClassXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{member?(char "a", cl2)\free{cl2 }}
\end{paste}\end{patch}

\begin{patch}{CharacterClassXmpPagePatch11}
\begin{paste}{CharacterClassXmpPageFull11}{CharacterClassXmpPageEmpty11}
\pastebutton{CharacterClassXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{intersect(cl1, cl2)\free{cl1 cl2 }}
\indentrel{3}\begin{verbatim}
   (11)  "y"
                                   Type: CharacterClass
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CharacterClassXmpPageEmpty11}
\begin{paste}{CharacterClassXmpPageEmpty11}{CharacterClassXmpPagePatch11}
\pastebutton{CharacterClassXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{intersect(cl1, cl2)\free{cl1 cl2 }}
\end{paste}\end{patch}

\begin{patch}{CharacterClassXmpPagePatch12}
\begin{paste}{CharacterClassXmpPageFull12}{CharacterClassXmpPageEmpty12}
\pastebutton{CharacterClassXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{union(cl1,cl2)\free{cl1 cl2 }}
\indentrel{3}\begin{verbatim}
   (12)  "abcdefghijklmnopqrstuvwxyz"
                                   Type: CharacterClass
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CharacterClassXmpPageEmpty12}
\begin{paste}{CharacterClassXmpPageEmpty12}{CharacterClassXmpPagePatch12}
\pastebutton{CharacterClassXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{union(cl1,cl2)\free{cl1 cl2 }}
\end{paste}\end{patch}

\begin{patch}{CharacterClassXmpPagePatch13}
\begin{paste}{CharacterClassXmpPageFull13}{CharacterClassXmpPageEmpty13}
\pastebutton{CharacterClassXmpPageFull13}{\hidepaste}
```

```
\tab{5}\spadcommand{difference(cl1,cl2)\free{cl1 cl2 }}
\indentrel{3}\begin{verbatim}
   (13)  "aeiou"
                                   Type: CharacterClass
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CharacterClassXmpPageEmpty13}
\begin{paste}{CharacterClassXmpPageEmpty13}{CharacterClassXmpPagePatch13}
\pastebutton{CharacterClassXmpPageEmpty13}{\showpaste}
\tab{5}\spadcommand{difference(cl1,cl2)\free{cl1 cl2 }}
\end{paste}\end{patch}

\begin{patch}{CharacterClassXmpPagePatch14}
\begin{paste}{CharacterClassXmpPageFull14}{CharacterClassXmpPageEmpty14}
\pastebutton{CharacterClassXmpPageFull14}{\hidepaste}
\tab{5}\spadcommand{intersect(complement(cl1),cl2)\free{cl1 cl2 }}
\indentrel{3}\begin{verbatim}
   (14)  "bcdfghjklmnpqrstvwxz"
                                   Type: CharacterClass
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CharacterClassXmpPageEmpty14}
\begin{paste}{CharacterClassXmpPageEmpty14}{CharacterClassXmpPagePatch14}
\pastebutton{CharacterClassXmpPageEmpty14}{\showpaste}
\tab{5}\spadcommand{intersect(complement(cl1),cl2)\free{cl1 cl2 }}
\end{paste}\end{patch}

\begin{patch}{CharacterClassXmpPagePatch15}
\begin{paste}{CharacterClassXmpPageFull15}{CharacterClassXmpPageEmpty15}
\pastebutton{CharacterClassXmpPageFull15}{\hidepaste}
\tab{5}\spadcommand{insert!(char "a", cl2)\free{cl2 }\bound{cl22 }}
\indentrel{3}\begin{verbatim}
   (15)  "abcdfghjklmnpqrstvwxyz"
                                   Type: CharacterClass
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CharacterClassXmpPageEmpty15}
\begin{paste}{CharacterClassXmpPageEmpty15}{CharacterClassXmpPagePatch15}
\pastebutton{CharacterClassXmpPageEmpty15}{\showpaste}
\tab{5}\spadcommand{insert!(char "a", cl2)\free{cl2 }\bound{cl22 }}
\end{paste}\end{patch}

\begin{patch}{CharacterClassXmpPagePatch16}
\begin{paste}{CharacterClassXmpPageFull16}{CharacterClassXmpPageEmpty16}
\pastebutton{CharacterClassXmpPageFull16}{\hidepaste}
\tab{5}\spadcommand{remove!(char "b", cl2)\free{cl22 }\bound{cl23 }}
\indentrel{3}\begin{verbatim}
```

```
  (16)  "acdfghjklmnpqrstvwxyz"
                                     Type: CharacterClass
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CharacterClassXmpPageEmpty16}
\begin{paste}{CharacterClassXmpPageEmpty16}{CharacterClassXmpPagePatch16}
\pastebutton{CharacterClassXmpPageEmpty16}{\showpaste}
\tab{5}\spadcommand{remove!(char "b", cl2)\free{cl22 }\bound{cl23 }}
\end{paste}\end{patch}
```

———-

3.15 char.ht

Character

⇒ "CharacterClass" (CharacterClassXmpPage) 3.14 on page 221
⇒ "String" (StringXmpPage) 3.103 on page 1269

— **char.ht** —

```
\begin{page}{CharacterXmpPage}{Character}
\beginscroll

The members of the domain \spadtype{Character} are values
representing letters, numerals and other text elements.
For more information on related topics, see
\downlink{`CharacterClass'}{CharacterClassXmpPage}\ignore{CharacterClass}
and \downlink{`String'}{StringXmpPage}\ignore{String}.

\xtc{
Characters can be obtained using \spadtype{String} notation.
}{
\spadpaste{chars := [char "a", char "A", char "X", char "8", char "+"]
\bound{chars}}
}
\xtc{
Certain characters are available by name.
This is the blank character.
}{
\spadpaste{space()}
}
\xtc{
This is the quote that is used in strings.
}{
```

```
\spadpaste{quote()}
}
\xtc{
This is the escape character that allows quotes and other characters
within strings.
}{
\spadpaste{escape()}
}
\xtc{
Characters are represented as integers in a machine-dependent way.
The integer value can be obtained using the \spadfunFrom{ord}{Character}
operation.
It is always true that \spad{char(ord c) = c} and \spad{ord(char i) = i},
provided that \spad{i} is in the range \spad{0..size()\$Character-1}.
}{
\spadpaste{[ord c for c in chars] \free{chars}}
}

\xtc{
The \spadfunFrom{lowerCase}{Character} operation converts an upper case
letter to the corresponding lower case letter.
If the argument is not an upper case letter, then it is returned
unchanged.
}{
\spadpaste{[upperCase c for c in chars] \free{chars}}
}
\xtc{
Likewise, the \spadfunFrom{upperCase}{Character} operation converts lower
case letters to upper case.
}{
\spadpaste{[lowerCase c for c in chars] \free{chars}}
}

\xtc{
A number of tests are available to determine whether characters
belong to certain families.
}{
\spadpaste{[alphabetic? c for c in chars] \free{chars}}
}
\xtc{
}{
\spadpaste{[upperCase? c for c in chars] \free{chars}}
}
\xtc{
}{
\spadpaste{[lowerCase? c for c in chars] \free{chars}}
}
\xtc{
}{
\spadpaste{[digit? c for c in chars] \free{chars}}
```

```
}
\xtc{
}{
\spadpaste{[hexDigit? c for c in chars] \free{chars}}
}
\xtc{
}{
\spadpaste{[alphanumeric? c for c in chars] \free{chars}}
}
\endscroll
\autobuttons
\end{page}

\begin{patch}{CharacterXmpPagePatch1}
\begin{paste}{CharacterXmpPageFull1}{CharacterXmpPageEmpty1}
\pastebutton{CharacterXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{
chars := [char "a", char "A", char "X", char "8", char "+"]\bound{chars }}
\indentrel{3}\begin{verbatim}
   (1)  [a,A,X,8,+]
                                   Type: List Character
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CharacterXmpPageEmpty1}
\begin{paste}{CharacterXmpPageEmpty1}{CharacterXmpPagePatch1}
\pastebutton{CharacterXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{
chars := [char "a", char "A", char "X", char "8", char "+"]\bound{chars }}
\end{paste}\end{patch}

\begin{patch}{CharacterXmpPagePatch2}
\begin{paste}{CharacterXmpPageFull2}{CharacterXmpPageEmpty2}
\pastebutton{CharacterXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{space()}
\indentrel{3}\begin{verbatim}
   (2)
                                        Type: Character
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CharacterXmpPageEmpty2}
\begin{paste}{CharacterXmpPageEmpty2}{CharacterXmpPagePatch2}
\pastebutton{CharacterXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{space()}
\end{paste}\end{patch}

\begin{patch}{CharacterXmpPagePatch3}
\begin{paste}{CharacterXmpPageFull3}{CharacterXmpPageEmpty3}
\pastebutton{CharacterXmpPageFull3}{\hidepaste}
```

```
\tab{5}\spadcommand{quote()}
\indentrel{3}\begin{verbatim}
   (3)  "
                                       Type: Character
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CharacterXmpPageEmpty3}
\begin{paste}{CharacterXmpPageEmpty3}{CharacterXmpPagePatch3}
\pastebutton{CharacterXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{quote()}
\end{paste}\end{patch}

\begin{patch}{CharacterXmpPagePatch4}
\begin{paste}{CharacterXmpPageFull4}{CharacterXmpPageEmpty4}
\pastebutton{CharacterXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{escape()}
\indentrel{3}\begin{verbatim}
   (4)  _
                                       Type: Character
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CharacterXmpPageEmpty4}
\begin{paste}{CharacterXmpPageEmpty4}{CharacterXmpPagePatch4}
\pastebutton{CharacterXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{escape()}
\end{paste}\end{patch}

\begin{patch}{CharacterXmpPagePatch5}
\begin{paste}{CharacterXmpPageFull5}{CharacterXmpPageEmpty5}
\pastebutton{CharacterXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{[ord c for c in chars]\free{chars }}
\indentrel{3}\begin{verbatim}
   (5)  [97,65,88,56,43]
                                    Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CharacterXmpPageEmpty5}
\begin{paste}{CharacterXmpPageEmpty5}{CharacterXmpPagePatch5}
\pastebutton{CharacterXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{[ord c for c in chars]\free{chars }}
\end{paste}\end{patch}

\begin{patch}{CharacterXmpPagePatch6}
\begin{paste}{CharacterXmpPageFull6}{CharacterXmpPageEmpty6}
\pastebutton{CharacterXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{[upperCase c for c in chars]\free{chars }}
\indentrel{3}\begin{verbatim}
```

```
   (6)  [A,A,X,8,+]
                                  Type: List Character
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CharacterXmpPageEmpty6}
\begin{paste}{CharacterXmpPageEmpty6}{CharacterXmpPagePatch6}
\pastebutton{CharacterXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{[upperCase c for c in chars]\free{chars }}
\end{paste}\end{patch}

\begin{patch}{CharacterXmpPagePatch7}
\begin{paste}{CharacterXmpPageFull7}{CharacterXmpPageEmpty7}
\pastebutton{CharacterXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{[lowerCase c for c in chars]\free{chars }}
\indentrel{3}\begin{verbatim}
   (7)  [a,a,x,8,+]
                                  Type: List Character
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CharacterXmpPageEmpty7}
\begin{paste}{CharacterXmpPageEmpty7}{CharacterXmpPagePatch7}
\pastebutton{CharacterXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{[lowerCase c for c in chars]\free{chars }}
\end{paste}\end{patch}

\begin{patch}{CharacterXmpPagePatch8}
\begin{paste}{CharacterXmpPageFull8}{CharacterXmpPageEmpty8}
\pastebutton{CharacterXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{[alphabetic? c for c in chars]\free{chars }}
\indentrel{3}\begin{verbatim}
   (8)  [true,true,true,false,false]
                                    Type: List Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CharacterXmpPageEmpty8}
\begin{paste}{CharacterXmpPageEmpty8}{CharacterXmpPagePatch8}
\pastebutton{CharacterXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{[alphabetic? c for c in chars]\free{chars }}
\end{paste}\end{patch}

\begin{patch}{CharacterXmpPagePatch9}
\begin{paste}{CharacterXmpPageFull9}{CharacterXmpPageEmpty9}
\pastebutton{CharacterXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{[upperCase? c for c in chars]\free{chars }}
\indentrel{3}\begin{verbatim}
   (9)  [false,true,true,false,false]
                                    Type: List Boolean
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CharacterXmpPageEmpty9}
\begin{paste}{CharacterXmpPageEmpty9}{CharacterXmpPagePatch9}
\pastebutton{CharacterXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{[upperCase? c for c in chars]\free{chars }}
\end{paste}\end{patch}

\begin{patch}{CharacterXmpPagePatch10}
\begin{paste}{CharacterXmpPageFull10}{CharacterXmpPageEmpty10}
\pastebutton{CharacterXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{[lowerCase? c for c in chars]\free{chars }}
\indentrel{3}\begin{verbatim}
   (10)  [true,false,false,false,false]
                                     Type: List Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CharacterXmpPageEmpty10}
\begin{paste}{CharacterXmpPageEmpty10}{CharacterXmpPagePatch10}
\pastebutton{CharacterXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{[lowerCase? c for c in chars]\free{chars }}
\end{paste}\end{patch}

\begin{patch}{CharacterXmpPagePatch11}
\begin{paste}{CharacterXmpPageFull11}{CharacterXmpPageEmpty11}
\pastebutton{CharacterXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{[digit? c for c in chars]\free{chars }}
\indentrel{3}\begin{verbatim}
   (11)  [false,false,false,true,false]
                                     Type: List Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CharacterXmpPageEmpty11}
\begin{paste}{CharacterXmpPageEmpty11}{CharacterXmpPagePatch11}
\pastebutton{CharacterXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{[digit? c for c in chars]\free{chars }}
\end{paste}\end{patch}

\begin{patch}{CharacterXmpPagePatch12}
\begin{paste}{CharacterXmpPageFull12}{CharacterXmpPageEmpty12}
\pastebutton{CharacterXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{[hexDigit? c for c in chars]\free{chars }}
\indentrel{3}\begin{verbatim}
   (12)  [true,true,false,true,false]
                                     Type: List Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{CharacterXmpPageEmpty12}
\begin{paste}{CharacterXmpPageEmpty12}{CharacterXmpPagePatch12}
\pastebutton{CharacterXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{[hexDigit? c for c in chars]\free{chars }}
\end{paste}\end{patch}

\begin{patch}{CharacterXmpPagePatch13}
\begin{paste}{CharacterXmpPageFull13}{CharacterXmpPageEmpty13}
\pastebutton{CharacterXmpPageFull13}{\hidepaste}
\tab{5}\spadcommand{[alphanumeric? c for c in chars]\free{chars }}
\indentrel{3}\begin{verbatim}
   (13)  [true,true,true,true,false]
                                     Type: List Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CharacterXmpPageEmpty13}
\begin{paste}{CharacterXmpPageEmpty13}{CharacterXmpPagePatch13}
\pastebutton{CharacterXmpPageEmpty13}{\showpaste}
\tab{5}\spadcommand{[alphanumeric? c for c in chars]\free{chars }}
\end{paste}\end{patch}
```

———-

CliffordAlgebra

⇒ "The Complex Numbers as a Clifford Algebra" (ugxCliffordComplexPage) 3.15 on page 235

⇒ "The Quaternion Numbers as a Clifford AlgebraNo" (ugxCliffordQuaternPage) 3.15 on page 239
⇒ "The Exterior Algebra on a Three Space" (ugxCliffordExteriorPage) 3.15 on page 244
⇒ "The Dirac Spin Algebra" (ugxCliffordDiracPage) 3.15 on page 250

— **clif.ht** —

```
\begin{page}{CliffordAlgebraXmpPage}{CliffordAlgebra}
\beginscroll

\noindent
\spadtype{CliffordAlgebra(n,K,Q)} defines a vector space of dimension
\texht{$2^n$}{\spad{2**n}} over the field \texht{$K$}{\spad{K}} with a
given quadratic form \spad{Q}.
If \texht{$\{e_1, \ldots, e_n\}$}{\spad{\{e(i), 1<=i<=n\}}}
is a basis for  \texht{$K^n$}{\spad{K**n}} then
\begin{verbatim}
```

```
{ 1,
  e(i) 1 <= i <= n,
  e(i1)*e(i2) 1 <= i1 < i2 <=n,
  ...,
  e(1)*e(2)*...*e(n) }
\end{verbatim}
is a basis for the Clifford algebra.
The algebra is defined by the relations
\begin{verbatim}
e(i)*e(i) = Q(e(i))
e(i)*e(j) = -e(j)*e(i),  i ^= j
\end{verbatim}
Examples of Clifford Algebras are
gaussians (complex numbers), quaternions,
exterior algebras and spin algebras.
%

\beginmenu
    \menudownlink{{9.10.1. The Complex Numbers as a Clifford Algebra}}
{ugxCliffordComplexPage}
    \menudownlink{{9.10.2. The Quaternion Numbers as a Clifford Algebra}}
{ugxCliffordQuaternPage}
    \menudownlink{{9.10.3. The Exterior Algebra on a Three Space}}
{ugxCliffordExteriorPage}
    \menudownlink{{9.10.4. The Dirac Spin Algebra}}
{ugxCliffordDiracPage}
\endmenu
\endscroll
\autobuttons
\end{page}
```

The Complex Numbers as a Clifford Algebra

⇒ "Complex" (ComplexXmpPage) 3.16 on page 254

— **clif.ht** —

```
\begin{page}{ugxCliffordComplexPage}
{The Complex Numbers as a Clifford Algebra}
\beginscroll

\labelSpace{5pc}
\xtc{
This is the field over which we will work, rational functions with
integer coefficients.
}{
```

```
\spadpaste{K := Fraction Polynomial Integer \bound{K}}
}
\xtc{
We use this matrix for the quadratic form.
}{
\spadpaste{m := matrix [[-1]] \bound{m}}
}
\xtc{
We get complex arithmetic by using this domain.
}{
\spadpaste{C := CliffordAlgebra(1, K, quadraticForm m) \free{K m}
\bound{C}}
}
\xtc{
Here is \spad{i}, the usual square root of \spad{-1.}
}{
\spadpaste{i: C := e(1)   \bound{i}\free{C}}
}
\xtc{
Here are some examples of the arithmetic.
}{
\spadpaste{x := a + b * i \bound{x}\free{i}}
}
\xtc{
}{
\spadpaste{y := c + d * i \bound{y}\free{i}}
}
\xtc{
See \downlink{`Complex'}{ComplexXmpPage}\ignore{Complex}
for examples of Axiom's constructor
implementing complex numbers.
}{
\spadpaste{x * y \free{x y}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxCliffordComplexPagePatch1}
\begin{paste}{ugxCliffordComplexPageFull1}{ugxCliffordComplexPageEmpty1}
\pastebutton{ugxCliffordComplexPageFull1}{\hidepaste}
\tab{5}\spadcommand{K := Fraction Polynomial Integer\bound{K }}
\indentrel{3}\begin{verbatim}
   (1)  Fraction Polynomial Integer
                                            Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxCliffordComplexPageEmpty1}
```

```
\begin{paste}{ugxCliffordComplexPageEmpty1}{ugxCliffordComplexPagePatch1}
\pastebutton{ugxCliffordComplexPageEmpty1}{\showpaste}
\tab{5}\spadcommand{K := Fraction Polynomial Integer\bound{K }}
\end{paste}\end{patch}

\begin{patch}{ugxCliffordComplexPagePatch2}
\begin{paste}{ugxCliffordComplexPageFull2}{ugxCliffordComplexPageEmpty2}
\pastebutton{ugxCliffordComplexPageFull2}{\hidepaste}
\tab{5}\spadcommand{m := matrix [[-1]]\bound{m }}
\indentrel{3}\begin{verbatim}
   (2)  [- 1]
                               Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxCliffordComplexPageEmpty2}
\begin{paste}{ugxCliffordComplexPageEmpty2}{ugxCliffordComplexPagePatch2}
\pastebutton{ugxCliffordComplexPageEmpty2}{\showpaste}
\tab{5}\spadcommand{m := matrix [[-1]]\bound{m }}
\end{paste}\end{patch}

\begin{patch}{ugxCliffordComplexPagePatch3}
\begin{paste}{ugxCliffordComplexPageFull3}{ugxCliffordComplexPageEmpty3}
\pastebutton{ugxCliffordComplexPageFull3}{\hidepaste}
\tab{5}\spadcommand{
C := CliffordAlgebra(1, K, quadraticForm m)\free{K m }\bound{C }}
\indentrel{3}\begin{verbatim}
   (3)
   CliffordAlgebra(1,Fraction Polynomial Integer,MATRIX)
                                       Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxCliffordComplexPageEmpty3}
\begin{paste}{ugxCliffordComplexPageEmpty3}{ugxCliffordComplexPagePatch3}
\pastebutton{ugxCliffordComplexPageEmpty3}{\showpaste}
\tab{5}\spadcommand{
C := CliffordAlgebra(1, K, quadraticForm m)\free{K m }\bound{C }}
\end{paste}\end{patch}

\begin{patch}{ugxCliffordComplexPagePatch4}
\begin{paste}{ugxCliffordComplexPageFull4}{ugxCliffordComplexPageEmpty4}
\pastebutton{ugxCliffordComplexPageFull4}{\hidepaste}
\tab{5}\spadcommand{i: C := e(1)\bound{i }\free{C }}
\indentrel{3}\begin{verbatim}
   (4)  e
         1
Type: CliffordAlgebra(1,Fraction Polynomial Integer,MATRIX)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugxCliffordComplexPageEmpty4}
\begin{paste}{ugxCliffordComplexPageEmpty4}{ugxCliffordComplexPagePatch4}
\pastebutton{ugxCliffordComplexPageEmpty4}{\showpaste}
\tab{5}\spadcommand{i: C := e(1)\bound{i }\free{C }}
\end{paste}\end{patch}

\begin{patch}{ugxCliffordComplexPagePatch5}
\begin{paste}{ugxCliffordComplexPageFull5}{ugxCliffordComplexPageEmpty5}
\pastebutton{ugxCliffordComplexPageFull5}{\hidepaste}
\tab{5}\spadcommand{x := a + b * i\bound{x }\free{i }}
\indentrel{3}\begin{verbatim}
   (5)  a + b e
               1
Type: CliffordAlgebra(1,Fraction Polynomial Integer,MATRIX)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxCliffordComplexPageEmpty5}
\begin{paste}{ugxCliffordComplexPageEmpty5}{ugxCliffordComplexPagePatch5}
\pastebutton{ugxCliffordComplexPageEmpty5}{\showpaste}
\tab{5}\spadcommand{x := a + b * i\bound{x }\free{i }}
\end{paste}\end{patch}

\begin{patch}{ugxCliffordComplexPagePatch6}
\begin{paste}{ugxCliffordComplexPageFull6}{ugxCliffordComplexPageEmpty6}
\pastebutton{ugxCliffordComplexPageFull6}{\hidepaste}
\tab{5}\spadcommand{y := c + d * i\bound{y }\free{i }}
\indentrel{3}\begin{verbatim}
   (6)  c + d e
               1
Type: CliffordAlgebra(1,Fraction Polynomial Integer,MATRIX)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxCliffordComplexPageEmpty6}
\begin{paste}{ugxCliffordComplexPageEmpty6}{ugxCliffordComplexPagePatch6}
\pastebutton{ugxCliffordComplexPageEmpty6}{\showpaste}
\tab{5}\spadcommand{y := c + d * i\bound{y }\free{i }}
\end{paste}\end{patch}

\begin{patch}{ugxCliffordComplexPagePatch7}
\begin{paste}{ugxCliffordComplexPageFull7}{ugxCliffordComplexPageEmpty7}
\pastebutton{ugxCliffordComplexPageFull7}{\hidepaste}
\tab{5}\spadcommand{x * y\free{x y }}
\indentrel{3}\begin{verbatim}
   (7)  - b d + a c + (a d + b c)e
                                   1
Type: CliffordAlgebra(1,Fraction Polynomial Integer,MATRIX)
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxCliffordComplexPageEmpty7}
\begin{paste}{ugxCliffordComplexPageEmpty7}{ugxCliffordComplexPagePatch7}
\pastebutton{ugxCliffordComplexPageEmpty7}{\showpaste}
\tab{5}\spadcommand{x * y\free{x y }}
\end{paste}\end{patch}
```

———-

The Quaternion Numbers as a Clifford Algebra

⇒ "Quaternion" (QuaternionXmpPage) 3.89 on page 1134

— **clif.ht** —

```
\begin{page}{ugxCliffordQuaternPage}
{The Quaternion Numbers as a Clifford Algebra}
\beginscroll

\labelSpace{3pc}
\xtc{
This is the field over which we will work, rational functions with
integer coefficients.
}{
\spadpaste{K := Fraction Polynomial Integer \bound{K}}
}
\xtc{
We use this matrix for the quadratic form.
}{
\spadpaste{m := matrix [[-1,0],[0,-1]] \bound{m}}
}
\xtc{
The resulting domain is the quaternions.
}{
\spadpaste{H  := CliffordAlgebra(2, K, quadraticForm m) \free{K m}\bound{H}}
}
\xtc{
We use Hamilton's notation for \spad{i},\spad{j},\spad{k}.
}{
\spadpaste{i: H  := e(1) \free{H}\bound{i}}
}
\xtc{
}{
\spadpaste{j: H  := e(2) \free{H}\bound{j}}
}
\xtc{
}{
```

```
\spadpaste{k: H  := i * j \free{H,i,j}\bound{k}}
}
\xtc{
}{
\spadpaste{x := a + b * i + c * j + d * k \free{i j k}\bound{x}}
}
\xtc{
}{
\spadpaste{y := e + f * i + g * j + h * k \free{i j k}\bound{y}}
}
\xtc{
}{
\spadpaste{x + y \free{x y}}
}
\xtc{
}{
\spadpaste{x * y \free{x y}}
}
\xtc{
See \downlink{`Quaternion'}{QuaternionXmpPage}
\ignore{Quaternion} for examples of Axiom's constructor
implementing quaternions.
}{
\spadpaste{y * x \free{x y}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxCliffordQuaternPagePatch1}
\begin{paste}{ugxCliffordQuaternPageFull1}{ugxCliffordQuaternPageEmpty1}
\pastebutton{ugxCliffordQuaternPageFull1}{\hidepaste}
\tab{5}\spadcommand{K := Fraction Polynomial Integer\bound{K }}
\indentrel{3}\begin{verbatim}
   (1)  Fraction Polynomial Integer
                                            Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxCliffordQuaternPageEmpty1}
\begin{paste}{ugxCliffordQuaternPageEmpty1}{ugxCliffordQuaternPagePatch1}
\pastebutton{ugxCliffordQuaternPageEmpty1}{\showpaste}
\tab{5}\spadcommand{K := Fraction Polynomial Integer\bound{K }}
\end{paste}\end{patch}

\begin{patch}{ugxCliffordQuaternPagePatch2}
\begin{paste}{ugxCliffordQuaternPageFull2}{ugxCliffordQuaternPageEmpty2}
\pastebutton{ugxCliffordQuaternPageFull2}{\hidepaste}
\tab{5}\spadcommand{m := matrix [[-1,0],[0,-1]]\bound{m }}
```

```
\indentrel{3}\begin{verbatim}
        - 1   0
   (2)
         0   - 1
                                   Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxCliffordQuaternPageEmpty2}
\begin{paste}{ugxCliffordQuaternPageEmpty2}{ugxCliffordQuaternPagePatch2}
\pastebutton{ugxCliffordQuaternPageEmpty2}{\showpaste}
\tab{5}\spadcommand{m := matrix [[-1,0],[0,-1]]\bound{m }}
\end{paste}\end{patch}

\begin{patch}{ugxCliffordQuaternPagePatch3}
\begin{paste}{ugxCliffordQuaternPageFull3}{ugxCliffordQuaternPageEmpty3}
\pastebutton{ugxCliffordQuaternPageFull3}{\hidepaste}
\tab{5}\spadcommand{
H := CliffordAlgebra(2, K, quadraticForm m)\free{K m }\bound{H }}
\indentrel{3}\begin{verbatim}
   (3)
   CliffordAlgebra(2,Fraction Polynomial Integer,MATRIX)
                                           Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxCliffordQuaternPageEmpty3}
\begin{paste}{ugxCliffordQuaternPageEmpty3}{ugxCliffordQuaternPagePatch3}
\pastebutton{ugxCliffordQuaternPageEmpty3}{\showpaste}
\tab{5}\spadcommand{
H := CliffordAlgebra(2, K, quadraticForm m)\free{K m }\bound{H }}
\end{paste}\end{patch}

\begin{patch}{ugxCliffordQuaternPagePatch4}
\begin{paste}{ugxCliffordQuaternPageFull4}{ugxCliffordQuaternPageEmpty4}
\pastebutton{ugxCliffordQuaternPageFull4}{\hidepaste}
\tab{5}\spadcommand{i: H := e(1)\free{H }\bound{i }}
\indentrel{3}\begin{verbatim}
   (4)  e
         1
Type: CliffordAlgebra(2,Fraction Polynomial Integer,MATRIX)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxCliffordQuaternPageEmpty4}
\begin{paste}{ugxCliffordQuaternPageEmpty4}{ugxCliffordQuaternPagePatch4}
\pastebutton{ugxCliffordQuaternPageEmpty4}{\showpaste}
\tab{5}\spadcommand{i: H := e(1)\free{H }\bound{i }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugxCliffordQuaternPagePatch5}
\begin{paste}{ugxCliffordQuaternPageFull5}{ugxCliffordQuaternPageEmpty5}
\pastebutton{ugxCliffordQuaternPageFull5}{\hidepaste}
\tab{5}\spadcommand{j: H := e(2)\free{H }\bound{j }}
\indentrel{3}\begin{verbatim}
   (5)  e
         2
Type: CliffordAlgebra(2,Fraction Polynomial Integer,MATRIX)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxCliffordQuaternPageEmpty5}
\begin{paste}{ugxCliffordQuaternPageEmpty5}{ugxCliffordQuaternPagePatch5}
\pastebutton{ugxCliffordQuaternPageEmpty5}{\showpaste}
\tab{5}\spadcommand{j: H := e(2)\free{H }\bound{j }}
\end{paste}\end{patch}

\begin{patch}{ugxCliffordQuaternPagePatch6}
\begin{paste}{ugxCliffordQuaternPageFull6}{ugxCliffordQuaternPageEmpty6}
\pastebutton{ugxCliffordQuaternPageFull6}{\hidepaste}
\tab{5}\spadcommand{k: H := i * j\free{H i j }\bound{k }}
\indentrel{3}\begin{verbatim}
   (6)  e e
         1 2
Type: CliffordAlgebra(2,Fraction Polynomial Integer,MATRIX)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxCliffordQuaternPageEmpty6}
\begin{paste}{ugxCliffordQuaternPageEmpty6}{ugxCliffordQuaternPagePatch6}
\pastebutton{ugxCliffordQuaternPageEmpty6}{\showpaste}
\tab{5}\spadcommand{k: H := i * j\free{H i j }\bound{k }}
\end{paste}\end{patch}

\begin{patch}{ugxCliffordQuaternPagePatch7}
\begin{paste}{ugxCliffordQuaternPageFull7}{ugxCliffordQuaternPageEmpty7}
\pastebutton{ugxCliffordQuaternPageFull7}{\hidepaste}
\tab{5}\spadcommand{x := a + b * i + c * j + d * k\free{i j k }\bound{x }}
\indentrel{3}\begin{verbatim}
   (7)  a + b e  + c e  + d e e
               1      2      1 2
Type: CliffordAlgebra(2,Fraction Polynomial Integer,MATRIX)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxCliffordQuaternPageEmpty7}
\begin{paste}{ugxCliffordQuaternPageEmpty7}{ugxCliffordQuaternPagePatch7}
\pastebutton{ugxCliffordQuaternPageEmpty7}{\showpaste}
\tab{5}\spadcommand{x := a + b * i + c * j + d * k\free{i j k }\bound{x }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugxCliffordQuaternPagePatch8}
\begin{paste}{ugxCliffordQuaternPageFull8}{ugxCliffordQuaternPageEmpty8}
\pastebutton{ugxCliffordQuaternPageFull8}{\hidepaste}
\tab{5}\spadcommand{y := e + f * i + g * j + h * k\free{i j k }\bound{y }}
\indentrel{3}\begin{verbatim}
   (8)  e + f e  + g e  + h e e
               1      2      1 2
Type: CliffordAlgebra(2,Fraction Polynomial Integer,MATRIX)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxCliffordQuaternPageEmpty8}
\begin{paste}{ugxCliffordQuaternPageEmpty8}{ugxCliffordQuaternPagePatch8}
\pastebutton{ugxCliffordQuaternPageEmpty8}{\showpaste}
\tab{5}\spadcommand{y := e + f * i + g * j + h * k\free{i j k }\bound{y }}
\end{paste}\end{patch}

\begin{patch}{ugxCliffordQuaternPagePatch9}
\begin{paste}{ugxCliffordQuaternPageFull9}{ugxCliffordQuaternPageEmpty9}
\pastebutton{ugxCliffordQuaternPageFull9}{\hidepaste}
\tab{5}\spadcommand{x + y\free{x y }}
\indentrel{3}\begin{verbatim}
   (9)  e + a + (f + b)e  + (g + c)e  + (h + d)e e
                        1           2           1 2
Type: CliffordAlgebra(2,Fraction Polynomial Integer,MATRIX)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxCliffordQuaternPageEmpty9}
\begin{paste}{ugxCliffordQuaternPageEmpty9}{ugxCliffordQuaternPagePatch9}
\pastebutton{ugxCliffordQuaternPageEmpty9}{\showpaste}
\tab{5}\spadcommand{x + y\free{x y }}
\end{paste}\end{patch}

\begin{patch}{ugxCliffordQuaternPagePatch10}
\begin{paste}{ugxCliffordQuaternPageFull10}{ugxCliffordQuaternPageEmpty10}
\pastebutton{ugxCliffordQuaternPageFull10}{\hidepaste}
\tab{5}\spadcommand{x * y\free{x y }}
\indentrel{3}\begin{verbatim}
   (10)
     - d h - c g - b f + a e + (c h - d g + a f + b e)e
                                                       1
   +
     (- b h + a g + d f + c e)e
                               2
   +
     (a h + b g - c f + d e)e e
                             1 2
Type: CliffordAlgebra(2,Fraction Polynomial Integer,MATRIX)
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxCliffordQuaternPageEmpty10}
\begin{paste}{ugxCliffordQuaternPageEmpty10}{ugxCliffordQuaternPagePatch10}
\pastebutton{ugxCliffordQuaternPageEmpty10}{\showpaste}
\tab{5}\spadcommand{x * y\free{x y }}
\end{paste}\end{patch}

\begin{patch}{ugxCliffordQuaternPagePatch11}
\begin{paste}{ugxCliffordQuaternPageFull11}{ugxCliffordQuaternPageEmpty11}
\pastebutton{ugxCliffordQuaternPageFull11}{\hidepaste}
\tab{5}\spadcommand{y * x\free{x y }}
\indentrel{3}\begin{verbatim}
   (11)
     - d h - c g - b f + a e + (- c h + d g + a f + b e)e
                                                         1
   +
   (b h + a g - d f + c e)e  + (a h - b g + c f + d e)e e
                           2                           1 2
Type: CliffordAlgebra(2,Fraction Polynomial Integer,MATRIX)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxCliffordQuaternPageEmpty11}
\begin{paste}{ugxCliffordQuaternPageEmpty11}{ugxCliffordQuaternPagePatch11}
\pastebutton{ugxCliffordQuaternPageEmpty11}{\showpaste}
\tab{5}\spadcommand{y * x\free{x y }}
\end{paste}\end{patch}
```

———-

The Exterior Algebra on a Three Space

— clif.ht —

```
\begin{page}{ugxCliffordExteriorPage}{The Exterior Algebra on a Three Space}
\beginscroll

\labelSpace{4pc}
\xtc{
This is the field over which we will work, rational functions with
integer coefficients.
}{
\spadpaste{K := Fraction Polynomial Integer \bound{K}}
}
\xtc{
```

```
If we chose the three by three zero quadratic form, we obtain
the exterior algebra on \spad{e(1),e(2),e(3)}.
}{
\spadpaste{Ext := CliffordAlgebra(3, K, quadraticForm 0) \bound{Ext}
\free{K}}
}
\xtc{
This is a three dimensional vector algebra.
We define \spad{i}, \spad{j}, \spad{k} as the unit vectors.
}{
\spadpaste{i: Ext := e(1) \free{Ext}\bound{i}}
}
\xtc{
}{
\spadpaste{j: Ext := e(2) \free{Ext}\bound{j}}
}
\xtc{
}{
\spadpaste{k: Ext := e(3) \free{Ext}\bound{k}}
}
\xtc{
Now it is possible to do arithmetic.
}{
\spadpaste{x := x1*i + x2*j + x3*k \free{i j k}\bound{x}}
}
\xtc{
}{
\spadpaste{y := y1*i + y2*j + y3*k \free{i j k}\bound{y}}
}
\xtc{
}{
\spadpaste{x + y         \free{x y}}
}
\xtc{
}{
\spadpaste{x * y + y * x \free{x y}}
}
\xtc{
On an \spad{n} space, a grade \spad{p} form has a dual \spad{n-p} form.
In particular, in three space the dual of a grade two element identifies
\spad{e1*e2->e3, e2*e3->e1, e3*e1->e2}.
}{
\spadpaste{dual2 a == coefficient(a,[2,3]) * i + coefficient(a,[3,1]) *
j + coefficient(a,[1,2]) * k \free{i j k}\bound{dual2}}
}
\xtc{
The vector cross product is then given by this.
}{
\spadpaste{dual2(x*y) \free{x y dual2}}
}
```

```
\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxCliffordExteriorPagePatch1}
\begin{paste}{ugxCliffordExteriorPageFull1}{ugxCliffordExteriorPageEmpty1}
\pastebutton{ugxCliffordExteriorPageFull1}{\hidepaste}
\tab{5}\spadcommand{K := Fraction Polynomial Integer\bound{K }}
\indentrel{3}\begin{verbatim}
   (1)  Fraction Polynomial Integer
                                            Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxCliffordExteriorPageEmpty1}
\begin{paste}{ugxCliffordExteriorPageEmpty1}{ugxCliffordExteriorPagePatch1}
\pastebutton{ugxCliffordExteriorPageEmpty1}{\showpaste}
\tab{5}\spadcommand{K := Fraction Polynomial Integer\bound{K }}
\end{paste}\end{patch}

\begin{patch}{ugxCliffordExteriorPagePatch2}
\begin{paste}{ugxCliffordExteriorPageFull2}{ugxCliffordExteriorPageEmpty2}
\pastebutton{ugxCliffordExteriorPageFull2}{\hidepaste}
\tab{5}\spadcommand{
Ext := CliffordAlgebra(3, K, quadraticForm 0)\bound{Ext }\free{K }}
\indentrel{3}\begin{verbatim}
   (2)
   CliffordAlgebra(3,Fraction Polynomial Integer,MATRIX)
                                            Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxCliffordExteriorPageEmpty2}
\begin{paste}{ugxCliffordExteriorPageEmpty2}{ugxCliffordExteriorPagePatch2}
\pastebutton{ugxCliffordExteriorPageEmpty2}{\showpaste}
\tab{5}\spadcommand{
Ext := CliffordAlgebra(3, K, quadraticForm 0)\bound{Ext }\free{K }}
\end{paste}\end{patch}

\begin{patch}{ugxCliffordExteriorPagePatch3}
\begin{paste}{ugxCliffordExteriorPageFull3}{ugxCliffordExteriorPageEmpty3}
\pastebutton{ugxCliffordExteriorPageFull3}{\hidepaste}
\tab{5}\spadcommand{i: Ext := e(1)\free{Ext }\bound{i }}
\indentrel{3}\begin{verbatim}
   (3)  e
         1
Type: CliffordAlgebra(3,Fraction Polynomial Integer,MATRIX)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugxCliffordExteriorPageEmpty3}
\begin{paste}{ugxCliffordExteriorPageEmpty3}{ugxCliffordExteriorPagePatch3}
\pastebutton{ugxCliffordExteriorPageEmpty3}{\showpaste}
\tab{5}\spadcommand{i: Ext := e(1)\free{Ext }\bound{i }}
\end{paste}\end{patch}

\begin{patch}{ugxCliffordExteriorPagePatch4}
\begin{paste}{ugxCliffordExteriorPageFull4}{ugxCliffordExteriorPageEmpty4}
\pastebutton{ugxCliffordExteriorPageFull4}{\hidepaste}
\tab{5}\spadcommand{j: Ext := e(2)\free{Ext }\bound{j }}
\indentrel{3}\begin{verbatim}
   (4)  e
         2
Type: CliffordAlgebra(3,Fraction Polynomial Integer,MATRIX)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxCliffordExteriorPageEmpty4}
\begin{paste}{ugxCliffordExteriorPageEmpty4}{ugxCliffordExteriorPagePatch4}
\pastebutton{ugxCliffordExteriorPageEmpty4}{\showpaste}
\tab{5}\spadcommand{j: Ext := e(2)\free{Ext }\bound{j }}
\end{paste}\end{patch}

\begin{patch}{ugxCliffordExteriorPagePatch5}
\begin{paste}{ugxCliffordExteriorPageFull5}{ugxCliffordExteriorPageEmpty5}
\pastebutton{ugxCliffordExteriorPageFull5}{\hidepaste}
\tab{5}\spadcommand{k: Ext := e(3)\free{Ext }\bound{k }}
\indentrel{3}\begin{verbatim}
   (5)  e
         3
Type: CliffordAlgebra(3,Fraction Polynomial Integer,MATRIX)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxCliffordExteriorPageEmpty5}
\begin{paste}{ugxCliffordExteriorPageEmpty5}{ugxCliffordExteriorPagePatch5}
\pastebutton{ugxCliffordExteriorPageEmpty5}{\showpaste}
\tab{5}\spadcommand{k: Ext := e(3)\free{Ext }\bound{k }}
\end{paste}\end{patch}

\begin{patch}{ugxCliffordExteriorPagePatch6}
\begin{paste}{ugxCliffordExteriorPageFull6}{ugxCliffordExteriorPageEmpty6}
\pastebutton{ugxCliffordExteriorPageFull6}{\hidepaste}
\tab{5}\spadcommand{x := x1*i + x2*j + x3*k\free{i j k }\bound{x }}
\indentrel{3}\begin{verbatim}
   (6)  x1 e  + x2 e  + x3 e
            1       2       3
Type: CliffordAlgebra(3,Fraction Polynomial Integer,MATRIX)
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxCliffordExteriorPageEmpty6}
\begin{paste}{ugxCliffordExteriorPageEmpty6}{ugxCliffordExteriorPagePatch6}
\pastebutton{ugxCliffordExteriorPageEmpty6}{\showpaste}
\tab{5}\spadcommand{x := x1*i + x2*j + x3*k\free{i j k }\bound{x }}
\end{paste}\end{patch}

\begin{patch}{ugxCliffordExteriorPagePatch7}
\begin{paste}{ugxCliffordExteriorPageFull7}{ugxCliffordExteriorPageEmpty7}
\pastebutton{ugxCliffordExteriorPageFull7}{\hidepaste}
\tab{5}\spadcommand{y := y1*i + y2*j + y3*k\free{i j k }\bound{y }}
\indentrel{3}\begin{verbatim}
   (7)  y1 e  + y2 e  + y3 e
            1       2       3
Type: CliffordAlgebra(3,Fraction Polynomial Integer,MATRIX)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxCliffordExteriorPageEmpty7}
\begin{paste}{ugxCliffordExteriorPageEmpty7}{ugxCliffordExteriorPagePatch7}
\pastebutton{ugxCliffordExteriorPageEmpty7}{\showpaste}
\tab{5}\spadcommand{y := y1*i + y2*j + y3*k\free{i j k }\bound{y }}
\end{paste}\end{patch}

\begin{patch}{ugxCliffordExteriorPagePatch8}
\begin{paste}{ugxCliffordExteriorPageFull8}{ugxCliffordExteriorPageEmpty8}
\pastebutton{ugxCliffordExteriorPageFull8}{\hidepaste}
\tab{5}\spadcommand{x + y\free{x y }}
\indentrel{3}\begin{verbatim}
   (8)  (y1 + x1)e  + (y2 + x2)e  + (y3 + x3)e
                  1             2             3
Type: CliffordAlgebra(3,Fraction Polynomial Integer,MATRIX)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxCliffordExteriorPageEmpty8}
\begin{paste}{ugxCliffordExteriorPageEmpty8}{ugxCliffordExteriorPagePatch8}
\pastebutton{ugxCliffordExteriorPageEmpty8}{\showpaste}
\tab{5}\spadcommand{x + y\free{x y }}
\end{paste}\end{patch}

\begin{patch}{ugxCliffordExteriorPagePatch9}
\begin{paste}{ugxCliffordExteriorPageFull9}{ugxCliffordExteriorPageEmpty9}
\pastebutton{ugxCliffordExteriorPageFull9}{\hidepaste}
\tab{5}\spadcommand{x * y + y * x\free{x y }}
\indentrel{3}\begin{verbatim}
   (9)  0
Type: CliffordAlgebra(3,Fraction Polynomial Integer,MATRIX)
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxCliffordExteriorPageEmpty9}
\begin{paste}{ugxCliffordExteriorPageEmpty9}{ugxCliffordExteriorPagePatch9}
\pastebutton{ugxCliffordExteriorPageEmpty9}{\showpaste}
\tab{5}\spadcommand{x * y + y * x\free{x y }}
\end{paste}\end{patch}

\begin{patch}{ugxCliffordExteriorPagePatch10}
\begin{paste}{ugxCliffordExteriorPageFull10}{ugxCliffordExteriorPageEmpty10}
\pastebutton{ugxCliffordExteriorPageFull10}{\hidepaste}
\tab{5}\spadcommand{
dual2 a == coefficient(a,[2,3]) * i + coefficient(a,[3,1]) * _
           j + coefficient(a,[1,2]) * k\free{i j k }\bound{dual2 }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxCliffordExteriorPageEmpty10}
\begin{paste}{ugxCliffordExteriorPageEmpty10}{ugxCliffordExteriorPagePatch10}
\pastebutton{ugxCliffordExteriorPageEmpty10}{\showpaste}
\tab{5}\spadcommand{
dual2 a == coefficient(a,[2,3]) * i + coefficient(a,[3,1]) * _
      j + coefficient(a,[1,2]) * k\free{i j k }\bound{dual2 }}
\end{paste}\end{patch}

\begin{patch}{ugxCliffordExteriorPagePatch11}
\begin{paste}{ugxCliffordExteriorPageFull11}{ugxCliffordExteriorPageEmpty11}
\pastebutton{ugxCliffordExteriorPageFull11}{\hidepaste}
\tab{5}\spadcommand{dual2(x*y)\free{x y dual2 }}
\indentrel{3}\begin{verbatim}
   (11)
     (x2 y3 - x3 y2)e  + (- x1 y3 + x3 y1)e
                     1                      2
   +
     (x1 y2 - x2 y1)e
                     3
Type: CliffordAlgebra(3,Fraction Polynomial Integer,MATRIX)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxCliffordExteriorPageEmpty11}
\begin{paste}{ugxCliffordExteriorPageEmpty11}{ugxCliffordExteriorPagePatch11}
\pastebutton{ugxCliffordExteriorPageEmpty11}{\showpaste}
\tab{5}\spadcommand{dual2(x*y)\free{x y dual2 }}
\end{paste}\end{patch}
```

The Dirac Spin Algebra

— **clif.ht** —

```
\begin{page}{ugxCliffordDiracPage}{The Dirac Spin Algebra}
\beginscroll

\labelSpace{4pc}
{The Dirac Spin Algebra}
%
\xtc{
In this section we will work over the field of rational numbers.
}{
\spadpaste{K := Fraction Integer \bound{K}}
}
\xtc{
We define the quadratic form to be the Minkowski space-time metric.
}{
\spadpaste{g := matrix [[1,0,0,0], [0,-1,0,0], [0,0,-1,0], [0,0,0,-1]]
\bound{g}}
}
\xtc{
We obtain the Dirac spin algebra
used in Relativistic Quantum Field Theory.
{9.10.4.}{The Dirac Spin Algebra}
}{
\spadpaste{D := CliffordAlgebra(4,K, quadraticForm g) \free{K g}\bound{D}}
}
\xtc{
The usual notation for the basis is \texht{$\gamma$}{\spad{gamma}}
with a superscript.
For Axiom input we will use \spad{gam(i)}:
}{
\spadpaste{gam := [e(i)\$D for i in 1..4] \free{D}\bound{gam}}
}
\noindent
There are various contraction identities of the form
\begin{verbatim}
g(l,t)*gam(l)*gam(m)*gam(n)*gam(r)*gam(s)*gam(t) =
      2*(gam(s)gam(m)gam(n)gam(r) + gam(r)*gam(n)*gam(m)*gam(s))
\end{verbatim}
where a sum over \spad{l} and \spad{t} is implied.
\xtc{
Verify this identity for particular values of \spad{m,n,r,s}.
}{
\spadpaste{m := 1; n:= 2; r := 3; s := 4; \bound{m,n,r,s}}
}
\xtc{
}{
```

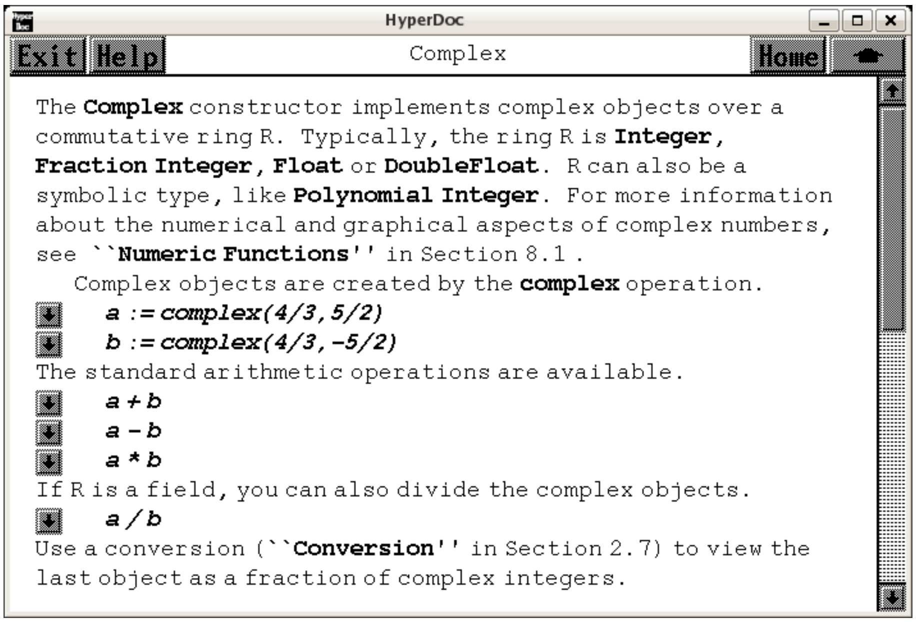
```
\spadpaste{lhs :=
reduce(+, [reduce(+,
[ g(l,t)*gam(l)*gam(m)*gam(n)*gam(r)*gam(s)*gam(t) for l in 1..4])
for t in 1..4]) \bound{lhs}\free{g gam m n r s}}
}
\xtc{
}{
\spadpaste{rhs := 2*(gam s * gam m*gam n*gam r + gam r*gam n*gam m*gam s)
\bound{rhs}\free{lhs g gam m n r s}}
}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxCliffordDiracPagePatch1}
\begin{paste}{ugxCliffordDiracPageFull1}{ugxCliffordDiracPageEmpty1}
\pastebutton{ugxCliffordDiracPageFull1}{\hidepaste}
\tab{5}\spadcommand{K := Fraction Integer\bound{K }}
\indentrel{3}\begin{verbatim}
   (1)  Fraction Integer
                                          Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxCliffordDiracPageEmpty1}
\begin{paste}{ugxCliffordDiracPageEmpty1}{ugxCliffordDiracPagePatch1}
\pastebutton{ugxCliffordDiracPageEmpty1}{\showpaste}
\tab{5}\spadcommand{K := Fraction Integer\bound{K }}
\end{paste}\end{patch}

\begin{patch}{ugxCliffordDiracPagePatch2}
\begin{paste}{ugxCliffordDiracPageFull2}{ugxCliffordDiracPageEmpty2}
\pastebutton{ugxCliffordDiracPageFull2}{\hidepaste}
\tab{5}\spadcommand{g := matrix [[1,0,0,0], [0,-1,0,0], [0,0,-1,0], [0,0,0,-1]]\bound{g }}
\indentrel{3}\begin{verbatim}
        1   0    0    0

        0  - 1   0    0
   (2)
        0   0   - 1   0

        0   0    0   - 1
                                  Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxCliffordDiracPageEmpty2}
\begin{paste}{ugxCliffordDiracPageEmpty2}{ugxCliffordDiracPagePatch2}
\pastebutton{ugxCliffordDiracPageEmpty2}{\showpaste}
\tab{5}\spadcommand{
```

```
g := matrix [[1,0,0,0], [0,-1,0,0], [0,0,-1,0], [0,0,0,-1]]\bound{g }}
\end{paste}\end{patch}

\begin{patch}{ugxCliffordDiracPagePatch3}
\begin{paste}{ugxCliffordDiracPageFull3}{ugxCliffordDiracPageEmpty3}
\pastebutton{ugxCliffordDiracPageFull3}{\hidepaste}
\tab{5}\spadcommand{
D := CliffordAlgebra(4,K, quadraticForm g)\free{K g }\bound{D }}
\indentrel{3}\begin{verbatim}
   (3)  CliffordAlgebra(4,Fraction Integer,MATRIX)
                                          Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxCliffordDiracPageEmpty3}
\begin{paste}{ugxCliffordDiracPageEmpty3}{ugxCliffordDiracPagePatch3}
\pastebutton{ugxCliffordDiracPageEmpty3}{\showpaste}
\tab{5}\spadcommand{
D := CliffordAlgebra(4,K, quadraticForm g)\free{K g }\bound{D }}
\end{paste}\end{patch}

\begin{patch}{ugxCliffordDiracPagePatch4}
\begin{paste}{ugxCliffordDiracPageFull4}{ugxCliffordDiracPageEmpty4}
\pastebutton{ugxCliffordDiracPageFull4}{\hidepaste}
\tab{5}\spadcommand{gam := [e(i)$D for i in 1..4]\free{D }\bound{gam }}
\indentrel{3}\begin{verbatim}
   (4)  [e ,e ,e ,e ]
          1  2  3  4
  Type: List CliffordAlgebra(4,Fraction Integer,MATRIX)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxCliffordDiracPageEmpty4}
\begin{paste}{ugxCliffordDiracPageEmpty4}{ugxCliffordDiracPagePatch4}
\pastebutton{ugxCliffordDiracPageEmpty4}{\showpaste}
\tab{5}\spadcommand{gam := [e(i)$D for i in 1..4]\free{D }\bound{gam }}
\end{paste}\end{patch}

\begin{patch}{ugxCliffordDiracPagePatch5}
\begin{paste}{ugxCliffordDiracPageFull5}{ugxCliffordDiracPageEmpty5}
\pastebutton{ugxCliffordDiracPageFull5}{\hidepaste}
\tab{5}\spadcommand{m := 1; n:= 2; r := 3; s := 4;\bound{m n r s }}
\indentrel{3}\begin{verbatim}
                                 Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxCliffordDiracPageEmpty5}
\begin{paste}{ugxCliffordDiracPageEmpty5}{ugxCliffordDiracPagePatch5}
\pastebutton{ugxCliffordDiracPageEmpty5}{\showpaste}
```

```
\tab{5}\spadcommand{m := 1; n:= 2; r := 3; s := 4;\bound{m n r s }}
\end{paste}\end{patch}

\begin{patch}{ugxCliffordDiracPagePatch6}
\begin{paste}{ugxCliffordDiracPageFull6}{ugxCliffordDiracPageEmpty6}
\pastebutton{ugxCliffordDiracPageFull6}{\hidepaste}
\tab{5}\spadcommand{
lhs := reduce(+, [reduce(+, _
  [ g(l,t)*gam(l)*gam(m)*gam(n)*gam(r)*gam(s)*gam(t) for l in 1..4]) _
  for t in 1..4])\bound{lhs }\free{g gam m n r s }}
\indentrel{3}\begin{verbatim}
   (6)  - 4e e e e
            1 2 3 4
       Type: CliffordAlgebra(4,Fraction Integer,MATRIX)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxCliffordDiracPageEmpty6}
\begin{paste}{ugxCliffordDiracPageEmpty6}{ugxCliffordDiracPagePatch6}
\pastebutton{ugxCliffordDiracPageEmpty6}{\showpaste}
\tab{5}\spadcommand{
lhs := reduce(+, [reduce(+, _
  [ g(l,t)*gam(l)*gam(m)*gam(n)*gam(r)*gam(s)*gam(t) for l in 1..4]) _
  for t in 1..4])\bound{lhs }\free{g gam m n r s }}
\end{paste}\end{patch}

\begin{patch}{ugxCliffordDiracPagePatch7}
\begin{paste}{ugxCliffordDiracPageFull7}{ugxCliffordDiracPageEmpty7}
\pastebutton{ugxCliffordDiracPageFull7}{\hidepaste}
\tab{5}\spadcommand{
rhs := 2*(gam s * gam m*gam n*gam r + gam r*gam n*gam m*gam s)
\bound{rhs }\free{lhs g gam m n r s }}
\indentrel{3}\begin{verbatim}
   (7)  - 4e e e e
            1 2 3 4
       Type: CliffordAlgebra(4,Fraction Integer,MATRIX)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxCliffordDiracPageEmpty7}
\begin{paste}{ugxCliffordDiracPageEmpty7}{ugxCliffordDiracPagePatch7}
\pastebutton{ugxCliffordDiracPageEmpty7}{\showpaste}
\tab{5}\spadcommand{
rhs := 2*(gam s * gam m*gam n*gam r + gam r*gam n*gam m*gam s)
\bound{rhs }\free{lhs g gam m n r s }}
\end{paste}\end{patch}
```

3.16 complex.ht

Complex



⇐ “Axiom Number Types” (NumberPage) 3.80 on page 1021
⇐ “Primes and Factorization” (ugxIntegerPrimesPage) 3.55 on page 712
⇒ “Numeric Functions” (ugProblemNumericPage) 12 on page 2079
⇒ “Conversion” (ugTypesConvertPage) 7 on page 1671

— **complex.ht** —

```
\begin{page}{ComplexXmpPage}{Complex}
\beginscroll
The \spadtype{Complex} constructor implements complex objects over a
commutative ring \spad{R}.
Typically, the ring \spad{R} is \spadtype{Integer}, \spadtype{Fraction
Integer}, \spadtype{Float} or \spadtype{DoubleFloat}.
\spad{R} can also be a symbolic type, like \spadtype{Polynomial Integer}.
For more information about the numerical and graphical aspects of complex
numbers,
see \downlink{‘‘Numeric Functions’’}{ugProblemNumericPage} in Section 8.1
\ignore{ugProblemNumeric}.

\xtc{
Complex objects are created by the \spadfunFrom{complex}{Complex} operation.
}{
\spadpaste{a := complex(4/3,5/2) \bound{a}}
```

```
}
\xtc{
}{
\spadpaste{b := complex(4/3,-5/2) \bound{b}}
}
\xtc{
The standard arithmetic operations are available.
}{
\spadpaste{a + b \free{a b}}
}
\xtc{
}{
\spadpaste{a - b \free{a b}}
}
\xtc{
}{
\spadpaste{a * b \free{a b}}
}
\xtc{
If \spad{R} is a field, you can also divide the complex objects.
}{
\spadpaste{a / b \free{a b}\bound{adb}}
}
\xtc{
Use a conversion (\downlink{``Conversion''}{ugTypesConvertPage}
in Section 2.7\ignore{ugTypesConvert}) to view the last
object as a fraction of complex integers.
}{
\spadpaste{\% :: Fraction Complex Integer \free{adb}}
}
\xtc{
The predefined macro \spad{\%i} is defined to be \spad{complex(0,1)}.
}{
\spadpaste{3.4 + 6.7 * \%i}
}
\xtc{
You can also compute the \spadfunFrom{conjugate}{Complex} and
\spadfunFrom{norm}{Complex} of a complex number.
}{
\spadpaste{conjugate a \free{a}}
}
\xtc{
}{
\spadpaste{norm a \free{a}}
}
\xtc{
The \spadfunFrom{real}{Complex} and \spadfunFrom{imag}{Complex} operations
are provided to extract the real and imaginary parts, respectively.
}{
\spadpaste{real a \free{a}}
```

```
}
\xtc{
}{
\spadpaste{imag a \free{a}}
}

\xtc{
The domain \spadtype{Complex Integer} is also called the Gaussian
integers.
If \spad{R} is the integers (or, more generally,
a \spadtype{EuclideanDomain}), you can compute greatest common divisors.
}{
\spadpaste{gcd(13 - 13*\%i,31 + 27*\%i)}
}
\xtc{
You can also compute least common multiples.
}{
\spadpaste{lcm(13 - 13*\%i,31 + 27*\%i)}
}
\xtc{
You can \spadfunFrom{factor}{Complex} Gaussian integers.
}{
\spadpaste{factor(13 - 13*\%i)}
}
\xtc{
}{
\spadpaste{factor complex(2,0)}
}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ComplexXmpPagePatch1}
\begin{paste}{ComplexXmpPageFull1}{ComplexXmpPageEmpty1}
\pastebutton{ComplexXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{a := complex(4/3,5/2)\bound{a }}
\indentrel{3}\begin{verbatim}
        4   5
   (1)   +  %i
        3   2
                         Type: Complex Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ComplexXmpPageEmpty1}
\begin{paste}{ComplexXmpPageEmpty1}{ComplexXmpPagePatch1}
\pastebutton{ComplexXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{a := complex(4/3,5/2)\bound{a }}
\end{paste}\end{patch}
```

```
\begin{patch}{ComplexXmpPagePatch2}
\begin{paste}{ComplexXmpPageFull2}{ComplexXmpPageEmpty2}
\pastebutton{ComplexXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{b := complex(4/3,-5/2)\bound{b }}
\indentrel{3}\begin{verbatim}
        4   5
   (2)   -  %i
        3   2
                         Type: Complex Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ComplexXmpPageEmpty2}
\begin{paste}{ComplexXmpPageEmpty2}{ComplexXmpPagePatch2}
\pastebutton{ComplexXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{b := complex(4/3,-5/2)\bound{b }}
\end{paste}\end{patch}

\begin{patch}{ComplexXmpPagePatch3}
\begin{paste}{ComplexXmpPageFull3}{ComplexXmpPageEmpty3}
\pastebutton{ComplexXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{a + b\free{a b }}
\indentrel{3}\begin{verbatim}
        8
   (3)
        3
                         Type: Complex Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ComplexXmpPageEmpty3}
\begin{paste}{ComplexXmpPageEmpty3}{ComplexXmpPagePatch3}
\pastebutton{ComplexXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{a + b\free{a b }}
\end{paste}\end{patch}

\begin{patch}{ComplexXmpPagePatch4}
\begin{paste}{ComplexXmpPageFull4}{ComplexXmpPageEmpty4}
\pastebutton{ComplexXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{a - b\free{a b }}
\indentrel{3}\begin{verbatim}
   (4)  5%i
                         Type: Complex Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ComplexXmpPageEmpty4}
\begin{paste}{ComplexXmpPageEmpty4}{ComplexXmpPagePatch4}
\pastebutton{ComplexXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{a - b\free{a b }}
```

```
\end{paste}\end{patch}

\begin{patch}{ComplexXmpPagePatch5}
\begin{paste}{ComplexXmpPageFull5}{ComplexXmpPageEmpty5}
\pastebutton{ComplexXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{a * b\free{a b }}
\indentrel{3}\begin{verbatim}
        289
   (5)
         36
                         Type: Complex Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ComplexXmpPageEmpty5}
\begin{paste}{ComplexXmpPageEmpty5}{ComplexXmpPagePatch5}
\pastebutton{ComplexXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{a * b\free{a b }}
\end{paste}\end{patch}

\begin{patch}{ComplexXmpPagePatch6}
\begin{paste}{ComplexXmpPageFull6}{ComplexXmpPageEmpty6}
\pastebutton{ComplexXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{a / b\free{a b }\bound{adb }}
\indentrel{3}\begin{verbatim}
          161   240
   (6)  -  +  %i
          289   289
                         Type: Complex Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ComplexXmpPageEmpty6}
\begin{paste}{ComplexXmpPageEmpty6}{ComplexXmpPagePatch6}
\pastebutton{ComplexXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{a / b\free{a b }\bound{adb }}
\end{paste}\end{patch}

\begin{patch}{ComplexXmpPagePatch7}
\begin{paste}{ComplexXmpPageFull7}{ComplexXmpPageEmpty7}
\pastebutton{ComplexXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{\% :: Fraction Complex Integer\free{adb }}
\indentrel{3}\begin{verbatim}
        - 15 + 8%i
   (7)
         15 + 8%i
                         Type: Fraction Complex Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ComplexXmpPageEmpty7}
\begin{paste}{ComplexXmpPageEmpty7}{ComplexXmpPagePatch7}
\pastebutton{ComplexXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{\% :: Fraction Complex Integer\free{adb }}
\end{paste}\end{patch}

\begin{patch}{ComplexXmpPagePatch8}
\begin{paste}{ComplexXmpPageFull8}{ComplexXmpPageEmpty8}
\pastebutton{ComplexXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{3.4 + 6.7 * \%i}
\indentrel{3}\begin{verbatim}
   (8)  3.4 + 6.7 %i
                                   Type: Complex Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ComplexXmpPageEmpty8}
\begin{paste}{ComplexXmpPageEmpty8}{ComplexXmpPagePatch8}
\pastebutton{ComplexXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{3.4 + 6.7 * \%i}
\end{paste}\end{patch}

\begin{patch}{ComplexXmpPagePatch9}
\begin{paste}{ComplexXmpPageFull9}{ComplexXmpPageEmpty9}
\pastebutton{ComplexXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{conjugate a\free{a }}
\indentrel{3}\begin{verbatim}
        4   5
   (9)   -  %i
        3   2
                         Type: Complex Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ComplexXmpPageEmpty9}
\begin{paste}{ComplexXmpPageEmpty9}{ComplexXmpPagePatch9}
\pastebutton{ComplexXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{conjugate a\free{a }}
\end{paste}\end{patch}

\begin{patch}{ComplexXmpPagePatch10}
\begin{paste}{ComplexXmpPageFull10}{ComplexXmpPageEmpty10}
\pastebutton{ComplexXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{norm a\free{a }}
\indentrel{3}\begin{verbatim}
         289
   (10)
          36
                                Type: Fraction Integer
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ComplexXmpPageEmpty10}
\begin{paste}{ComplexXmpPageEmpty10}{ComplexXmpPagePatch10}
\pastebutton{ComplexXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{norm a\free{a }}
\end{paste}\end{patch}

\begin{patch}{ComplexXmpPagePatch11}
\begin{paste}{ComplexXmpPageFull11}{ComplexXmpPageEmpty11}
\pastebutton{ComplexXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{real a\free{a }}
\indentrel{3}\begin{verbatim}
         4
   (11)
         3
                                  Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ComplexXmpPageEmpty11}
\begin{paste}{ComplexXmpPageEmpty11}{ComplexXmpPagePatch11}
\pastebutton{ComplexXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{real a\free{a }}
\end{paste}\end{patch}

\begin{patch}{ComplexXmpPagePatch12}
\begin{paste}{ComplexXmpPageFull12}{ComplexXmpPageEmpty12}
\pastebutton{ComplexXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{imag a\free{a }}
\indentrel{3}\begin{verbatim}
         5
   (12)
         2
                                  Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ComplexXmpPageEmpty12}
\begin{paste}{ComplexXmpPageEmpty12}{ComplexXmpPagePatch12}
\pastebutton{ComplexXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{imag a\free{a }}
\end{paste}\end{patch}

\begin{patch}{ComplexXmpPagePatch13}
\begin{paste}{ComplexXmpPageFull13}{ComplexXmpPageEmpty13}
\pastebutton{ComplexXmpPageFull13}{\hidepaste}
\tab{5}\spadcommand{gcd(13 - 13*\%i,31 + 27*\%i)}
\indentrel{3}\begin{verbatim}
   (13)  5 + %i
```

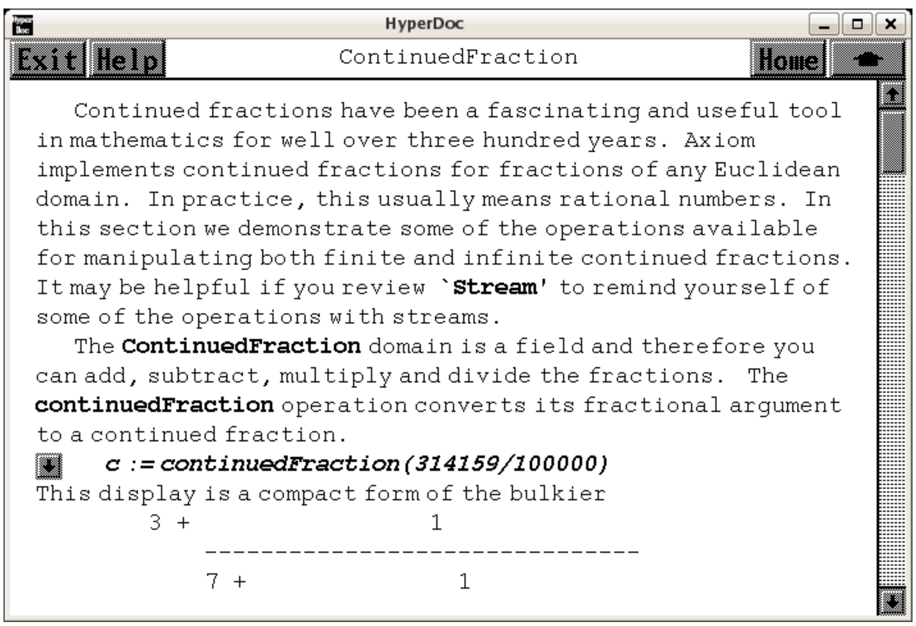
```
                                   Type: Complex Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ComplexXmpPageEmpty13}
\begin{paste}{ComplexXmpPageEmpty13}{ComplexXmpPagePatch13}
\pastebutton{ComplexXmpPageEmpty13}{\showpaste}
\tab{5}\spadcommand{gcd(13 - 13*\%i,31 + 27*\%i)}
\end{paste}\end{patch}

\begin{patch}{ComplexXmpPagePatch14}
\begin{paste}{ComplexXmpPageFull14}{ComplexXmpPageEmpty14}
\pastebutton{ComplexXmpPageFull14}{\hidepaste}
\tab{5}\spadcommand{lcm(13 - 13*\%i,31 + 27*\%i)}
\indentrel{3}\begin{verbatim}
   (14)  143 - 39%i
                                   Type: Complex Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ComplexXmpPageEmpty14}
\begin{paste}{ComplexXmpPageEmpty14}{ComplexXmpPagePatch14}
\pastebutton{ComplexXmpPageEmpty14}{\showpaste}
\tab{5}\spadcommand{lcm(13 - 13*\%i,31 + 27*\%i)}
\end{paste}\end{patch}

\begin{patch}{ComplexXmpPagePatch15}
\begin{paste}{ComplexXmpPageFull15}{ComplexXmpPageEmpty15}
\pastebutton{ComplexXmpPageFull15}{\hidepaste}
\tab{5}\spadcommand{factor(13 - 13*\%i)}
\indentrel{3}\begin{verbatim}
   (15)  - (1 + %i)(2 + 3%i)(3 + 2%i)
                          Type: Factored Complex Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ComplexXmpPageEmpty15}
\begin{paste}{ComplexXmpPageEmpty15}{ComplexXmpPagePatch15}
\pastebutton{ComplexXmpPageEmpty15}{\showpaste}
\tab{5}\spadcommand{factor(13 - 13*\%i)}
\end{paste}\end{patch}

\begin{patch}{ComplexXmpPagePatch16}
\begin{paste}{ComplexXmpPageFull16}{ComplexXmpPageEmpty16}
\pastebutton{ComplexXmpPageFull16}{\hidepaste}
\tab{5}\spadcommand{factor complex(2,0)}
\indentrel{3}\begin{verbatim}
                      2
   (16)  - %i (1 + %i)
                          Type: Factored Complex Integer
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ComplexXmpPageEmpty16}
\begin{paste}{ComplexXmpPageEmpty16}{ComplexXmpPagePatch16}
\pastebutton{ComplexXmpPageEmpty16}{\showpaste}
\tab{5}\spadcommand{factor complex(2,0)}
\end{paste}\end{patch}
```

———-

3.17 contfrac.ht

ContinuedFraction



⇐ "Axiom Number Types" (NumberPage) 3.80 on page 1021
⇒ "Stream" (StreamXmpPage) 3.102 on page 1263

— contfrac.ht —

```
\begin{page}{ContinuedFractionXmpPage}{ContinuedFraction}
\beginscroll

Continued fractions have been a fascinating and useful tool in
mathematics for well over three hundred years.  Axiom implements
```

```
continued fractions for fractions of any Euclidean domain.  In
practice, this usually means rational numbers.  In this section we
demonstrate some of the operations available for manipulating both
finite and infinite continued fractions.  It may be helpful if you
review \downlink{`Stream'}{StreamXmpPage}\ignore{Stream} to remind
yourself of some of the operations with streams.

The \spadtype{ContinuedFraction} domain is a field and therefore you can
add, subtract, multiply and divide the fractions.
\xtc{
The \spadfunFrom{continuedFraction}{ContinuedFraction} operation
converts its fractional argument to a continued fraction.
}{
\spadpaste{c := continuedFraction(314159/100000) \bound{c}}
}
%
This display is a compact form of the bulkier
\texht{\narrowDisplay{%
3 + {\displaystyle 1 \over {\displaystyle
7 + {1 \over {\displaystyle
15 + {1 \over {\displaystyle
1 + {1 \over {\displaystyle
25 + {1 \over {\displaystyle
1 + {1 \over {\displaystyle
7 + {1 \over 4}}}}}}}}}}}}}%
}{
\begin{verbatim}
        3 +              1
            -------------------------------
            7 +              1
                ---------------------------
                15 +            1
                     ----------------------
                     1 +          1
                         ------------------
                         25 +       1
                              -------------
                              1 +     1
                                  ---------
                                  7 +   1
                                      -----
                                        4
\end{verbatim}
}
You can write any rational number in a similar form.  The fraction
will be finite and you can always take the ``numerators'' to be
\spad{1}.  That is, any rational number can be written as a simple,
finite continued fraction of the form
\texht{\narrowDisplay{%
a_1 + {\displaystyle 1 \over {\displaystyle
```

```
a_2 + {1 \over {\displaystyle
a_3 + {1 \over {\displaystyle \ddots
a_{n-1} + {1 \over a_n}}}}}}}}%
}{
\begin{verbatim}
        a(1) +           1
               -------------------------
               a(2) +          1
                      --------------------
                      a(3) +
                             .
                              .
                               .
                                     1
                               -------------
                               a(n-1) +  1
                                        ----
                                        a(n)
\end{verbatim}
}
\xtc{
The \texht{$a_i$}{\spad{a(i)}} are called partial quotients and the
operation \spadfunFrom{partialQuotients}{ContinuedFraction} creates a
stream of them.
}{
\spadpaste{partialQuotients c \free{c}}
}
\xtc{
By considering more and more of the fraction, you get the
\spadfunFrom{convergents}{ContinuedFraction}.
For example, the first convergent is \texht{$a_1$}{\spad{a(1)}},
the second is
\texht{$a_1 + 1/a_2$}{\spad{a(1) + 1/a(2)}} and so on.
}{
\spadpaste{convergents c \free{c}}
}
%
\xtc{
Since this is a finite continued fraction, the last convergent is
the original rational number, in reduced form.
The result of \spadfunFrom{approximants}{ContinuedFraction}
is always an infinite stream, though it may just repeat the ``last''
value.
}{
\spadpaste{approximants c \free{c}}
}
\xtc{
Inverting \spad{c} only changes the partial quotients of its fraction
by inserting a \spad{0} at the beginning of the list.
}{
```

```
\spadpaste{pq := partialQuotients(1/c) \free{c}\bound{pq}}
}
\xtc{
Do this to recover the original continued fraction from this list of
partial quotients.
The three-argument form of the
\spadfunFrom{continuedFraction}{ContinuedFraction} operation takes an
element which is the whole part of the fraction, a stream of elements
which are the numerators of the fraction, and a stream of elements which
are the denominators of the fraction.
}{
\spadpaste{continuedFraction(first pq,repeating [1],rest pq) \free{pq}}
}
\xtc{
The streams need not be finite for
\spadfunFrom{continuedFraction}{ContinuedFraction}.
Can you guess which irrational number has the following continued
fraction?
See the end of this section for the answer.
}{
\spadpaste{z:=continuedFraction(3,repeating [1],repeating [3,6]) \bound{z}}
}
%

In 1737 Euler discovered the infinite continued fraction expansion
\texht{\narrowDisplay{%
{{e - 1} \over 2} =
{1 \over {\displaystyle
1 + {1 \over {\displaystyle
6 + {1 \over {\displaystyle
10 + {1 \over {\displaystyle
14 + \cdots}}}}}}}}%
}{
\begin{verbatim}
        e - 1             1
        ----- = ---------------------
          2     1 +         1
                    -----------------
                    6 +       1
                        -------------
                        10 +    1
                             --------
                             14 + ...
\end{verbatim}
}
We use this expansion to compute rational and floating point
approximations of \spad{e}.\footnote{For this and other interesting
expansions, see C. D. Olds, {\it Continued Fractions,}
New Mathematical Library, (New York: Random House, 1963), pp.
134--139.}
```

```
\xtc{
By looking at the above expansion, we see that the whole part is
\spad{0} and the numerators are all equal to \spad{1}.  This
constructs the stream of denominators.
}{
\spadpaste{dens:Stream Integer := cons(1,generate((x+->x+4),6))
\bound{dens}}
}
\xtc{
Therefore this is the continued fraction expansion for
\texht{$(e - 1) / 2$}{\spad{(e-1)/2}}.
}{
\spadpaste{cf := continuedFraction(0,repeating [1],dens)
\free{dens}\bound{cf}}
}
\xtc{
These are the rational number convergents.
}{
\spadpaste{ccf := convergents cf \free{cf}\bound{ccf}}
}
\xtc{
You can get rational convergents for \spad{e} by multiplying by \spad{2} and
adding \spad{1}.
}{
\spadpaste{eConvergents := [2*e + 1 for e in ccf] \bound{ec}\free{ccf}}
}
%
\xtc{
You can also compute the floating point approximations to these convergents.
}{
\spadpaste{eConvergents :: Stream Float \free{ec}}
}
\xtc{
Compare this to the value of \spad{e} computed by the
\spadfunFrom{exp}{Float} operation in \spadtype{Float}.
}{
\spadpaste{exp 1.0}
}

In about 1658, Lord Brouncker established the following expansion
for \texht{$4 / \pi$}{\spad{4/pi}}.
\texht{\narrowDisplay{%
1 + {\displaystyle
1 \over {\displaystyle
2 + {9 \over {\displaystyle
2 + {25 \over {\displaystyle
2 + {49 \over {\displaystyle
2 + {81 \over {\displaystyle
2 + \cdots}}}}}}}}}}}%
```

```
}{
\begin{verbatim}
        1 +           1
            -----------------------
            2 +         9
                -------------------
                2 +       25
                    ---------------
                    2 +      49
                        -----------
                        2 +    81
                            -------
                            2 + ...
\end{verbatim}
}
\xtc{
Let's use this expansion to compute rational and floating point
approximations for \texht{$\pi$}{\spad{pi}}.
}{
\spadpaste{
cf := continuedFraction(1,[(2*i+1)**2 for i in 0..],repeating [2])
\bound{cf1}}
}
\xtc{
}{
\spadpaste{ccf := convergents cf \free{cf1}\bound{ccf1}}
}
\xtc{
}{
\spadpaste{
piConvergents := [4/p for p in ccf] \bound{piConvergents}\free{ccf1}}
}
\xtc{
As you can see, the values are converging to
\texht{$\pi$}{\spad{pi}} = 3.14159265358979323846...,
but not very quickly.
}{
\spadpaste{piConvergents :: Stream Float \free{piConvergents}}
}

\xtc{
You need not restrict yourself to continued fractions of integers.
Here is an expansion for a quotient of Gaussian integers.
}{
\spadpaste{continuedFraction((- 122 + 597*\%i)/(4 - 4*\%i))}
}
\xtc{
This is an expansion for a quotient of polynomials in one variable
with rational number coefficients.
}{
```

```
\spadpaste{r : Fraction UnivariatePolynomial(x,Fraction Integer)
\bound{rdec}}
}
\xtc{
}{
\spadpaste{r := ((x - 1) * (x - 2)) / ((x-3) * (x-4)) \free{rdec}
\bound{r}}
}
\xtc{
}{
\spadpaste{continuedFraction r \free{r}}
}

To conclude this section, we give you evidence that
\texht{\narrowDisplay{%
z =
{3+\zag{1}{3}+\zag{1}{6}+\zag{1}{3}+\zag{1}{6}+\zag{1}{3}+\zag{1}{6}+
\zag{1}{3}+\zag{1}{6}+\zag{1}{3}+\zag{1}{6}+...}}%
}{
\begin{verbatim}
    z = 3 +           1
            -----------------------
            3 +         1
                -------------------
                6 +       1
                    ---------------
                    3 +     1
                        -----------
                        6 +   1
                            -------
                            3 + ...
\end{verbatim}
}
is the expansion of \texht{$\sqrt{11}$}{the square root of \spad{11}}.
%
\xtc{
}{
\spadpaste{[i*i for i in convergents(z) :: Stream Float] \free{z}}
}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ContinuedFractionXmpPagePatch1}
\begin{paste}{ContinuedFractionXmpPageFull1}{ContinuedFractionXmpPageEmpty1}
\pastebutton{ContinuedFractionXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{c := continuedFraction(314159/100000)\bound{c }}
\indentrel{3}\begin{verbatim}
   (1)
           1      1       1      1       1      1
```

```
     3 +  +  +  +  +  +
          7      15      1      25      1      7
  +
      1

     4
                        Type: ContinuedFraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ContinuedFractionXmpPageEmpty1}
\begin{paste}{ContinuedFractionXmpPageEmpty1}
{ContinuedFractionXmpPagePatch1}
\pastebutton{ContinuedFractionXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{c := continuedFraction(314159/100000)\bound{c }}
\end{paste}\end{patch}

\begin{patch}{ContinuedFractionXmpPagePatch2}
\begin{paste}{ContinuedFractionXmpPageFull2}
{ContinuedFractionXmpPageEmpty2}
\pastebutton{ContinuedFractionXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{partialQuotients c\free{c }}
\indentrel{3}\begin{verbatim}
   (2)  [3,7,15,1,25,1,7,4]
                                   Type: Stream Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ContinuedFractionXmpPageEmpty2}
\begin{paste}{ContinuedFractionXmpPageEmpty2}
{ContinuedFractionXmpPagePatch2}
\pastebutton{ContinuedFractionXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{partialQuotients c\free{c }}
\end{paste}\end{patch}

\begin{patch}{ContinuedFractionXmpPagePatch3}
\begin{paste}{ContinuedFractionXmpPageFull3}
{ContinuedFractionXmpPageEmpty3}
\pastebutton{ContinuedFractionXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{convergents c\free{c }}
\indentrel{3}\begin{verbatim}
           22 333 355 9208 9563 76149 314159
   (3)  [3,,,,,,,]
            7 106 113 2931 3044 24239 100000
                          Type: Stream Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ContinuedFractionXmpPageEmpty3}
\begin{paste}{ContinuedFractionXmpPageEmpty3}
```

```
{ContinuedFractionXmpPagePatch3}
\pastebutton{ContinuedFractionXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{convergents c\free{c }}
\end{paste}\end{patch}

\begin{patch}{ContinuedFractionXmpPagePatch4}
\begin{paste}{ContinuedFractionXmpPageFull4}
{ContinuedFractionXmpPageEmpty4}
\pastebutton{ContinuedFractionXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{approximants c\free{c }}
\indentrel{3}\begin{verbatim}
                                         ------
           22 333 355 9208 9563 76149 314159
   (4)  [3,,,,,,,,]
            7 106 113 2931 3044 24239 100000
                          Type: Stream Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ContinuedFractionXmpPageEmpty4}
\begin{paste}{ContinuedFractionXmpPageEmpty4}
{ContinuedFractionXmpPagePatch4}
\pastebutton{ContinuedFractionXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{approximants c\free{c }}
\end{paste}\end{patch}

\begin{patch}{ContinuedFractionXmpPagePatch5}
\begin{paste}{ContinuedFractionXmpPageFull5}
{ContinuedFractionXmpPageEmpty5}
\pastebutton{ContinuedFractionXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{pq := partialQuotients(1/c)\free{c }\bound{pq }}
\indentrel{3}\begin{verbatim}
   (5)  [0,3,7,15,1,25,1,7,4]
                                   Type: Stream Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ContinuedFractionXmpPageEmpty5}
\begin{paste}{ContinuedFractionXmpPageEmpty5}
{ContinuedFractionXmpPagePatch5}
\pastebutton{ContinuedFractionXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{pq := partialQuotients(1/c)\free{c }\bound{pq }}
\end{paste}\end{patch}

\begin{patch}{ContinuedFractionXmpPagePatch6}
\begin{paste}{ContinuedFractionXmpPageFull6}
{ContinuedFractionXmpPageEmpty6}
\pastebutton{ContinuedFractionXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{
continuedFraction(first pq,repeating [1],rest pq)\free{pq }}
```

```
\indentrel{3}\begin{verbatim}
   (6)
       1      1      1       1      1       1
      +  +  +  +  +
      3      7      15      1      25      1
   +
       1      1
      +
      7      4
                        Type: ContinuedFraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ContinuedFractionXmpPageEmpty6}
\begin{paste}{ContinuedFractionXmpPageEmpty6}
{ContinuedFractionXmpPagePatch6}
\pastebutton{ContinuedFractionXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{
continuedFraction(first pq,repeating [1],rest pq)\free{pq }}
\end{paste}\end{patch}

\begin{patch}{ContinuedFractionXmpPagePatch7}
\begin{paste}{ContinuedFractionXmpPageFull7}
{ContinuedFractionXmpPageEmpty7}
\pastebutton{ContinuedFractionXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{
z:=continuedFraction(3,repeating [1],repeating [3,6])\bound{z }}
\indentrel{3}\begin{verbatim}
   (7)
          1      1      1      1      1      1
     3 +  +  +  +  +  +
          3      6      3      6      3      6
   +
       1      1      1      1
      +  +  +  + ...
      3      6      3      6
                        Type: ContinuedFraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ContinuedFractionXmpPageEmpty7}
\begin{paste}{ContinuedFractionXmpPageEmpty7}
{ContinuedFractionXmpPagePatch7}
\pastebutton{ContinuedFractionXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{
z:=continuedFraction(3,repeating [1],repeating [3,6])\bound{z }}
\end{paste}\end{patch}

\begin{patch}{ContinuedFractionXmpPagePatch8}
\begin{paste}{ContinuedFractionXmpPageFull8}
```

```
{ContinuedFractionXmpPageEmpty8}
\pastebutton{ContinuedFractionXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{
dens:Stream Integer := cons(1,generate((x+->x+4),6))\bound{dens }}
\indentrel{3}\begin{verbatim}
   (8)  [1,6,10,14,18,22,26,30,34,38,...]
                                  Type: Stream Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ContinuedFractionXmpPageEmpty8}
\begin{paste}{ContinuedFractionXmpPageEmpty8}
{ContinuedFractionXmpPagePatch8}
\pastebutton{ContinuedFractionXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{
dens:Stream Integer := cons(1,generate((x+->x+4),6))\bound{dens }}
\end{paste}\end{patch}

\begin{patch}{ContinuedFractionXmpPagePatch9}
\begin{paste}{ContinuedFractionXmpPageFull9}
{ContinuedFractionXmpPageEmpty9}
\pastebutton{ContinuedFractionXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{
cf := continuedFraction(0,repeating [1],dens)\free{dens }\bound{cf }}
\indentrel{3}\begin{verbatim}
   (9)
       1      1      1       1       1       1
      +  +  +  +  +
      1      6      10      14      18      22
   +
       1       1       1       1
      +  +  +  + ...
      26      30      34      38
                       Type: ContinuedFraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ContinuedFractionXmpPageEmpty9}
\begin{paste}{ContinuedFractionXmpPageEmpty9}
{ContinuedFractionXmpPagePatch9}
\pastebutton{ContinuedFractionXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{
cf := continuedFraction(0,repeating [1],dens)\free{dens }\bound{cf }}
\end{paste}\end{patch}

\begin{patch}{ContinuedFractionXmpPagePatch10}
\begin{paste}{ContinuedFractionXmpPageFull10}
{ContinuedFractionXmpPageEmpty10}
\pastebutton{ContinuedFractionXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{ccf := convergents cf\free{cf }\bound{ccf }}
```

```
\indentrel{3}\begin{verbatim}
   (10)
          6  61   860  15541  342762   8927353
   [0, 1, , , , , , ,
          7  71  1001  18089  398959  10391023
    268163352   9126481321
     , , ...]
    312129649  10622799089
                          Type: Stream Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ContinuedFractionXmpPageEmpty10}
\begin{paste}{ContinuedFractionXmpPageEmpty10}
{ContinuedFractionXmpPagePatch10}
\pastebutton{ContinuedFractionXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{ccf := convergents cf\free{cf }\bound{ccf }}
\end{paste}\end{patch}

\begin{patch}{ContinuedFractionXmpPagePatch11}
\begin{paste}{ContinuedFractionXmpPageFull11}
{ContinuedFractionXmpPageEmpty11}
\pastebutton{ContinuedFractionXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{
eConvergents := [2*e + 1 for e in ccf]\bound{ec }\free{ccf }}
\indentrel{3}\begin{verbatim}
   (11)
          19  193  2721  49171  1084483  28245729
   [1, 3, , , , , , ,
           7   71  1001  18089   398959  10391023
    848456353  28875761731
     , , ...]
    312129649  10622799089
                          Type: Stream Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ContinuedFractionXmpPageEmpty11}
\begin{paste}{ContinuedFractionXmpPageEmpty11}
{ContinuedFractionXmpPagePatch11}
\pastebutton{ContinuedFractionXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{
eConvergents := [2*e + 1 for e in ccf]\bound{ec }\free{ccf }}
\end{paste}\end{patch}

\begin{patch}{ContinuedFractionXmpPagePatch12}
\begin{paste}{ContinuedFractionXmpPageFull12}
{ContinuedFractionXmpPageEmpty12}
\pastebutton{ContinuedFractionXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{eConvergents :: Stream Float\free{ec }}
```

```
\indentrel{3}\begin{verbatim}
   (12)
   [1.0, 3.0, 2.7142857142 857142857,
    2.7183098591 549295775, 2.7182817182 817182817,
    2.7182818287 356957267, 2.7182818284 585634113,
    2.7182818284 590458514, 2.7182818284 590452348,
    2.7182818284 590452354, ...]
                                      Type: Stream Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ContinuedFractionXmpPageEmpty12}
\begin{paste}{ContinuedFractionXmpPageEmpty12}
{ContinuedFractionXmpPagePatch12}
\pastebutton{ContinuedFractionXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{eConvergents :: Stream Float\free{ec }}
\end{paste}\end{patch}

\begin{patch}{ContinuedFractionXmpPagePatch13}
\begin{paste}{ContinuedFractionXmpPageFull13}
{ContinuedFractionXmpPageEmpty13}
\pastebutton{ContinuedFractionXmpPageFull13}{\hidepaste}
\tab{5}\spadcommand{exp 1.0}
\indentrel{3}\begin{verbatim}
   (13)  2.7182818284 590452354
                                             Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ContinuedFractionXmpPageEmpty13}
\begin{paste}{ContinuedFractionXmpPageEmpty13}
{ContinuedFractionXmpPagePatch13}
\pastebutton{ContinuedFractionXmpPageEmpty13}{\showpaste}
\tab{5}\spadcommand{exp 1.0}
\end{paste}\end{patch}

\begin{patch}{ContinuedFractionXmpPagePatch14}
\begin{paste}{ContinuedFractionXmpPageFull14}
{ContinuedFractionXmpPageEmpty14}
\pastebutton{ContinuedFractionXmpPageFull14}{\hidepaste}
\tab{5}\spadcommand{
cf := continuedFraction(1,[(2*i+1)**2 for i in 0..],repeating [2])
\bound{cf1 }}
\indentrel{3}\begin{verbatim}
   (14)
           1      9      25      49      81
     1 +  +  +  +  +
          2      2      2       2       2
   +
     121      169      225      289      361
```

```
    +  +  +  +  + ...
     2       2       2       2       2
                        Type: ContinuedFraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ContinuedFractionXmpPageEmpty14}
\begin{paste}{ContinuedFractionXmpPageEmpty14}
{ContinuedFractionXmpPagePatch14}
\pastebutton{ContinuedFractionXmpPageEmpty14}{\showpaste}
\tab{5}\spadcommand{
cf := continuedFraction(1,[(2*i+1)**2 for i in 0..],repeating [2])
\bound{cf1 }}
\end{paste}\end{patch}

\begin{patch}{ContinuedFractionXmpPagePatch15}
\begin{paste}{ContinuedFractionXmpPageFull15}
{ContinuedFractionXmpPageEmpty15}
\pastebutton{ContinuedFractionXmpPageFull15}{\hidepaste}
\tab{5}\spadcommand{ccf := convergents cf\free{cf1 }\bound{ccf1 }}
\indentrel{3}\begin{verbatim}
   (15)
      3 15 105 315 3465 45045 45045 765765 14549535
   [1,,,,,,,,,,,...]
      2 13  76 263 2578 36979 33976 622637 11064338
                          Type: Stream Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ContinuedFractionXmpPageEmpty15}
\begin{paste}{ContinuedFractionXmpPageEmpty15}
{ContinuedFractionXmpPagePatch15}
\pastebutton{ContinuedFractionXmpPageEmpty15}{\showpaste}
\tab{5}\spadcommand{ccf := convergents cf\free{cf1 }\bound{ccf1 }}
\end{paste}\end{patch}

\begin{patch}{ContinuedFractionXmpPagePatch16}
\begin{paste}{ContinuedFractionXmpPageFull16}
{ContinuedFractionXmpPageEmpty16}
\pastebutton{ContinuedFractionXmpPageFull16}{\hidepaste}
\tab{5}\spadcommand{
piConvergents := [4/p for p in ccf]\bound{piConvergents }\free{ccf1 }}
\indentrel{3}\begin{verbatim}
   (16)
       8  52  304  1052  10312  147916  135904  2490548
   [4, , , , , , , , ,
       3  15  105   315   3465   45045   45045   765765
    44257352
    , ...]
    14549535
```

```
                              Type: Stream Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ContinuedFractionXmpPageEmpty16}
\begin{paste}{ContinuedFractionXmpPageEmpty16}
{ContinuedFractionXmpPagePatch16}
\pastebutton{ContinuedFractionXmpPageEmpty16}{\showpaste}
\tab{5}\spadcommand{
piConvergents := [4/p for p in ccf]\bound{piConvergents }\free{ccf1 }}
\end{paste}\end{patch}

\begin{patch}{ContinuedFractionXmpPagePatch17}
\begin{paste}{ContinuedFractionXmpPageFull17}
{ContinuedFractionXmpPageEmpty17}
\pastebutton{ContinuedFractionXmpPageFull17}{\hidepaste}
\tab{5}\spadcommand{piConvergents :: Stream Float\free{piConvergents }}
\indentrel{3}\begin{verbatim}
   (17)
   [4.0, 2.6666666666 666666667, 3.4666666666 666666667,
    2.8952380952 380952381, 3.3396825396 825396825,
    2.9760461760 461760462, 3.2837384837 384837385,
    3.0170718170 718170718, 3.2523659347 188758953,
    3.0418396189 294022111, ...]
                                         Type: Stream Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ContinuedFractionXmpPageEmpty17}
\begin{paste}{ContinuedFractionXmpPageEmpty17}
{ContinuedFractionXmpPagePatch17}
\pastebutton{ContinuedFractionXmpPageEmpty17}{\showpaste}
\tab{5}\spadcommand{piConvergents :: Stream Float\free{piConvergents }}
\end{paste}\end{patch}

\begin{patch}{ContinuedFractionXmpPagePatch18}
\begin{paste}{ContinuedFractionXmpPageFull18}
{ContinuedFractionXmpPageEmpty18}
\pastebutton{ContinuedFractionXmpPageFull18}{\hidepaste}
\tab{5}\spadcommand{continuedFraction((- 122 + 597*\%i)/(4 - 4*\%i))}
\indentrel{3}\begin{verbatim}
                                1             1
   (18)  - 90 + 59%i +  +
                            1 - 2%i      - 1 + 2%i
                Type: ContinuedFraction Complex Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ContinuedFractionXmpPageEmpty18}
\begin{paste}{ContinuedFractionXmpPageEmpty18}
```

```
{ContinuedFractionXmpPagePatch18}
\pastebutton{ContinuedFractionXmpPageEmpty18}{\showpaste}
\tab{5}\spadcommand{continuedFraction((- 122 + 597*\%i)/(4 - 4*\%i))}
\end{paste}\end{patch}

\begin{patch}{ContinuedFractionXmpPagePatch19}
\begin{paste}{ContinuedFractionXmpPageFull19}
{ContinuedFractionXmpPageEmpty19}
\pastebutton{ContinuedFractionXmpPageFull19}{\hidepaste}
\tab{5}\spadcommand{
r : Fraction UnivariatePolynomial(x,Fraction Integer)\bound{rdec }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ContinuedFractionXmpPageEmpty19}
\begin{paste}{ContinuedFractionXmpPageEmpty19}
{ContinuedFractionXmpPagePatch19}
\pastebutton{ContinuedFractionXmpPageEmpty19}{\showpaste}
\tab{5}\spadcommand{
r : Fraction UnivariatePolynomial(x,Fraction Integer)\bound{rdec }}
\end{paste}\end{patch}

\begin{patch}{ContinuedFractionXmpPagePatch20}
\begin{paste}{ContinuedFractionXmpPageFull20}
{ContinuedFractionXmpPageEmpty20}
\pastebutton{ContinuedFractionXmpPageFull20}{\hidepaste}
\tab{5}\spadcommand{
r := ((x - 1) * (x - 2)) / ((x-3) * (x-4))\free{rdec }\bound{r }}
\indentrel{3}\begin{verbatim}
           2
          x  - 3x + 2
   (20)
          2
         x  - 7x + 12
Type: Fraction UnivariatePolynomial(x,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ContinuedFractionXmpPageEmpty20}
\begin{paste}{ContinuedFractionXmpPageEmpty20}
{ContinuedFractionXmpPagePatch20}
\pastebutton{ContinuedFractionXmpPageEmpty20}{\showpaste}
\tab{5}\spadcommand{
r := ((x - 1) * (x - 2)) / ((x-3) * (x-4))\free{rdec }\bound{r }}
\end{paste}\end{patch}

\begin{patch}{ContinuedFractionXmpPagePatch21}
\begin{paste}{ContinuedFractionXmpPageFull21}
```

```
{ContinuedFractionXmpPageEmpty21}
\pastebutton{ContinuedFractionXmpPageFull21}{\hidepaste}
\tab{5}\spadcommand{continuedFraction r\free{r }}
\indentrel{3}\begin{verbatim}
                 1             1
   (21)  1 +  +
             1     9      16     40
              x -         x -
             4     8       3      3
Type: ContinuedFraction UnivariatePolynomial(x,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ContinuedFractionXmpPageEmpty21}
\begin{paste}{ContinuedFractionXmpPageEmpty21}
{ContinuedFractionXmpPagePatch21}
\pastebutton{ContinuedFractionXmpPageEmpty21}{\showpaste}
\tab{5}\spadcommand{continuedFraction r\free{r }}
\end{paste}\end{patch}

\begin{patch}{ContinuedFractionXmpPagePatch22}
\begin{paste}{ContinuedFractionXmpPageFull22}
{ContinuedFractionXmpPageEmpty22}
\pastebutton{ContinuedFractionXmpPageFull22}{\hidepaste}
\tab{5}\spadcommand{
[i*i for i in convergents(z) :: Stream Float]\free{z }}
\indentrel{3}\begin{verbatim}
   (22)
   [9.0, 11.1111111111 11111111, 10.9944598337 9501385,
    11.0002777777 77777778, 10.9999860763 98799786,
    11.0000006979 29731039, 10.9999999650 15834446,
    11.0000000017 53603304, 10.9999999999 12099531,
    11.0000000000 04406066, ...]
                                     Type: Stream Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ContinuedFractionXmpPageEmpty22}
\begin{paste}{ContinuedFractionXmpPageEmpty22}
{ContinuedFractionXmpPagePatch22}
\pastebutton{ContinuedFractionXmpPageEmpty22}{\showpaste}
\tab{5}\spadcommand{
[i*i for i in convergents(z) :: Stream Float]\free{z }}
\end{paste}\end{patch}
```

———

3.18 cphelp.ht

Control Panel Bits

— cphelp.ht —

```
\begin{page}{CPHelp}{Control Panel Bits}
\beginscroll

Here are some stuff from a Three Dimensional Viewport's Control Panel
\newline

Main Control Panel: \newline \newline

Rotate: \helpbit{rotate3D}
Zoom: \helpbit{zoom3D}
Translate up/down left/right in the window: \helpbit{translate3D}
Changing the color of the rendered surface: \helpbit{color3D}
Turn the axes on and off: \helpbit{axes3D}
Display surface as a transparent wire mesh: \helpbit{transparent3D}
Display surface with hidden surface removed and the core dumped:
\helpbit{opaque3D}
Display rendered surface: \helpbit{render3D}
Show region within which the function is defined: \helpbit{region3D}
Change position of light source: \helpbit{lighting3D}
Change Perspective/Clipping of surface: \helpbit{volume3D}
Reset to original viewpoint: \helpbit{reset3D}
Show grid lines on the rendered surface: \helpbit{outline3D}
Hide the menu: \helpbit{hide3D}
Close the viewport: \helpbit{close3D}

\newline
\newline
\endscroll
\autobuttons
\end{page}
```

————-

3.19 cycles.ht

CycleIndicators

— cycles.ht —

```
\begin{page}{CycleIndicatorsXmpPage}{CycleIndicators}
\beginscroll
This section is based upon the paper
J. H. Redfield, ``The Theory of Group-Reduced Distributions'',
and is an application of group theory to enumeration problems.
It is a development of the work by P. A. MacMahon on the

The theory is based upon the power sum symmetric functions
\subscriptIt{s}{i}
which are the sum of the \eth{\it i} powers of the variables.
The cycle index of a permutation is an expression that specifies
the sizes of the cycles of a permutation, and
may be represented as a partition.
A partition of a non-negative integer \spad{n} is a collection
of positive integers called its parts whose sum is \spad{n}.
For example, the partition
\texht{$(3^2 \  2 \ 1^2)$}{\spad{3**2 2 1**2)}}
will be used to represent
\texht{$s^2_3 s_2 s^2_1$}{\spad{(s_3)**2 s_2 (s_1)**2}}
and will indicate that the permutation has two cycles of length 3,
one of length 2 and two of length 1.
The cycle index of a permutation group is the sum of the cycle indices
of its permutations divided by the number of permutations.
The cycle indices of certain groups are provided.
\xtc{
We first expose something from the library.
}{
\spadpaste{)expose EVALCYC}
}
\xtc{
The operation \spadfun{complete} returns the cycle index of the
symmetric group of order \spad{n} for argument \spad{n}.
Alternatively, it is the \eth{\spad{n}} complete homogeneous symmetric
function expressed in terms of power sum symmetric functions.
}{
\spadpaste{complete 1}
}
\xtc{
}{
\spadpaste{complete 2}
}
\xtc{
}{
\spadpaste{complete 3}
}
\xtc{
}{
\spadpaste{complete 7}
}
\xtc{
```

```
The operation \spadfun{elementary} computes the \eth{\spad{n}}
elementary symmetric function for argument \spad{n.}
}{
\spadpaste{elementary 7}
}
\xtc{
The operation \spadfun{alternating} returns
the cycle index of the alternating group
having an even number of even parts in each cycle partition.
}{
\spadpaste{alternating 7}
}
\xtc{
The operation \spadfun{cyclic} returns the cycle index of the cyclic group.
}{
\spadpaste{cyclic 7}
}
\xtc{
The operation \spadfun{dihedral} is the cycle index of the
dihedral group.
}{
\spadpaste{dihedral 7}
}
\xtc{
The operation \spadfun{graphs} for argument \spad{n} returns
the cycle index of the group of permutations on
the edges of the complete graph with \spad{n} nodes induced by
applying the symmetric group to the nodes.
}{
\spadpaste{graphs 5}
}

The cycle index of a direct product of two groups is the product
of the cycle indices of the groups.
Redfield provided two operations on two cycle indices which will
be called ``cup'' and ``cap'' here.
The \spadfun{cup} of two cycle indices is a kind of scalar product
that combines monomials for permutations with the same cycles.
The \spadfun{cap} operation provides the sum of the coefficients
of the result of the \spadfun{cup} operation which will be an
integer that enumerates what Redfield called
group-reduced distributions.

We can, for example, represent  \spad{complete 2 * complete 2}
as the set of objects \spad{a a b b} and
\spad{complete 2 * complete 1 * complete 1} as \spad{c c d e.}

\xtc{
This integer
is the number of different sets of four pairs.
```

```
}{
\spadpaste{cap(complete 2**2, complete 2*complete 1**2)}
}
For example,
\begin{verbatim}
a a b b    a a b b   a a b b   a a b b
c c d e    c d c e   c e c d   d e c c
\end{verbatim}

\xtc{
This integer
is the number of different sets of four pairs no two pairs being equal.
}{
\spadpaste{cap(elementary 2**2, complete 2*complete 1**2)}
}
For example,
\begin{verbatim}
a a b b   a a b b
c d c e   c e c d
\end{verbatim}
In this case the configurations enumerated are easily constructed,
however the theory merely enumerates them providing little help in
actually constructing them.
\xtc{
Here are the
number of 6-pairs, first from \spad{a a a b b c,} second from
\spad{d d e e f g.}
}{
\spadpaste{cap(complete 3*complete 2*complete 1,
complete 2**2*complete 1**2)}
}
\xtc{
Here it is again, but with no equal pairs.
}{
\spadpaste{cap(elementary 3*elementary 2*elementary 1,
complete 2**2*complete 1**2)}
}
\xtc{
}{
\spadpaste{cap(complete 3*complete 2*complete 1,
elementary 2**2*elementary 1**2)}
}
\xtc{
The number of 6-triples, first from \spad{a a a b b c,} second from
\spad{d d e e f g,} third from \spad{h h i i j j.}
}{
\spadpaste{eval(cup(complete 3*complete 2*complete 1,
cup(complete 2**2*complete 1**2,complete 2**3)))}
}
\xtc{
```

```
The cycle index of vertices of a square is dihedral 4.
}{
\spadpaste{square:=dihedral 4}
}
\xtc{
The number of different squares with 2 red vertices and 2 blue vertices.
}{
\spadpaste{cap(complete 2**2,square)}
}
\xtc{
The number of necklaces with 3 red beads, 2 blue beads and 2 green beads.
}{
\spadpaste{cap(complete 3*complete 2**2,dihedral 7)}
}
\xtc{
The number of graphs with 5 nodes and 7 edges.
}{
\spadpaste{cap(graphs 5,complete 7*complete 3)}
}
\xtc{
The cycle index of rotations of vertices of a cube.
}{
\spadpaste{s(x) == powerSum(x)}
}
\xtc{
}{
\spadpaste{cube:=(1/24)*(s 1**8+9*s 2**4 + 8*s 3**2*s 1**2+6*s 4**2)}
}
\xtc{
The number of cubes with 4 red vertices and 4 blue vertices.
}{
\spadpaste{cap(complete 4**2,cube)}
}
\xtc{
The number of labeled graphs with degree sequence \spad{2 2 2 1 1}
with no loops or multiple edges.
}{
\spadpaste{
cap(complete 2**3*complete 1**2,wreath(elementary 4,elementary 2))}
}
\xtc{
Again, but
with loops allowed but not multiple edges.
}{
\spadpaste{
cap(complete 2**3*complete 1**2,wreath(elementary 4,complete 2))}
}
\xtc{
Again, but
with multiple edges allowed, but not loops
```

```
}{
\spadpaste{cap(complete 2**3*complete 1**2,wreath(complete 4,elementary 2))}
}
\xtc{
Again, but
with both multiple edges and loops allowed
}{
\spadpaste{cap(complete 2**3*complete 1**2,wreath(complete 4,complete 2))}
}

Having constructed a cycle index for a configuration
we are at liberty to evaluate the
\subscriptIt{s}{i}
components any way we please.
For example we can produce enumerating generating functions.
{CycleIndicators}
This is done by providing a function \spad{f} on an integer \spad{i}
to the value required of \subscriptIt{s}{i},
and then evaluating \spad{eval(f, cycleindex)}.
\xtc{
}{
\spadpaste{x: ULS(FRAC INT,'x,0) := 'x \bound{x}}
}
\xtc{
}{
\spadpaste{ZeroOrOne: INT -> ULS(FRAC INT, 'x, 0) \bound{zodec}}
}
\xtc{
}{
\spadpaste{Integers: INT -> ULS(FRAC INT, 'x, 0) \bound{idec}}
}
\xtc{
For the integers 0 and 1, or two colors.
}{
\spadpaste{ZeroOrOne n == 1+x**n \free{x zodec}\bound{zo}}
}
\xtc{
}{
\spadpaste{ZeroOrOne 5 \free{zo}}
}
\xtc{
For the integers \spad{0, 1, 2, ...} we have this.
}{
\spadpaste{Integers n == 1/(1-x**n) \free{x idec}\bound{i}}
}
\xtc{
}{
\spadpaste{Integers 5 \free{i}}
}
```

```
\xtc{
The coefficient of \texht{$x^n$}{\spad{x**n}}
is the number of graphs with 5 nodes
and \spad{n} edges.
}{
\spadpaste{eval(ZeroOrOne, graphs 5) \free{zo}}
}
\xtc{
The coefficient of \texht{$x^n$}{\spad{x**n}}  is the number of
necklaces with \spad{n} red beads and \spad{n-8} green beads.
}{
\spadpaste{eval(ZeroOrOne,dihedral 8) \free{zo}}
}
\xtc{
The coefficient of \texht{$x^n$}{\spad{x**n}} is the number of
partitions of \spad{n} into 4 or fewer parts.
}{
\spadpaste{eval(Integers,complete 4) \free{i}}
}
\xtc{
The coefficient of \texht{$x^n$}{\spad{x**n}} is the number of
partitions of \spad{n} into 4
boxes containing ordered distinct parts.
}{
\spadpaste{eval(Integers,elementary 4) \free{i}}
}
\xtc{
The coefficient of \texht{$x^n$}{\spad{x**n}} is the number of
different cubes with \spad{n} red vertices and \spad{8-n} green ones.
}{
\spadpaste{eval(ZeroOrOne,cube) \free{zo}}
}
\xtc{
The coefficient of \texht{$x^n$}{\spad{x**n}}
is the number of different cubes with integers
on the vertices whose sum is \spad{n.}
}{
\spadpaste{eval(Integers,cube) \free{i}}
}
\xtc{
The coefficient of \texht{$x^n$}{\spad{x**n}} is the number of
graphs with 5 nodes and with integers on the edges whose sum is
\spad{n.}
In other words, the enumeration is of multigraphs with 5 nodes and
\spad{n} edges.
}{
\spadpaste{eval(Integers,graphs 5) \free{i}}
}
\xtc{
Graphs with 15 nodes enumerated with respect to number of edges.
```

```
}{
\spadpaste{eval(ZeroOrOne ,graphs 15) \free{zo}}
}
\xtc{
Necklaces with 7 green beads, 8 white beads, 5 yellow beads and 10
red beads.
}{
\spadpaste{cap(dihedral 30,complete 7*complete 8*complete 5*complete 10)}
}
The operation \spadfun{SFunction} is the S-function or Schur function
of a partition written
as a descending list of integers expressed in terms of power sum
symmetric functions.
\xtc{
In this case the argument partition represents a tableau shape.
For example \spad{3,2,2,1} represents a tableau with three boxes in the
first row, two boxes in the second and third rows, and one box in the
fourth row.
\spad{SFunction [3,2,2,1]}
counts the number of different tableaux of shape \spad{3, 2, 2, 1} filled
with objects with an ascending order in the columns and a
non-descending order in the rows.
}{
\spadpaste{sf3221:= SFunction [3,2,2,1] \bound{sf3221}}
}
\xtc{
This is the number filled with \spad{a a b b c c d d.}
}{
\spadpaste{cap(sf3221,complete 2**4) \free{sf3221}}
}
The configurations enumerated above are:
\begin{verbatim}
a a b    a a c    a a d
b c      b b      b b
c d      c d      c c
d        d        d
\end{verbatim}
\xtc{
This is the number of tableaux filled with \spad{1..8.}
}{
\spadpaste{cap(sf3221, powerSum 1**8)\free{sf3221}}
}
\xtc{
The coefficient of \texht{$x^n$}{\spad{x**n}} is the number
of column strict reverse plane partitions of \spad{n} of shape
\spad{3 2 2 1.}
}{
\spadpaste{eval(Integers, sf3221)\free{i sf3221}}
}
The smallest is
```

```
\begin{verbatim}
0 0 0
1 1
2 2
3
\end{verbatim}
\endscroll
\autobuttons
\end{page}

\begin{patch}{CycleIndicatorsXmpPagePatch1}
\begin{paste}{CycleIndicatorsXmpPageFull1}{CycleIndicatorsXmpPageEmpty1}
\pastebutton{CycleIndicatorsXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{)expose EVALCYC}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty1}
\begin{paste}{CycleIndicatorsXmpPageEmpty1}{CycleIndicatorsXmpPagePatch1}
\pastebutton{CycleIndicatorsXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{)expose EVALCYC}
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch2}
\begin{paste}{CycleIndicatorsXmpPageFull2}{CycleIndicatorsXmpPageEmpty2}
\pastebutton{CycleIndicatorsXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{complete 1}
\indentrel{3}\begin{verbatim}
   (1)  (1)
             Type: SymmetricPolynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty2}
\begin{paste}{CycleIndicatorsXmpPageEmpty2}{CycleIndicatorsXmpPagePatch2}
\pastebutton{CycleIndicatorsXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{complete 1}
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch3}
\begin{paste}{CycleIndicatorsXmpPageFull3}{CycleIndicatorsXmpPageEmpty3}
\pastebutton{CycleIndicatorsXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{complete 2}
\indentrel{3}\begin{verbatim}
        1       1   2
   (2)   (2) +  (1 )
        2       2
             Type: SymmetricPolynomial Fraction Integer
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty3}
\begin{paste}{CycleIndicatorsXmpPageEmpty3}{CycleIndicatorsXmpPagePatch3}
\pastebutton{CycleIndicatorsXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{complete 2}
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch4}
\begin{paste}{CycleIndicatorsXmpPageFull4}{CycleIndicatorsXmpPageEmpty4}
\pastebutton{CycleIndicatorsXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{complete 3}
\indentrel{3}\begin{verbatim}
        1       1         1   3
   (3)   (3) +  (2 1) +  (1 )
        3       2         6
             Type: SymmetricPolynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty4}
\begin{paste}{CycleIndicatorsXmpPageEmpty4}{CycleIndicatorsXmpPagePatch4}
\pastebutton{CycleIndicatorsXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{complete 3}
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch5}
\begin{paste}{CycleIndicatorsXmpPageFull5}{CycleIndicatorsXmpPageEmpty5}
\pastebutton{CycleIndicatorsXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{complete 7}
\indentrel{3}\begin{verbatim}
   (4)
     1       1          1          1     2     1
      (7) +  (6 1) +  (5 2) +  (5 1 ) +  (4 3)
     7       6         10         10          12
   +
     1           1     3     1   2      1     2
      (4 2 1) +  (4 1 ) +  (3 1) +  (3 2 )
     8           24         18         24
   +
      1       2     1     4     1   3      1   2 3
      (3 2 1 ) +  (3 1 ) +  (2 1) +  (2 1 )
     12           72          48         48
   +
      1      5      1    7
      (2 1 ) +  (1 )
     240          5040
             Type: SymmetricPolynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{CycleIndicatorsXmpPageEmpty5}
\begin{paste}{CycleIndicatorsXmpPageEmpty5}{CycleIndicatorsXmpPagePatch5}
\pastebutton{CycleIndicatorsXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{complete 7}
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch6}
\begin{paste}{CycleIndicatorsXmpPageFull6}{CycleIndicatorsXmpPageEmpty6}
\pastebutton{CycleIndicatorsXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{elementary 7}
\indentrel{3}\begin{verbatim}
   (5)
     1       1          1          1     2     1
      (7) -  (6 1) -  (5 2) +  (5 1 ) -  (4 3)
     7       6         10         10          12
   +
     1            1     3     1   2      1     2
      (4 2 1) -  (4 1 ) +  (3 1) +  (3 2 )
     8           24          18         24
   +
        1       2     1     4     1   3      1   2 3
     -  (3 2 1 ) +  (3 1 ) -  (2 1) +  (2 1 )
       12           72          48         48
   +
        1      5      1    7
     -  (2 1 ) +  (1 )
       240          5040
            Type: SymmetricPolynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty6}
\begin{paste}{CycleIndicatorsXmpPageEmpty6}{CycleIndicatorsXmpPagePatch6}
\pastebutton{CycleIndicatorsXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{elementary 7}
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch7}
\begin{paste}{CycleIndicatorsXmpPageFull7}{CycleIndicatorsXmpPageEmpty7}
\pastebutton{CycleIndicatorsXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{alternating 7}
\indentrel{3}\begin{verbatim}
   (6)
     2       1     2    1           1   2      1     2
      (7) +  (5 1 ) +  (4 2 1) +  (3 1) +  (3 2 )
     7       5          4           9         12
   +
      1     4     1   2 3      1    7
      (3 1 ) +  (2 1 ) +  (1 )
```

```
     36          24          2520
             Type: SymmetricPolynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty7}
\begin{paste}{CycleIndicatorsXmpPageEmpty7}{CycleIndicatorsXmpPagePatch7}
\pastebutton{CycleIndicatorsXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{alternating 7}
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch8}
\begin{paste}{CycleIndicatorsXmpPageFull8}{CycleIndicatorsXmpPageEmpty8}
\pastebutton{CycleIndicatorsXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{cyclic 7}
\indentrel{3}\begin{verbatim}
        6       1   7
   (7)   (7) +  (1 )
        7       7
             Type: SymmetricPolynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty8}
\begin{paste}{CycleIndicatorsXmpPageEmpty8}{CycleIndicatorsXmpPagePatch8}
\pastebutton{CycleIndicatorsXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{cyclic 7}
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch9}
\begin{paste}{CycleIndicatorsXmpPageFull9}{CycleIndicatorsXmpPageEmpty9}
\pastebutton{CycleIndicatorsXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{dihedral 7}
\indentrel{3}\begin{verbatim}
        3       1   3      1   7
   (8)   (7) +  (2 1) +  (1 )
        7       2        14
             Type: SymmetricPolynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty9}
\begin{paste}{CycleIndicatorsXmpPageEmpty9}{CycleIndicatorsXmpPagePatch9}
\pastebutton{CycleIndicatorsXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{dihedral 7}
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch10}
\begin{paste}{CycleIndicatorsXmpPageFull10}{CycleIndicatorsXmpPageEmpty10}
\pastebutton{CycleIndicatorsXmpPageFull10}{\hidepaste}
```

```
\tab{5}\spadcommand{graphs 5}
\indentrel{3}\begin{verbatim}
   (9)
     1           1   2    1   2     1   3     1   4 2
      (6 3 1) +  (5 ) +  (4 2) +  (3 1) +  (2 1 )
     6           5        4         6         8
   +
      1   3 4     1    10
      (2 1 ) +  (1  )
     12          120
             Type: SymmetricPolynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty10}
\begin{paste}{CycleIndicatorsXmpPageEmpty10}{CycleIndicatorsXmpPagePatch10}
\pastebutton{CycleIndicatorsXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{graphs 5}
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch11}
\begin{paste}{CycleIndicatorsXmpPageFull11}{CycleIndicatorsXmpPageEmpty11}
\pastebutton{CycleIndicatorsXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{cap(complete 2**2, complete 2*complete 1**2)}
\indentrel{3}\begin{verbatim}
   (10)  4
                                 Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty11}
\begin{paste}{CycleIndicatorsXmpPageEmpty11}{CycleIndicatorsXmpPagePatch11}
\pastebutton{CycleIndicatorsXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{cap(complete 2**2, complete 2*complete 1**2)}
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch12}
\begin{paste}{CycleIndicatorsXmpPageFull12}{CycleIndicatorsXmpPageEmpty12}
\pastebutton{CycleIndicatorsXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{cap(elementary 2**2, complete 2*complete 1**2)}
\indentrel{3}\begin{verbatim}
   (11)  2
                                 Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty12}
\begin{paste}{CycleIndicatorsXmpPageEmpty12}{CycleIndicatorsXmpPagePatch12}
\pastebutton{CycleIndicatorsXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{cap(elementary 2**2, complete 2*complete 1**2)}
```

```
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch13}
\begin{paste}{CycleIndicatorsXmpPageFull13}{CycleIndicatorsXmpPageEmpty13}
\pastebutton{CycleIndicatorsXmpPageFull13}{\hidepaste}
\tab{5}\spadcommand{
cap(complete 3*complete 2*complete 1,complete 2**2*complete 1**2)}
\indentrel{3}\begin{verbatim}
   (12)  24
                                 Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty13}
\begin{paste}{CycleIndicatorsXmpPageEmpty13}{CycleIndicatorsXmpPagePatch13}
\pastebutton{CycleIndicatorsXmpPageEmpty13}{\showpaste}
\tab{5}\spadcommand{
cap(complete 3*complete 2*complete 1,complete 2**2*complete 1**2)}
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch14}
\begin{paste}{CycleIndicatorsXmpPageFull14}{CycleIndicatorsXmpPageEmpty14}
\pastebutton{CycleIndicatorsXmpPageFull14}{\hidepaste}
\tab{5}\spadcommand{
cap(elementary 3*elementary 2*elementary 1,complete 2**2*complete 1**2)}
\indentrel{3}\begin{verbatim}
   (13)  8
                                 Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty14}
\begin{paste}{CycleIndicatorsXmpPageEmpty14}{CycleIndicatorsXmpPagePatch14}
\pastebutton{CycleIndicatorsXmpPageEmpty14}{\showpaste}
\tab{5}\spadcommand{
cap(elementary 3*elementary 2*elementary 1,complete 2**2*complete 1**2)}
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch15}
\begin{paste}{CycleIndicatorsXmpPageFull15}{CycleIndicatorsXmpPageEmpty15}
\pastebutton{CycleIndicatorsXmpPageFull15}{\hidepaste}
\tab{5}\spadcommand{
cap(complete 3*complete 2*complete 1,elementary 2**2*elementary 1**2)}
\indentrel{3}\begin{verbatim}
   (14)  8
                                 Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty15}
```

```
\begin{paste}{CycleIndicatorsXmpPageEmpty15}{CycleIndicatorsXmpPagePatch15}
\pastebutton{CycleIndicatorsXmpPageEmpty15}{\showpaste}
\tab{5}\spadcommand{
cap(complete 3*complete 2*complete 1,elementary 2**2*elementary 1**2)}
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch16}
\begin{paste}{CycleIndicatorsXmpPageFull16}{CycleIndicatorsXmpPageEmpty16}
\pastebutton{CycleIndicatorsXmpPageFull16}{\hidepaste}
\tab{5}\spadcommand{
eval(cup(complete 3*complete 2*complete 1, _
     cup(complete 2**2*complete 1**2,complete 2**3)))}
\indentrel{3}\begin{verbatim}
   (15)  1500
                                 Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty16}
\begin{paste}{CycleIndicatorsXmpPageEmpty16}{CycleIndicatorsXmpPagePatch16}
\pastebutton{CycleIndicatorsXmpPageEmpty16}{\showpaste}
\tab{5}\spadcommand{
eval(cup(complete 3*complete 2*complete 1, _
     cup(complete 2**2*complete 1**2,complete 2**3)))}
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch17}
\begin{paste}{CycleIndicatorsXmpPageFull17}{CycleIndicatorsXmpPageEmpty17}
\pastebutton{CycleIndicatorsXmpPageFull17}{\hidepaste}
\tab{5}\spadcommand{square:=dihedral 4}
\indentrel{3}\begin{verbatim}
         1       3   2    1     2    1   4
   (16)   (4) +  (2 ) +  (2 1 ) +  (1 )
         4       8        4          8
             Type: SymmetricPolynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty17}
\begin{paste}{CycleIndicatorsXmpPageEmpty17}{CycleIndicatorsXmpPagePatch17}
\pastebutton{CycleIndicatorsXmpPageEmpty17}{\showpaste}
\tab{5}\spadcommand{square:=dihedral 4}
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch18}
\begin{paste}{CycleIndicatorsXmpPageFull18}{CycleIndicatorsXmpPageEmpty18}
\pastebutton{CycleIndicatorsXmpPageFull18}{\hidepaste}
\tab{5}\spadcommand{cap(complete 2**2,square)}
\indentrel{3}\begin{verbatim}
   (17)  2
```

```
                                 Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty18}
\begin{paste}{CycleIndicatorsXmpPageEmpty18}{CycleIndicatorsXmpPagePatch18}
\pastebutton{CycleIndicatorsXmpPageEmpty18}{\showpaste}
\tab{5}\spadcommand{cap(complete 2**2,square)}
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch19}
\begin{paste}{CycleIndicatorsXmpPageFull19}{CycleIndicatorsXmpPageEmpty19}
\pastebutton{CycleIndicatorsXmpPageFull19}{\hidepaste}
\tab{5}\spadcommand{cap(complete 3*complete 2**2,dihedral 7)}
\indentrel{3}\begin{verbatim}
   (18)  18
                                 Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty19}
\begin{paste}{CycleIndicatorsXmpPageEmpty19}{CycleIndicatorsXmpPagePatch19}
\pastebutton{CycleIndicatorsXmpPageEmpty19}{\showpaste}
\tab{5}\spadcommand{cap(complete 3*complete 2**2,dihedral 7)}
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch20}
\begin{paste}{CycleIndicatorsXmpPageFull20}{CycleIndicatorsXmpPageEmpty20}
\pastebutton{CycleIndicatorsXmpPageFull20}{\hidepaste}
\tab{5}\spadcommand{cap(graphs 5,complete 7*complete 3)}
\indentrel{3}\begin{verbatim}
   (19)  4
                                 Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty20}
\begin{paste}{CycleIndicatorsXmpPageEmpty20}{CycleIndicatorsXmpPagePatch20}
\pastebutton{CycleIndicatorsXmpPageEmpty20}{\showpaste}
\tab{5}\spadcommand{cap(graphs 5,complete 7*complete 3)}
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch21}
\begin{paste}{CycleIndicatorsXmpPageFull21}{CycleIndicatorsXmpPageEmpty21}
\pastebutton{CycleIndicatorsXmpPageFull21}{\hidepaste}
\tab{5}\spadcommand{s(x) == powerSum(x)}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{CycleIndicatorsXmpPageEmpty21}
\begin{paste}{CycleIndicatorsXmpPageEmpty21}{CycleIndicatorsXmpPagePatch21}
\pastebutton{CycleIndicatorsXmpPageEmpty21}{\showpaste}
\tab{5}\spadcommand{s(x) == powerSum(x)}
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch22}
\begin{paste}{CycleIndicatorsXmpPageFull22}{CycleIndicatorsXmpPageEmpty22}
\pastebutton{CycleIndicatorsXmpPageFull22}{\hidepaste}
\tab{5}\spadcommand{
cube:=(1/24)*(s 1**8+9*s 2**4 + 8*s 3**2*s 1**2+6*s 4**2)}
\indentrel{3}\begin{verbatim}
         1   2    1   2 2    3   4     1   8
   (21)   (4 ) +  (3 1 ) +  (2 ) +  (1 )
         4        3          8        24
             Type: SymmetricPolynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty22}
\begin{paste}{CycleIndicatorsXmpPageEmpty22}{CycleIndicatorsXmpPagePatch22}
\pastebutton{CycleIndicatorsXmpPageEmpty22}{\showpaste}
\tab{5}\spadcommand{
cube:=(1/24)*(s 1**8+9*s 2**4 + 8*s 3**2*s 1**2+6*s 4**2)}
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch23}
\begin{paste}{CycleIndicatorsXmpPageFull23}{CycleIndicatorsXmpPageEmpty23}
\pastebutton{CycleIndicatorsXmpPageFull23}{\hidepaste}
\tab{5}\spadcommand{cap(complete 4**2,cube)}
\indentrel{3}\begin{verbatim}
   (22)  7
                                 Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty23}
\begin{paste}{CycleIndicatorsXmpPageEmpty23}{CycleIndicatorsXmpPagePatch23}
\pastebutton{CycleIndicatorsXmpPageEmpty23}{\showpaste}
\tab{5}\spadcommand{cap(complete 4**2,cube)}
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch24}
\begin{paste}{CycleIndicatorsXmpPageFull24}{CycleIndicatorsXmpPageEmpty24}
\pastebutton{CycleIndicatorsXmpPageFull24}{\hidepaste}
\tab{5}\spadcommand{
cap(complete 2**3*complete 1**2,wreath(elementary 4,elementary 2))}
\indentrel{3}\begin{verbatim}
   (23)  7
```

```
                                  Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty24}
\begin{paste}{CycleIndicatorsXmpPageEmpty24}{CycleIndicatorsXmpPagePatch24}
\pastebutton{CycleIndicatorsXmpPageEmpty24}{\showpaste}
\tab{5}\spadcommand{
cap(complete 2**3*complete 1**2,wreath(elementary 4,elementary 2))}
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch25}
\begin{paste}{CycleIndicatorsXmpPageFull25}{CycleIndicatorsXmpPageEmpty25}
\pastebutton{CycleIndicatorsXmpPageFull25}{\hidepaste}
\tab{5}\spadcommand{
cap(complete 2**3*complete 1**2,wreath(elementary 4,complete 2))}
\indentrel{3}\begin{verbatim}
   (24)  17
                                  Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty25}
\begin{paste}{CycleIndicatorsXmpPageEmpty25}{CycleIndicatorsXmpPagePatch25}
\pastebutton{CycleIndicatorsXmpPageEmpty25}{\showpaste}
\tab{5}\spadcommand{
cap(complete 2**3*complete 1**2,wreath(elementary 4,complete 2))}
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch26}
\begin{paste}{CycleIndicatorsXmpPageFull26}{CycleIndicatorsXmpPageEmpty26}
\pastebutton{CycleIndicatorsXmpPageFull26}{\hidepaste}
\tab{5}\spadcommand{
cap(complete 2**3*complete 1**2,wreath(complete 4,elementary 2))}
\indentrel{3}\begin{verbatim}
   (25)  10
                                  Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty26}
\begin{paste}{CycleIndicatorsXmpPageEmpty26}{CycleIndicatorsXmpPagePatch26}
\pastebutton{CycleIndicatorsXmpPageEmpty26}{\showpaste}
\tab{5}\spadcommand{
cap(complete 2**3*complete 1**2,wreath(complete 4,elementary 2))}
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch27}
\begin{paste}{CycleIndicatorsXmpPageFull27}{CycleIndicatorsXmpPageEmpty27}
\pastebutton{CycleIndicatorsXmpPageFull27}{\hidepaste}
```

```
\tab{5}\spadcommand{
cap(complete 2**3*complete 1**2,wreath(complete 4,complete 2))}
\indentrel{3}\begin{verbatim}
   (26)  23
                                 Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty27}
\begin{paste}{CycleIndicatorsXmpPageEmpty27}{CycleIndicatorsXmpPagePatch27}
\pastebutton{CycleIndicatorsXmpPageEmpty27}{\showpaste}
\tab{5}\spadcommand{
cap(complete 2**3*complete 1**2,wreath(complete 4,complete 2))}
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch28}
\begin{paste}{CycleIndicatorsXmpPageFull28}{CycleIndicatorsXmpPageEmpty28}
\pastebutton{CycleIndicatorsXmpPageFull28}{\hidepaste}
\tab{5}\spadcommand{x: ULS(FRAC INT,'x,0) := 'x\bound{x }}
\indentrel{3}\begin{verbatim}
   (27)  x
    Type: UnivariateLaurentSeries(Fraction Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty28}
\begin{paste}{CycleIndicatorsXmpPageEmpty28}{CycleIndicatorsXmpPagePatch28}
\pastebutton{CycleIndicatorsXmpPageEmpty28}{\showpaste}
\tab{5}\spadcommand{x: ULS(FRAC INT,'x,0) := 'x\bound{x }}
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch29}
\begin{paste}{CycleIndicatorsXmpPageFull29}{CycleIndicatorsXmpPageEmpty29}
\pastebutton{CycleIndicatorsXmpPageFull29}{\hidepaste}
\tab{5}\spadcommand{ZeroOrOne: INT -> ULS(FRAC INT, 'x, 0)\bound{zodec }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty29}
\begin{paste}{CycleIndicatorsXmpPageEmpty29}{CycleIndicatorsXmpPagePatch29}
\pastebutton{CycleIndicatorsXmpPageEmpty29}{\showpaste}
\tab{5}\spadcommand{ZeroOrOne: INT -> ULS(FRAC INT, 'x, 0)\bound{zodec }}
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch30}
\begin{paste}{CycleIndicatorsXmpPageFull30}{CycleIndicatorsXmpPageEmpty30}
\pastebutton{CycleIndicatorsXmpPageFull30}{\hidepaste}
\tab{5}\spadcommand{Integers: INT -> ULS(FRAC INT, 'x, 0)\bound{idec }}
```

```
\indentrel{3}\begin{verbatim}
                                              Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty30}
\begin{paste}{CycleIndicatorsXmpPageEmpty30}{CycleIndicatorsXmpPagePatch30}
\pastebutton{CycleIndicatorsXmpPageEmpty30}{\showpaste}
\tab{5}\spadcommand{Integers: INT -> ULS(FRAC INT, 'x, 0)\bound{idec }}
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch31}
\begin{paste}{CycleIndicatorsXmpPageFull31}{CycleIndicatorsXmpPageEmpty31}
\pastebutton{CycleIndicatorsXmpPageFull31}{\hidepaste}
\tab{5}\spadcommand{ZeroOrOne n == 1+x**n\free{x zodec }\bound{zo }}
\indentrel{3}\begin{verbatim}
                                              Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty31}
\begin{paste}{CycleIndicatorsXmpPageEmpty31}{CycleIndicatorsXmpPagePatch31}
\pastebutton{CycleIndicatorsXmpPageEmpty31}{\showpaste}
\tab{5}\spadcommand{ZeroOrOne n == 1+x**n\free{x zodec }\bound{zo }}
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch32}
\begin{paste}{CycleIndicatorsXmpPageFull32}{CycleIndicatorsXmpPageEmpty32}
\pastebutton{CycleIndicatorsXmpPageFull32}{\hidepaste}
\tab{5}\spadcommand{ZeroOrOne 5\free{zo }}
\indentrel{3}\begin{verbatim}
              5
   (31)  1 + x
    Type: UnivariateLaurentSeries(Fraction Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty32}
\begin{paste}{CycleIndicatorsXmpPageEmpty32}{CycleIndicatorsXmpPagePatch32}
\pastebutton{CycleIndicatorsXmpPageEmpty32}{\showpaste}
\tab{5}\spadcommand{ZeroOrOne 5\free{zo }}
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch33}
\begin{paste}{CycleIndicatorsXmpPageFull33}{CycleIndicatorsXmpPageEmpty33}
\pastebutton{CycleIndicatorsXmpPageFull33}{\hidepaste}
\tab{5}\spadcommand{Integers n == 1/(1-x**n)\free{x idec }\bound{i }}
\indentrel{3}\begin{verbatim}
                                              Type: Void
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty33}
\begin{paste}{CycleIndicatorsXmpPageEmpty33}{CycleIndicatorsXmpPagePatch33}
\pastebutton{CycleIndicatorsXmpPageEmpty33}{\showpaste}
\tab{5}\spadcommand{Integers n == 1/(1-x**n)\free{x idec }\bound{i }}
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch34}
\begin{paste}{CycleIndicatorsXmpPageFull34}{CycleIndicatorsXmpPageEmpty34}
\pastebutton{CycleIndicatorsXmpPageFull34}{\hidepaste}
\tab{5}\spadcommand{Integers 5\free{i }}
\indentrel{3}\begin{verbatim}
              5    10      11
   (33)  1 + x  + x   + O(x  )
    Type: UnivariateLaurentSeries(Fraction Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty34}
\begin{paste}{CycleIndicatorsXmpPageEmpty34}{CycleIndicatorsXmpPagePatch34}
\pastebutton{CycleIndicatorsXmpPageEmpty34}{\showpaste}
\tab{5}\spadcommand{Integers 5\free{i }}
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch35}
\begin{paste}{CycleIndicatorsXmpPageFull35}{CycleIndicatorsXmpPageEmpty35}
\pastebutton{CycleIndicatorsXmpPageFull35}{\hidepaste}
\tab{5}\spadcommand{eval(ZeroOrOne, graphs 5)\free{zo }}
\indentrel{3}\begin{verbatim}
   (34)
               2     3     4     5     6     7     8    9
     1 + x + 2x  + 4x  + 6x  + 6x  + 6x  + 4x  + 2x  + x
   +
      10      11
     x   + O(x  )
    Type: UnivariateLaurentSeries(Fraction Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty35}
\begin{paste}{CycleIndicatorsXmpPageEmpty35}{CycleIndicatorsXmpPagePatch35}
\pastebutton{CycleIndicatorsXmpPageEmpty35}{\showpaste}
\tab{5}\spadcommand{eval(ZeroOrOne, graphs 5)\free{zo }}
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch36}
\begin{paste}{CycleIndicatorsXmpPageFull36}{CycleIndicatorsXmpPageEmpty36}
\pastebutton{CycleIndicatorsXmpPageFull36}{\hidepaste}
\tab{5}\spadcommand{eval(ZeroOrOne,dihedral 8)\free{zo }}
```

```
\indentrel{3}\begin{verbatim}
                   2     3     4     5     6    7    8
   (35)  1 + x + 4x  + 5x  + 8x  + 5x  + 4x  + x  + x
    Type: UnivariateLaurentSeries(Fraction Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty36}
\begin{paste}{CycleIndicatorsXmpPageEmpty36}{CycleIndicatorsXmpPagePatch36}
\pastebutton{CycleIndicatorsXmpPageEmpty36}{\showpaste}
\tab{5}\spadcommand{eval(ZeroOrOne,dihedral 8)\free{zo }}
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch37}
\begin{paste}{CycleIndicatorsXmpPageFull37}{CycleIndicatorsXmpPageEmpty37}
\pastebutton{CycleIndicatorsXmpPageFull37}{\hidepaste}
\tab{5}\spadcommand{eval(Integers,complete 4)\free{i }}
\indentrel{3}\begin{verbatim}
   (36)
               2     3     4     5     6      7      8
     1 + x + 2x  + 3x  + 5x  + 6x  + 9x  + 11x  + 15x
   +
        9      10      11
     18x  + 23x   + O(x  )
    Type: UnivariateLaurentSeries(Fraction Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty37}
\begin{paste}{CycleIndicatorsXmpPageEmpty37}{CycleIndicatorsXmpPagePatch37}
\pastebutton{CycleIndicatorsXmpPageEmpty37}{\showpaste}
\tab{5}\spadcommand{eval(Integers,complete 4)\free{i }}
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch38}
\begin{paste}{CycleIndicatorsXmpPageFull38}{CycleIndicatorsXmpPageEmpty38}
\pastebutton{CycleIndicatorsXmpPageFull38}{\hidepaste}
\tab{5}\spadcommand{eval(Integers,elementary 4)\free{i }}
\indentrel{3}\begin{verbatim}
   (37)
      6    7     8     9     10     11     12      13
     x  + x  + 2x  + 3x  + 5x   + 6x   + 9x   + 11x
   +
        14      15      16      17
     15x   + 18x   + 23x   + O(x  )
    Type: UnivariateLaurentSeries(Fraction Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty38}
```

```
\begin{paste}{CycleIndicatorsXmpPageEmpty38}{CycleIndicatorsXmpPagePatch38}
\pastebutton{CycleIndicatorsXmpPageEmpty38}{\showpaste}
\tab{5}\spadcommand{eval(Integers,elementary 4)\free{i }}
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch39}
\begin{paste}{CycleIndicatorsXmpPageFull39}{CycleIndicatorsXmpPageEmpty39}
\pastebutton{CycleIndicatorsXmpPageFull39}{\hidepaste}
\tab{5}\spadcommand{eval(ZeroOrOne,cube)\free{zo }}
\indentrel{3}\begin{verbatim}
                   2     3     4     5     6    7    8
   (38)  1 + x + 3x  + 3x  + 7x  + 3x  + 3x  + x  + x
    Type: UnivariateLaurentSeries(Fraction Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty39}
\begin{paste}{CycleIndicatorsXmpPageEmpty39}{CycleIndicatorsXmpPagePatch39}
\pastebutton{CycleIndicatorsXmpPageEmpty39}{\showpaste}
\tab{5}\spadcommand{eval(ZeroOrOne,cube)\free{zo }}
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch40}
\begin{paste}{CycleIndicatorsXmpPageFull40}{CycleIndicatorsXmpPageEmpty40}
\pastebutton{CycleIndicatorsXmpPageFull40}{\hidepaste}
\tab{5}\spadcommand{eval(Integers,cube)\free{i }}
\indentrel{3}\begin{verbatim}
   (39)
               2     3      4      5      6       7
     1 + x + 4x  + 7x  + 21x  + 37x  + 85x  + 151x
   +
         8       9       10      11
     292x  + 490x  + 848x   + O(x  )
    Type: UnivariateLaurentSeries(Fraction Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty40}
\begin{paste}{CycleIndicatorsXmpPageEmpty40}{CycleIndicatorsXmpPagePatch40}
\pastebutton{CycleIndicatorsXmpPageEmpty40}{\showpaste}
\tab{5}\spadcommand{eval(Integers,cube)\free{i }}
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch41}
\begin{paste}{CycleIndicatorsXmpPageFull41}{CycleIndicatorsXmpPageEmpty41}
\pastebutton{CycleIndicatorsXmpPageFull41}{\hidepaste}
\tab{5}\spadcommand{eval(Integers,graphs 5)\free{i }}
\indentrel{3}\begin{verbatim}
   (40)
               2     3      4      5      6       7
```

```
     1 + x + 3x  + 7x  + 17x  + 35x  + 76x  + 149x
   +
         8       9       10      11
     291x  + 539x  + 974x   + O(x  )
    Type: UnivariateLaurentSeries(Fraction Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty41}
\begin{paste}{CycleIndicatorsXmpPageEmpty41}{CycleIndicatorsXmpPagePatch41}
\pastebutton{CycleIndicatorsXmpPageEmpty41}{\showpaste}
\tab{5}\spadcommand{eval(Integers,graphs 5)\free{i }}
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch42}
\begin{paste}{CycleIndicatorsXmpPageFull42}{CycleIndicatorsXmpPageEmpty42}
\pastebutton{CycleIndicatorsXmpPageFull42}{\hidepaste}
\tab{5}\spadcommand{eval(ZeroOrOne ,graphs 15)\free{zo }}
\indentrel{3}\begin{verbatim}
   (41)
              2     3      4      5      6       7
     1 + x + 2x  + 5x  + 11x  + 26x  + 68x  + 177x
   +
         8        9        10      11
     496x  + 1471x  + 4583x   + O(x  )
    Type: UnivariateLaurentSeries(Fraction Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty42}
\begin{paste}{CycleIndicatorsXmpPageEmpty42}{CycleIndicatorsXmpPagePatch42}
\pastebutton{CycleIndicatorsXmpPageEmpty42}{\showpaste}
\tab{5}\spadcommand{eval(ZeroOrOne ,graphs 15)\free{zo }}
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch43}
\begin{paste}{CycleIndicatorsXmpPageFull43}{CycleIndicatorsXmpPageEmpty43}
\pastebutton{CycleIndicatorsXmpPageFull43}{\hidepaste}
\tab{5}\spadcommand{cap(dihedral 30,complete 7*complete 8*complete 5*complete 10)}
\indentrel{3}\begin{verbatim}
   (42)  49958972383320
                                 Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty43}
\begin{paste}{CycleIndicatorsXmpPageEmpty43}{CycleIndicatorsXmpPagePatch43}
\pastebutton{CycleIndicatorsXmpPageEmpty43}{\showpaste}
\tab{5}\spadcommand{
cap(dihedral 30,complete 7*complete 8*complete 5*complete 10)}
```

```
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch44}
\begin{paste}{CycleIndicatorsXmpPageFull44}{CycleIndicatorsXmpPageEmpty44}
\pastebutton{CycleIndicatorsXmpPageFull44}{\hidepaste}
\tab{5}\spadcommand{sf3221:= SFunction [3,2,2,1]\bound{sf3221 }}
\indentrel{3}\begin{verbatim}
   (43)
     1          1     2     1   2     1
     (6 2) -  (6 1 ) -  (4 ) +  (4 3 1)
    12         12          16        12
   +
     1     4     1   2      1   2 2     1     2
     (4 1 ) -  (3 2) +  (3 1 ) -  (3 2 1)
    24          36         36          24
   +
       1       3     1     5     1    4     1   3 2
    -  (3 2 1 ) -  (3 1 ) -  (2 ) +  (2 1 )
      36            72          192        48
   +
     1   2 4     1      6     1    8
     (2 1 ) -  (2 1 ) +  (1 )
    96          144          576
             Type: SymmetricPolynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty44}
\begin{paste}{CycleIndicatorsXmpPageEmpty44}{CycleIndicatorsXmpPagePatch44}
\pastebutton{CycleIndicatorsXmpPageEmpty44}{\showpaste}
\tab{5}\spadcommand{sf3221:= SFunction [3,2,2,1]\bound{sf3221 }}
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch45}
\begin{paste}{CycleIndicatorsXmpPageFull45}{CycleIndicatorsXmpPageEmpty45}
\pastebutton{CycleIndicatorsXmpPageFull45}{\hidepaste}
\tab{5}\spadcommand{cap(sf3221,complete 2**4)\free{sf3221 }}
\indentrel{3}\begin{verbatim}
   (44)  3
                                 Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty45}
\begin{paste}{CycleIndicatorsXmpPageEmpty45}{CycleIndicatorsXmpPagePatch45}
\pastebutton{CycleIndicatorsXmpPageEmpty45}{\showpaste}
\tab{5}\spadcommand{cap(sf3221,complete 2**4)\free{sf3221 }}
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch46}
```

```
\begin{paste}{CycleIndicatorsXmpPageFull46}{CycleIndicatorsXmpPageEmpty46}
\pastebutton{CycleIndicatorsXmpPageFull46}{\hidepaste}
\tab{5}\spadcommand{cap(sf3221, powerSum 1**8)\free{sf3221 }}
\indentrel{3}\begin{verbatim}
   (45)  70
                                Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty46}
\begin{paste}{CycleIndicatorsXmpPageEmpty46}{CycleIndicatorsXmpPagePatch46}
\pastebutton{CycleIndicatorsXmpPageEmpty46}{\showpaste}
\tab{5}\spadcommand{cap(sf3221, powerSum 1**8)\free{sf3221 }}
\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPagePatch47}
\begin{paste}{CycleIndicatorsXmpPageFull47}{CycleIndicatorsXmpPageEmpty47}
\pastebutton{CycleIndicatorsXmpPageFull47}{\hidepaste}
\tab{5}\spadcommand{eval(Integers, sf3221)\free{i sf3221 }}
\indentrel{3}\begin{verbatim}
   (46)
      9     10     11      12      13      14      15
     x  + 3x   + 7x   + 14x   + 27x   + 47x   + 79x
   +
        16       17       18       19      20
     126x   + 196x   + 294x   + 432x   + O(x  )
    Type: UnivariateLaurentSeries(Fraction Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{CycleIndicatorsXmpPageEmpty47}
\begin{paste}{CycleIndicatorsXmpPageEmpty47}{CycleIndicatorsXmpPagePatch47}
\pastebutton{CycleIndicatorsXmpPageEmpty47}{\showpaste}
\tab{5}\spadcommand{eval(Integers, sf3221)\free{i sf3221 }}
\end{paste}\end{patch}
```

3.20 coverex.ht

Examples Of Axiom Commands

⇒ "Differentiation" (Menuexdiff) 3.20 on page 305
⇒ "Integration" (Menuexint) 3.20 on page 310
⇒ "Laplace Transforms" (Menuexlap) 3.20 on page 317

⇒ “Limits” (Menuexlimit) 3.20 on page 320
⇒ “Matrices” (Menuexmatrix) 3.20 on page 325
⇒ “2-D Graphics” (Menuexplot2d) 3.20 on page 333
⇒ “3-D Graphics” (Menuexplot3d) 3.20 on page 335
⇒ “Series” (Menuexseries) 3.20 on page 337
⇒ “Summations” (Menuexsum) 3.20 on page 342

— coverex.ht —

```
\begin{page}{ExampleCoverPage}{Examples Of Axiom Commands}
\beginscroll\table{
{\downlink{Differentiation}{Menuexdiff}}
{\downlink{Integration}{Menuexint}}
{\downlink{Laplace Transforms}{Menuexlap}}
{\downlink{Limits}{Menuexlimit}}
{\downlink{Matrices}{Menuexmatrix}}
{\downlink{2-D Graphics}{Menuexplot2d}}
{\downlink{3-D Graphics}{Menuexplot3d}}
{\downlink{Series}{Menuexseries}}
{\downlink{Summations}{Menuexsum}}
}\endscroll\end{page}
```

———-

Differentiation

⇒ “Computing Derivatives” (ExDiffBasic) 3.28 on page 391
⇒ “Derivatives of Functions of Several Variables” (ExDiffSeveralVariables) 3.28 on page 392
⇒ “Derivatives of Higher Order” (ExDiffHigherOrder) 3.28 on page 393
⇒ “Multiple Derivatives I” (ExDiffMultipleI) 3.28 on page 394
⇒ “Multiple Derivatives II” (ExDiffMultipleII) 3.28 on page 396
⇒ “Derivatives of Functions Involving Formal Integrals” (ExDiffFormalIntegral) 3.28 on page 396

— coverex.ht —

```
\begin{page}{Menuexdiff}{Differentiation}
\beginscroll\beginmenu
\menudownlink{Computing Derivatives}{ExDiffBasic}
\spadpaste{differentiate(sin(x) * exp(x**2),x)}
\menudownlink{Derivatives of Functions of Several Variables}
{ExDiffSeveralVariables}
\spadpaste{differentiate(sin(x) * tan(y)/(x**2 + y**2),x)}
\spadpaste{differentiate(sin(x) * tan(y)/(x**2 + y**2),y)}
\menudownlink{Derivatives of Higher Order}{ExDiffHigherOrder}
\spadpaste{differentiate(exp(x**2),x,4)}
\menudownlink{Multiple Derivatives I}{ExDiffMultipleI}
```

```
\spadpaste{differentiate(sin(x)/(x**2 + y**2),[x,y])}
\spadpaste{differentiate(sin(x)/(x**2 + y**2),[x,y,y])}
\menudownlink{Multiple Derivatives II}{ExDiffMultipleII}
\spadpaste{differentiate(cos(z)/(x**2 + y**3),[x,y,z],[1,2,3])}
\menudownlink{Derivatives of Functions Involving Formal Integrals}
{ExDiffFormalIntegral}
\spadpaste{f := integrate(sqrt(1 + t**3),t) \bound{f}}
\spadpaste{differentiate(f,t) \free{f}}
\spadpaste{differentiate(f * t**2,t) \free{f}}
\endmenu\endscroll
\end{page}

\begin{patch}{MenuexdiffPatch1}
\begin{paste}{MenuexdiffFull1}{MenuexdiffEmpty1}
\pastebutton{MenuexdiffFull1}{\hidepaste}
\tab{5}\spadcommand{differentiate(sin(x) * exp(x**2),x)}
\indentrel{3}\begin{verbatim}
              2                  2
             x                  x
   (1)  2x %e  sin(x) + cos(x)%e
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexdiffEmpty1}
\begin{paste}{MenuexdiffEmpty1}{MenuexdiffPatch1}
\pastebutton{MenuexdiffEmpty1}{\showpaste}
\tab{5}\spadcommand{differentiate(sin(x) * exp(x**2),x)}
\end{paste}\end{patch}

\begin{patch}{MenuexdiffPatch2}
\begin{paste}{MenuexdiffFull2}{MenuexdiffEmpty2}
\pastebutton{MenuexdiffFull2}{\hidepaste}
\tab{5}\spadcommand{differentiate(sin(x) * tan(y)/(x**2 + y**2),x)}
\indentrel{3}\begin{verbatim}
                         2    2
        (- 2x sin(x) + (y  + x )cos(x))tan(y)
   (2)
                    4     2 2    4
                   y  + 2x y  + x
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexdiffEmpty2}
\begin{paste}{MenuexdiffEmpty2}{MenuexdiffPatch2}
\pastebutton{MenuexdiffEmpty2}{\showpaste}
\tab{5}\spadcommand{differentiate(sin(x) * tan(y)/(x**2 + y**2),x)}
\end{paste}\end{patch}
```

```
\begin{patch}{MenuexdiffPatch3}
\begin{paste}{MenuexdiffFull3}{MenuexdiffEmpty3}
\pastebutton{MenuexdiffFull3}{\hidepaste}
\tab{5}\spadcommand{differentiate(sin(x) * tan(y)/(x**2 + y**2),y)}
\indentrel{3}\begin{verbatim}
   (3)
         2    2              2
       (y  + x )sin(x)tan(y)  - 2y sin(x)tan(y)
     +
         2    2
       (y  + x )sin(x)
  /
      4     2 2    4
     y  + 2x y  + x
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexdiffEmpty3}
\begin{paste}{MenuexdiffEmpty3}{MenuexdiffPatch3}
\pastebutton{MenuexdiffEmpty3}{\showpaste}
\tab{5}\spadcommand{differentiate(sin(x) * tan(y)/(x**2 + y**2),y)}
\end{paste}\end{patch}

\begin{patch}{MenuexdiffPatch4}
\begin{paste}{MenuexdiffFull4}{MenuexdiffEmpty4}
\pastebutton{MenuexdiffFull4}{\hidepaste}
\tab{5}\spadcommand{differentiate(exp(x**2),x,4)}
\indentrel{3}\begin{verbatim}
                             2
           4      2        x
   (4)  (16x  + 48x  + 12)%e
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexdiffEmpty4}
\begin{paste}{MenuexdiffEmpty4}{MenuexdiffPatch4}
\pastebutton{MenuexdiffEmpty4}{\showpaste}
\tab{5}\spadcommand{differentiate(exp(x**2),x,4)}
\end{paste}\end{patch}

\begin{patch}{MenuexdiffPatch5}
\begin{paste}{MenuexdiffFull5}{MenuexdiffEmpty5}
\pastebutton{MenuexdiffFull5}{\hidepaste}
\tab{5}\spadcommand{differentiate(sin(x)/(x**2 + y**2),[x,y])}
\indentrel{3}\begin{verbatim}
                            3     2
        8x y sin(x) + (- 2y  - 2x y)cos(x)
   (5)
```

```
              6     2 4     4 2    6
             y  + 3x y  + 3x y  + x
                                Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexdiffEmpty5}
\begin{paste}{MenuexdiffEmpty5}{MenuexdiffPatch5}
\pastebutton{MenuexdiffEmpty5}{\showpaste}
\tab{5}\spadcommand{differentiate(sin(x)/(x**2 + y**2),[x,y])}
\end{paste}\end{patch}

\begin{patch}{MenuexdiffPatch6}
\begin{paste}{MenuexdiffFull6}{MenuexdiffEmpty6}
\pastebutton{MenuexdiffFull6}{\hidepaste}
\tab{5}\spadcommand{differentiate(sin(x)/(x**2 + y**2),[x,y,y])}
\indentrel{3}\begin{verbatim}
   (6)
           2     3            4     2 2     4
   (- 40x y  + 8x )sin(x) + (6y  + 4x y  - 2x )cos(x)

             8     2 6     4 4     6 2    8
            y  + 4x y  + 6x y  + 4x y  + x
                                Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexdiffEmpty6}
\begin{paste}{MenuexdiffEmpty6}{MenuexdiffPatch6}
\pastebutton{MenuexdiffEmpty6}{\showpaste}
\tab{5}\spadcommand{differentiate(sin(x)/(x**2 + y**2),[x,y,y])}
\end{paste}\end{patch}

\begin{patch}{MenuexdiffPatch7}
\begin{paste}{MenuexdiffFull7}{MenuexdiffEmpty7}
\pastebutton{MenuexdiffFull7}{\hidepaste}
\tab{5}\spadcommand{differentiate(cos(z)/(x**2 + y**3),[x,y,z],[1,2,3])}
\indentrel{3}\begin{verbatim}
                  4      3
            (- 84x y  + 24x y)sin(z)
   (7)
         12     2 9     4 6     6 3    8
        y   + 4x y  + 6x y  + 4x y  + x
                                Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexdiffEmpty7}
\begin{paste}{MenuexdiffEmpty7}{MenuexdiffPatch7}
\pastebutton{MenuexdiffEmpty7}{\showpaste}
```

```
\tab{5}\spadcommand{differentiate(cos(z)/(x**2 + y**3),[x,y,z],[1,2,3])}
\end{paste}\end{patch}

\begin{patch}{MenuexdiffPatch8}
\begin{paste}{MenuexdiffFull8}{MenuexdiffEmpty8}
\pastebutton{MenuexdiffFull8}{\hidepaste}
\tab{5}\spadcommand{f := integrate(sqrt(1 + t**3),t)\bound{f }}
\indentrel{3}\begin{verbatim}
           t
             3
   (8)      \%N  + 1 d%N

                    Type: Union(Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexdiffEmpty8}
\begin{paste}{MenuexdiffEmpty8}{MenuexdiffPatch8}
\pastebutton{MenuexdiffEmpty8}{\showpaste}
\tab{5}\spadcommand{f := integrate(sqrt(1 + t**3),t)\bound{f }}
\end{paste}\end{patch}

\begin{patch}{MenuexdiffPatch9}
\begin{paste}{MenuexdiffFull9}{MenuexdiffEmpty9}
\pastebutton{MenuexdiffFull9}{\hidepaste}
\tab{5}\spadcommand{differentiate(f,t)\free{f }}
\indentrel{3}\begin{verbatim}

          3
   (9)  \t  + 1
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexdiffEmpty9}
\begin{paste}{MenuexdiffEmpty9}{MenuexdiffPatch9}
\pastebutton{MenuexdiffEmpty9}{\showpaste}
\tab{5}\spadcommand{differentiate(f,t)\free{f }}
\end{paste}\end{patch}

\begin{patch}{MenuexdiffPatch10}
\begin{paste}{MenuexdiffFull10}{MenuexdiffEmpty10}
\pastebutton{MenuexdiffFull10}{\hidepaste}
\tab{5}\spadcommand{differentiate(f * t**2,t)\free{f }}
\indentrel{3}\begin{verbatim}
              t
                3             2  3
   (10)  2t    \%N  + 1 d%N  + t \t  + 1

                               Type: Expression Integer
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexdiffEmpty10}
\begin{paste}{MenuexdiffEmpty10}{MenuexdiffPatch10}
\pastebutton{MenuexdiffEmpty10}{\showpaste}
\tab{5}\spadcommand{differentiate(f * t**2,t)\free{f }}
\end{paste}\end{patch}
```

———-

Integration

— **coverex.ht** —

```
\begin{page}{Menuexint}{Integration}
\beginscroll\beginmenu
\menudownlink{Integral of a Rational Function}{ExIntRationalFunction}
\spadpaste{integrate((x**2+2*x+1)/((x+1)**6+1),x)}
\spadpaste{integrate(1/(x**3+x+1),x) \bound{i}}
\spadpaste{definingPolynomial(tower(\%).2::EXPR INT) \free{i}}
\menudownlink{Integral of a Rational Function with a Real Parameter}
{ExIntRationalWithRealParameter}
\spadpaste{integrate(1/(x**2 + a),x)}
\menudownlink{Integral of a Rational Function with a Complex Parameter}
{ExIntRationalWithComplexParameter}
\spadpaste{complexIntegrate(1/(x**2 + a),x)}
\menudownlink{Two Similar Integrands Producing Very Different Results}
{ExIntTwoSimilarIntegrands}
```

```
\spadpaste{integrate(x**3 / (a+b*x)**(1/3),x)}
\spadpaste{integrate(1 / (x**3 * (a+b*x)**(1/3)),x)}
\menudownlink{An Integral Which Does Not Exist}{ExIntNoSolution}
\spadpaste{integrate(log(1 + sqrt(a*x + b)) / x,x)}
\menudownlink{A Trigonometric Function of a Quadratic}{ExIntTrig}
\spadpaste{integrate((sinh(1+sqrt(x+b))+2*sqrt(x+b))/
(sqrt(x+b)*(x+cosh(1+sqrt(x+b)))),x)}
\menudownlink{Integrating a Function with a Hidden Algebraic Relation}
{ExIntAlgebraicRelation}
\spadpaste{integrate(tan(atan(x)/3),x)}
\menudownlink
{Details for integrating a function with a Hidden Algebraic Relation}
{ExIntAlgebraicRelationExplain}
\menudownlink{An Integral Involving a Root of a Transcendental Function}
{ExIntRadicalOfTranscendental}
\spadpaste{integrate((x + 1) / (x * (x + log x)**(3/2)),x)}
\menudownlink{An Integral of a Non-elementary Function}{ExIntNonElementary}
\spadpaste{integrate(exp(-x**2) * erf(x) /
(erf(x)**3 - erf(x)**2 - erf(x) + 1),x)}
\endmenu\endscroll
\end{page}

\begin{patch}{MenuexintPatch1}
\begin{paste}{MenuexintFull1}{MenuexintEmpty1}
\pastebutton{MenuexintFull1}{\hidepaste}
\tab{5}\spadcommand{integrate((x**2+2*x+1)/((x+1)**6+1),x)}
\indentrel{3}\begin{verbatim}
              3     2
        atan(x  + 3x  + 3x + 1)
   (1)  
                   3
                    Type: Union(Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexintEmpty1}
\begin{paste}{MenuexintEmpty1}{MenuexintPatch1}
\pastebutton{MenuexintEmpty1}{\showpaste}
\tab{5}\spadcommand{integrate((x**2+2*x+1)/((x+1)**6+1),x)}
\end{paste}\end{patch}

\begin{patch}{MenuexintPatch2}
\begin{paste}{MenuexintFull2}{MenuexintEmpty2}
\pastebutton{MenuexintFull2}{\hidepaste}
\tab{5}\spadcommand{integrate(1/(x**3+x+1),x)\bound{i }}
\indentrel{3}\begin{verbatim}
   (2)

                    2
           - 93%%CE0  + 12
```

```
        (    - %%CE0)
         \       31
     *
        log

                                   2
                         - 93%%CE0  + 12           2
             (62%%CE0 + 31)    + 62%%CE0
                        \       31
           +
             - 31%%CE0 + 18x - 4
    +

                      2
            - 93%%CE0  + 12
        (-     - %%CE0)
           \       31
     *
        log

                                     2
                           - 93%%CE0  + 12
             (- 62%%CE0 - 31)
                          \       31
           +
                     2
             62%%CE0  - 31%%CE0 + 18x - 4
    +
                           2
      2%%CE0 log(- 62%%CE0  + 31%%CE0 + 9x + 4)
  /
    2
                   Type: Union(Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexintEmpty2}
\begin{paste}{MenuexintEmpty2}{MenuexintPatch2}
\pastebutton{MenuexintEmpty2}{\showpaste}
\tab{5}\spadcommand{integrate(1/(x**3+x+1),x)\bound{i }}
\end{paste}\end{patch}

\begin{patch}{MenuexintPatch3}
\begin{paste}{MenuexintFull3}{MenuexintEmpty3}
\pastebutton{MenuexintFull3}{\hidepaste}
\tab{5}\spadcommand{definingPolynomial(tower(\%).2::EXPR INT)\free{i }}
\indentrel{3}\begin{verbatim}
               3
       31%%CE0  - 3%%CE0 - 1
   (3)
```

```
                 31
                              Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexintEmpty3}
\begin{paste}{MenuexintEmpty3}{MenuexintPatch3}
\pastebutton{MenuexintEmpty3}{\showpaste}
\tab{5}\spadcommand{definingPolynomial(tower(\%).2::EXPR INT)\free{i }}
\end{paste}\end{patch}

\begin{patch}{MenuexintPatch4}
\begin{paste}{MenuexintFull4}{MenuexintEmpty4}
\pastebutton{MenuexintFull4}{\hidepaste}
\tab{5}\spadcommand{integrate(1/(x**2 + a),x)}
\indentrel{3}\begin{verbatim}
               2
             (x  - a)\- a  + 2a x
         log()      x\a
                      2             atan()
                     x  + a                a
   (4)  [,]

                  2\- a               \a
               Type: Union(List Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexintEmpty4}
\begin{paste}{MenuexintEmpty4}{MenuexintPatch4}
\pastebutton{MenuexintEmpty4}{\showpaste}
\tab{5}\spadcommand{integrate(1/(x**2 + a),x)}
\end{paste}\end{patch}

\begin{patch}{MenuexintPatch5}
\begin{paste}{MenuexintFull5}{MenuexintEmpty5}
\pastebutton{MenuexintFull5}{\hidepaste}
\tab{5}\spadcommand{complexIntegrate(1/(x**2 + a),x)}
\indentrel{3}\begin{verbatim}

            x\- a  + a        x\- a  - a
        log() - log()

               \- a              \- a
   (5)

                      2\- a
                              Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{MenuexintEmpty5}
\begin{paste}{MenuexintEmpty5}{MenuexintPatch5}
\pastebutton{MenuexintEmpty5}{\showpaste}
\tab{5}\spadcommand{complexIntegrate(1/(x**2 + a),x)}
\end{paste}\end{patch}

\begin{patch}{MenuexintPatch6}
\begin{paste}{MenuexintFull6}{MenuexintEmpty6}
\pastebutton{MenuexintFull6}{\hidepaste}
\tab{5}\spadcommand{integrate(x**3 / (a+b*x)**(1/3),x)}
\indentrel{3}\begin{verbatim}
   (6)
        3 3         2 2       2          3 32
   (120b x  - 135a b x  + 162a b x - 243a )\b x + a

                              4
                          440b
                    Type: Union(Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexintEmpty6}
\begin{paste}{MenuexintEmpty6}{MenuexintPatch6}
\pastebutton{MenuexintEmpty6}{\showpaste}
\tab{5}\spadcommand{integrate(x**3 / (a+b*x)**(1/3),x)}
\end{paste}\end{patch}

\begin{patch}{MenuexintPatch7}
\begin{paste}{MenuexintFull7}{MenuexintEmpty7}
\pastebutton{MenuexintFull7}{\hidepaste}
\tab{5}\spadcommand{integrate(1 / (x**3 * (a+b*x)**(1/3)),x)}
\indentrel{3}\begin{verbatim}
   (7)
       -
              2 2
            2b x \3
         *
                332   32 3
            log(\a \b x + a   + \a   \b x + a + a)
     +
         2 2     32 3
       4b x \3 log(\a   \b x + a - a)
     +
                     32 3
          2 2     2\3 \a   \b x + a + a\3
       12b x atan()
                              3a
     +
                     332
```

```
       (12b x - 9a)\3 \a \b x + a
  /
        2 2 3
     18a x \3 \a
                    Type: Union(Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexintEmpty7}
\begin{paste}{MenuexintEmpty7}{MenuexintPatch7}
\pastebutton{MenuexintEmpty7}{\showpaste}
\tab{5}\spadcommand{integrate(1 / (x**3 * (a+b*x)**(1/3)),x)}
\end{paste}\end{patch}

\begin{patch}{MenuexintPatch8}
\begin{paste}{MenuexintFull8}{MenuexintEmpty8}
\pastebutton{MenuexintFull8}{\hidepaste}
\tab{5}\spadcommand{integrate(log(1 + sqrt(a*x + b)) / x,x)}
\indentrel{3}\begin{verbatim}
           x
           log(\b + %N a  + 1)
   (8)       d%N
                    %N
                    Type: Union(Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexintEmpty8}
\begin{paste}{MenuexintEmpty8}{MenuexintPatch8}
\pastebutton{MenuexintEmpty8}{\showpaste}
\tab{5}\spadcommand{integrate(log(1 + sqrt(a*x + b)) / x,x)}
\end{paste}\end{patch}

\begin{patch}{MenuexintPatch9}
\begin{paste}{MenuexintFull9}{MenuexintEmpty9}
\pastebutton{MenuexintFull9}{\hidepaste}
\tab{5}\spadcommand{
integrate((sinh(1+sqrt(x+b))+2*sqrt(x+b))/_
          (sqrt(x+b)*(x+cosh(1+sqrt(x+b)))),x)}
\indentrel{3}\begin{verbatim}
   (9)

                 - 2cosh(\x + b  + 1) - 2x
     2log()

          sinh(\x + b  + 1) - cosh(\x + b  + 1)
   +

     - 2\x + b
                    Type: Union(Expression Integer,...)
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexintEmpty9}
\begin{paste}{MenuexintEmpty9}{MenuexintPatch9}
\pastebutton{MenuexintEmpty9}{\showpaste}
\tab{5}\spadcommand{
integrate((sinh(1+sqrt(x+b))+2*sqrt(x+b))/_
          (sqrt(x+b)*(x+cosh(1+sqrt(x+b)))),x)}
\end{paste}\end{patch}

\begin{patch}{MenuexintPatch10}
\begin{paste}{MenuexintFull10}{MenuexintEmpty10}
\pastebutton{MenuexintFull10}{\hidepaste}
\tab{5}\spadcommand{integrate(tan(atan(x)/3),x)}
\indentrel{3}\begin{verbatim}
   (10)
                 atan(x) 2             atan(x) 2
       8log(3tan()  - 1) - 3tan()
                   3                     3
     +
               atan(x)
       18x tan()
                  3
  /
     18
                    Type: Union(Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexintEmpty10}
\begin{paste}{MenuexintEmpty10}{MenuexintPatch10}
\pastebutton{MenuexintEmpty10}{\showpaste}
\tab{5}\spadcommand{integrate(tan(atan(x)/3),x)}
\end{paste}\end{patch}

\begin{patch}{MenuexintPatch11}
\begin{paste}{MenuexintFull11}{MenuexintEmpty11}
\pastebutton{MenuexintFull11}{\hidepaste}
\tab{5}\spadcommand{integrate((x + 1) / (x * (x + log x)**(3/2)),x)}
\indentrel{3}\begin{verbatim}

           2\log(x) + x
   (11)  -
             log(x) + x
                    Type: Union(Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexintEmpty11}
```

```
\begin{paste}{MenuexintEmpty11}{MenuexintPatch11}
\pastebutton{MenuexintEmpty11}{\showpaste}
\tab{5}\spadcommand{integrate((x + 1) / (x * (x + log x)**(3/2)),x)}
\end{paste}\end{patch}

\begin{patch}{MenuexintPatch12}
\begin{paste}{MenuexintFull12}{MenuexintEmpty12}
\pastebutton{MenuexintFull12}{\hidepaste}
\tab{5}\spadcommand{integrate(exp(-x**2) * erf(x) / (erf(x)**3 - erf(x)**2 - erf(x) + 1),x)}
\indentrel{3}\begin{verbatim}
                         erf(x) - 1
         (erf(x) - 1)\%pi log() - 2\%pi
                              erf(x) + 1
   (12)
                        8erf(x) - 8
                   Type: Union(Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexintEmpty12}
\begin{paste}{MenuexintEmpty12}{MenuexintPatch12}
\pastebutton{MenuexintEmpty12}{\showpaste}
\tab{5}\spadcommand{integrate(exp(-x**2) * erf(x) /
 (erf(x)**3 - erf(x)**2 - erf(x) + 1),x)}
\end{paste}\end{patch}
```

——————

Laplace Transforms

⇒ "Laplace transform with a single pole" (ExLapSimplePole) 3.29 on page 402
⇒ "Laplace transform of a trigonometric function" (ExLapTrigTrigh) 3.29 on page 402
⇒ "Laplace transform requiring a definite integration" (ExLapDefInt) 3.29 on page 403
⇒ "Laplace transform of exponentials" (ExLapExpExp) 3.29 on page 404
⇒ "Laplace transform of an exponential integral" (ExLapSpecial1) 3.29 on page 405
⇒ "Laplace transform of special functions" (ExLapSpecial2) 3.29 on page 406

— **coverex.ht** —

```
\begin{page}{Menuexlap}{Laplace Transforms}
\beginscroll\beginmenu
\menudownlink
\spadpaste{laplace(t**4 * exp(-a*t) / factorial(4), t, s)}
\menudownlink{Laplace transform of a trigonometric function}
{ExLapTrigTrigh}
\spadpaste{laplace(sin(a*t) * cosh(a*t) - cos(a*t) * sinh(a*t), t, s)}
\menudownlink{Laplace transform requiring a definite integration}
```

```
{ExLapDefInt}
\spadpaste{laplace(2/t * (1 - cos(a*t)), t, s)}
\menudownlink{Laplace transform of exponentials}{ExLapExpExp}
\spadpaste{laplace((exp(a*t) - exp(b*t))/t, t, s)}
\menudownlink{Laplace transform of an exponential integral}{ExLapSpecial1}
\spadpaste{laplace(exp(a*t+b)*Ei(c*t), t, s)}
\menudownlink{Laplace transform of special functions}{ExLapSpecial2}
\spadpaste{laplace(a*Ci(b*t) + c*Si(d*t), t, s)}
\endmenu\endscroll
\end{page}

\begin{patch}{MenuexlapPatch1}
\begin{paste}{MenuexlapFull1}{MenuexlapEmpty1}
\pastebutton{MenuexlapFull1}{\hidepaste}
\tab{5}\spadcommand{laplace(t**4 * exp(-a*t) / factorial(4), t, s)}
\indentrel{3}\begin{verbatim}
                            1
   (1)
         5       4      2 3      3 2     4     5
        s  + 5a s  + 10a s  + 10a s  + 5a s + a
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexlapEmpty1}
\begin{paste}{MenuexlapEmpty1}{MenuexlapPatch1}
\pastebutton{MenuexlapEmpty1}{\showpaste}
\tab{5}\spadcommand{laplace(t**4 * exp(-a*t) / factorial(4), t, s)}
\end{paste}\end{patch}

\begin{patch}{MenuexlapPatch2}
\begin{paste}{MenuexlapFull2}{MenuexlapEmpty2}
\pastebutton{MenuexlapFull2}{\hidepaste}
\tab{5}\spadcommand{laplace(sin(a*t) * cosh(a*t) - cos(a*t) * sinh(a*t), t, s)}
\indentrel{3}\begin{verbatim}
             3
           4a
   (2)
         4     4
        s  + 4a
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexlapEmpty2}
\begin{paste}{MenuexlapEmpty2}{MenuexlapPatch2}
\pastebutton{MenuexlapEmpty2}{\showpaste}
\tab{5}\spadcommand{laplace(sin(a*t) * cosh(a*t) - cos(a*t) * sinh(a*t), t, s)}
\end{paste}\end{patch}
```

```
\begin{patch}{MenuexlapPatch3}
\begin{paste}{MenuexlapFull3}{MenuexlapEmpty3}
\pastebutton{MenuexlapFull3}{\hidepaste}
\tab{5}\spadcommand{laplace(2/t * (1 - cos(a*t)), t, s)}
\indentrel{3}\begin{verbatim}
             2    2
   (3)  log(s  + a ) - 2log(s)
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexlapEmpty3}
\begin{paste}{MenuexlapEmpty3}{MenuexlapPatch3}
\pastebutton{MenuexlapEmpty3}{\showpaste}
\tab{5}\spadcommand{laplace(2/t * (1 - cos(a*t)), t, s)}
\end{paste}\end{patch}

\begin{patch}{MenuexlapPatch4}
\begin{paste}{MenuexlapFull4}{MenuexlapEmpty4}
\pastebutton{MenuexlapFull4}{\hidepaste}
\tab{5}\spadcommand{laplace((exp(a*t) - exp(b*t))/t, t, s)}
\indentrel{3}\begin{verbatim}
   (4)  - log(s - a) + log(s - b)
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexlapEmpty4}
\begin{paste}{MenuexlapEmpty4}{MenuexlapPatch4}
\pastebutton{MenuexlapEmpty4}{\showpaste}
\tab{5}\spadcommand{laplace((exp(a*t) - exp(b*t))/t, t, s)}
\end{paste}\end{patch}

\begin{patch}{MenuexlapPatch5}
\begin{paste}{MenuexlapFull5}{MenuexlapEmpty5}
\pastebutton{MenuexlapFull5}{\hidepaste}
\tab{5}\spadcommand{laplace(exp(a*t+b)*Ei(c*t), t, s)}
\indentrel{3}\begin{verbatim}
          b    s + c - a
        %e log()
                   c
   (5)
              s - a
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexlapEmpty5}
\begin{paste}{MenuexlapEmpty5}{MenuexlapPatch5}
\pastebutton{MenuexlapEmpty5}{\showpaste}
```

```
\tab{5}\spadcommand{laplace(exp(a*t+b)*Ei(c*t), t, s)}
\end{paste}\end{patch}

\begin{patch}{MenuexlapPatch6}
\begin{paste}{MenuexlapFull6}{MenuexlapEmpty6}
\pastebutton{MenuexlapFull6}{\hidepaste}
\tab{5}\spadcommand{laplace(a*Ci(b*t) + c*Si(d*t), t, s)}
\indentrel{3}\begin{verbatim}
               2    2
              s   + b             d
        a log() + 2c atan()
                  2               s
                 b
   (6)
                     2s
                                Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexlapEmpty6}
\begin{paste}{MenuexlapEmpty6}{MenuexlapPatch6}
\pastebutton{MenuexlapEmpty6}{\showpaste}
\tab{5}\spadcommand{laplace(a*Ci(b*t) + c*Si(d*t), t, s)}
\end{paste}\end{patch}
```

———-

Limits

⇒ “Computing Limits” (ExLimitBasic) 3.31 on page 417
⇒ “Limits of Functions with Parameters” (ExLimitParameter) 3.31 on page 418
⇒ “One-sided Limits” (ExLimitOneSided) 3.31 on page 419
⇒ “Two-sided Limits” (ExLimitTwoSided) 3.31 on page 420
⇒ “Limits at Infinity” (ExLimitInfinite) 3.31 on page 422
⇒ “Real Limits vs. Complex Limits” (ExLimitRealComplex) 3.31 on page 423
⇒ “Complex Limits at Infinity” (ExLimitComplexInfinite) 3.31 on page 424

— coverex.ht —

```
\begin{page}{Menuexlimit}{Limits}
\beginscroll\beginmenu
\menudownlink{Computing Limits}{ExLimitBasic}
\spadpaste{limit((x**2 - 3*x + 2)/(x**2 - 1),x = 1)}
\menudownlink{Limits of Functions with Parameters}{ExLimitParameter}
\spadpaste{limit(sinh(a*x)/tan(b*x),x = 0)}
\menudownlink{One-sided Limits}{ExLimitOneSided}
\spadpaste{limit(x * log(x),x = 0,"right")}
```

```
\spadpaste{limit(x * log(x),x = 0)}
\menudownlink{Two-sided Limits}{ExLimitTwoSided}
\spadpaste{limit(sqrt(y**2)/y,y = 0)}
\spadpaste{limit(sqrt(1 - cos(t))/t,t = 0)}
\menudownlink{Limits at Infinity}{ExLimitInfinite}
\spadpaste{limit(sqrt(3*x**2 + 1)/(5*x),x = \%plusInfinity)}
\spadpaste{limit(sqrt(3*x**2 + 1)/(5*x),x = \%minusInfinity)}
\menudownlink{Real Limits vs. Complex Limits}{ExLimitRealComplex}
\spadpaste{limit(z * sin(1/z),z = 0)}
\spadpaste{complexLimit(z * sin(1/z),z = 0)}
\menudownlink{Complex Limits at Infinity}{ExLimitComplexInfinite}
\spadpaste{complexLimit((2 + z)/(1 - z),z = \%infinity)}
\spadpaste{limit(sin(x)/x,x = \%plusInfinity)}
\spadpaste{complexLimit(sin(x)/x,x = \%infinity)}
\endmenu\endscroll
\end{page}

\begin{patch}{MenuexlimitPatch1}
\begin{paste}{MenuexlimitFull1}{MenuexlimitEmpty1}
\pastebutton{MenuexlimitFull1}{\hidepaste}
\tab{5}\spadcommand{limit((x**2 - 3*x + 2)/(x**2 - 1),x = 1)}
\indentrel{3}\begin{verbatim}
          1
   (1)  -
          2
Type: Union(OrderedCompletion Fraction Polynomial Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexlimitEmpty1}
\begin{paste}{MenuexlimitEmpty1}{MenuexlimitPatch1}
\pastebutton{MenuexlimitEmpty1}{\showpaste}
\tab{5}\spadcommand{limit((x**2 - 3*x + 2)/(x**2 - 1),x = 1)}
\end{paste}\end{patch}

\begin{patch}{MenuexlimitPatch2}
\begin{paste}{MenuexlimitFull2}{MenuexlimitEmpty2}
\pastebutton{MenuexlimitFull2}{\hidepaste}
\tab{5}\spadcommand{limit(sinh(a*x)/tan(b*x),x = 0)}
\indentrel{3}\begin{verbatim}
        a
   (2)
        b
  Type: Union(OrderedCompletion Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexlimitEmpty2}
\begin{paste}{MenuexlimitEmpty2}{MenuexlimitPatch2}
\pastebutton{MenuexlimitEmpty2}{\showpaste}
```

```
\tab{5}\spadcommand{limit(sinh(a*x)/tan(b*x),x = 0)}
\end{paste}\end{patch}

\begin{patch}{MenuexlimitPatch3}
\begin{paste}{MenuexlimitFull3}{MenuexlimitEmpty3}
\pastebutton{MenuexlimitFull3}{\hidepaste}
\tab{5}\spadcommand{limit(x * log(x),x = 0,"right")}
\indentrel{3}\begin{verbatim}
   (3)  0
  Type: Union(OrderedCompletion Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexlimitEmpty3}
\begin{paste}{MenuexlimitEmpty3}{MenuexlimitPatch3}
\pastebutton{MenuexlimitEmpty3}{\showpaste}
\tab{5}\spadcommand{limit(x * log(x),x = 0,"right")}
\end{paste}\end{patch}

\begin{patch}{MenuexlimitPatch4}
\begin{paste}{MenuexlimitFull4}{MenuexlimitEmpty4}
\pastebutton{MenuexlimitFull4}{\hidepaste}
\tab{5}\spadcommand{limit(x * log(x),x = 0)}
\indentrel{3}\begin{verbatim}
   (4)  [leftHandLimit= "failed",rightHandLimit= 0]
Type: Union(Record(leftHandLimit: Union(OrderedCompletion Expression Integer,"failed"),rightHand
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexlimitEmpty4}
\begin{paste}{MenuexlimitEmpty4}{MenuexlimitPatch4}
\pastebutton{MenuexlimitEmpty4}{\showpaste}
\tab{5}\spadcommand{limit(x * log(x),x = 0)}
\end{paste}\end{patch}

\begin{patch}{MenuexlimitPatch5}
\begin{paste}{MenuexlimitFull5}{MenuexlimitEmpty5}
\pastebutton{MenuexlimitFull5}{\hidepaste}
\tab{5}\spadcommand{limit(sqrt(y**2)/y,y = 0)}
\indentrel{3}\begin{verbatim}
   (5)  [leftHandLimit= - 1,rightHandLimit= 1]
Type: Union(Record(leftHandLimit: Union(OrderedCompletion Expression Integer,"failed"),rightHand
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexlimitEmpty5}
\begin{paste}{MenuexlimitEmpty5}{MenuexlimitPatch5}
\pastebutton{MenuexlimitEmpty5}{\showpaste}
\tab{5}\spadcommand{limit(sqrt(y**2)/y,y = 0)}
\end{paste}\end{patch}
```

```
\begin{patch}{MenuexlimitPatch6}
\begin{paste}{MenuexlimitFull6}{MenuexlimitEmpty6}
\pastebutton{MenuexlimitFull6}{\hidepaste}
\tab{5}\spadcommand{limit(sqrt(1 - cos(t))/t,t = 0)}
\indentrel{3}\begin{verbatim}
                            1                    1
   (6)  [leftHandLimit= - ,rightHandLimit= ]

                          \2                   \2
Type: Union(Record(leftHandLimit: Union(OrderedCompletion Expression Integer,"failed"),rightHandLimit: Unio
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexlimitEmpty6}
\begin{paste}{MenuexlimitEmpty6}{MenuexlimitPatch6}
\pastebutton{MenuexlimitEmpty6}{\showpaste}
\tab{5}\spadcommand{limit(sqrt(1 - cos(t))/t,t = 0)}
\end{paste}\end{patch}

\begin{patch}{MenuexlimitPatch7}
\begin{paste}{MenuexlimitFull7}{MenuexlimitEmpty7}
\pastebutton{MenuexlimitFull7}{\hidepaste}
\tab{5}\spadcommand{limit(sqrt(3*x**2 + 1)/(5*x),x = \%plusInfinity)}
\indentrel{3}\begin{verbatim}

        \3
   (7)
          5
  Type: Union(OrderedCompletion Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexlimitEmpty7}
\begin{paste}{MenuexlimitEmpty7}{MenuexlimitPatch7}
\pastebutton{MenuexlimitEmpty7}{\showpaste}
\tab{5}\spadcommand{limit(sqrt(3*x**2 + 1)/(5*x),x = \%plusInfinity)}
\end{paste}\end{patch}

\begin{patch}{MenuexlimitPatch8}
\begin{paste}{MenuexlimitFull8}{MenuexlimitEmpty8}
\pastebutton{MenuexlimitFull8}{\hidepaste}
\tab{5}\spadcommand{limit(sqrt(3*x**2 + 1)/(5*x),x = \%minusInfinity)}
\indentrel{3}\begin{verbatim}

          \3
   (8)  -
            5
  Type: Union(OrderedCompletion Expression Integer,...)
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexlimitEmpty8}
\begin{paste}{MenuexlimitEmpty8}{MenuexlimitPatch8}
\pastebutton{MenuexlimitEmpty8}{\showpaste}
\tab{5}\spadcommand{limit(sqrt(3*x**2 + 1)/(5*x),x = \%minusInfinity)}
\end{paste}\end{patch}

\begin{patch}{MenuexlimitPatch9}
\begin{paste}{MenuexlimitFull9}{MenuexlimitEmpty9}
\pastebutton{MenuexlimitFull9}{\hidepaste}
\tab{5}\spadcommand{limit(z * sin(1/z),z = 0)}
\indentrel{3}\begin{verbatim}
   (9)  0
  Type: Union(OrderedCompletion Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexlimitEmpty9}
\begin{paste}{MenuexlimitEmpty9}{MenuexlimitPatch9}
\pastebutton{MenuexlimitEmpty9}{\showpaste}
\tab{5}\spadcommand{limit(z * sin(1/z),z = 0)}
\end{paste}\end{patch}

\begin{patch}{MenuexlimitPatch10}
\begin{paste}{MenuexlimitFull10}{MenuexlimitEmpty10}
\pastebutton{MenuexlimitFull10}{\hidepaste}
\tab{5}\spadcommand{complexLimit(z * sin(1/z),z = 0)}
\indentrel{3}\begin{verbatim}
   (10)  "failed"
                              Type: Union("failed",...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexlimitEmpty10}
\begin{paste}{MenuexlimitEmpty10}{MenuexlimitPatch10}
\pastebutton{MenuexlimitEmpty10}{\showpaste}
\tab{5}\spadcommand{complexLimit(z * sin(1/z),z = 0)}
\end{paste}\end{patch}

\begin{patch}{MenuexlimitPatch11}
\begin{paste}{MenuexlimitFull11}{MenuexlimitEmpty11}
\pastebutton{MenuexlimitFull11}{\hidepaste}
\tab{5}\spadcommand{complexLimit((2 + z)/(1 - z),z = \%infinity)}
\indentrel{3}\begin{verbatim}
   (11)  - 1
   Type: OnePointCompletion Fraction Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{MenuexlimitEmpty11}
\begin{paste}{MenuexlimitEmpty11}{MenuexlimitPatch11}
\pastebutton{MenuexlimitEmpty11}{\showpaste}
\tab{5}\spadcommand{complexLimit((2 + z)/(1 - z),z = \%infinity)}
\end{paste}\end{patch}

\begin{patch}{MenuexlimitPatch12}
\begin{paste}{MenuexlimitFull12}{MenuexlimitEmpty12}
\pastebutton{MenuexlimitFull12}{\hidepaste}
\tab{5}\spadcommand{limit(sin(x)/x,x = \%plusInfinity)}
\indentrel{3}\begin{verbatim}
   (12)  0
  Type: Union(OrderedCompletion Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexlimitEmpty12}
\begin{paste}{MenuexlimitEmpty12}{MenuexlimitPatch12}
\pastebutton{MenuexlimitEmpty12}{\showpaste}
\tab{5}\spadcommand{limit(sin(x)/x,x = \%plusInfinity)}
\end{paste}\end{patch}

\begin{patch}{MenuexlimitPatch13}
\begin{paste}{MenuexlimitFull13}{MenuexlimitEmpty13}
\pastebutton{MenuexlimitFull13}{\hidepaste}
\tab{5}\spadcommand{complexLimit(sin(x)/x,x = \%infinity)}
\indentrel{3}\begin{verbatim}
   (13)  "failed"
                              Type: Union("failed",...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexlimitEmpty13}
\begin{paste}{MenuexlimitEmpty13}{MenuexlimitPatch13}
\pastebutton{MenuexlimitEmpty13}{\showpaste}
\tab{5}\spadcommand{complexLimit(sin(x)/x,x = \%infinity)}
\end{paste}\end{patch}
```

Matrices

⇒ "Basic Arithmetic Operations on Matrices" (ExMatrixBasicFunction) 3.32 on page 426
⇒ "Constructing new Matrices" (ExConstructMatrix) 3.32 on page 429
⇒ "Trace of a Matrix" (ExTraceMatrix) 3.32 on page 433
⇒ "Determinant of a Matrix" (ExDeterminantMatrix) 3.32 on page 433

⇒ "Inverse of a Matrix" (ExInverseMatrix) 3.32 on page 434
⇒ "Rank of a Matrix" (ExRankMatrix) 3.32 on page 435

— **coverex.ht** —

```
\begin{page}{Menuexmatrix}{Matrices}
\beginscroll\beginmenu
\menudownlink{Basic Arithmetic Operations on Matrices}
{ExMatrixBasicFunction}
\spadpaste{m1 := matrix([[1,-2,1],[4,2,-4]]) \bound{m1}}
\spadpaste{m2 := matrix([[1,0,2],[20,30,10],[0,200,100]]) \bound{m2}}
\spadpaste{m3 := matrix([[1,2,3],[2,4,6]]) \bound{m3}}
\spadpaste{m1 + m3 \free{m1} \free{m3}}
\spadpaste{100 * m1 \free{m1}}
\spadpaste{m1 * m2 \free{m1} \free{m2}}
\spadpaste{-m1 + m3 * m2 \free{m1} \free{m2} \free{m3}}
\spadpaste{m3 *vector([1,0,1]) \free{m3}}
\menudownlink{Constructing new Matrices}{ExConstructMatrix}
\spadpaste{diagonalMatrix([1,2,3,2,1])}
\spadpaste{subMatrix(matrix([[0,1,2,3,4],[5,6,7,8,9],[10,11,12,13,14]]),
1,3,2,4)}
\spadpaste{horizConcat(matrix([[1,2,3],[6,7,8]]),
matrix([[11,12,13],[55,77,88]])) }
\spadpaste{vertConcat(matrix([[1,2,3],[6,7,8]]),
matrix([[11,12,13],[55,77,88]])) }
\spadpaste{b:=matrix([[0,1,2,3,4],[5,6,7,8,9],[10,11,12,13,14]]) \bound{b}}
\spadpaste{setsubMatrix!(b,1,1,transpose(subMatrix(b,1,3,1,3)))\free{b}}
\menudownlink{Trace of a Matrix}{ExTraceMatrix}
\spadpaste{trace(
matrix([[1,x,x**2,x**3],[1,y,y**2,y**3],
[1,z,z**2,z**3],[1,u,u**2,u**3]]) )}
\menudownlink{Determinant of a Matrix}{ExDeterminantMatrix}
\spadpaste{
determinant(matrix([[1,2,3,4],[2,3,2,5],[3,4,5,6],[4,1,6,7]]))}
\menudownlink{Inverse of a Matrix}{ExInverseMatrix}
\spadpaste{inverse(matrix([[1,2,1],[-2,3,4],[-1,5,6]])) }
\menudownlink{Rank of a Matrix}{ExRankMatrix}
\spadpaste{rank(matrix([[0,4,1],[5,3,-7],[-5,5,9]]))}
\endmenu\endscroll
\end{page}

\begin{patch}{MenuexmatrixPatch1}
\begin{paste}{MenuexmatrixFull1}{MenuexmatrixEmpty1}
\pastebutton{MenuexmatrixFull1}{\hidepaste}
\tab{5}\spadcommand{m1 := matrix([[1,-2,1],[4,2,-4]])\bound{m1 }}
\indentrel{3}\begin{verbatim}
        1  - 2   1
   (1)
        4   2   - 4
                                   Type: Matrix Integer
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexmatrixEmpty1}
\begin{paste}{MenuexmatrixEmpty1}{MenuexmatrixPatch1}
\pastebutton{MenuexmatrixEmpty1}{\showpaste}
\tab{5}\spadcommand{m1 := matrix([[1,-2,1],[4,2,-4]])\bound{m1 }}
\end{paste}\end{patch}

\begin{patch}{MenuexmatrixPatch2}
\begin{paste}{MenuexmatrixFull2}{MenuexmatrixEmpty2}
\pastebutton{MenuexmatrixFull2}{\hidepaste}
\tab{5}\spadcommand{m2 := matrix([[1,0,2],[20,30,10],[0,200,100]])\bound{m2 }}
\indentrel{3}\begin{verbatim}
        1    0    2

   (2)  20  30   10

        0   200  100
                                   Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexmatrixEmpty2}
\begin{paste}{MenuexmatrixEmpty2}{MenuexmatrixPatch2}
\pastebutton{MenuexmatrixEmpty2}{\showpaste}
\tab{5}\spadcommand{m2 := matrix([[1,0,2],[20,30,10],[0,200,100]])\bound{m2 }}
\end{paste}\end{patch}

\begin{patch}{MenuexmatrixPatch3}
\begin{paste}{MenuexmatrixFull3}{MenuexmatrixEmpty3}
\pastebutton{MenuexmatrixFull3}{\hidepaste}
\tab{5}\spadcommand{m3 := matrix([[1,2,3],[2,4,6]])\bound{m3 }}
\indentrel{3}\begin{verbatim}
        1  2  3
   (3)
        2  4  6
                                   Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexmatrixEmpty3}
\begin{paste}{MenuexmatrixEmpty3}{MenuexmatrixPatch3}
\pastebutton{MenuexmatrixEmpty3}{\showpaste}
\tab{5}\spadcommand{m3 := matrix([[1,2,3],[2,4,6]])\bound{m3 }}
\end{paste}\end{patch}

\begin{patch}{MenuexmatrixPatch4}
\begin{paste}{MenuexmatrixFull4}{MenuexmatrixEmpty4}
\pastebutton{MenuexmatrixFull4}{\hidepaste}
\tab{5}\spadcommand{m1 + m3\free{m1 }\free{m3 }}
```

```
\indentrel{3}\begin{verbatim}
        2  0  4
   (4)
        6  6  2
                                   Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexmatrixEmpty4}
\begin{paste}{MenuexmatrixEmpty4}{MenuexmatrixPatch4}
\pastebutton{MenuexmatrixEmpty4}{\showpaste}
\tab{5}\spadcommand{m1 + m3\free{m1 }\free{m3 }}
\end{paste}\end{patch}

\begin{patch}{MenuexmatrixPatch5}
\begin{paste}{MenuexmatrixFull5}{MenuexmatrixEmpty5}
\pastebutton{MenuexmatrixFull5}{\hidepaste}
\tab{5}\spadcommand{100 * m1\free{m1 }}
\indentrel{3}\begin{verbatim}
        100  - 200   100
   (5)
        400   200   - 400
                                   Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexmatrixEmpty5}
\begin{paste}{MenuexmatrixEmpty5}{MenuexmatrixPatch5}
\pastebutton{MenuexmatrixEmpty5}{\showpaste}
\tab{5}\spadcommand{100 * m1\free{m1 }}
\end{paste}\end{patch}

\begin{patch}{MenuexmatrixPatch6}
\begin{paste}{MenuexmatrixFull6}{MenuexmatrixEmpty6}
\pastebutton{MenuexmatrixFull6}{\hidepaste}
\tab{5}\spadcommand{m1 * m2\free{m1 }\free{m2 }}
\indentrel{3}\begin{verbatim}
        - 39   140    82
   (6)
         44   - 740  - 372
                                   Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexmatrixEmpty6}
\begin{paste}{MenuexmatrixEmpty6}{MenuexmatrixPatch6}
\pastebutton{MenuexmatrixEmpty6}{\showpaste}
\tab{5}\spadcommand{m1 * m2\free{m1 }\free{m2 }}
\end{paste}\end{patch}
```

```
\begin{patch}{MenuexmatrixPatch7}
\begin{paste}{MenuexmatrixFull7}{MenuexmatrixEmpty7}
\pastebutton{MenuexmatrixFull7}{\hidepaste}
\tab{5}\spadcommand{-m1 + m3 * m2\free{m1 }\free{m2 }\free{m3 }}
\indentrel{3}\begin{verbatim}
        40  662   321
   (7)
        78  1318  648
                                  Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexmatrixEmpty7}
\begin{paste}{MenuexmatrixEmpty7}{MenuexmatrixPatch7}
\pastebutton{MenuexmatrixEmpty7}{\showpaste}
\tab{5}\spadcommand{-m1 + m3 * m2\free{m1 }\free{m2 }\free{m3 }}
\end{paste}\end{patch}

\begin{patch}{MenuexmatrixPatch8}
\begin{paste}{MenuexmatrixFull8}{MenuexmatrixEmpty8}
\pastebutton{MenuexmatrixFull8}{\hidepaste}
\tab{5}\spadcommand{m3 *vector([1,0,1])\free{m3 }}
\indentrel{3}\begin{verbatim}
   (8)  [4,8]
                                  Type: Vector Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexmatrixEmpty8}
\begin{paste}{MenuexmatrixEmpty8}{MenuexmatrixPatch8}
\pastebutton{MenuexmatrixEmpty8}{\showpaste}
\tab{5}\spadcommand{m3 *vector([1,0,1])\free{m3 }}
\end{paste}\end{patch}

\begin{patch}{MenuexmatrixPatch9}
\begin{paste}{MenuexmatrixFull9}{MenuexmatrixEmpty9}
\pastebutton{MenuexmatrixFull9}{\hidepaste}
\tab{5}\spadcommand{diagonalMatrix([1,2,3,2,1])}
\indentrel{3}\begin{verbatim}
        1  0  0  0  0

        0  2  0  0  0

   (9)  0  0  3  0  0

        0  0  0  2  0

        0  0  0  0  1
                                  Type: Matrix Integer
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexmatrixEmpty9}
\begin{paste}{MenuexmatrixEmpty9}{MenuexmatrixPatch9}
\pastebutton{MenuexmatrixEmpty9}{\showpaste}
\tab{5}\spadcommand{diagonalMatrix([1,2,3,2,1])}
\end{paste}\end{patch}

\begin{patch}{MenuexmatrixPatch10}
\begin{paste}{MenuexmatrixFull10}{MenuexmatrixEmpty10}
\pastebutton{MenuexmatrixFull10}{\hidepaste}
\tab{5}\spadcommand{subMatrix(matrix([[0,1,2,3,4],[5,6,7,8,9],[10,11,12,13,14]]), 1,3,2,4)}
\indentrel{3}\begin{verbatim}
         1   2   3

   (10)  6   7   8

         11  12  13
                                   Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexmatrixEmpty10}
\begin{paste}{MenuexmatrixEmpty10}{MenuexmatrixPatch10}
\pastebutton{MenuexmatrixEmpty10}{\showpaste}
\tab{5}\spadcommand{subMatrix(matrix([[0,1,2,3,4],[5,6,7,8,9],[10,11,12,13,14]]), 1,3,2,4)}
\end{paste}\end{patch}

\begin{patch}{MenuexmatrixPatch11}
\begin{paste}{MenuexmatrixFull11}{MenuexmatrixEmpty11}
\pastebutton{MenuexmatrixFull11}{\hidepaste}
\tab{5}\spadcommand{horizConcat(matrix([[1,2,3],[6,7,8]]),matrix([[11,12,13],[55,77,88]]))}
\indentrel{3}\begin{verbatim}
         1  2  3  11  12  13
   (11)
         6  7  8  55  77  88
                                   Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexmatrixEmpty11}
\begin{paste}{MenuexmatrixEmpty11}{MenuexmatrixPatch11}
\pastebutton{MenuexmatrixEmpty11}{\showpaste}
\tab{5}\spadcommand{horizConcat(matrix([[1,2,3],[6,7,8]]),matrix([[11,12,13],[55,77,88]]))}
\end{paste}\end{patch}

\begin{patch}{MenuexmatrixPatch12}
\begin{paste}{MenuexmatrixFull12}{MenuexmatrixEmpty12}
\pastebutton{MenuexmatrixFull12}{\hidepaste}
\tab{5}\spadcommand{vertConcat(matrix([[1,2,3],[6,7,8]]),matrix([[11,12,13],[55,77,88]]))}
```

```
\indentrel{3}\begin{verbatim}
        1   2   3

        6   7   8
   (12)
        11  12  13

        55  77  88
                                   Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexmatrixEmpty12}
\begin{paste}{MenuexmatrixEmpty12}{MenuexmatrixPatch12}
\pastebutton{MenuexmatrixEmpty12}{\showpaste}
\tab{5}\spadcommand{vertConcat(matrix([[1,2,3],[6,7,8]]),matrix([[11,12,13],[55,77,88]]))}
\end{paste}\end{patch}

\begin{patch}{MenuexmatrixPatch13}
\begin{paste}{MenuexmatrixFull13}{MenuexmatrixEmpty13}
\pastebutton{MenuexmatrixFull13}{\hidepaste}
\tab{5}\spadcommand{b:=matrix([[0,1,2,3,4],[5,6,7,8,9],[10,11,12,13,14]])\bound{b }}
\indentrel{3}\begin{verbatim}
        0   1   2   3   4

   (13)  5   6   7   8   9

        10  11  12  13  14
                                   Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexmatrixEmpty13}
\begin{paste}{MenuexmatrixEmpty13}{MenuexmatrixPatch13}
\pastebutton{MenuexmatrixEmpty13}{\showpaste}
\tab{5}\spadcommand{b:=matrix([[0,1,2,3,4],[5,6,7,8,9],[10,11,12,13,14]])\bound{b }}
\end{paste}\end{patch}

\begin{patch}{MenuexmatrixPatch14}
\begin{paste}{MenuexmatrixFull14}{MenuexmatrixEmpty14}
\pastebutton{MenuexmatrixFull14}{\hidepaste}
\tab{5}\spadcommand{setsubMatrix!(b,1,1,transpose(subMatrix(b,1,3,1,3)))\free{b }}
\indentrel{3}\begin{verbatim}
        0  5  10  3   4

   (14)  1  6  11  8   9

        2  7  12  13  14
                                   Type: Matrix Integer
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexmatrixEmpty14}
\begin{paste}{MenuexmatrixEmpty14}{MenuexmatrixPatch14}
\pastebutton{MenuexmatrixEmpty14}{\showpaste}
\tab{5}\spadcommand{setsubMatrix!(b,1,1,transpose(subMatrix(b,1,3,1,3)))\free{b }}
\end{paste}\end{patch}

\begin{patch}{MenuexmatrixPatch15}
\begin{paste}{MenuexmatrixFull15}{MenuexmatrixEmpty15}
\pastebutton{MenuexmatrixFull15}{\hidepaste}
\tab{5}\spadcommand{trace( matrix([[1,x,x**2,x**3],[1,y,y**2,y**3],[1,z,z**2,z**3],[1,u,u**2,u**
\indentrel{3}\begin{verbatim}
          2       3
   (15)  z  + y + u  + 1
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexmatrixEmpty15}
\begin{paste}{MenuexmatrixEmpty15}{MenuexmatrixPatch15}
\pastebutton{MenuexmatrixEmpty15}{\showpaste}
\tab{5}\spadcommand{trace( matrix([[1,x,x**2,x**3],[1,y,y**2,y**3],[1,z,z**2,z**3],[1,u,u**2,u**
\end{paste}\end{patch}

\begin{patch}{MenuexmatrixPatch16}
\begin{paste}{MenuexmatrixFull16}{MenuexmatrixEmpty16}
\pastebutton{MenuexmatrixFull16}{\hidepaste}
\tab{5}\spadcommand{determinant(matrix([[1,2,3,4],[2,3,2,5],[3,4,5,6],[4,1,6,7]]))}
\indentrel{3}\begin{verbatim}
   (16)  - 48
                                          Type: Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexmatrixEmpty16}
\begin{paste}{MenuexmatrixEmpty16}{MenuexmatrixPatch16}
\pastebutton{MenuexmatrixEmpty16}{\showpaste}
\tab{5}\spadcommand{determinant(matrix([[1,2,3,4],[2,3,2,5],[3,4,5,6],[4,1,6,7]]))}
\end{paste}\end{patch}

\begin{patch}{MenuexmatrixPatch17}
\begin{paste}{MenuexmatrixFull17}{MenuexmatrixEmpty17}
\pastebutton{MenuexmatrixFull17}{\hidepaste}
\tab{5}\spadcommand{inverse(matrix([[1,2,1],[-2,3,4],[-1,5,6]]))}
\indentrel{3}\begin{verbatim}
           2        5
         -   - 1
           7        7
```

```
   (17)   8          6
              1   -
          7          7

        - 1  - 1   1
               Type: Union(Matrix Fraction Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexmatrixEmpty17}
\begin{paste}{MenuexmatrixEmpty17}{MenuexmatrixPatch17}
\pastebutton{MenuexmatrixEmpty17}{\showpaste}
\tab{5}\spadcommand{inverse(matrix([[1,2,1],[-2,3,4],[-1,5,6]]))}
\end{paste}\end{patch}

\begin{patch}{MenuexmatrixPatch18}
\begin{paste}{MenuexmatrixFull18}{MenuexmatrixEmpty18}
\pastebutton{MenuexmatrixFull18}{\hidepaste}
\tab{5}\spadcommand{rank(matrix([[0,4,1],[5,3,-7],[-5,5,9]]))}
\indentrel{3}\begin{verbatim}
   (18)  2
                                   Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexmatrixEmpty18}
\begin{paste}{MenuexmatrixEmpty18}{MenuexmatrixPatch18}
\pastebutton{MenuexmatrixEmpty18}{\showpaste}
\tab{5}\spadcommand{rank(matrix([[0,4,1],[5,3,-7],[-5,5,9]]))}
\end{paste}\end{patch}
```

———-

2-D Graphics

⇒ "Plotting Functions of One Variable" (ExPlot2DFunctions) 3.34 on page 449
⇒ "Plotting Parametric Curves" (ExPlot2DParametric) 3.34 on page 449
⇒ "Plotting Using Polar Coordinates" (ExPlot2DPolar) 3.34 on page 450
⇒ "Plotting Plane Algebraic Curves" (ExPlot2DAlgebraic) 3.34 on page 451

— **coverex.ht** —

```
\begin{page}{Menuexplot2d}{2-D Graphics}
\beginscroll\beginmenu
\menudownlink{Plotting Functions of One Variable}{ExPlot2DFunctions}
\graphpaste{draw(sin(tan(x)) - tan(sin(x)),x = 0..6)}
\menudownlink{Plotting Parametric Curves}{ExPlot2DParametric}
```

```
\graphpaste{draw(curve(9 * sin(3*t/4),8 * sin(t)),t = -4*\%pi..4*\%pi)}
\menudownlink{Plotting Using Polar Coordinates}{ExPlot2DPolar}
\graphpaste{draw(sin(4*t/7),t = 0..14*\%pi,coordinates == polar)}
\menudownlink{Plotting Plane Algebraic Curves}{ExPlot2DAlgebraic}
\graphpaste{draw(y**2 + y - (x**3 - x) = 0, x, y, range == [-2..2,-2..1])}
\endmenu\endscroll
\end{page}

\begin{patch}{Menuexplot2dPatch1}
\begin{paste}{Menuexplot2dFull1}{Menuexplot2dEmpty1}
\pastebutton{Menuexplot2dFull1}{\hidepaste}
\tab{5}\spadgraph{draw(sin(tan(x)) - tan(sin(x)),x = 0..6)}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/menuexplot2d1.view/image}}{viewalone\
\end{paste}\end{patch}

\begin{patch}{Menuexplot2dEmpty1}
\begin{paste}{Menuexplot2dEmpty1}{Menuexplot2dPatch1}
\pastebutton{Menuexplot2dEmpty1}{\showpaste}
\tab{5}\spadgraph{draw(sin(tan(x)) - tan(sin(x)),x = 0..6)}
\end{paste}\end{patch}

\begin{patch}{Menuexplot2dPatch2}
\begin{paste}{Menuexplot2dFull2}{Menuexplot2dEmpty2}
\pastebutton{Menuexplot2dFull2}{\hidepaste}
\tab{5}\spadgraph{draw(curve(9 * sin(3*t/4),8 * sin(t)),t = -4*\%pi..4*\%pi)}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/menuexplot2d2.view/image}}{viewalone\
\end{paste}\end{patch}

\begin{patch}{Menuexplot2dEmpty2}
\begin{paste}{Menuexplot2dEmpty2}{Menuexplot2dPatch2}
\pastebutton{Menuexplot2dEmpty2}{\showpaste}
\tab{5}\spadgraph{draw(curve(9 * sin(3*t/4),8 * sin(t)),t = -4*\%pi..4*\%pi)}
\end{paste}\end{patch}

\begin{patch}{Menuexplot2dPatch3}
\begin{paste}{Menuexplot2dFull3}{Menuexplot2dEmpty3}
\pastebutton{Menuexplot2dFull3}{\hidepaste}
\tab{5}\spadgraph{draw(sin(4*t/7),t = 0..14*\%pi,coordinates == polar)}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/menuexplot2d3.view/image}}{viewalone\
\end{paste}\end{patch}

\begin{patch}{Menuexplot2dEmpty3}
\begin{paste}{Menuexplot2dEmpty3}{Menuexplot2dPatch3}
\pastebutton{Menuexplot2dEmpty3}{\showpaste}
\tab{5}\spadgraph{draw(sin(4*t/7),t = 0..14*\%pi,coordinates == polar)}
\end{paste}\end{patch}

\begin{patch}{Menuexplot2dPatch4}
\begin{paste}{Menuexplot2dFull4}{Menuexplot2dEmpty4}
\pastebutton{Menuexplot2dFull4}{\hidepaste}
```

```
\tab{5}\spadgraph{draw(y**2 + y - (x**3 - x) = 0, x, y, range == [-2..2,-2..1])}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/menuexplot2d4.view/image}}{viewalone\space{1} \en
\end{paste}\end{patch}

\begin{patch}{Menuexplot2dEmpty4}
\begin{paste}{Menuexplot2dEmpty4}{Menuexplot2dPatch4}
\pastebutton{Menuexplot2dEmpty4}{\showpaste}
\tab{5}\spadgraph{draw(y**2 + y - (x**3 - x) = 0, x, y, range == [-2..2,-2..1])}
\end{paste}\end{patch}
```

———-

3-D Graphics

⇒ "Plotting Functions of Two Variables" (ExPlot3DFunctions) 3.35 on page 451
⇒ "Plotting Parametric Surfaces" (ExPlot3DParametricSurface) 3.35 on page 452
⇒ "Plotting Parametric Curves" (ExPlot3DParametricCurve) 3.35 on page 453

— **coverex.ht** —

```
\begin{page}{Menuexplot3d}{3-D Graphics}
\beginscroll\beginmenu
\menudownlink{Plotting Functions of Two Variables}{ExPlot3DFunctions}
\graphpaste{draw(cos(x*y),x = -3..3,y = -3..3)}
\menudownlink{Plotting Parametric Surfaces}{ExPlot3DParametricSurface}
\graphpaste{draw(surface(5*sin(u)*cos(v),4*sin(u)*sin(v),3*cos(u)),
u=0..\%pi,v=0..2*\%pi)}
\graphpaste{draw(surface(u*cos(v),u*sin(v),u),u=0..4,v=0..2*\%pi)}
\menudownlink{Plotting Parametric Curves}{ExPlot3DParametricCurve}
\graphpaste{draw(curve(cos(t),sin(t),t),t=0..6)}
\graphpaste{draw(curve(t,t**2,t**3),t=-3..3)}
\endmenu\endscroll
\end{page}

\begin{patch}{Menuexplot3dPatch1}
\begin{paste}{Menuexplot3dFull1}{Menuexplot3dEmpty1}
\pastebutton{Menuexplot3dFull1}{\hidepaste}
\tab{5}\spadgraph{draw(cos(x*y),x = -3..3,y = -3..3)}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/menuexplot3d1.view/image}}{viewalone\space{1} \en
\end{paste}\end{patch}

\begin{patch}{Menuexplot3dEmpty1}
\begin{paste}{Menuexplot3dEmpty1}{Menuexplot3dPatch1}
\pastebutton{Menuexplot3dEmpty1}{\showpaste}
\tab{5}\spadgraph{draw(cos(x*y),x = -3..3,y = -3..3)}
\end{paste}\end{patch}
```

```
\begin{patch}{Menuexplot3dPatch2}
\begin{paste}{Menuexplot3dFull2}{Menuexplot3dEmpty2}
\pastebutton{Menuexplot3dFull2}{\hidepaste}
\tab{5}\spadgraph{draw(surface(5*sin(u)*cos(v),4*sin(u)*sin(v),3*cos(u)),u=0..\%pi,v=0..2*\%pi)}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/menuexplot3d2.view/image}}{viewalone\
\end{paste}\end{patch}

\begin{patch}{Menuexplot3dEmpty2}
\begin{paste}{Menuexplot3dEmpty2}{Menuexplot3dPatch2}
\pastebutton{Menuexplot3dEmpty2}{\showpaste}
\tab{5}\spadgraph{draw(surface(5*sin(u)*cos(v),4*sin(u)*sin(v),3*cos(u)),u=0..\%pi,v=0..2*\%pi)}
\end{paste}\end{patch}

\begin{patch}{Menuexplot3dPatch3}
\begin{paste}{Menuexplot3dFull3}{Menuexplot3dEmpty3}
\pastebutton{Menuexplot3dFull3}{\hidepaste}
\tab{5}\spadgraph{draw(surface(u*cos(v),u*sin(v),u),u=0..4,v=0..2*\%pi)}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/menuexplot3d3.view/image}}{viewalone\
\end{paste}\end{patch}

\begin{patch}{Menuexplot3dEmpty3}
\begin{paste}{Menuexplot3dEmpty3}{Menuexplot3dPatch3}
\pastebutton{Menuexplot3dEmpty3}{\showpaste}
\tab{5}\spadgraph{draw(surface(u*cos(v),u*sin(v),u),u=0..4,v=0..2*\%pi)}
\end{paste}\end{patch}

\begin{patch}{Menuexplot3dPatch4}
\begin{paste}{Menuexplot3dFull4}{Menuexplot3dEmpty4}
\pastebutton{Menuexplot3dFull4}{\hidepaste}
\tab{5}\spadgraph{draw(curve(cos(t),sin(t),t),t=0..6)}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/menuexplot3d4.view/image}}{viewalone\
\end{paste}\end{patch}

\begin{patch}{Menuexplot3dEmpty4}
\begin{paste}{Menuexplot3dEmpty4}{Menuexplot3dPatch4}
\pastebutton{Menuexplot3dEmpty4}{\showpaste}
\tab{5}\spadgraph{draw(curve(cos(t),sin(t),t),t=0..6)}
\end{paste}\end{patch}

\begin{patch}{Menuexplot3dPatch5}
\begin{paste}{Menuexplot3dFull5}{Menuexplot3dEmpty5}
\pastebutton{Menuexplot3dFull5}{\hidepaste}
\tab{5}\spadgraph{draw(curve(t,t**2,t**3),t=-3..3)}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/menuexplot3d5.view/image}}{viewalone\
\end{paste}\end{patch}

\begin{patch}{Menuexplot3dEmpty5}
\begin{paste}{Menuexplot3dEmpty5}{Menuexplot3dPatch5}
\pastebutton{Menuexplot3dEmpty5}{\showpaste}
\tab{5}\spadgraph{draw(curve(t,t**2,t**3),t=-3..3)}
```

```
\end{paste}\end{patch}
```

———-

Series

⇒ "Converting Expressions to Series" (ExSeriesConvert) 3.37 on page 457
⇒ "Manipulating Power Series" (ExSeriesManipulate) 3.37 on page 459
⇒ "Functions on Power Series" (ExSeriesFunctions) 3.37 on page 461
⇒ "Substituting Numerical Values in Power Series" (ExSeriesSubstitution) 3.37 on page 462

— **coverex.ht** —

```
\begin{page}{Menuexseries}{Series}
\beginscroll\beginmenu
\menudownlink{Converting Expressions to Series}{ExSeriesConvert}
\spadpaste{series(sin(a*x),x = 0)}
\spadpaste{series(sin(a*x),a = \%pi/4)}
\menudownlink{Manipulating Power Series}{ExSeriesManipulate}
\spadpaste{f := series(1/(1-x),x = 0) \bound{f}}
\spadpaste{f ** 2 \free{f}}
\menudownlink{Functions on Power Series}{ExSeriesFunctions}
\spadpaste{f := series(1/(1-x),x = 0) \bound{f1}}
\spadpaste{g := log(f) \free{f1} \bound{g}}
\spadpaste{exp(g) \free{g}}
\menudownlink{Substituting Numerical Values in Power Series}
{ExSeriesSubstitution}
\spadpaste{f := taylor(exp(x)) \bound{f2}}
\spadpaste{eval(f,1.0) \free{f2}}
\endmenu\endscroll
\end{page}

\begin{patch}{MenuexseriesPatch1}
\begin{paste}{MenuexseriesFull1}{MenuexseriesEmpty1}
\pastebutton{MenuexseriesFull1}{\hidepaste}
\tab{5}\spadcommand{series(sin(a*x),x = 0)}
\indentrel{3}\begin{verbatim}
   (1)
            3         5         7           9
           a   3     a   5     a    7      a     9
     a x - x  + x  - x  + x
            6       120       5040       362880
   +
           11
          a       11       12
     - x    + O(x   )
       39916800
```

```
  Type: UnivariatePuiseuxSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexseriesEmpty1}
\begin{paste}{MenuexseriesEmpty1}{MenuexseriesPatch1}
\pastebutton{MenuexseriesEmpty1}{\showpaste}
\tab{5}\spadcommand{series(sin(a*x),x = 0)}
\end{paste}\end{patch}

\begin{patch}{MenuexseriesPatch2}
\begin{paste}{MenuexseriesFull2}{MenuexseriesEmpty2}
\pastebutton{MenuexseriesFull2}{\hidepaste}
\tab{5}\spadcommand{series(sin(a*x),a = \%pi/4)}
\indentrel{3}\begin{verbatim}
   (2)
         %pi x          %pi x      %pi
     sin() + x cos()(a - )
           4              4         4
   +
        2    %pi x                 3    %pi x
      x sin()               x cos()
               4         %pi 2           4         %pi 3
     -  (a - )  -  (a - )
             2            4            6            4
   +
      4    %pi x                 5    %pi x
     x sin()               x cos()
             4         %pi 4           4         %pi 5
      (a - )  +  (a - )
          24            4           120           4
   +
        6    %pi x                 7    %pi x
      x sin()               x cos()
               4         %pi 6           4         %pi 7
     -  (a - )  -  (a - )
            720           4          5040           4
   +
      8    %pi x                 9    %pi x
     x sin()               x cos()
             4         %pi 8           4         %pi 9
      (a - )  +  (a - )
         40320          4        362880          4
   +
        10    %pi x
      x  sin()
                4         %pi 10          %pi 11
     -  (a - )   + O((a - )  )
          3628800         4               4
Type: UnivariatePuiseuxSeries(Expression Integer,a,pi/4)
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexseriesEmpty2}
\begin{paste}{MenuexseriesEmpty2}{MenuexseriesPatch2}
\pastebutton{MenuexseriesEmpty2}{\showpaste}
\tab{5}\spadcommand{series(sin(a*x),a = \%pi/4)}
\end{paste}\end{patch}

\begin{patch}{MenuexseriesPatch3}
\begin{paste}{MenuexseriesFull3}{MenuexseriesEmpty3}
\pastebutton{MenuexseriesFull3}{\hidepaste}
\tab{5}\spadcommand{f := series(1/(1-x),x = 0)\bound{f }}
\indentrel{3}\begin{verbatim}
   (3)
             2    3    4    5    6    7    8    9    10
     1 + x + x  + x  + x  + x  + x  + x  + x  + x  + x
   +
        11
     O(x  )
  Type: UnivariatePuiseuxSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexseriesEmpty3}
\begin{paste}{MenuexseriesEmpty3}{MenuexseriesPatch3}
\pastebutton{MenuexseriesEmpty3}{\showpaste}
\tab{5}\spadcommand{f := series(1/(1-x),x = 0)\bound{f }}
\end{paste}\end{patch}

\begin{patch}{MenuexseriesPatch4}
\begin{paste}{MenuexseriesFull4}{MenuexseriesEmpty4}
\pastebutton{MenuexseriesFull4}{\hidepaste}
\tab{5}\spadcommand{f ** 2\free{f }}
\indentrel{3}\begin{verbatim}
   (4)
               2     3     4     5     6     7     8
     1 + 2x + 3x  + 4x  + 5x  + 6x  + 7x  + 8x  + 9x
   +
        9      10      11
     10x  + 11x   + O(x  )
  Type: UnivariatePuiseuxSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexseriesEmpty4}
\begin{paste}{MenuexseriesEmpty4}{MenuexseriesPatch4}
\pastebutton{MenuexseriesEmpty4}{\showpaste}
\tab{5}\spadcommand{f ** 2\free{f }}
\end{paste}\end{patch}
```

```
\begin{patch}{MenuexseriesPatch5}
\begin{paste}{MenuexseriesFull5}{MenuexseriesEmpty5}
\pastebutton{MenuexseriesFull5}{\hidepaste}
\tab{5}\spadcommand{f := series(1/(1-x),x = 0)\bound{f1 }}
\indentrel{3}\begin{verbatim}
   (5)
              2    3    4    5    6    7    8    9    10
     1 + x + x  + x  + x  + x  + x  + x  + x  + x  + x
   +
        11
     O(x  )
  Type: UnivariatePuiseuxSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexseriesEmpty5}
\begin{paste}{MenuexseriesEmpty5}{MenuexseriesPatch5}
\pastebutton{MenuexseriesEmpty5}{\showpaste}
\tab{5}\spadcommand{f := series(1/(1-x),x = 0)\bound{f1 }}
\end{paste}\end{patch}

\begin{patch}{MenuexseriesPatch6}
\begin{paste}{MenuexseriesFull6}{MenuexseriesEmpty6}
\pastebutton{MenuexseriesFull6}{\hidepaste}
\tab{5}\spadcommand{g := log(f)\free{f1 }\bound{g }}
\indentrel{3}\begin{verbatim}
   (6)
        1  2   1  3   1  4   1  5   1  6   1  7   1  8
     x +  x  +  x  +  x  +  x  +  x  +  x  +  x
        2      3      4      5      6      7      8
   +
     1  9    1  10    1  11      12
      x  +  x   +  x   + O(x  )
     9      10       11
  Type: UnivariatePuiseuxSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexseriesEmpty6}
\begin{paste}{MenuexseriesEmpty6}{MenuexseriesPatch6}
\pastebutton{MenuexseriesEmpty6}{\showpaste}
\tab{5}\spadcommand{g := log(f)\free{f1 }\bound{g }}
\end{paste}\end{patch}

\begin{patch}{MenuexseriesPatch7}
\begin{paste}{MenuexseriesFull7}{MenuexseriesEmpty7}
\pastebutton{MenuexseriesFull7}{\hidepaste}
\tab{5}\spadcommand{exp(g)\free{g }}
\indentrel{3}\begin{verbatim}
```

```
   (7)
             2    3    4    5    6    7    8    9    10
     1 + x + x  + x  + x  + x  + x  + x  + x  + x  + x
   +
        11
     O(x  )
  Type: UnivariatePuiseuxSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexseriesEmpty7}
\begin{paste}{MenuexseriesEmpty7}{MenuexseriesPatch7}
\pastebutton{MenuexseriesEmpty7}{\showpaste}
\tab{5}\spadcommand{exp(g)\free{g }}
\end{paste}\end{patch}

\begin{patch}{MenuexseriesPatch8}
\begin{paste}{MenuexseriesFull8}{MenuexseriesEmpty8}
\pastebutton{MenuexseriesFull8}{\hidepaste}
\tab{5}\spadcommand{f := taylor(exp(x))\bound{f2 }}
\indentrel{3}\begin{verbatim}
   (8)
            1  2   1  3    1  4    1   5    1   6
     1 + x +  x  +  x  +  x  +  x  +  x
            2      6      24      120      720
   +
     1   7     1    8      1    9      1     10      11
    x  +  x  +  x  +  x   + O(x  )
   5040      40320      362880      3628800
   Type: UnivariateTaylorSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexseriesEmpty8}
\begin{paste}{MenuexseriesEmpty8}{MenuexseriesPatch8}
\pastebutton{MenuexseriesEmpty8}{\showpaste}
\tab{5}\spadcommand{f := taylor(exp(x))\bound{f2 }}
\end{paste}\end{patch}

\begin{patch}{MenuexseriesPatch9}
\begin{paste}{MenuexseriesFull9}{MenuexseriesEmpty9}
\pastebutton{MenuexseriesFull9}{\hidepaste}
\tab{5}\spadcommand{eval(f,1.0)\free{f2 }}
\indentrel{3}\begin{verbatim}
   (9)
   [1.0, 2.0, 2.5, 2.6666666666 666666667,
    2.7083333333 333333333, 2.7166666666 666666667,
    2.7180555555 555555556, 2.7182539682 53968254,
    2.7182787698 412698413, 2.7182815255 731922399, ...]
                         Type: Stream Expression Float
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexseriesEmpty9}
\begin{paste}{MenuexseriesEmpty9}{MenuexseriesPatch9}
\pastebutton{MenuexseriesEmpty9}{\showpaste}
\tab{5}\spadcommand{eval(f,1.0)\free{f2 }}
\end{paste}\end{patch}
```

———-

Summations

⇒ “notitle” (ExSumListEntriesI) 3.38 on page 464
⇒ “notitle” (ExSumListEntriesII) 3.38 on page 465
⇒ “notitle” (ExSumApproximateE) 3.38 on page 466
⇒ “notitle” (ExSumClosedForm) 3.38 on page 467
⇒ “notitle” (ExSumCubes) 3.38 on page 468
⇒ “notitle” (ExSumPolynomial) 3.38 on page 470
⇒ “notitle” (ExSumGeneralFunction) 3.38 on page 471
⇒ “notitle” (ExSumInfinite) 3.38 on page 472

— **coverex.ht** —

```
\begin{page}{Menuexsum}{Summations}
\beginscroll\beginmenu
\menudownlink{Summing the Entries of a List I}{ExSumListEntriesI}
\spadpaste{[i for i in 1..15]}
\spadpaste{reduce(+,[i for i in 1..15])}
\menudownlink{Summing the Entries of a List II}{ExSumListEntriesII}
\spadpaste{[n**2 for n in 5..20]}
\spadpaste{reduce(+,[n**2 for n in 5..20])}
\menudownlink{Approximating e}{ExSumApproximateE}
\spadpaste{reduce(+,[1.0/factorial(n) for n in 0..20])}
\menudownlink{Closed Form Summations}{ExSumClosedForm}
\spadpaste{s := sum(k**2,k = a..b) \bound{s}}
\spadpaste{eval(s,[a,b],[1,25]) \free{s}}
\spadpaste{reduce(+,[i**2 for i in 1..25])}
\menudownlink{Sums of Cubes}{ExSumCubes}
\spadpaste{sum(k**3,k = 1..n)}
\spadpaste{sum(k,k = 1..n) ** 2}
\menudownlink{Sums of Polynomials}{ExSumPolynomial}
\spadpaste{sum(3*k**2/(c**2 + 1) + 12*k/d,k = (3*a)..(4*b))}
\menudownlink{Sums of General Functions}{ExSumGeneralFunction}
\spadpaste{sum(k * x**k,k = 1..n)}
\menudownlink{Infinite Sums}{ExSumInfinite}
\spadpaste{limit( sum(1/(k * (k + 2)),k = 1..n) ,n = \%plusInfinity)}
```

```
\endmenu\endscroll
\end{page}

\begin{patch}{MenuexsumPatch1}
\begin{paste}{MenuexsumFull1}{MenuexsumEmpty1}
\pastebutton{MenuexsumFull1}{\hidepaste}
\tab{5}\spadcommand{[i for i in 1..15]}
\indentrel{3}\begin{verbatim}
   (1)  [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
                             Type: List PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexsumEmpty1}
\begin{paste}{MenuexsumEmpty1}{MenuexsumPatch1}
\pastebutton{MenuexsumEmpty1}{\showpaste}
\tab{5}\spadcommand{[i for i in 1..15]}
\end{paste}\end{patch}

\begin{patch}{MenuexsumPatch2}
\begin{paste}{MenuexsumFull2}{MenuexsumEmpty2}
\pastebutton{MenuexsumFull2}{\hidepaste}
\tab{5}\spadcommand{reduce(+,[i for i in 1..15])}
\indentrel{3}\begin{verbatim}
   (2)  120
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexsumEmpty2}
\begin{paste}{MenuexsumEmpty2}{MenuexsumPatch2}
\pastebutton{MenuexsumEmpty2}{\showpaste}
\tab{5}\spadcommand{reduce(+,[i for i in 1..15])}
\end{paste}\end{patch}

\begin{patch}{MenuexsumPatch3}
\begin{paste}{MenuexsumFull3}{MenuexsumEmpty3}
\pastebutton{MenuexsumFull3}{\hidepaste}
\tab{5}\spadcommand{[n**2 for n in 5..20]}
\indentrel{3}\begin{verbatim}
   (3)
   [25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225,
    256, 289, 324, 361, 400]
                             Type: List PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexsumEmpty3}
\begin{paste}{MenuexsumEmpty3}{MenuexsumPatch3}
\pastebutton{MenuexsumEmpty3}{\showpaste}
```

```
\tab{5}\spadcommand{[n**2 for n in 5..20]}
\end{paste}\end{patch}

\begin{patch}{MenuexsumPatch4}
\begin{paste}{MenuexsumFull4}{MenuexsumEmpty4}
\pastebutton{MenuexsumFull4}{\hidepaste}
\tab{5}\spadcommand{reduce(+,[n**2 for n in 5..20])}
\indentrel{3}\begin{verbatim}
   (4)  2840
                                 Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexsumEmpty4}
\begin{paste}{MenuexsumEmpty4}{MenuexsumPatch4}
\pastebutton{MenuexsumEmpty4}{\showpaste}
\tab{5}\spadcommand{reduce(+,[n**2 for n in 5..20])}
\end{paste}\end{patch}

\begin{patch}{MenuexsumPatch5}
\begin{paste}{MenuexsumFull5}{MenuexsumEmpty5}
\pastebutton{MenuexsumFull5}{\hidepaste}
\tab{5}\spadcommand{reduce(+,[1.0/factorial(n) for n in 0..20])}
\indentrel{3}\begin{verbatim}
   (5)  2.7182818284 590452354
                                           Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexsumEmpty5}
\begin{paste}{MenuexsumEmpty5}{MenuexsumPatch5}
\pastebutton{MenuexsumEmpty5}{\showpaste}
\tab{5}\spadcommand{reduce(+,[1.0/factorial(n) for n in 0..20])}
\end{paste}\end{patch}

\begin{patch}{MenuexsumPatch6}
\begin{paste}{MenuexsumFull6}{MenuexsumEmpty6}
\pastebutton{MenuexsumFull6}{\hidepaste}
\tab{5}\spadcommand{s := sum(k**2,k = a..b)\bound{s }}
\indentrel{3}\begin{verbatim}
          3     2         3     2
        2b  + 3b  + b - 2a  + 3a  - a
   (6)
                      6
                      Type: Fraction Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexsumEmpty6}
\begin{paste}{MenuexsumEmpty6}{MenuexsumPatch6}
```

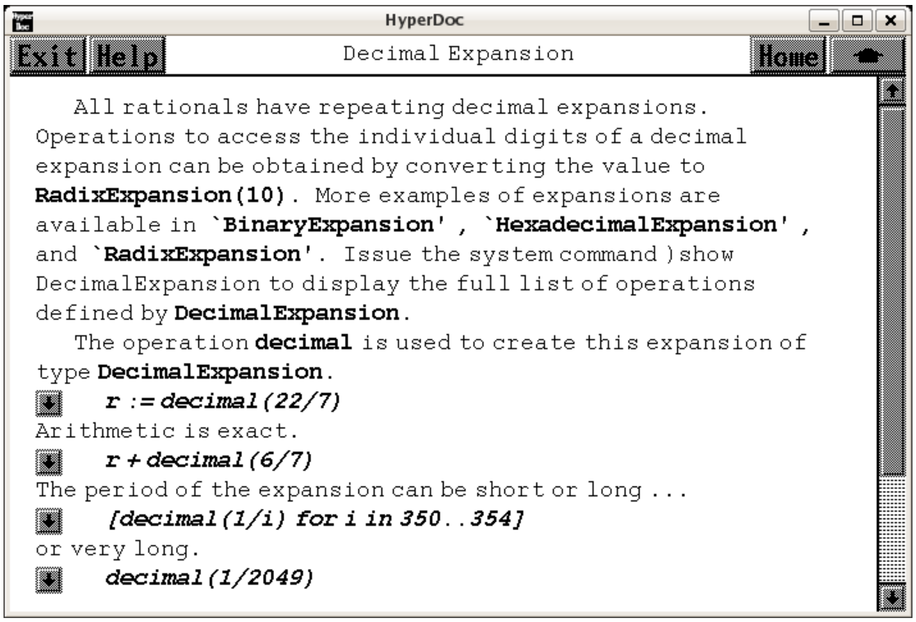
```
\pastebutton{MenuexsumEmpty6}{\showpaste}
\tab{5}\spadcommand{s := sum(k**2,k = a..b)\bound{s }}
\end{paste}\end{patch}

\begin{patch}{MenuexsumPatch7}
\begin{paste}{MenuexsumFull7}{MenuexsumEmpty7}
\pastebutton{MenuexsumFull7}{\hidepaste}
\tab{5}\spadcommand{eval(s,[a,b],[1,25])\free{s }}
\indentrel{3}\begin{verbatim}
   (7)  5525
                      Type: Fraction Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexsumEmpty7}
\begin{paste}{MenuexsumEmpty7}{MenuexsumPatch7}
\pastebutton{MenuexsumEmpty7}{\showpaste}
\tab{5}\spadcommand{eval(s,[a,b],[1,25])\free{s }}
\end{paste}\end{patch}

\begin{patch}{MenuexsumPatch8}
\begin{paste}{MenuexsumFull8}{MenuexsumEmpty8}
\pastebutton{MenuexsumFull8}{\hidepaste}
\tab{5}\spadcommand{reduce(+,[i**2 for i in 1..25])}
\indentrel{3}\begin{verbatim}
   (8)  5525
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexsumEmpty8}
\begin{paste}{MenuexsumEmpty8}{MenuexsumPatch8}
\pastebutton{MenuexsumEmpty8}{\showpaste}
\tab{5}\spadcommand{reduce(+,[i**2 for i in 1..25])}
\end{paste}\end{patch}

\begin{patch}{MenuexsumPatch9}
\begin{paste}{MenuexsumFull9}{MenuexsumEmpty9}
\pastebutton{MenuexsumFull9}{\hidepaste}
\tab{5}\spadcommand{sum(k**3,k = 1..n)}
\indentrel{3}\begin{verbatim}
         4     3    2
        n  + 2n  + n
   (9)  ------------
              4
                      Type: Fraction Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexsumEmpty9}
```

```
\begin{paste}{MenuexsumEmpty9}{MenuexsumPatch9}
\pastebutton{MenuexsumEmpty9}{\showpaste}
\tab{5}\spadcommand{sum(k**3,k = 1..n)}
\end{paste}\end{patch}

\begin{patch}{MenuexsumPatch10}
\begin{paste}{MenuexsumFull10}{MenuexsumEmpty10}
\pastebutton{MenuexsumFull10}{\hidepaste}
\tab{5}\spadcommand{sum(k,k = 1..n) ** 2}
\indentrel{3}\begin{verbatim}
          4     3    2
         n  + 2n  + n
   (10)
              4
                     Type: Fraction Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexsumEmpty10}
\begin{paste}{MenuexsumEmpty10}{MenuexsumPatch10}
\pastebutton{MenuexsumEmpty10}{\showpaste}
\tab{5}\spadcommand{sum(k,k = 1..n) ** 2}
\end{paste}\end{patch}

\begin{patch}{MenuexsumPatch11}
\begin{paste}{MenuexsumFull11}{MenuexsumEmpty11}
\pastebutton{MenuexsumFull11}{\hidepaste}
\tab{5}\spadcommand{sum(3*k**2/(c**2 + 1) + 12*k/d,k = (3*a)..(4*b))}
\indentrel{3}\begin{verbatim}
   (11)
            3      2           3      2
       (128b  + 48b  + 4b - 54a  + 27a  - 3a)d
     +
            2             2        2       2             2
       (192b  + 48b - 108a  + 36a)c  + 192b  + 48b - 108a
     +
       36a
  /
        2
     (2c  + 2)d
           Type: Union(Fraction Polynomial Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexsumEmpty11}
\begin{paste}{MenuexsumEmpty11}{MenuexsumPatch11}
\pastebutton{MenuexsumEmpty11}{\showpaste}
\tab{5}\spadcommand{sum(3*k**2/(c**2 + 1) + 12*k/d,k = (3*a)..(4*b))}
\end{paste}\end{patch}
```

```
\begin{patch}{MenuexsumPatch12}
\begin{paste}{MenuexsumFull12}{MenuexsumEmpty12}
\pastebutton{MenuexsumFull12}{\hidepaste}
\tab{5}\spadcommand{sum(k * x**k,k = 1..n)}
\indentrel{3}\begin{verbatim}
             2                n
         (n x  + (- n - 1)x)x  + x
   (12)
                 2
                x  - 2x + 1
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexsumEmpty12}
\begin{paste}{MenuexsumEmpty12}{MenuexsumPatch12}
\pastebutton{MenuexsumEmpty12}{\showpaste}
\tab{5}\spadcommand{sum(k * x**k,k = 1..n)}
\end{paste}\end{patch}

\begin{patch}{MenuexsumPatch13}
\begin{paste}{MenuexsumFull13}{MenuexsumEmpty13}
\pastebutton{MenuexsumFull13}{\hidepaste}
\tab{5}\spadcommand{limit( sum(1/(k * (k + 2)),k = 1..n) ,n = \%plusInfinity)}
\indentrel{3}\begin{verbatim}
         3
   (13)
         4
Type: Union(OrderedCompletion Fraction Polynomial Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MenuexsumEmpty13}
\begin{paste}{MenuexsumEmpty13}{MenuexsumPatch13}
\pastebutton{MenuexsumEmpty13}{\showpaste}
\tab{5}\spadcommand{limit( sum(1/(k * (k + 2)),k = 1..n) ,n = \%plusInfinity)}
\end{paste}\end{patch}
```

3.21 decimal.ht

Decimal Expansion



⇐ "Axiom Number Types" (NumberPage) 3.80 on page 1021
⇐ "Integer" (IntegerXmpPage) 3.55 on page 696
⇒ "BinaryExpansion" (BinaryExpansionXmpPage) 3.8 on page 163
⇒ "HexadecimalExpansion" (HexExpansionXmpPage) 3.54 on page 692
⇒ "RadixExpansion" (RadixExpansionXmpPage) 3.90 on page 1140

— decimal.ht —

```
\begin{page}{DecimalExpansionXmpPage}{Decimal Expansion}
\beginscroll
All rationals have repeating decimal expansions.
Operations to access the individual digits of a decimal expansion can
be obtained by converting the value to \spadtype{RadixExpansion(10)}.
More examples of expansions are available in
\downlink{`BinaryExpansion'}{BinaryExpansionXmpPage}
\ignore{BinaryExpansion},
\downlink{`HexadecimalExpansion'}{HexExpansionXmpPage}
\ignore{HexadecimalExpansion}, and
\downlink{`RadixExpansion'}{RadixExpansionXmpPage}\ignore{RadixExpansion}.
\showBlurb{DecimalExpansion}

\xtc{
The operation \spadfunFrom{decimal}{DecimalExpansion} is used to create
```

```
this expansion of type \spadtype{DecimalExpansion}.
}{
\spadpaste{r := decimal(22/7) \bound{r}}
}
\xtc{
Arithmetic is exact.
}{
\spadpaste{r + decimal(6/7) \free{r}}
}
\xtc{
The period of the expansion can be short or long \ldots
}{
\spadpaste{[decimal(1/i) for i in 350..354] }
}
\xtc{
or very long.
}{
\spadpaste{decimal(1/2049) }
}
\xtc{
These numbers are bona fide algebraic objects.
}{
\spadpaste{p := decimal(1/4)*x**2 + decimal(2/3)*x + decimal(4/9)
\bound{p}}
}
\xtc{
}{
\spadpaste{q := differentiate(p, x) \free{p}\bound{q}}
}
\xtc{
}{
\spadpaste{g := gcd(p, q)            \free{p q} \bound{g}}
}
\endscroll
\autobuttons
\end{page}

\begin{patch}{DecimalExpansionXmpPagePatch1}
\begin{paste}{DecimalExpansionXmpPageFull1}{DecimalExpansionXmpPageEmpty1}
\pastebutton{DecimalExpansionXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{r := decimal(22/7)\bound{r }}
\indentrel{3}\begin{verbatim}
          ------
   (1)  3.142857
                                 Type: DecimalExpansion
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DecimalExpansionXmpPageEmpty1}
\begin{paste}{DecimalExpansionXmpPageEmpty1}{DecimalExpansionXmpPagePatch1}
```

```
\pastebutton{DecimalExpansionXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{r := decimal(22/7)\bound{r }}
\end{paste}\end{patch}

\begin{patch}{DecimalExpansionXmpPagePatch2}
\begin{paste}{DecimalExpansionXmpPageFull2}{DecimalExpansionXmpPageEmpty2}
\pastebutton{DecimalExpansionXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{r + decimal(6/7)\free{r }}
\indentrel{3}\begin{verbatim}
   (2)  4
                                 Type: DecimalExpansion
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DecimalExpansionXmpPageEmpty2}
\begin{paste}{DecimalExpansionXmpPageEmpty2}{DecimalExpansionXmpPagePatch2}
\pastebutton{DecimalExpansionXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{r + decimal(6/7)\free{r }}
\end{paste}\end{patch}

\begin{patch}{DecimalExpansionXmpPagePatch3}
\begin{paste}{DecimalExpansionXmpPageFull3}{DecimalExpansionXmpPageEmpty3}
\pastebutton{DecimalExpansionXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{[decimal(1/i) for i in 350..354]}
\indentrel{3}\begin{verbatim}
   (3)

        ------    ------         --
   [0.00285714, 0.002849, 0.0028409,

      --------------------------------
    0.00283286118980169971671388101983,

     0.0
       OVERBAR
          028248587570621468926553672316384180790960451977
            4011299435
     ]
                            Type: List DecimalExpansion
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DecimalExpansionXmpPageEmpty3}
\begin{paste}{DecimalExpansionXmpPageEmpty3}{DecimalExpansionXmpPagePatch3}
\pastebutton{DecimalExpansionXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{[decimal(1/i) for i in 350..354]}
\end{paste}\end{patch}

\begin{patch}{DecimalExpansionXmpPagePatch4}
\begin{paste}{DecimalExpansionXmpPageFull4}{DecimalExpansionXmpPageEmpty4}
\pastebutton{DecimalExpansionXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{decimal(1/2049)}
```

```
\indentrel{3}\begin{verbatim}
   (4)
   0.
     OVERBAR
        00048804294777940458760370912640312347486578818936
          6066373840897999023914104441190824792581747193
          5305026842362127867252318204001952171791117618
          5041483650561249389946315275744265495363591996
          9565641776476329917032698877501220107369448511
          6900927281600780868716447047340165934602244997
          5978526110297706198145436798438262567105905319
          6813079551
                                  Type: DecimalExpansion
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DecimalExpansionXmpPageEmpty4}
\begin{paste}{DecimalExpansionXmpPageEmpty4}{DecimalExpansionXmpPagePatch4}
\pastebutton{DecimalExpansionXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{decimal(1/2049)}
\end{paste}\end{patch}

\begin{patch}{DecimalExpansionXmpPagePatch5}
\begin{paste}{DecimalExpansionXmpPageFull5}{DecimalExpansionXmpPageEmpty5}
\pastebutton{DecimalExpansionXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{p := decimal(1/4)*x**2 + decimal(2/3)*x + decimal(4/9)\bound{p }}
\indentrel{3}\begin{verbatim}
             2     _     _
   (5)  0.25x  + 0.6x + 0.4
                      Type: Polynomial DecimalExpansion
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DecimalExpansionXmpPageEmpty5}
\begin{paste}{DecimalExpansionXmpPageEmpty5}{DecimalExpansionXmpPagePatch5}
\pastebutton{DecimalExpansionXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{p := decimal(1/4)*x**2 + decimal(2/3)*x + decimal(4/9)\bound{p }}
\end{paste}\end{patch}

\begin{patch}{DecimalExpansionXmpPagePatch6}
\begin{paste}{DecimalExpansionXmpPageFull6}{DecimalExpansionXmpPageEmpty6}
\pastebutton{DecimalExpansionXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{q := differentiate(p, x)\free{p }\bound{q }}
\indentrel{3}\begin{verbatim}
                 _
   (6)  0.5x + 0.6
                      Type: Polynomial DecimalExpansion
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{DecimalExpansionXmpPageEmpty6}
\begin{paste}{DecimalExpansionXmpPageEmpty6}{DecimalExpansionXmpPagePatch6}
\pastebutton{DecimalExpansionXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{q := differentiate(p, x)\free{p }\bound{q }}
\end{paste}\end{patch}

\begin{patch}{DecimalExpansionXmpPagePatch7}
\begin{paste}{DecimalExpansionXmpPageFull7}{DecimalExpansionXmpPageEmpty7}
\pastebutton{DecimalExpansionXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{g := gcd(p, q)\free{p q }\bound{g }}
\indentrel{3}\begin{verbatim}
              -
   (7)  x + 1.3
                      Type: Polynomial DecimalExpansion
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DecimalExpansionXmpPageEmpty7}
\begin{paste}{DecimalExpansionXmpPageEmpty7}{DecimalExpansionXmpPagePatch7}
\pastebutton{DecimalExpansionXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{g := gcd(p, q)\free{p q }\bound{g }}
\end{paste}\end{patch}
```

3.22 derham.ht

DeRhamComplex

— derham.ht —

```
\begin{page}{DeRhamComplexXmpPage}{DeRhamComplex}
\beginscroll

The domain constructor \spadtype{DeRhamComplex} creates the
class of differential forms of arbitrary degree over a coefficient ring.
The De Rham complex constructor takes two arguments: a ring,
\spad{coefRing,} and a list of coordinate variables.

\xtc{
This is the ring of coefficients.
}{
\spadpaste{coefRing := Integer \bound{coefRing}}
}
\xtc{
These are the coordinate variables.
```

```
}{
\spadpaste{lv : List Symbol := [x,y,z] \bound{lv}}
}
\xtc{
This is the De Rham complex of Euclidean three-space using
coordinates \spad{x, y} and \spad{z.}
}{
\spadpaste{der := DERHAM(coefRing,lv) \free{coefRing}\free{lv}
\bound{der}}
}

This complex allows us to describe differential forms having
expressions of integers as coefficients.
These coefficients can involve any number of variables, for example,
\spad{f(x,t,r,y,u,z).}
As we've chosen to work with ordinary
Euclidean three-space, expressions involving these forms
are treated as functions of
\spad{x, y} and \spad{z} with the additional arguments
\spad{t, r} and \spad{u} regarded as symbolic constants.
\xtc{
Here are some examples of coefficients.
}{
\spadpaste{R := Expression coefRing \free{coefRing}\bound{R}}
}
\xtc{
}{
\spadpaste{f : R := x**2*y*z-5*x**3*y**2*z**5 \free{R}\bound{f}}
}
\xtc{
}{
\spadpaste{g : R := z**2*y*cos(z)-7*sin(x**3*y**2)*z**2 \free{R}\bound{g}}
}
\xtc{
}{
\spadpaste{h : R :=x*y*z-2*x**3*y*z**2 \free{R}\bound{h}}
}
\xtc{
We now define
the multiplicative basis elements for the exterior algebra over \spad{R}.
}{
\spadpaste{dx : der := generator(1) \free{der}\bound{dx}}
}
\xtc{
}{
\spadpaste{dy : der := generator(2)\free{der}\bound{dy}}
}
\xtc{
}{
\spadpaste{dz : der := generator(3)\free{der}\bound{dz}}
```

```
}
\xtc{
This is an alternative way to give the above assignments.
}{
\spadpaste{[dx,dy,dz] := [generator(i)\$der for i in 1..3] \free{der}
\bound{dxyz}}
}
\xtc{
Now we define some one-forms.
}{
\spadpaste{alpha : der := f*dx + g*dy + h*dz \bound{alpha}
\free{der f g h dxyz}}
}
\xtc{
}{
\spadpaste{beta  : der := cos(tan(x*y*z)+x*y*z)*dx + x*dy
\bound{beta}\free{der f g h dxyz}}
}
\xtc{
A well-known theorem states that the composition of
\spadfunFrom{exteriorDifferential}{DeRhamComplex}
with itself is the zero map for continuous forms.
Let's verify this theorem for \spad{alpha}.
}{
\spadpaste{exteriorDifferential alpha; \free{alpha}\bound{ed}}
}
\xtc{
We suppressed the lengthy output of the last expression, but nevertheless, the
composition is zero.
}{
\spadpaste{exteriorDifferential \% \free{ed}}
}

\xtc{
Now we check that \spadfunFrom{exteriorDifferential}{DeRhamComplex}
is a ``graded derivation'' \spad{D,} that is, \spad{D} satisfies:
\begin{verbatim}
D(a*b) = D(a)*b + (-1)**degree(a)*a*D(b)
\end{verbatim}
}{
\spadpaste{gamma := alpha * beta \bound{gamma}\free{alpha}\free{beta}}
}
\xtc{
We try this for the one-forms \spad{alpha} and \spad{beta}.
}{
\spadpaste{exteriorDifferential(gamma) -
(exteriorDifferential(alpha)*beta - alpha * exteriorDifferential(beta))
\free{alpha beta gamma}}
}
\xtc{
```

```
Now we define some ‘‘basic operators’’ (see
\downlink{‘Operator’}{OperatorXmpPage}\ignore{Operator}).
}{
\spadpaste{a : BOP := operator(’a) \bound{ao}}
}
\xtc{
}{
\spadpaste{b : BOP := operator(’b) \bound{bo}}
}
\xtc{
}{
\spadpaste{c : BOP := operator(’c) \bound{co}}
}
\xtc{
We also define
some indeterminate one- and two-forms using these operators.
}{
\spadpaste{sigma := a(x,y,z) * dx + b(x,y,z) * dy + c(x,y,z) * dz
\bound{sigma}\free{ao bo co dxyz}}
}
\xtc{
}{
\spadpaste{theta  := a(x,y,z) * dx * dy + b(x,y,z) * dx *
dz + c(x,y,z) * dy * dz \bound{theta}\free{ao bo co dxyz}}
}

\xtc{
This allows us to get formal definitions for the ‘‘gradient’’ \ldots
}{
\spadpaste{totalDifferential(a(x,y,z))\$der \free{ao der}}
}
\xtc{
the ‘‘curl’’ \ldots
}{
\spadpaste{exteriorDifferential sigma \free{sigma}}
}
\xtc{
and the ‘‘divergence.’’
}{
\spadpaste{exteriorDifferential theta \free{theta}}
}

\xtc{
Note that the De Rham complex is an algebra with unity.
This element \spad{1} is the basis for elements for zero-forms, that is,
functions in our space.
}{
\spadpaste{one : der := 1 \bound{one}\free{der}}
}
```

```
\xtc{
To convert a function to a function lying in the De Rham complex,
multiply the function by ``one.''
}{
\spadpaste{g1 : der := a([x,t,y,u,v,z,e]) * one \free{der one ao}\bound{g1}}
}
\xtc{
A current limitation of Axiom forces you to write
functions with more than four arguments using square brackets in this way.
}{
\spadpaste{h1 : der := a([x,y,x,t,x,z,y,r,u,x]) * one
\free{der one ao}\bound{h1}}
}

\xtc{
Now note how the system keeps track of where your coordinate functions
are located in expressions.
}{
\spadpaste{exteriorDifferential g1 \free{g1}}
}
\xtc{
}{
\spadpaste{exteriorDifferential h1 \free{h1}}
}

\xtc{
In this example of Euclidean three-space, the basis for the De Rham complex
consists of the eight forms: \spad{1}, \spad{dx}, \spad{dy}, \spad{dz},
\spad{dx*dy}, \spad{dx*dz}, \spad{dy*dz}, and \spad{dx*dy*dz}.
}{
\spadpaste{coefficient(gamma, dx*dy) \free{gamma dxyz}}
}
\xtc{
}{
\spadpaste{coefficient(gamma, one) \free{gamma one}}
}
\xtc{
}{
\spadpaste{coefficient(g1,one) \free{g1 one}}
}
\endscroll
\autobuttons
\end{page}

\begin{patch}{DeRhamComplexXmpPagePatch1}
\begin{paste}{DeRhamComplexXmpPageFull1}{DeRhamComplexXmpPageEmpty1}
\pastebutton{DeRhamComplexXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{coefRing := Integer\bound{coefRing }}
\indentrel{3}\begin{verbatim}
   (1)  Integer
```

```
                                        Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPageEmpty1}
\begin{paste}{DeRhamComplexXmpPageEmpty1}{DeRhamComplexXmpPagePatch1}
\pastebutton{DeRhamComplexXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{coefRing := Integer\bound{coefRing }}
\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPagePatch2}
\begin{paste}{DeRhamComplexXmpPageFull2}{DeRhamComplexXmpPageEmpty2}
\pastebutton{DeRhamComplexXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{lv : List Symbol := [x,y,z]\bound{lv }}
\indentrel{3}\begin{verbatim}
   (2)  [x,y,z]
                                   Type: List Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPageEmpty2}
\begin{paste}{DeRhamComplexXmpPageEmpty2}{DeRhamComplexXmpPagePatch2}
\pastebutton{DeRhamComplexXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{lv : List Symbol := [x,y,z]\bound{lv }}
\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPagePatch3}
\begin{paste}{DeRhamComplexXmpPageFull3}{DeRhamComplexXmpPageEmpty3}
\pastebutton{DeRhamComplexXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{der := DERHAM(coefRing,lv)\free{coefRing }\free{lv }\bound{der }}
\indentrel{3}\begin{verbatim}
   (3)  DeRhamComplex(Integer,[x,y,z])
                                        Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPageEmpty3}
\begin{paste}{DeRhamComplexXmpPageEmpty3}{DeRhamComplexXmpPagePatch3}
\pastebutton{DeRhamComplexXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{der := DERHAM(coefRing,lv)\free{coefRing }\free{lv }\bound{der }}
\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPagePatch4}
\begin{paste}{DeRhamComplexXmpPageFull4}{DeRhamComplexXmpPageEmpty4}
\pastebutton{DeRhamComplexXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{R := Expression coefRing\free{coefRing }\bound{R }}
\indentrel{3}\begin{verbatim}
   (4)  Expression Integer
                                        Type: Domain
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPageEmpty4}
\begin{paste}{DeRhamComplexXmpPageEmpty4}{DeRhamComplexXmpPagePatch4}
\pastebutton{DeRhamComplexXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{R := Expression coefRing\free{coefRing }\bound{R }}
\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPagePatch5}
\begin{paste}{DeRhamComplexXmpPageFull5}{DeRhamComplexXmpPageEmpty5}
\pastebutton{DeRhamComplexXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{f : R := x**2*y*z-5*x**3*y**2*z**5\free{R }\bound{f }}
\indentrel{3}\begin{verbatim}
            3 2 5    2
   (5)  - 5x y z  + x y z
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPageEmpty5}
\begin{paste}{DeRhamComplexXmpPageEmpty5}{DeRhamComplexXmpPagePatch5}
\pastebutton{DeRhamComplexXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{f : R := x**2*y*z-5*x**3*y**2*z**5\free{R }\bound{f }}
\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPagePatch6}
\begin{paste}{DeRhamComplexXmpPageFull6}{DeRhamComplexXmpPageEmpty6}
\pastebutton{DeRhamComplexXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{g : R := z**2*y*cos(z)-7*sin(x**3*y**2)*z**2\free{R }\bound{g }}
\indentrel{3}\begin{verbatim}
            2     3 2       2
   (6)  - 7z sin(x y ) + y z cos(z)
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPageEmpty6}
\begin{paste}{DeRhamComplexXmpPageEmpty6}{DeRhamComplexXmpPagePatch6}
\pastebutton{DeRhamComplexXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{g : R := z**2*y*cos(z)-7*sin(x**3*y**2)*z**2\free{R }\bound{g }}
\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPagePatch7}
\begin{paste}{DeRhamComplexXmpPageFull7}{DeRhamComplexXmpPageEmpty7}
\pastebutton{DeRhamComplexXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{h : R :=x*y*z-2*x**3*y*z**2\free{R }\bound{h }}
\indentrel{3}\begin{verbatim}
            3   2
   (7)  - 2x y z  + x y z
                               Type: Expression Integer
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPageEmpty7}
\begin{paste}{DeRhamComplexXmpPageEmpty7}{DeRhamComplexXmpPagePatch7}
\pastebutton{DeRhamComplexXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{h : R :=x*y*z-2*x**3*y*z**2\free{R }\bound{h }}
\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPagePatch8}
\begin{paste}{DeRhamComplexXmpPageFull8}{DeRhamComplexXmpPageEmpty8}
\pastebutton{DeRhamComplexXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{dx : der := generator(1)\free{der }\bound{dx }}
\indentrel{3}\begin{verbatim}
   (8)  dx
                  Type: DeRhamComplex(Integer,[x,y,z])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPageEmpty8}
\begin{paste}{DeRhamComplexXmpPageEmpty8}{DeRhamComplexXmpPagePatch8}
\pastebutton{DeRhamComplexXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{dx : der := generator(1)\free{der }\bound{dx }}
\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPagePatch9}
\begin{paste}{DeRhamComplexXmpPageFull9}{DeRhamComplexXmpPageEmpty9}
\pastebutton{DeRhamComplexXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{dy : der := generator(2)\free{der }\bound{dy }}
\indentrel{3}\begin{verbatim}
   (9)  dy
                  Type: DeRhamComplex(Integer,[x,y,z])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPageEmpty9}
\begin{paste}{DeRhamComplexXmpPageEmpty9}{DeRhamComplexXmpPagePatch9}
\pastebutton{DeRhamComplexXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{dy : der := generator(2)\free{der }\bound{dy }}
\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPagePatch10}
\begin{paste}{DeRhamComplexXmpPageFull10}{DeRhamComplexXmpPageEmpty10}
\pastebutton{DeRhamComplexXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{dz : der := generator(3)\free{der }\bound{dz }}
\indentrel{3}\begin{verbatim}
   (10)  dz
                  Type: DeRhamComplex(Integer,[x,y,z])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{DeRhamComplexXmpPageEmpty10}
\begin{paste}{DeRhamComplexXmpPageEmpty10}{DeRhamComplexXmpPagePatch10}
\pastebutton{DeRhamComplexXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{dz : der := generator(3)\free{der }\bound{dz }}
\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPagePatch11}
\begin{paste}{DeRhamComplexXmpPageFull11}{DeRhamComplexXmpPageEmpty11}
\pastebutton{DeRhamComplexXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{[dx,dy,dz] := [generator(i)$der for i in 1..3]\free{der }\bound{dxyz }}
\indentrel{3}\begin{verbatim}
   (11)  [dx,dy,dz]
              Type: List DeRhamComplex(Integer,[x,y,z])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPageEmpty11}
\begin{paste}{DeRhamComplexXmpPageEmpty11}{DeRhamComplexXmpPagePatch11}
\pastebutton{DeRhamComplexXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{[dx,dy,dz] := [generator(i)$der for i in 1..3]\free{der }\bound{dxyz }}
\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPagePatch12}
\begin{paste}{DeRhamComplexXmpPageFull12}{DeRhamComplexXmpPageEmpty12}
\pastebutton{DeRhamComplexXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{alpha : der := f*dx + g*dy + h*dz\bound{alpha }\free{der f g h dxyz }}
\indentrel{3}\begin{verbatim}
   (12)
          3   2
     (- 2x y z  + x y z)dz
   +
        2     3 2       2                 3 2 5    2
   (- 7z sin(x y ) + y z cos(z))dy + (- 5x y z  + x y z)dx
                   Type: DeRhamComplex(Integer,[x,y,z])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPageEmpty12}
\begin{paste}{DeRhamComplexXmpPageEmpty12}{DeRhamComplexXmpPagePatch12}
\pastebutton{DeRhamComplexXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{alpha : der := f*dx + g*dy + h*dz\bound{alpha }\free{der f g h dxyz }}
\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPagePatch13}
\begin{paste}{DeRhamComplexXmpPageFull13}{DeRhamComplexXmpPageEmpty13}
\pastebutton{DeRhamComplexXmpPageFull13}{\hidepaste}
\tab{5}\spadcommand{beta : der := cos(tan(x*y*z)+x*y*z)*dx + x*dy\bound{beta }\free{der f g h dx
\indentrel{3}\begin{verbatim}
   (13)  x dy + cos(tan(x y z) + x y z)dx
```

```
                  Type: DeRhamComplex(Integer,[x,y,z])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPageEmpty13}
\begin{paste}{DeRhamComplexXmpPageEmpty13}{DeRhamComplexXmpPagePatch13}
\pastebutton{DeRhamComplexXmpPageEmpty13}{\showpaste}
\tab{5}\spadcommand{beta : der := cos(tan(x*y*z)+x*y*z)*dx + x*dy\bound{beta }\free{der f g h dxyz }}
\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPagePatch14}
\begin{paste}{DeRhamComplexXmpPageFull14}{DeRhamComplexXmpPageEmpty14}
\pastebutton{DeRhamComplexXmpPageFull14}{\hidepaste}
\tab{5}\spadcommand{exteriorDifferential alpha;\free{alpha }\bound{ed }}
\indentrel{3}\begin{verbatim}
                  Type: DeRhamComplex(Integer,[x,y,z])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPageEmpty14}
\begin{paste}{DeRhamComplexXmpPageEmpty14}{DeRhamComplexXmpPagePatch14}
\pastebutton{DeRhamComplexXmpPageEmpty14}{\showpaste}
\tab{5}\spadcommand{exteriorDifferential alpha;\free{alpha }\bound{ed }}
\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPagePatch15}
\begin{paste}{DeRhamComplexXmpPageFull15}{DeRhamComplexXmpPageEmpty15}
\pastebutton{DeRhamComplexXmpPageFull15}{\hidepaste}
\tab{5}\spadcommand{exteriorDifferential \%\free{ed }}
\indentrel{3}\begin{verbatim}
   (15)  0
                  Type: DeRhamComplex(Integer,[x,y,z])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPageEmpty15}
\begin{paste}{DeRhamComplexXmpPageEmpty15}{DeRhamComplexXmpPagePatch15}
\pastebutton{DeRhamComplexXmpPageEmpty15}{\showpaste}
\tab{5}\spadcommand{exteriorDifferential \%\free{ed }}
\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPagePatch16}
\begin{paste}{DeRhamComplexXmpPageFull16}{DeRhamComplexXmpPageEmpty16}
\pastebutton{DeRhamComplexXmpPageFull16}{\hidepaste}
\tab{5}\spadcommand{gamma := alpha * beta\bound{gamma }\free{alpha }\free{beta }}
\indentrel{3}\begin{verbatim}
   (16)
        4   2    2
     (2x y z  - x y z)dy dz
   +
```

```
       3   2
     (2x y z  - x y z)cos(tan(x y z) + x y z)dx dz
   +
             2     3 2       2
           (7z sin(x y ) - y z cos(z))
        *
           cos(tan(x y z) + x y z)
       +
            4 2 5    3
         - 5x y z  + x y z
    *
       dx dy
                  Type: DeRhamComplex(Integer,[x,y,z])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPageEmpty16}
\begin{paste}{DeRhamComplexXmpPageEmpty16}{DeRhamComplexXmpPagePatch16}
\pastebutton{DeRhamComplexXmpPageEmpty16}{\showpaste}
\tab{5}\spadcommand{gamma := alpha * beta\bound{gamma }\free{alpha }\free{beta }}
\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPagePatch17}
\begin{paste}{DeRhamComplexXmpPageFull17}{DeRhamComplexXmpPageEmpty17}
\pastebutton{DeRhamComplexXmpPageFull17}{\hidepaste}
\tab{5}\spadcommand{exteriorDifferential(gamma) - (exteriorDifferential(alpha)*beta - alpha * ex
\indentrel{3}\begin{verbatim}
   (17)  0
                  Type: DeRhamComplex(Integer,[x,y,z])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPageEmpty17}
\begin{paste}{DeRhamComplexXmpPageEmpty17}{DeRhamComplexXmpPagePatch17}
\pastebutton{DeRhamComplexXmpPageEmpty17}{\showpaste}
\tab{5}\spadcommand{exteriorDifferential(gamma) - (exteriorDifferential(alpha)*beta - alpha * ex
\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPagePatch18}
\begin{paste}{DeRhamComplexXmpPageFull18}{DeRhamComplexXmpPageEmpty18}
\pastebutton{DeRhamComplexXmpPageFull18}{\hidepaste}
\tab{5}\spadcommand{a : BOP := operator('a)\bound{ao }}
\indentrel{3}\begin{verbatim}
   (18)  a
                                   Type: BasicOperator
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPageEmpty18}
\begin{paste}{DeRhamComplexXmpPageEmpty18}{DeRhamComplexXmpPagePatch18}
```

```
\pastebutton{DeRhamComplexXmpPageEmpty18}{\showpaste}
\tab{5}\spadcommand{a : BOP := operator('a)\bound{ao }}
\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPagePatch19}
\begin{paste}{DeRhamComplexXmpPageFull19}{DeRhamComplexXmpPageEmpty19}
\pastebutton{DeRhamComplexXmpPageFull19}{\hidepaste}
\tab{5}\spadcommand{b : BOP := operator('b)\bound{bo }}
\indentrel{3}\begin{verbatim}
   (19)  b
                                   Type: BasicOperator
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPageEmpty19}
\begin{paste}{DeRhamComplexXmpPageEmpty19}{DeRhamComplexXmpPagePatch19}
\pastebutton{DeRhamComplexXmpPageEmpty19}{\showpaste}
\tab{5}\spadcommand{b : BOP := operator('b)\bound{bo }}
\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPagePatch20}
\begin{paste}{DeRhamComplexXmpPageFull20}{DeRhamComplexXmpPageEmpty20}
\pastebutton{DeRhamComplexXmpPageFull20}{\hidepaste}
\tab{5}\spadcommand{c : BOP := operator('c)\bound{co }}
\indentrel{3}\begin{verbatim}
   (20)  c
                                   Type: BasicOperator
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPageEmpty20}
\begin{paste}{DeRhamComplexXmpPageEmpty20}{DeRhamComplexXmpPagePatch20}
\pastebutton{DeRhamComplexXmpPageEmpty20}{\showpaste}
\tab{5}\spadcommand{c : BOP := operator('c)\bound{co }}
\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPagePatch21}
\begin{paste}{DeRhamComplexXmpPageFull21}{DeRhamComplexXmpPageEmpty21}
\pastebutton{DeRhamComplexXmpPageFull21}{\hidepaste}
\tab{5}\spadcommand{sigma := a(x,y,z) * dx + b(x,y,z) * dy + c(x,y,z) * dz\bound{sigma }\free{ao bo co dxyz
\indentrel{3}\begin{verbatim}
   (21)  c(x,y,z)dz + b(x,y,z)dy + a(x,y,z)dx
                    Type: DeRhamComplex(Integer,[x,y,z])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPageEmpty21}
\begin{paste}{DeRhamComplexXmpPageEmpty21}{DeRhamComplexXmpPagePatch21}
\pastebutton{DeRhamComplexXmpPageEmpty21}{\showpaste}
\tab{5}\spadcommand{sigma := a(x,y,z) * dx + b(x,y,z) * dy + c(x,y,z) * dz\bound{sigma }\free{ao bo co dxyz
```

```
\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPagePatch22}
\begin{paste}{DeRhamComplexXmpPageFull22}{DeRhamComplexXmpPageEmpty22}
\pastebutton{DeRhamComplexXmpPageFull22}{\hidepaste}
\tab{5}\spadcommand{theta := a(x,y,z) * dx * dy + b(x,y,z) * dx * dz + c(x,y,z) * dy * dz\bound{
\indentrel{3}\begin{verbatim}
   (22)  c(x,y,z)dy dz + b(x,y,z)dx dz + a(x,y,z)dx dy
                   Type: DeRhamComplex(Integer,[x,y,z])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPageEmpty22}
\begin{paste}{DeRhamComplexXmpPageEmpty22}{DeRhamComplexXmpPagePatch22}
\pastebutton{DeRhamComplexXmpPageEmpty22}{\showpaste}
\tab{5}\spadcommand{theta := a(x,y,z) * dx * dy + b(x,y,z) * dx * dz + c(x,y,z) * dy * dz\bound{
\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPagePatch23}
\begin{paste}{DeRhamComplexXmpPageFull23}{DeRhamComplexXmpPageEmpty23}
\pastebutton{DeRhamComplexXmpPageFull23}{\hidepaste}
\tab{5}\spadcommand{totalDifferential(a(x,y,z))$der\free{ao der }}
\indentrel{3}\begin{verbatim}
   (23)  a  (x,y,z)dz + a  (x,y,z)dy + a  (x,y,z)dx
          ,3             ,2             ,1
                   Type: DeRhamComplex(Integer,[x,y,z])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPageEmpty23}
\begin{paste}{DeRhamComplexXmpPageEmpty23}{DeRhamComplexXmpPagePatch23}
\pastebutton{DeRhamComplexXmpPageEmpty23}{\showpaste}
\tab{5}\spadcommand{totalDifferential(a(x,y,z))$der\free{ao der }}
\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPagePatch24}
\begin{paste}{DeRhamComplexXmpPageFull24}{DeRhamComplexXmpPageEmpty24}
\pastebutton{DeRhamComplexXmpPageFull24}{\hidepaste}
\tab{5}\spadcommand{exteriorDifferential sigma\free{sigma }}
\indentrel{3}\begin{verbatim}
   (24)
     (c  (x,y,z) - b  (x,y,z))dy dz
       ,2           ,3
   +
     (c  (x,y,z) - a  (x,y,z))dx dz
       ,1           ,3
   +
     (b  (x,y,z) - a  (x,y,z))dx dy
       ,1           ,2
                   Type: DeRhamComplex(Integer,[x,y,z])
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPageEmpty24}
\begin{paste}{DeRhamComplexXmpPageEmpty24}{DeRhamComplexXmpPagePatch24}
\pastebutton{DeRhamComplexXmpPageEmpty24}{\showpaste}
\tab{5}\spadcommand{exteriorDifferential sigma\free{sigma }}
\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPagePatch25}
\begin{paste}{DeRhamComplexXmpPageFull25}{DeRhamComplexXmpPageEmpty25}
\pastebutton{DeRhamComplexXmpPageFull25}{\hidepaste}
\tab{5}\spadcommand{exteriorDifferential theta\free{theta }}
\indentrel{3}\begin{verbatim}
   (25)  (c  (x,y,z) - b  (x,y,z) + a  (x,y,z))dx dy dz
           ,1           ,2           ,3
                   Type: DeRhamComplex(Integer,[x,y,z])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPageEmpty25}
\begin{paste}{DeRhamComplexXmpPageEmpty25}{DeRhamComplexXmpPagePatch25}
\pastebutton{DeRhamComplexXmpPageEmpty25}{\showpaste}
\tab{5}\spadcommand{exteriorDifferential theta\free{theta }}
\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPagePatch26}
\begin{paste}{DeRhamComplexXmpPageFull26}{DeRhamComplexXmpPageEmpty26}
\pastebutton{DeRhamComplexXmpPageFull26}{\hidepaste}
\tab{5}\spadcommand{one : der := 1\bound{one }\free{der }}
\indentrel{3}\begin{verbatim}
   (26)  1
                   Type: DeRhamComplex(Integer,[x,y,z])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPageEmpty26}
\begin{paste}{DeRhamComplexXmpPageEmpty26}{DeRhamComplexXmpPagePatch26}
\pastebutton{DeRhamComplexXmpPageEmpty26}{\showpaste}
\tab{5}\spadcommand{one : der := 1\bound{one }\free{der }}
\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPagePatch27}
\begin{paste}{DeRhamComplexXmpPageFull27}{DeRhamComplexXmpPageEmpty27}
\pastebutton{DeRhamComplexXmpPageFull27}{\hidepaste}
\tab{5}\spadcommand{g1 : der := a([x,t,y,u,v,z,e]) * one\free{der one ao }\bound{g1 }}
\indentrel{3}\begin{verbatim}
   (27)  a(x,t,y,u,v,z,e)
                   Type: DeRhamComplex(Integer,[x,y,z])
\end{verbatim}
```

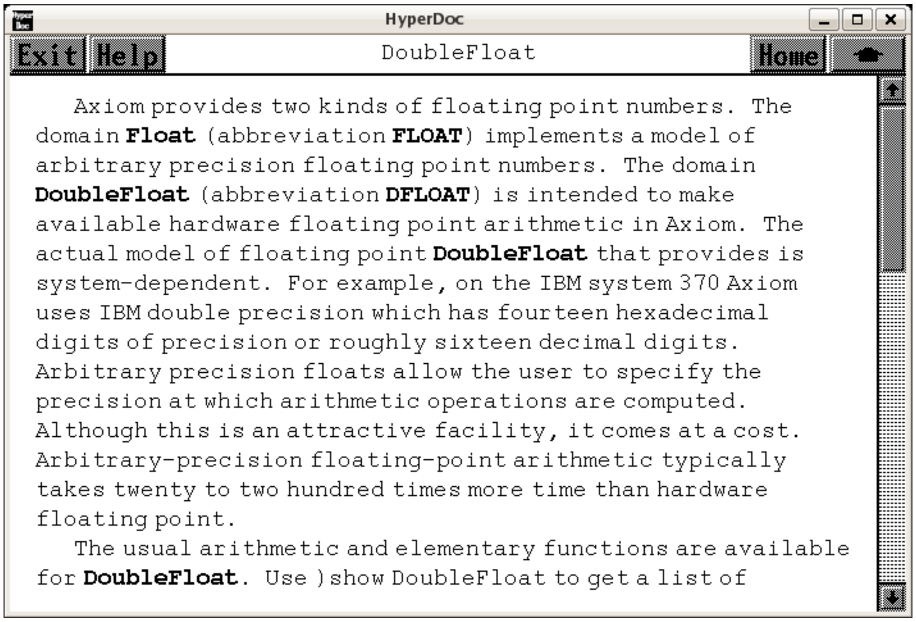
```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPageEmpty27}
\begin{paste}{DeRhamComplexXmpPageEmpty27}{DeRhamComplexXmpPagePatch27}
\pastebutton{DeRhamComplexXmpPageEmpty27}{\showpaste}
\tab{5}\spadcommand{g1 : der := a([x,t,y,u,v,z,e]) * one\free{der one ao }\bound{g1 }}
\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPagePatch28}
\begin{paste}{DeRhamComplexXmpPageFull28}{DeRhamComplexXmpPageEmpty28}
\pastebutton{DeRhamComplexXmpPageFull28}{\hidepaste}
\tab{5}\spadcommand{h1 : der := a([x,y,x,t,x,z,y,r,u,x]) * one\free{der one ao }\bound{h1 }}
\indentrel{3}\begin{verbatim}
   (28)  a(x,y,x,t,x,z,y,r,u,x)
                  Type: DeRhamComplex(Integer,[x,y,z])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPageEmpty28}
\begin{paste}{DeRhamComplexXmpPageEmpty28}{DeRhamComplexXmpPagePatch28}
\pastebutton{DeRhamComplexXmpPageEmpty28}{\showpaste}
\tab{5}\spadcommand{h1 : der := a([x,y,x,t,x,z,y,r,u,x]) * one\free{der one ao }\bound{h1 }}
\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPagePatch29}
\begin{paste}{DeRhamComplexXmpPageFull29}{DeRhamComplexXmpPageEmpty29}
\pastebutton{DeRhamComplexXmpPageFull29}{\hidepaste}
\tab{5}\spadcommand{exteriorDifferential g1\free{g1 }}
\indentrel{3}\begin{verbatim}
   (29)
     a  (x,t,y,u,v,z,e)dz + a  (x,t,y,u,v,z,e)dy
      ,6                     ,3
   +
     a  (x,t,y,u,v,z,e)dx
      ,1
                  Type: DeRhamComplex(Integer,[x,y,z])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPageEmpty29}
\begin{paste}{DeRhamComplexXmpPageEmpty29}{DeRhamComplexXmpPagePatch29}
\pastebutton{DeRhamComplexXmpPageEmpty29}{\showpaste}
\tab{5}\spadcommand{exteriorDifferential g1\free{g1 }}
\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPagePatch30}
\begin{paste}{DeRhamComplexXmpPageFull30}{DeRhamComplexXmpPageEmpty30}
\pastebutton{DeRhamComplexXmpPageFull30}{\hidepaste}
\tab{5}\spadcommand{exteriorDifferential h1\free{h1 }}
\indentrel{3}\begin{verbatim}
```

```
   (30)
     a  (x,y,x,t,x,z,y,r,u,x)dz
      ,6
   +
           a  (x,y,x,t,x,z,y,r,u,x)
            ,7
         +
           a  (x,y,x,t,x,z,y,r,u,x)
            ,2
    *
       dy
   +
         a   (x,y,x,t,x,z,y,r,u,x)
          ,10
       +
         a  (x,y,x,t,x,z,y,r,u,x)
          ,5
       +
       a  (x,y,x,t,x,z,y,r,u,x) + a  (x,y,x,t,x,z,y,r,u,x)
        ,3                         ,1
    *
       dx
                   Type: DeRhamComplex(Integer,[x,y,z])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPageEmpty30}
\begin{paste}{DeRhamComplexXmpPageEmpty30}{DeRhamComplexXmpPagePatch30}
\pastebutton{DeRhamComplexXmpPageEmpty30}{\showpaste}
\tab{5}\spadcommand{exteriorDifferential h1\free{h1 }}
\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPagePatch31}
\begin{paste}{DeRhamComplexXmpPageFull31}{DeRhamComplexXmpPageEmpty31}
\pastebutton{DeRhamComplexXmpPageFull31}{\hidepaste}
\tab{5}\spadcommand{coefficient(gamma, dx*dy)\free{gamma dxyz }}
\indentrel{3}\begin{verbatim}
   (31)
        2     3 2       2
     (7z sin(x y ) - y z cos(z))cos(tan(x y z) + x y z)
   +
         4 2 5    3
     - 5x y z  + x y z
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPageEmpty31}
\begin{paste}{DeRhamComplexXmpPageEmpty31}{DeRhamComplexXmpPagePatch31}
\pastebutton{DeRhamComplexXmpPageEmpty31}{\showpaste}
```

```
\tab{5}\spadcommand{coefficient(gamma, dx*dy)\free{gamma dxyz }}
\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPagePatch32}
\begin{paste}{DeRhamComplexXmpPageFull32}{DeRhamComplexXmpPageEmpty32}
\pastebutton{DeRhamComplexXmpPageFull32}{\hidepaste}
\tab{5}\spadcommand{coefficient(gamma, one)\free{gamma one }}
\indentrel{3}\begin{verbatim}
   (32)  0
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPageEmpty32}
\begin{paste}{DeRhamComplexXmpPageEmpty32}{DeRhamComplexXmpPagePatch32}
\pastebutton{DeRhamComplexXmpPageEmpty32}{\showpaste}
\tab{5}\spadcommand{coefficient(gamma, one)\free{gamma one }}
\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPagePatch33}
\begin{paste}{DeRhamComplexXmpPageFull33}{DeRhamComplexXmpPageEmpty33}
\pastebutton{DeRhamComplexXmpPageFull33}{\hidepaste}
\tab{5}\spadcommand{coefficient(g1,one)\free{g1 one }}
\indentrel{3}\begin{verbatim}
   (33)  a(x,t,y,u,v,z,e)
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DeRhamComplexXmpPageEmpty33}
\begin{paste}{DeRhamComplexXmpPageEmpty33}{DeRhamComplexXmpPagePatch33}
\pastebutton{DeRhamComplexXmpPageEmpty33}{\showpaste}
\tab{5}\spadcommand{coefficient(g1,one)\free{g1 one }}
\end{paste}\end{patch}
```

3.23 dfloat.ht

DoubleFloat



⇐ “Axiom Number Types” (NumberPage) 3.80 on page 1021
⇒ “Graphics” (ugGraphPage) 11 on page 1967
⇒ “Numeric Functions” (ugProblemNumericPage) 12 on page 2079
⇒ “Float” (FloatXmpPage) 3.41 on page 487

— **dfloat.ht** —

```
\begin{page}{DoubleFloatXmpPage}{DoubleFloat}
\beginscroll
Axiom provides two kinds of floating point numbers.
The domain \spadtype{Float} (abbreviation \spadtype{FLOAT})
implements a model of arbitrary
precision floating point numbers.
The domain \spadtype{DoubleFloat} (abbreviation \spadtype{DFLOAT})
is intended to make available
hardware floating point arithmetic in Axiom.
The actual model of floating point \spadtype{DoubleFloat} that provides is
system-dependent.
For example, on the IBM system 370 Axiom uses IBM double
precision which has fourteen hexadecimal digits of precision or roughly
sixteen decimal digits.
Arbitrary precision floats allow the user to specify the
precision at which arithmetic operations are computed.
Although this is an attractive facility, it comes at a cost.
```

```
Arbitrary-precision floating-point arithmetic typically takes
twenty to two hundred times more time than hardware floating point.

The usual arithmetic and elementary functions are available for
\spadtype{DoubleFloat}.
Use \spadsys{)show DoubleFloat} to get a list of operations or
the Hyperdoc \Browse{} facility to get more extensive
documentation about \spadtype{DoubleFloat}.

\xtc{
By default, floating point numbers that you enter into Axiom
are of type \spadtype{Float}.
}{
\spadpaste{2.71828}
}
You must therefore tell Axiom that you want to use
\spadtype{DoubleFloat} values and operations.
The following are some conservative guidelines for getting
Axiom to use \spadtype{DoubleFloat}.

\xtc{
To get a value of type \spadtype{DoubleFloat}, use a target with
\spadSyntax{@}, \ldots
}{
\spadpaste{2.71828@DoubleFloat}
}
\xtc{
a conversion, \ldots
}{
\spadpaste{2.71828 :: DoubleFloat}
}
\xtc{
or an assignment to a declared variable.
It is more efficient if you use a target rather than an explicit or
implicit conversion.
}{
\spadpaste{eApprox : DoubleFloat := 2.71828 \bound{eApprox}}
}

\xtc{
You also need to declare functions that work with
\spadtype{DoubleFloat}.
}{
\spadpaste{avg : List DoubleFloat -> DoubleFloat \bound{avgDec}}
}
\xtc{
}{
\begin{spadsrc}[\bound{avg}\free{avgDec}]
avg l ==
  empty? l => 0 :: DoubleFloat
```

```
  reduce(_+,l) / #l
\end{spadsrc}
}
\xtc{
}{
\spadpaste{avg [] \free{avg}}
}
\xtc{
}{
\spadpaste{avg [3.4,9.7,-6.8] \free{avg}}
}
\xtc{
Use package-calling for operations from \spadtype{DoubleFloat} unless
the arguments themselves are already of type \spadtype{DoubleFloat}.
}{
\spadpaste{cos(3.1415926)\$DoubleFloat}
}
\xtc{
}{
\spadpaste{cos(3.1415926 :: DoubleFloat)}
}

By far, the most common usage of \spadtype{DoubleFloat} is for functions
to be graphed.
For more information about Axiom's numerical and graphical
facilities, see
\downlink{``Graphics''}{ugGraphPage} in Section 7.
\ignore{ugGraph},
\downlink{``Numeric Functions''}{ugProblemNumericPage} in
Section 8.1\ignore{ugProblemNumeric}, and
\downlink{`Float'}{FloatXmpPage}\ignore{Float}.
\endscroll
\autobuttons
\end{page}

\begin{patch}{DoubleFloatXmpPagePatch1}
\begin{paste}{DoubleFloatXmpPageFull1}{DoubleFloatXmpPageEmpty1}
\pastebutton{DoubleFloatXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{2.71828}
\indentrel{3}\begin{verbatim}
   (1)  2.71828
                                             Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DoubleFloatXmpPageEmpty1}
\begin{paste}{DoubleFloatXmpPageEmpty1}{DoubleFloatXmpPagePatch1}
\pastebutton{DoubleFloatXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{2.71828}
\end{paste}\end{patch}
```

```
\begin{patch}{DoubleFloatXmpPagePatch2}
\begin{paste}{DoubleFloatXmpPageFull2}{DoubleFloatXmpPageEmpty2}
\pastebutton{DoubleFloatXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{2.71828@DoubleFloat}
\indentrel{3}\begin{verbatim}
   (2)  2.71828
                                       Type: DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DoubleFloatXmpPageEmpty2}
\begin{paste}{DoubleFloatXmpPageEmpty2}{DoubleFloatXmpPagePatch2}
\pastebutton{DoubleFloatXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{2.71828@DoubleFloat}
\end{paste}\end{patch}

\begin{patch}{DoubleFloatXmpPagePatch3}
\begin{paste}{DoubleFloatXmpPageFull3}{DoubleFloatXmpPageEmpty3}
\pastebutton{DoubleFloatXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{2.71828 :: DoubleFloat}
\indentrel{3}\begin{verbatim}
   (3)  2.71828
                                       Type: DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DoubleFloatXmpPageEmpty3}
\begin{paste}{DoubleFloatXmpPageEmpty3}{DoubleFloatXmpPagePatch3}
\pastebutton{DoubleFloatXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{2.71828 :: DoubleFloat}
\end{paste}\end{patch}

\begin{patch}{DoubleFloatXmpPagePatch4}
\begin{paste}{DoubleFloatXmpPageFull4}{DoubleFloatXmpPageEmpty4}
\pastebutton{DoubleFloatXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{eApprox : DoubleFloat := 2.71828\bound{eApprox }}
\indentrel{3}\begin{verbatim}
   (4)  2.71828
                                       Type: DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DoubleFloatXmpPageEmpty4}
\begin{paste}{DoubleFloatXmpPageEmpty4}{DoubleFloatXmpPagePatch4}
\pastebutton{DoubleFloatXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{eApprox : DoubleFloat := 2.71828\bound{eApprox }}
\end{paste}\end{patch}

\begin{patch}{DoubleFloatXmpPagePatch5}
```

```
\begin{paste}{DoubleFloatXmpPageFull5}{DoubleFloatXmpPageEmpty5}
\pastebutton{DoubleFloatXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{avg : List DoubleFloat -> DoubleFloat\bound{avgDec }}
\indentrel{3}\begin{verbatim}
                                              Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DoubleFloatXmpPageEmpty5}
\begin{paste}{DoubleFloatXmpPageEmpty5}{DoubleFloatXmpPagePatch5}
\pastebutton{DoubleFloatXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{avg : List DoubleFloat -> DoubleFloat\bound{avgDec }}
\end{paste}\end{patch}

\begin{patch}{DoubleFloatXmpPagePatch6}
\begin{paste}{DoubleFloatXmpPageFull6}{DoubleFloatXmpPageEmpty6}
\pastebutton{DoubleFloatXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{avg l ==
  empty? l => 0 :: DoubleFloat
  reduce(_+,l) / \#l
\bound{avg }\free{avgDec }}
\indentrel{3}\begin{verbatim}
                                              Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DoubleFloatXmpPageEmpty6}
\begin{paste}{DoubleFloatXmpPageEmpty6}{DoubleFloatXmpPagePatch6}
\pastebutton{DoubleFloatXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{avg l ==
  empty? l => 0 :: DoubleFloat
  reduce(_+,l) / \#l
\bound{avg }\free{avgDec }}
\end{paste}\end{patch}

\begin{patch}{DoubleFloatXmpPagePatch7}
\begin{paste}{DoubleFloatXmpPageFull7}{DoubleFloatXmpPageEmpty7}
\pastebutton{DoubleFloatXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{avg []\free{avg }}
\indentrel{3}\begin{verbatim}
   (7)  0.0
                                       Type: DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DoubleFloatXmpPageEmpty7}
\begin{paste}{DoubleFloatXmpPageEmpty7}{DoubleFloatXmpPagePatch7}
\pastebutton{DoubleFloatXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{avg []\free{avg }}
\end{paste}\end{patch}
```

```
\begin{patch}{DoubleFloatXmpPagePatch8}
\begin{paste}{DoubleFloatXmpPageFull8}{DoubleFloatXmpPageEmpty8}
\pastebutton{DoubleFloatXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{avg [3.4,9.7,-6.8]\free{avg }}
\indentrel{3}\begin{verbatim}
   (8)  2.1
                                      Type: DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DoubleFloatXmpPageEmpty8}
\begin{paste}{DoubleFloatXmpPageEmpty8}{DoubleFloatXmpPagePatch8}
\pastebutton{DoubleFloatXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{avg [3.4,9.7,-6.8]\free{avg }}
\end{paste}\end{patch}

\begin{patch}{DoubleFloatXmpPagePatch9}
\begin{paste}{DoubleFloatXmpPageFull9}{DoubleFloatXmpPageEmpty9}
\pastebutton{DoubleFloatXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{cos(3.1415926)$DoubleFloat}
\indentrel{3}\begin{verbatim}
   (9)  - 0.999999999999999
                                      Type: DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DoubleFloatXmpPageEmpty9}
\begin{paste}{DoubleFloatXmpPageEmpty9}{DoubleFloatXmpPagePatch9}
\pastebutton{DoubleFloatXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{cos(3.1415926)$DoubleFloat}
\end{paste}\end{patch}

\begin{patch}{DoubleFloatXmpPagePatch10}
\begin{paste}{DoubleFloatXmpPageFull10}{DoubleFloatXmpPageEmpty10}
\pastebutton{DoubleFloatXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{cos(3.1415926 :: DoubleFloat)}
\indentrel{3}\begin{verbatim}
   (10)  - 0.999999999999999
                                      Type: DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DoubleFloatXmpPageEmpty10}
\begin{paste}{DoubleFloatXmpPageEmpty10}{DoubleFloatXmpPagePatch10}
\pastebutton{DoubleFloatXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{cos(3.1415926 :: DoubleFloat)}
\end{paste}\end{patch}
```

———-

3.24 dmp.ht

DistributedMultivariatePoly

⇒ “notitle” (ugIntroVariablesPage) 6 on page 1566
⇒ “notitle” (ugTypesConvertPage) 7 on page 1671
⇒ “notitle” (PolynomialXmpPage) 3.88 on page 1110
⇒ “notitle” (UnivariatePolyXmpPage) 3.112 on page 1327
⇒ “notitle” (MultivariatePolyXmpPage) 3.77 on page 1011

— **dmp.ht** —

```
\begin{page}{DistributedMultivariatePolyXmpPage}
{DistributedMultivariatePoly}
\beginscroll

\texht{\hyphenation{
Homo-gen-eous-Dis-tributed-Multi-var-i-ate-Pol-y-nomial
}}{}

\spadtype{DistributedMultivariatePoly} and
\spadtype{HomogeneousDistributedMultivariatePoly}, abbreviated
\spadtype{DMP} and \spadtype{HDMP}, respectively, are very similar to
\spadtype{MultivariatePolynomial} except that they are represented and
displayed in a non-recursive manner.
\xtc{
}{
\spadpaste{(d1,d2,d3) : DMP([z,y,x],FRAC INT) \bound{d1dec d2dec d3dec}}
}
\xtc{
The constructor
\spadtype{DMP} orders its monomials lexicographically while
\spadtype{HDMP} orders them by total order refined by reverse
lexicographic order.
}{
\spadpaste{d1 := -4*z + 4*y**2*x + 16*x**2 + 1 \bound{d1}\free{d1dec}}
}
\xtc{
}{
\spadpaste{d2 := 2*z*y**2 + 4*x + 1 \bound{d2}\free{d2dec}}
}
\xtc{
}{
\spadpaste{d3 := 2*z*x**2 - 2*y**2 - x \bound{d3}\free{d3dec}}
}
```

```
\xtc{
These constructors are mostly used in \texht{Gr\"{o}bner}{Groebner}
basis calculations.
}{
\spadpaste{groebner [d1,d2,d3] \free{d1 d2 d3}}
}
\xtc{
}{
\spadpaste{(n1,n2,n3) : HDMP([z,y,x],FRAC INT) \bound{ndec}}
}
\xtc{
}{
\spadpaste{(n1,n2,n3) := (d1,d2,d3) \free{ndec}\bound{n}\free{d1 d2 d3}}
}
\xtc{
Note that we get a different
\texht{Gr\"{o}bner}{Groebner} basis
when we use the \spadtype{HDMP} polynomials, as expected.
}{
\spadpaste{groebner [n1,n2,n3] \free{n}}
}

\spadtype{GeneralDistributedMultivariatePoly} is somewhat
more flexible in the sense that as well as accepting a list of
variables to specify the variable ordering, it also takes a
predicate on exponent vectors to specify the term ordering.
With this polynomial type the user can experiment with the effect
of using completely arbitrary term orderings.
This flexibility is mostly important for algorithms such as
\texht{Gr\"{o}bner}{Groebner} basis calculations which can be very
sensitive to term ordering.

For more information on related topics, see
\downlink{``Polynomials''}{ugIntroVariablesPage}
in Section 1.9\ignore{ugIntroVariables},
\downlink{``Conversion''}{ugTypesConvertPage}
in Section 2.7\ignore{ugTypesConvert},
\downlink{`Polynomial'}{PolynomialXmpPage}\ignore{Polynomial},
\downlink{`UnivariatePolynomial'}{UnivariatePolyXmpPage}
\ignore{UnivariatePolynomial}, and
\downlink{`MultivariatePolynomial'}{MultivariatePolyXmpPage}
\ignore{MultivariatePolynomial}.
%
\showBlurb{DistributedMultivariatePoly}
\endscroll
\autobuttons
\end{page}

\begin{patch}{DistributedMultivariatePolynomialXmpPagePatch1}
\begin{paste}{DistributedMultivariatePolynomialXmpPageFull1}{DistributedMultivariatePolynomialXm
```

```
\pastebutton{DistributedMultivariatePolynomialXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{(d1,d2,d3) : DMP([z,y,x],FRAC INT)\bound{d1dec d2dec d3dec }}
\indentrel{3}\begin{verbatim}
                                              Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DistributedMultivariatePolynomialXmpPageEmpty1}
\begin{paste}{DistributedMultivariatePolynomialXmpPageEmpty1}{DistributedMultivariatePolynomialXmpPagePatch1
\pastebutton{DistributedMultivariatePolynomialXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{(d1,d2,d3) : DMP([z,y,x],FRAC INT)\bound{d1dec d2dec d3dec }}
\end{paste}\end{patch}

\begin{patch}{DistributedMultivariatePolynomialXmpPagePatch2}
\begin{paste}{DistributedMultivariatePolynomialXmpPageFull2}{DistributedMultivariatePolynomialXmpPageEmpty2}
\pastebutton{DistributedMultivariatePolynomialXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{d1 := -4*z + 4*y**2*x + 16*x**2 + 1\bound{d1 }\free{d1dec }}
\indentrel{3}\begin{verbatim}
                2       2
   (2)  - 4z + 4y x + 16x  + 1
Type: DistributedMultivariatePolynomial([z,y,x],Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DistributedMultivariatePolynomialXmpPageEmpty2}
\begin{paste}{DistributedMultivariatePolynomialXmpPageEmpty2}{DistributedMultivariatePolynomialXmpPagePatch2
\pastebutton{DistributedMultivariatePolynomialXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{d1 := -4*z + 4*y**2*x + 16*x**2 + 1\bound{d1 }\free{d1dec }}
\end{paste}\end{patch}

\begin{patch}{DistributedMultivariatePolynomialXmpPagePatch3}
\begin{paste}{DistributedMultivariatePolynomialXmpPageFull3}{DistributedMultivariatePolynomialXmpPageEmpty3}
\pastebutton{DistributedMultivariatePolynomialXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{d2 := 2*z*y**2 + 4*x + 1\bound{d2 }\free{d2dec }}
\indentrel{3}\begin{verbatim}
           2
   (3)  2z y  + 4x + 1
Type: DistributedMultivariatePolynomial([z,y,x],Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DistributedMultivariatePolynomialXmpPageEmpty3}
\begin{paste}{DistributedMultivariatePolynomialXmpPageEmpty3}{DistributedMultivariatePolynomialXmpPagePatch3
\pastebutton{DistributedMultivariatePolynomialXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{d2 := 2*z*y**2 + 4*x + 1\bound{d2 }\free{d2dec }}
\end{paste}\end{patch}

\begin{patch}{DistributedMultivariatePolynomialXmpPagePatch4}
\begin{paste}{DistributedMultivariatePolynomialXmpPageFull4}{DistributedMultivariatePolynomialXmpPageEmpty4}
\pastebutton{DistributedMultivariatePolynomialXmpPageFull4}{\hidepaste}
```

```
\tab{5}\spadcommand{d3 := 2*z*x**2 - 2*y**2 - x\bound{d3 }\free{d3dec }}
\indentrel{3}\begin{verbatim}
            2     2
   (4)  2z x  - 2y  - x
Type: DistributedMultivariatePolynomial([z,y,x],Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DistributedMultivariatePolynomialXmpPageEmpty4}
\begin{paste}{DistributedMultivariatePolynomialXmpPageEmpty4}{DistributedMultivariatePolynomialX
\pastebutton{DistributedMultivariatePolynomialXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{d3 := 2*z*x**2 - 2*y**2 - x\bound{d3 }\free{d3dec }}
\end{paste}\end{patch}

\begin{patch}{DistributedMultivariatePolynomialXmpPagePatch5}
\begin{paste}{DistributedMultivariatePolynomialXmpPageFull5}{DistributedMultivariatePolynomialXm
\pastebutton{DistributedMultivariatePolynomialXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{groebner [d1,d2,d3]\free{d1 d2 d3 }}
\indentrel{3}\begin{verbatim}
   (5)
   [
           1568  6   1264  5    6   4   182  3   2047  2
       z -  x  -  x  +  x  +  x  -  x
           2745       305      305      549       610
     +
          103      2857
       -  x -
         2745     10980
     ,

        2    112  6    84  5   1264  4    13  3    84  2
       y  +  x  -  x  -  x  -  x  +  x
            2745      305       305      549      305
     +
       1772       2
        x +
       2745     2745
     ,
     7   29  6   17  4   11  3    1  2   15     1
    x  +  x  -  x  -  x  +  x  +  x + ]
          4      16       8      32      16     4
Type: List DistributedMultivariatePolynomial([z,y,x],Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DistributedMultivariatePolynomialXmpPageEmpty5}
\begin{paste}{DistributedMultivariatePolynomialXmpPageEmpty5}{DistributedMultivariatePolynomialX
\pastebutton{DistributedMultivariatePolynomialXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{groebner [d1,d2,d3]\free{d1 d2 d3 }}
\end{paste}\end{patch}
```

```
\begin{patch}{DistributedMultivariatePolynomialXmpPagePatch6}
\begin{paste}{DistributedMultivariatePolynomialXmpPageFull6}{DistributedMultivariatePolynomialXmpPageEmpty6}
\pastebutton{DistributedMultivariatePolynomialXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{(n1,n2,n3) : HDMP([z,y,x],FRAC INT)\bound{ndec }}
\indentrel{3}\begin{verbatim}
                                              Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DistributedMultivariatePolynomialXmpPageEmpty6}
\begin{paste}{DistributedMultivariatePolynomialXmpPageEmpty6}{DistributedMultivariatePolynomialXmpPagePatch6}
\pastebutton{DistributedMultivariatePolynomialXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{(n1,n2,n3) : HDMP([z,y,x],FRAC INT)\bound{ndec }}
\end{paste}\end{patch}

\begin{patch}{DistributedMultivariatePolynomialXmpPagePatch7}
\begin{paste}{DistributedMultivariatePolynomialXmpPageFull7}{DistributedMultivariatePolynomialXmpPageEmpty7}
\pastebutton{DistributedMultivariatePolynomialXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{(n1,n2,n3) := (d1,d2,d3)\free{ndec }\bound{n }\free{d1 d2 d3 }}
\indentrel{3}\begin{verbatim}
            2     2
   (7)  2z x  - 2y  - x
Type: HomogeneousDistributedMultivariatePolynomial([z,y,x],Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DistributedMultivariatePolynomialXmpPageEmpty7}
\begin{paste}{DistributedMultivariatePolynomialXmpPageEmpty7}{DistributedMultivariatePolynomialXmpPagePatch7}
\pastebutton{DistributedMultivariatePolynomialXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{(n1,n2,n3) := (d1,d2,d3)\free{ndec }\bound{n }\free{d1 d2 d3 }}
\end{paste}\end{patch}

\begin{patch}{DistributedMultivariatePolynomialXmpPagePatch8}
\begin{paste}{DistributedMultivariatePolynomialXmpPageFull8}{DistributedMultivariatePolynomialXmpPageEmpty8}
\pastebutton{DistributedMultivariatePolynomialXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{groebner [n1,n2,n3]\free{n }}
\indentrel{3}\begin{verbatim}
   (8)
    4     3   3  2   1     1
   [y  + 2x  -  x  +  z - ,
             2     2     8
    4   29  3   1  2   7        9     1     2        1
    x  +  x  -  y  -  z x -  x - , z y  + 2x + ,
        4      8      4       16     4             2
    2     2       1     2    2   1
    y x + 4x  - z + , z x  - y  -  x,
                  4             2
    2     2     2   1     3
    z  - 4y  + 2x  -  z -  x]
```

```
                    4     2
Type: List HomogeneousDistributedMultivariatePolynomial([z,y,x],Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{DistributedMultivariatePolynomialXmpPageEmpty8}
\begin{paste}{DistributedMultivariatePolynomialXmpPageEmpty8}{DistributedMultivariatePolynomialX
\pastebutton{DistributedMultivariatePolynomialXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{groebner [n1,n2,n3]\free{n }}
\end{paste}\end{patch}
```

————-

3.25 eq.ht

Equation

— eq.ht —

```
\begin{page}{EquationXmpPage}{Equation}
\beginscroll

The \spadtype{Equation} domain provides equations as mathematical objects.
These are used, for example, as the input to various
\spadfunFrom{solve}{TransSolvePackage} operations.

\xtc{
Equations are created using the equals symbol, \spadopFrom{=}{Equation}.
}{
\spadpaste{eq1 := 3*x + 4*y = 5 \bound{eq1}}
}
\xtc{
}{
\spadpaste{eq2 := 2*x + 2*y = 3 \bound{eq2}}
}
\xtc{
The left- and right-hand sides of an equation are accessible using
the operations \spadfunFrom{lhs}{Equation} and \spadfunFrom{rhs}{Equation}.
}{
\spadpaste{lhs eq1 \free{eq1}}
}
\xtc{
}{
\spadpaste{rhs eq1 \free{eq1}}
}
```

```
\xtc{
Arithmetic operations are supported
and operate on both sides of the equation.
}{
\spadpaste{eq1 + eq2   \free{eq1 eq2}}
}
\xtc{
}{
\spadpaste{eq1 * eq2   \free{eq1 eq2}}
}
\xtc{
}{
\spadpaste{2*eq2 - eq1 \free{eq1 eq2}}
}
\xtc{
Equations may be created for any type so the arithmetic operations
will be defined only when they make sense.  For example,
exponentiation is not defined for equations involving non-square matrices.
}{
\spadpaste{eq1**2 \free{eq1}}
}

\xtc{
Note that an equals symbol is also used to {\it test}
for equality of values in certain contexts.
For example, \spad{x+1} and \spad{y} are unequal as polynomials.
}{
\spadpaste{if x+1 = y then "equal" else "unequal"}
}
\xtc{
}{
\spadpaste{eqpol := x+1 = y \bound{eqpol}}
}
\xtc{
If an equation is used where a \spadtype{Boolean} value
is required, then it is evaluated using the equality
test from the operand type.
}{
\spadpaste{if eqpol then "equal" else "unequal" \free{eqpol}}
}
\xtc{
If one wants a \spadtype{Boolean} value rather than an equation,
all one has to do is ask!
}{
\spadpaste{eqpol::Boolean \free{eqpol}}
}
\endscroll
\autobuttons
\end{page}
```

```
\begin{patch}{EquationXmpPagePatch1}
\begin{paste}{EquationXmpPageFull1}{EquationXmpPageEmpty1}
\pastebutton{EquationXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{eq1 := 3*x + 4*y = 5\bound{eq1 }}
\indentrel{3}\begin{verbatim}
   (1)  4y + 3x= 5
                      Type: Equation Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{EquationXmpPageEmpty1}
\begin{paste}{EquationXmpPageEmpty1}{EquationXmpPagePatch1}
\pastebutton{EquationXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{eq1 := 3*x + 4*y = 5\bound{eq1 }}
\end{paste}\end{patch}

\begin{patch}{EquationXmpPagePatch2}
\begin{paste}{EquationXmpPageFull2}{EquationXmpPageEmpty2}
\pastebutton{EquationXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{eq2 := 2*x + 2*y = 3\bound{eq2 }}
\indentrel{3}\begin{verbatim}
   (2)  2y + 2x= 3
                      Type: Equation Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{EquationXmpPageEmpty2}
\begin{paste}{EquationXmpPageEmpty2}{EquationXmpPagePatch2}
\pastebutton{EquationXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{eq2 := 2*x + 2*y = 3\bound{eq2 }}
\end{paste}\end{patch}

\begin{patch}{EquationXmpPagePatch3}
\begin{paste}{EquationXmpPageFull3}{EquationXmpPageEmpty3}
\pastebutton{EquationXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{lhs eq1\free{eq1 }}
\indentrel{3}\begin{verbatim}
   (3)  4y + 3x
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{EquationXmpPageEmpty3}
\begin{paste}{EquationXmpPageEmpty3}{EquationXmpPagePatch3}
\pastebutton{EquationXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{lhs eq1\free{eq1 }}
\end{paste}\end{patch}

\begin{patch}{EquationXmpPagePatch4}
\begin{paste}{EquationXmpPageFull4}{EquationXmpPageEmpty4}
```

```
\pastebutton{EquationXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{rhs eq1\free{eq1 }}
\indentrel{3}\begin{verbatim}
   (4)  5
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{EquationXmpPageEmpty4}
\begin{paste}{EquationXmpPageEmpty4}{EquationXmpPagePatch4}
\pastebutton{EquationXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{rhs eq1\free{eq1 }}
\end{paste}\end{patch}

\begin{patch}{EquationXmpPagePatch5}
\begin{paste}{EquationXmpPageFull5}{EquationXmpPageEmpty5}
\pastebutton{EquationXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{eq1 + eq2\free{eq1 eq2 }}
\indentrel{3}\begin{verbatim}
   (5)  6y + 5x= 8
                      Type: Equation Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{EquationXmpPageEmpty5}
\begin{paste}{EquationXmpPageEmpty5}{EquationXmpPagePatch5}
\pastebutton{EquationXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{eq1 + eq2\free{eq1 eq2 }}
\end{paste}\end{patch}

\begin{patch}{EquationXmpPagePatch6}
\begin{paste}{EquationXmpPageFull6}{EquationXmpPageEmpty6}
\pastebutton{EquationXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{eq1 * eq2\free{eq1 eq2 }}
\indentrel{3}\begin{verbatim}
          2             2
   (6)  8y  + 14x y + 6x = 15
                      Type: Equation Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{EquationXmpPageEmpty6}
\begin{paste}{EquationXmpPageEmpty6}{EquationXmpPagePatch6}
\pastebutton{EquationXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{eq1 * eq2\free{eq1 eq2 }}
\end{paste}\end{patch}

\begin{patch}{EquationXmpPagePatch7}
\begin{paste}{EquationXmpPageFull7}{EquationXmpPageEmpty7}
\pastebutton{EquationXmpPageFull7}{\hidepaste}
```

```
\tab{5}\spadcommand{2*eq2 - eq1\free{eq1 eq2 }}
\indentrel{3}\begin{verbatim}
   (7)  x= 1
                       Type: Equation Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{EquationXmpPageEmpty7}
\begin{paste}{EquationXmpPageEmpty7}{EquationXmpPagePatch7}
\pastebutton{EquationXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{2*eq2 - eq1\free{eq1 eq2 }}
\end{paste}\end{patch}

\begin{patch}{EquationXmpPagePatch8}
\begin{paste}{EquationXmpPageFull8}{EquationXmpPageEmpty8}
\pastebutton{EquationXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{eq1**2\free{eq1 }}
\indentrel{3}\begin{verbatim}
           2              2
   (8)  16y  + 24x y + 9x = 25
                       Type: Equation Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{EquationXmpPageEmpty8}
\begin{paste}{EquationXmpPageEmpty8}{EquationXmpPagePatch8}
\pastebutton{EquationXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{eq1**2\free{eq1 }}
\end{paste}\end{patch}

\begin{patch}{EquationXmpPagePatch9}
\begin{paste}{EquationXmpPageFull9}{EquationXmpPageEmpty9}
\pastebutton{EquationXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{if x+1 = y then "equal" else "unequal"}
\indentrel{3}\begin{verbatim}
   (9)  "unequal"
                                          Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{EquationXmpPageEmpty9}
\begin{paste}{EquationXmpPageEmpty9}{EquationXmpPagePatch9}
\pastebutton{EquationXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{if x+1 = y then "equal" else "unequal"}
\end{paste}\end{patch}

\begin{patch}{EquationXmpPagePatch10}
\begin{paste}{EquationXmpPageFull10}{EquationXmpPageEmpty10}
\pastebutton{EquationXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{eqpol := x+1 = y\bound{eqpol }}
```

```
\indentrel{3}\begin{verbatim}
   (10)  x + 1= y
                      Type: Equation Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{EquationXmpPageEmpty10}
\begin{paste}{EquationXmpPageEmpty10}{EquationXmpPagePatch10}
\pastebutton{EquationXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{eqpol := x+1 = y\bound{eqpol }}
\end{paste}\end{patch}

\begin{patch}{EquationXmpPagePatch11}
\begin{paste}{EquationXmpPageFull11}{EquationXmpPageEmpty11}
\pastebutton{EquationXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{if eqpol then "equal" else "unequal"\free{eqpol }}
\indentrel{3}\begin{verbatim}
   (11)  "unequal"
                                          Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{EquationXmpPageEmpty11}
\begin{paste}{EquationXmpPageEmpty11}{EquationXmpPagePatch11}
\pastebutton{EquationXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{if eqpol then "equal" else "unequal"\free{eqpol }}
\end{paste}\end{patch}

\begin{patch}{EquationXmpPagePatch12}
\begin{paste}{EquationXmpPageFull12}{EquationXmpPageEmpty12}
\pastebutton{EquationXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{eqpol::Boolean\free{eqpol }}
\indentrel{3}\begin{verbatim}
   (12)  false
                                         Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{EquationXmpPageEmpty12}
\begin{paste}{EquationXmpPageEmpty12}{EquationXmpPagePatch12}
\pastebutton{EquationXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{eqpol::Boolean\free{eqpol }}
\end{paste}\end{patch}
```

3.26 eqtbl.ht

EqTable

⇒ “notitle” (TableXmpPage) 3.106 on page 1298

— eqtbl.ht —

```
\begin{page}{EqTableXmpPage}{EqTable}
\beginscroll

The \spadtype{EqTable} domain provides tables where the keys are compared
using \spadfunFrom{eq?}{EqTable}.
Keys are considered equal only if they are the same instance of a
structure.
This is useful if the keys are themselves updatable structures.
Otherwise, all operations are the same as for type \spadtype{Table}.
See \downlink{‘Table’}{TableXmpPage}\ignore{Table}
for general information about tables.
\showBlurb{EqTable}

\xtc{
The operation \spadfunFrom{table}{EqTable} is here used to create a table
where the keys are lists of integers.
}{
\spadpaste{e: EqTable(List Integer, Integer) := table() \bound{e}}
}
\xtc{
These two lists are equal according to \spadopFrom{=}{List},
but not according to \spadfunFrom{eq?}{List}.
}{
\spadpaste{l1 := [1,2,3] \bound{l1}}
}
\xtc{
}{
\spadpaste{l2 := [1,2,3] \bound{l2}}
}
\xtc{
Because the two lists are not \spadfunFrom{eq?}{List}, separate values
can be stored under each.
}{
\spadpaste{e.l1 := 111   \free{e l1} \bound{e1}}
}
\xtc{
}{
\spadpaste{e.l2 := 222   \free{e1 l2} \bound{e2}}
}
\xtc{
}{
```

```
\spadpaste{e.l1          \free{e2 l1}}
}
\endscroll
\autobuttons
\end{page}

\begin{patch}{EqTableXmpPagePatch1}
\begin{paste}{EqTableXmpPageFull1}{EqTableXmpPageEmpty1}
\pastebutton{EqTableXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{e: EqTable(List Integer, Integer) := table()\bound{e }}
\indentrel{3}\begin{verbatim}
   (1)  table()
                    Type: EqTable(List Integer,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{EqTableXmpPageEmpty1}
\begin{paste}{EqTableXmpPageEmpty1}{EqTableXmpPagePatch1}
\pastebutton{EqTableXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{e: EqTable(List Integer, Integer) := table()\bound{e }}
\end{paste}\end{patch}

\begin{patch}{EqTableXmpPagePatch2}
\begin{paste}{EqTableXmpPageFull2}{EqTableXmpPageEmpty2}
\pastebutton{EqTableXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{l1 := [1,2,3]\bound{l1 }}
\indentrel{3}\begin{verbatim}
   (2)  [1,2,3]
                             Type: List PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{EqTableXmpPageEmpty2}
\begin{paste}{EqTableXmpPageEmpty2}{EqTableXmpPagePatch2}
\pastebutton{EqTableXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{l1 := [1,2,3]\bound{l1 }}
\end{paste}\end{patch}

\begin{patch}{EqTableXmpPagePatch3}
\begin{paste}{EqTableXmpPageFull3}{EqTableXmpPageEmpty3}
\pastebutton{EqTableXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{l2 := [1,2,3]\bound{l2 }}
\indentrel{3}\begin{verbatim}
   (3)  [1,2,3]
                             Type: List PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{EqTableXmpPageEmpty3}
\begin{paste}{EqTableXmpPageEmpty3}{EqTableXmpPagePatch3}
```

```
\pastebutton{EqTableXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{l2 := [1,2,3]\bound{l2 }}
\end{paste}\end{patch}

\begin{patch}{EqTableXmpPagePatch4}
\begin{paste}{EqTableXmpPageFull4}{EqTableXmpPageEmpty4}
\pastebutton{EqTableXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{e.l1 := 111\free{e l1 }\bound{e1 }}
\indentrel{3}\begin{verbatim}
   (4)  111
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{EqTableXmpPageEmpty4}
\begin{paste}{EqTableXmpPageEmpty4}{EqTableXmpPagePatch4}
\pastebutton{EqTableXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{e.l1 := 111\free{e l1 }\bound{e1 }}
\end{paste}\end{patch}

\begin{patch}{EqTableXmpPagePatch5}
\begin{paste}{EqTableXmpPageFull5}{EqTableXmpPageEmpty5}
\pastebutton{EqTableXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{e.l2 := 222\free{e1 l2 }\bound{e2 }}
\indentrel{3}\begin{verbatim}
   (5)  222
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{EqTableXmpPageEmpty5}
\begin{paste}{EqTableXmpPageEmpty5}{EqTableXmpPagePatch5}
\pastebutton{EqTableXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{e.l2 := 222\free{e1 l2 }\bound{e2 }}
\end{paste}\end{patch}

\begin{patch}{EqTableXmpPagePatch6}
\begin{paste}{EqTableXmpPageFull6}{EqTableXmpPageEmpty6}
\pastebutton{EqTableXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{e.l1\free{e2 l1 }}
\indentrel{3}\begin{verbatim}
   (6)  111
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{EqTableXmpPageEmpty6}
\begin{paste}{EqTableXmpPageEmpty6}{EqTableXmpPagePatch6}
\pastebutton{EqTableXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{e.l1\free{e2 l1 }}
```

```
\end{paste}\end{patch}
```

———-

3.27 evalex.ht

Example of Standard Evaluation

— evalex.ht —

```
\begin{page}{PrefixEval}{Example of Standard Evaluation}
\beginscroll
We illustrate the general evaluation of {\em op a} for some
prefix operator {\em op} and operand {\em a}
by the example: {\em cos(2)}.
The evaluation steps are as follows:
\vspace{1}\newline
1.\tab{3}{\em a} evaluates to a value of some type.
\newline\tab{3}{\em Example:} {\em 2} evaluates to {\em 2} of type
\spadtype{Integer}
\newline
2.\tab{3}Axiom then chooses a function {\em op} based on the
type of {\em a}.
\newline\tab{3}{\em Example:} The function {\em cos:} \spadtype{Float} {\em ->}
\spadtype{Float} is chosen.
\newline
3.\tab{3}If the argument type of the function is different from that
of {\em a},
then the system coerces
%\downlink{coerces}{Coercion}
the value of {\em a} to the
argument type.
\newline\tab{3}{\em Example:} The integer {\em 2} is coerced to the
float {\em 2.0}.
\newline
4.\tab{3}The function is then applied to the value of {\em a} to
produce the value
for {\em op a}.
\newline\tab{3}{\em Example:} The function {\em cos} is applied to {\em 2.0}.
\vspace{1}\newline
Try it:
\example{cos(2)}
\endscroll
\autobuttons
\end{page}
```

```
\begin{patch}{PrefixEvalPatch1}
\begin{paste}{PrefixEvalFull1}{PrefixEvalEmpty1}
\pastebutton{PrefixEvalFull1}{\hidepaste}
\tab{5}\spadcommand{cos(2)}
\indentrel{3}\begin{verbatim}
   (1)  cos(2)
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PrefixEvalEmpty1}
\begin{paste}{PrefixEvalEmpty1}{PrefixEvalPatch1}
\pastebutton{PrefixEvalEmpty1}{\showpaste}
\tab{5}\spadcommand{cos(2)}
\end{paste}\end{patch}
```

———-

Example of Standard Evaluation

— evalex.ht —

```
\begin{page}{InfixEval}{Example of Standard Evaluation}
\beginscroll
We illustrate the general evaluation of {\em a op b} for some
infix operator {\em op} with operands {\em a} and {\em b}
by the example: {\em 2 + 3.4}.
The evaluation steps are as follows:
\vspace{1}\newline
1.\tab{3}{\em a} and {\em b} are evaluated, each producing
a value and a type.
\newline\tab{3}{\em Example:} {\em 2} evaluates to {\em 2} of type
\spadtype{Integer};
{\em 3.4} evaluates to {\em 3.4} of type \spadtype{Float}.
\vspace{1}\newline
2.\tab{3}Axiom then chooses a function {\em op} based on the
types of {\em a} and {\em b}.
\newline\tab{3}{\em Example:} The function {\em +: (D,D) -> D}
is chosen requiring a common type {\em D} for both arguments to {\em +}.
An operation called {\em resolve} determines the ‘smallest common
type’ \spadtype{Float}.
\vspace{1}\newline
3.\tab{3}If the argument types for the function are different from
those of {\em a} and {\em b},
then the system coerces
%\downlink{coerces}{Coercion}
the values to the argument types.
```

```
\newline\tab{3}{\em Example:} The integer {\em 2} is coerced to the
float {\em 2.0}.
\vspace{1}\newline
4.\tab{3}The function is then applied to the values of {\em a} and {\em b}
to produce the value for {\em a op b}.
\newline\tab{3}{\em Example:} The function {\em +: (D,D) -> D}, where
{\em D} = \spadtype{Float} is applied to {\em 2.0} and {\em 3.4} to
produce {\em 5.4}.
\vspace{1}\newline
Try it:
\example{2 + 3.4}
\endscroll
\autobuttons
\end{page}

\begin{patch}{InfixEvalPatch1}
\begin{paste}{InfixEvalFull1}{InfixEvalEmpty1}
\pastebutton{InfixEvalFull1}{\hidepaste}
\tab{5}\spadcommand{2 + 3.4}
\indentrel{3}\begin{verbatim}
   (1)  5.4
                                          Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{InfixEvalEmpty1}
\begin{paste}{InfixEvalEmpty1}{InfixEvalPatch1}
\pastebutton{InfixEvalEmpty1}{\showpaste}
\tab{5}\spadcommand{2 + 3.4}
\end{paste}\end{patch}
```

3.28 exdiff.ht

Computing Derivatives

— exdiff.ht —

```
\begin{page}{ExDiffBasic}{Computing Derivatives}
\beginscroll
To compute a derivative, you must specify an expression and a variable
of differentiation.
For example, to compute the derivative of {\em sin(x) * exp(x**2)}
with respect to the variable {\em x}, issue the following command:
\spadpaste{differentiate(sin(x) * exp(x**2),x)}
```

```
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExDiffBasicPatch1}
\begin{paste}{ExDiffBasicFull1}{ExDiffBasicEmpty1}
\pastebutton{ExDiffBasicFull1}{\hidepaste}
\tab{5}\spadcommand{differentiate(sin(x) * exp(x**2),x)}
\indentrel{3}\begin{verbatim}
              2                  2
             x                  x
   (1)  2x %e  sin(x) + cos(x)%e
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExDiffBasicEmpty1}
\begin{paste}{ExDiffBasicEmpty1}{ExDiffBasicPatch1}
\pastebutton{ExDiffBasicEmpty1}{\showpaste}
\tab{5}\spadcommand{differentiate(sin(x) * exp(x**2),x)}
\end{paste}\end{patch}
```

———-

Derivatives of Functions of Several Variables

— **exdiff.ht** —

```
\begin{page}{ExDiffSeveralVariables}
{Derivatives of Functions of Several Variables}
\beginscroll
Partial derivatives are computed in the same way as derivatives of functions
of one variable: you specify the function and a variable of differentiation.
For example:
\spadpaste{differentiate(sin(x) * tan(y)/(x**2 + y**2),x)}
\spadpaste{differentiate(sin(x) * tan(y)/(x**2 + y**2),y)}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExDiffSeveralVariablesPatch1}
\begin{paste}{ExDiffSeveralVariablesFull1}{ExDiffSeveralVariablesEmpty1}
\pastebutton{ExDiffSeveralVariablesFull1}{\hidepaste}
\tab{5}\spadcommand{differentiate(sin(x) * tan(y)/(x**2 + y**2),x)}
\indentrel{3}\begin{verbatim}
                         2    2
        (- 2x sin(x) + (y  + x )cos(x))tan(y)
```

```
   (1)
                    4     2 2    4
                   y  + 2x y  + x
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExDiffSeveralVariablesEmpty1}
\begin{paste}{ExDiffSeveralVariablesEmpty1}{ExDiffSeveralVariablesPatch1}
\pastebutton{ExDiffSeveralVariablesEmpty1}{\showpaste}
\tab{5}\spadcommand{differentiate(sin(x) * tan(y)/(x**2 + y**2),x)}
\end{paste}\end{patch}

\begin{patch}{ExDiffSeveralVariablesPatch2}
\begin{paste}{ExDiffSeveralVariablesFull2}{ExDiffSeveralVariablesEmpty2}
\pastebutton{ExDiffSeveralVariablesFull2}{\hidepaste}
\tab{5}\spadcommand{differentiate(sin(x) * tan(y)/(x**2 + y**2),y)}
\indentrel{3}\begin{verbatim}
   (2)
         2    2             2
       (y  + x )sin(x)tan(y)  - 2y sin(x)tan(y)
     +
         2    2
       (y  + x )sin(x)
  /
      4     2 2    4
     y  + 2x y  + x
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExDiffSeveralVariablesEmpty2}
\begin{paste}{ExDiffSeveralVariablesEmpty2}{ExDiffSeveralVariablesPatch2}
\pastebutton{ExDiffSeveralVariablesEmpty2}{\showpaste}
\tab{5}\spadcommand{differentiate(sin(x) * tan(y)/(x**2 + y**2),y)}
\end{paste}\end{patch}
```

———

Derivatives of Higher Order

— exdiff.ht —

```
\begin{page}{ExDiffHigherOrder}{Derivatives of Higher Order}
\beginscroll
To compute a derivative of higher order (e.g. a second or third derivative),
pass the order as the third argument of the function 'differentiate'.
```

```
For example, to compute the fourth derivative of {\em exp(x**2)}, issue the
following command:
\spadpaste{differentiate(exp(x**2),x,4)}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExDiffHigherOrderPatch1}
\begin{paste}{ExDiffHigherOrderFull1}{ExDiffHigherOrderEmpty1}
\pastebutton{ExDiffHigherOrderFull1}{\hidepaste}
\tab{5}\spadcommand{differentiate(exp(x**2),x,4)}
\indentrel{3}\begin{verbatim}
                             2
            4      2        x
   (1)  (16x  + 48x  + 12)%e
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExDiffHigherOrderEmpty1}
\begin{paste}{ExDiffHigherOrderEmpty1}{ExDiffHigherOrderPatch1}
\pastebutton{ExDiffHigherOrderEmpty1}{\showpaste}
\tab{5}\spadcommand{differentiate(exp(x**2),x,4)}
\end{paste}\end{patch}
```

Multiple Derivatives I

— exdiff.ht —

```
\begin{page}{ExDiffMultipleI}{Multiple Derivatives I}
\beginscroll
When given a function of several variables, you may take derivatives
repeatedly and with respect to different variables.
The following command differentiates the function
{\em sin(x)/(x**2 + y**2)} first with respect to {\em x} and then with
respect to {\em y}:
\spadpaste{differentiate(sin(x)/(x**2 + y**2),[x,y])}
As you can see, we first specify the function and then a list of
the variables of differentiation.
Variables may appear on the list more than once.
For example, the following command differentiates the same function with
respect to {\em x} and then twice with respect to {\em y}.
\spadpaste{differentiate(sin(x)/(x**2 + y**2),[x,y,y])}
\endscroll
\autobuttons
```

```
\end{page}

\begin{patch}{ExDiffMultipleIPatch1}
\begin{paste}{ExDiffMultipleIFull1}{ExDiffMultipleIEmpty1}
\pastebutton{ExDiffMultipleIFull1}{\hidepaste}
\tab{5}\spadcommand{differentiate(sin(x)/(x**2 + y**2),[x,y])}
\indentrel{3}\begin{verbatim}
                          3     2
        8x y sin(x) + (- 2y  - 2x y)cos(x)
   (1)
               6     2 4     4 2    6
              y  + 3x y  + 3x y  + x
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExDiffMultipleIEmpty1}
\begin{paste}{ExDiffMultipleIEmpty1}{ExDiffMultipleIPatch1}
\pastebutton{ExDiffMultipleIEmpty1}{\showpaste}
\tab{5}\spadcommand{differentiate(sin(x)/(x**2 + y**2),[x,y])}
\end{paste}\end{patch}

\begin{patch}{ExDiffMultipleIPatch2}
\begin{paste}{ExDiffMultipleIFull2}{ExDiffMultipleIEmpty2}
\pastebutton{ExDiffMultipleIFull2}{\hidepaste}
\tab{5}\spadcommand{differentiate(sin(x)/(x**2 + y**2),[x,y,y])}
\indentrel{3}\begin{verbatim}
   (2)
           2     3             4     2 2     4
   (- 40x y  + 8x )sin(x) + (6y  + 4x y  - 2x )cos(x)

              8     2 6     4 4     6 2    8
             y  + 4x y  + 6x y  + 4x y  + x
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExDiffMultipleIEmpty2}
\begin{paste}{ExDiffMultipleIEmpty2}{ExDiffMultipleIPatch2}
\pastebutton{ExDiffMultipleIEmpty2}{\showpaste}
\tab{5}\spadcommand{differentiate(sin(x)/(x**2 + y**2),[x,y,y])}
\end{paste}\end{patch}
```

Multiple Derivatives II

— exdiff.ht —

```
\begin{page}{ExDiffMultipleII}{Multiple Derivatives II}
\beginscroll
You may also compute multiple derivatives by specifying a list of
variables together with a list of multiplicities.
For example, to differentiate {\em cos(z)/(x**2 + y**3)}
first with respect to {\em x}, then twice with respect to {\em y},
then three times with respect to {\em z},
issue the following command:
\spadpaste{differentiate(cos(z)/(x**2 + y**3),[x,y,z],[1,2,3])}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExDiffMultipleIIPatch1}
\begin{paste}{ExDiffMultipleIIFull1}{ExDiffMultipleIIEmpty1}
\pastebutton{ExDiffMultipleIIFull1}{\hidepaste}
\tab{5}\spadcommand{differentiate(cos(z)/(x**2 + y**3),[x,y,z],[1,2,3])}
\indentrel{3}\begin{verbatim}
                   4      3
           (- 84x y  + 24x y)sin(z)
   (1)
        12     2 9     4 6     6 3    8
       y   + 4x y  + 6x y  + 4x y  + x
                              Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExDiffMultipleIIEmpty1}
\begin{paste}{ExDiffMultipleIIEmpty1}{ExDiffMultipleIIPatch1}
\pastebutton{ExDiffMultipleIIEmpty1}{\showpaste}
\tab{5}\spadcommand{differentiate(cos(z)/(x**2 + y**3),[x,y,z],[1,2,3])}
\end{paste}\end{patch}
```

———-

Derivatives of Functions Involving Formal Integrals

— exdiff.ht —

```
\begin{page}{ExDiffFormalIntegral}
{Derivatives of Functions Involving Formal Integrals}
```

```
\beginscroll
When a function does not have a closed-form antiderivative, Axiom
returns a formal integral.
A typical example is
\spadpaste{f := integrate(sqrt(1 + t**3),t) \bound{f}}
This formal integral may be differentiated, either by itself or in any
combination with other functions:
\spadpaste{differentiate(f,t) \free{f}}
\spadpaste{differentiate(f * t**2,t) \free{f}}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExDiffFormalIntegralPatch1}
\begin{paste}{ExDiffFormalIntegralFull1}{ExDiffFormalIntegralEmpty1}
\pastebutton{ExDiffFormalIntegralFull1}{\hidepaste}
\tab{5}\spadcommand{f := integrate(sqrt(1 + t**3),t)\bound{f }}
\indentrel{3}\begin{verbatim}
           t
             3
   (1)      \%N  + 1 d%N

                    Type: Union(Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExDiffFormalIntegralEmpty1}
\begin{paste}{ExDiffFormalIntegralEmpty1}{ExDiffFormalIntegralPatch1}
\pastebutton{ExDiffFormalIntegralEmpty1}{\showpaste}
\tab{5}\spadcommand{f := integrate(sqrt(1 + t**3),t)\bound{f }}
\end{paste}\end{patch}

\begin{patch}{ExDiffFormalIntegralPatch2}
\begin{paste}{ExDiffFormalIntegralFull2}{ExDiffFormalIntegralEmpty2}
\pastebutton{ExDiffFormalIntegralFull2}{\hidepaste}
\tab{5}\spadcommand{differentiate(f,t)\free{f }}
\indentrel{3}\begin{verbatim}

          3
   (2)  \t  + 1
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExDiffFormalIntegralEmpty2}
\begin{paste}{ExDiffFormalIntegralEmpty2}{ExDiffFormalIntegralPatch2}
\pastebutton{ExDiffFormalIntegralEmpty2}{\showpaste}
\tab{5}\spadcommand{differentiate(f,t)\free{f }}
\end{paste}\end{patch}
```

```
\begin{patch}{ExDiffFormalIntegralPatch3}
\begin{paste}{ExDiffFormalIntegralFull3}{ExDiffFormalIntegralEmpty3}
\pastebutton{ExDiffFormalIntegralFull3}{\hidepaste}
\tab{5}\spadcommand{differentiate(f * t**2,t)\free{f }}
\indentrel{3}\begin{verbatim}
             t
               3              2  3
   (3)  2t    \%N  + 1 d%N  + t \t  + 1

                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExDiffFormalIntegralEmpty3}
\begin{paste}{ExDiffFormalIntegralEmpty3}{ExDiffFormalIntegralPatch3}
\pastebutton{ExDiffFormalIntegralEmpty3}{\showpaste}
\tab{5}\spadcommand{differentiate(f * t**2,t)\free{f }}
\end{paste}\end{patch}
```

———-

Exit

— exit.ht —

```
\begin{page}{ExitXmpPage}{Exit}
\beginscroll

A function that does not return directly to its caller has
\spadtype{Exit} as its return type.
The operation \spadfun{error} is an example of one which does not return
to its caller.
Instead, it causes a return to top-level.
\xtc{
}{
\spadpaste{n := 0 \bound{n}}
}
\xtc{
The function \userfun{gasp} is given return type \spadtype{Exit} since
it is guaranteed never to return a value to its caller.
}{
\begin{spadsrc}[\bound{gasp}\free{n}]
gasp(): Exit ==
    free n
    n := n + 1
    error "Oh no!"
\end{spadsrc}
```

```
}
\xtc{
The return type of \userfun{half} is determined by resolving the
types of the two branches of the \spad{if}.
}{
\begin{spadsrc}[\bound{half}\free{gasp}]
half(k) ==
  if odd? k then gasp()
  else k quo 2
\end{spadsrc}
}
\xtc{
Because
\userfun{gasp} has the return type \spadtype{Exit}, the type of
\spad{if} in \userfun{half} is resolved to be \spadtype{Integer}.
}{
\spadpaste{half 4 \free{half}\bound{app1}}
}
\xtc{
}{
\spadpaste{half 3 \free{half app1}\bound{app2}}
}
\xtc{
}{
\spadpaste{n \free{app2}}
}

For functions which return no value at all, use \spadtype{Void}.
See \downlink{``User-Defined Functions, Macros and Rules''}
{ugUserPage} in Section 6\ignore{ugUser} and
\downlink{`Void'}{VoidXmpPage}\ignore{Void} for more information.
%
\showBlurb{Exit}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExitXmpPagePatch1}
\begin{paste}{ExitXmpPageFull1}{ExitXmpPageEmpty1}
\pastebutton{ExitXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{n := 0\bound{n }}
\indentrel{3}\begin{verbatim}
   (1)  0
                              Type: NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExitXmpPageEmpty1}
\begin{paste}{ExitXmpPageEmpty1}{ExitXmpPagePatch1}
\pastebutton{ExitXmpPageEmpty1}{\showpaste}
```

```
\tab{5}\spadcommand{n := 0\bound{n }}
\end{paste}\end{patch}

\begin{patch}{ExitXmpPagePatch2}
\begin{paste}{ExitXmpPageFull2}{ExitXmpPageEmpty2}
\pastebutton{ExitXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{gasp(): Exit ==
    free n
    n := n + 1
    error "Oh no!"
\bound{gasp }\free{n }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExitXmpPageEmpty2}
\begin{paste}{ExitXmpPageEmpty2}{ExitXmpPagePatch2}
\pastebutton{ExitXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{gasp(): Exit ==
    free n
    n := n + 1
    error "Oh no!"
\bound{gasp }\free{n }}
\end{paste}\end{patch}

\begin{patch}{ExitXmpPagePatch3}
\begin{paste}{ExitXmpPageFull3}{ExitXmpPageEmpty3}
\pastebutton{ExitXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{half(k) ==
  if odd? k then gasp()
  else k quo 2
\bound{half }\free{gasp }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExitXmpPageEmpty3}
\begin{paste}{ExitXmpPageEmpty3}{ExitXmpPagePatch3}
\pastebutton{ExitXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{half(k) ==
  if odd? k then gasp()
  else k quo 2
\bound{half }\free{gasp }}
\end{paste}\end{patch}

\begin{patch}{ExitXmpPagePatch4}
\begin{paste}{ExitXmpPageFull4}{ExitXmpPageEmpty4}
\pastebutton{ExitXmpPageFull4}{\hidepaste}
```

```
\tab{5}\spadcommand{half 4\free{half }\bound{app1 }}
\indentrel{3}\begin{verbatim}
   (4)  2
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExitXmpPageEmpty4}
\begin{paste}{ExitXmpPageEmpty4}{ExitXmpPagePatch4}
\pastebutton{ExitXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{half 4\free{half }\bound{app1 }}
\end{paste}\end{patch}

\begin{patch}{ExitXmpPagePatch5}
\begin{paste}{ExitXmpPageFull5}{ExitXmpPageEmpty5}
\pastebutton{ExitXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{half 3\free{half app1 }\bound{app2 }}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExitXmpPageEmpty5}
\begin{paste}{ExitXmpPageEmpty5}{ExitXmpPagePatch5}
\pastebutton{ExitXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{half 3\free{half app1 }\bound{app2 }}
\end{paste}\end{patch}

\begin{patch}{ExitXmpPagePatch6}
\begin{paste}{ExitXmpPageFull6}{ExitXmpPageEmpty6}
\pastebutton{ExitXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{n\free{app2 }}
\indentrel{3}\begin{verbatim}
   (5)  1
                               Type: NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExitXmpPageEmpty6}
\begin{paste}{ExitXmpPageEmpty6}{ExitXmpPagePatch6}
\pastebutton{ExitXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{n\free{app2 }}
\end{paste}\end{patch}
```

———-

3.29 exlap.ht

Laplace transform with a single pole

— exlap.ht —

```
\begin{page}{ExLapSimplePole}{Laplace transform with a single pole}
\beginscroll
The Laplace transform of t^n e^(a t) has a pole of order n+1 at x = a
and no other pole. We divide by n! to get a monic denominator in the
answer.
\spadpaste{laplace(t**4 * exp(-a*t) / factorial(4), t, s)}
\endscroll
\autobuttons\end{page}

\begin{patch}{ExLapSimplePolePatch1}
\begin{paste}{ExLapSimplePoleFull1}{ExLapSimplePoleEmpty1}
\pastebutton{ExLapSimplePoleFull1}{\hidepaste}
\tab{5}\spadcommand{laplace(t**4 * exp(-a*t) / factorial(4), t, s)}
\indentrel{3}\begin{verbatim}
                            1
   (1)
         5       4      2 3      3 2     4     5
        s  + 5a s  + 10a s  + 10a s  + 5a s + a
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExLapSimplePoleEmpty1}
\begin{paste}{ExLapSimplePoleEmpty1}{ExLapSimplePolePatch1}
\pastebutton{ExLapSimplePoleEmpty1}{\showpaste}
\tab{5}\spadcommand{laplace(t**4 * exp(-a*t) / factorial(4), t, s)}
\end{paste}\end{patch}
```

————

Laplace transform of a trigonometric function

— exlap.ht —

```
\begin{page}{ExLapTrigTrigh}{Laplace transform of a trigonometric function}
\beginscroll
Rather than looking up into a table, we use the normalizer to rewrite
the trigs and hyperbolic trigs to complex exponentials and
logarithms.
```

```
\spadpaste{laplace(sin(a*t) * cosh(a*t) - cos(a*t) * sinh(a*t), t, s)}
\endscroll
\autobuttons\end{page}

\begin{patch}{ExLapTrigTrighPatch1}
\begin{paste}{ExLapTrigTrighFull1}{ExLapTrigTrighEmpty1}
\pastebutton{ExLapTrigTrighFull1}{\hidepaste}
\tab{5}\spadcommand{laplace(sin(a*t) * cosh(a*t) - cos(a*t) * sinh(a*t), t, s)}
\indentrel{3}\begin{verbatim}
             3
           4a
   (1)
         4     4
        s  + 4a
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExLapTrigTrighEmpty1}
\begin{paste}{ExLapTrigTrighEmpty1}{ExLapTrigTrighPatch1}
\pastebutton{ExLapTrigTrighEmpty1}{\showpaste}
\tab{5}\spadcommand{laplace(sin(a*t) * cosh(a*t) - cos(a*t) * sinh(a*t), t, s)}
\end{paste}\end{patch}
```

———-

Laplace transform requiring a definite integration

— exlap.ht —

```
\begin{page}{ExLapDefInt}
{Laplace transform requiring a definite integration}
\beginscroll
When powers of t appear in the denominator, computing the laplace
transform requires integrating the result of another laplace
transform between a symbol and infinity. We use the full power of
Axiom's integrator in such cases.
\spadpaste{laplace(2/t * (1 - cos(a*t)), t, s)}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExLapDefIntPatch1}
\begin{paste}{ExLapDefIntFull1}{ExLapDefIntEmpty1}
\pastebutton{ExLapDefIntFull1}{\hidepaste}
\tab{5}\spadcommand{laplace(2/t * (1 - cos(a*t)), t, s)}
\indentrel{3}\begin{verbatim}
```

```
            2    2
   (1)  log(s  + a ) - 2log(s)
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExLapDefIntEmpty1}
\begin{paste}{ExLapDefIntEmpty1}{ExLapDefIntPatch1}
\pastebutton{ExLapDefIntEmpty1}{\showpaste}
\tab{5}\spadcommand{laplace(2/t * (1 - cos(a*t)), t, s)}
\end{paste}\end{patch}
```

———

Laplace transform of exponentials

— **exlap.ht** —

```
\begin{page}{ExLapExpExp}{Laplace transform of exponentials}
\beginscroll
This is another example where it is necessary to
integrate the result of another laplace transform.
\spadpaste{laplace((exp(a*t) - exp(b*t))/t, t, s)}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExLapExpExpPatch1}
\begin{paste}{ExLapExpExpFull1}{ExLapExpExpEmpty1}
\pastebutton{ExLapExpExpFull1}{\hidepaste}
\tab{5}\spadcommand{laplace((exp(a*t) - exp(b*t))/t, t, s)}
\indentrel{3}\begin{verbatim}
   (1)  - log(s - a) + log(s - b)
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExLapExpExpEmpty1}
\begin{paste}{ExLapExpExpEmpty1}{ExLapExpExpPatch1}
\pastebutton{ExLapExpExpEmpty1}{\showpaste}
\tab{5}\spadcommand{laplace((exp(a*t) - exp(b*t))/t, t, s)}
\end{paste}\end{patch}
```

———

Laplace transform of an exponential integral

— **exlap.ht** —

```
\begin{page}{ExLapSpecial1}
{Laplace transform of an exponential integral}
We can handle some restricted cases of special functions,
linear exponential integrals among them.
\beginscroll
\spadpaste{laplace(exp(a*t+b)*Ei(c*t), t, s)}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExLapSpecial1Patch1}
\begin{paste}{ExLapSpecial1Full1}{ExLapSpecial1Empty1}
\pastebutton{ExLapSpecial1Full1}{\hidepaste}
\tab{5}\spadcommand{laplace(exp(a*t+b)*Ei(c*t), t, s)}
\indentrel{3}\begin{verbatim}
          b    s + c - a
        %e log()
                   c
   (1)
              s - a
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExLapSpecial1Empty1}
\begin{paste}{ExLapSpecial1Empty1}{ExLapSpecial1Patch1}
\pastebutton{ExLapSpecial1Empty1}{\showpaste}
\tab{5}\spadcommand{laplace(exp(a*t+b)*Ei(c*t), t, s)}
\end{paste}\end{patch}

\begin{patch}{ExLapSpecial2Patch1}
\begin{paste}{ExLapSpecial2Full1}{ExLapSpecial2Empty1}
\pastebutton{ExLapSpecial2Full1}{\hidepaste}
\tab{5}\spadcommand{laplace(a*Ci(b*t) + c*Si(d*t), t, s)}
\indentrel{3}\begin{verbatim}
               2    2
              s  + b            d
        a log() + 2c atan()
                  2             s
                 b
   (1)
                    2s
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ExLapSpecial2Empty1}
\begin{paste}{ExLapSpecial2Empty1}{ExLapSpecial2Patch1}
\pastebutton{ExLapSpecial2Empty1}{\showpaste}
\tab{5}\spadcommand{laplace(a*Ci(b*t) + c*Si(d*t), t, s)}
\end{paste}\end{patch}
```

———-

Laplace transform of special functions

— exlap.ht —

```
\begin{page}{ExLapSpecial2}{Laplace transform of special functions}
\beginscroll
An example with some interesting special functions.
\spadpaste{laplace(a*Ci(b*t) + c*Si(d*t), t, s)}
\endscroll
\autobuttons\end{page}
```

———-

3.30 exint.ht

Integral of a Rational Function

— exint.ht —

```
\begin{page}{ExIntRationalFunction}{Integral of a Rational Function}
\beginscroll
The following fraction has a denominator which factors into
a quadratic and a quartic irreducible polynomial. The usual
partial fraction approach used by most other computer algebra
systems either fails or introduces expensive unneeded algebraic
numbers.
We use a factorization-free algorithm.
\spadpaste{integrate((x**2+2*x+1)/((x+1)**6+1),x)}
There are cases where algebraic numbers are absolutely necessary.
In that case the answer to the \spadfun{integrate} command will contain
symbols like \spad{\%\%F0} denoting the algebraic numbers used.
To find out what the definitions for these numbers is use the
\spadfun{definingPolynomial} operation on these numbers.
```

```
\spadpaste{integrate(1/(x**3+x+1),x) \bound{i}}
For example, if a symbol like \spad{\%\%F0} appears in the result of this last
integration, then \spad{definingPolynomial \%\%F0} will return the
polynomial that \spad{\%\%F0} is a root of. The next command isolates the
algebraic number from the expression and displays its defining polynomial.
\spadpaste{definingPolynomial(tower(\%).2::EXPR INT) \free{i}}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ExIntRationalFunctionPatch1}
\begin{paste}{ExIntRationalFunctionFull1}{ExIntRationalFunctionEmpty1}
\pastebutton{ExIntRationalFunctionFull1}{\hidepaste}
\tab{5}\spadcommand{integrate((x**2+2*x+1)/((x+1)**6+1),x)}
\indentrel{3}\begin{verbatim}
              3     2
        atan(x  + 3x  + 3x + 1)
   (1)
                   3
                    Type: Union(Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExIntRationalFunctionEmpty1}
\begin{paste}{ExIntRationalFunctionEmpty1}{ExIntRationalFunctionPatch1}
\pastebutton{ExIntRationalFunctionEmpty1}{\showpaste}
\tab{5}\spadcommand{integrate((x**2+2*x+1)/((x+1)**6+1),x)}
\end{paste}\end{patch}

\begin{patch}{ExIntRationalFunctionPatch2}
\begin{paste}{ExIntRationalFunctionFull2}{ExIntRationalFunctionEmpty2}
\pastebutton{ExIntRationalFunctionFull2}{\hidepaste}
\tab{5}\spadcommand{integrate(1/(x**3+x+1),x)\bound{i }}
\indentrel{3}\begin{verbatim}
   (2)

                   2
          - 93%%H0  + 12
        (   - %%H0)
         \      31
      *
         log

                                  2
                        - 93%%H0  + 12          2
               (62%%H0 + 31)    + 62%%H0
                            \      31
            +
               - 31%%H0 + 18x - 4
```

```
     +
                          2
             - 93%%H0  + 12
         (-     - %%H0)
            \      31
      *
         log

                                         2
                               - 93%%H0  + 12          2
              (- 62%%H0 - 31)   + 62%%H0
                              \      31
            +
              - 31%%H0 + 18x - 4
     +
                          2
       2%%H0 log(- 62%%H0  + 31%%H0 + 9x + 4)
  /
     2
                    Type: Union(Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExIntRationalFunctionEmpty2}
\begin{paste}{ExIntRationalFunctionEmpty2}{ExIntRationalFunctionPatch2}
\pastebutton{ExIntRationalFunctionEmpty2}{\showpaste}
\tab{5}\spadcommand{integrate(1/(x**3+x+1),x)\bound{i }}
\end{paste}\end{patch}

\begin{patch}{ExIntRationalFunctionPatch3}
\begin{paste}{ExIntRationalFunctionFull3}{ExIntRationalFunctionEmpty3}
\pastebutton{ExIntRationalFunctionFull3}{\hidepaste}
\tab{5}\spadcommand{definingPolynomial(tower(\%).2::EXPR INT)\free{i }}
\indentrel{3}\begin{verbatim}
              3
        31%%H0  - 3%%H0 - 1
   (3)
                 31
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExIntRationalFunctionEmpty3}
\begin{paste}{ExIntRationalFunctionEmpty3}{ExIntRationalFunctionPatch3}
\pastebutton{ExIntRationalFunctionEmpty3}{\showpaste}
\tab{5}\spadcommand{definingPolynomial(tower(\%).2::EXPR INT)\free{i }}
\end{paste}\end{patch}
```

———-

Integral of a Rational Function with a Real Parameter

— exint.ht —

```
\begin{page}{ExIntRationalWithRealParameter}
{Integral of a Rational Function with a Real Parameter}
\beginscroll
When real parameters are present, the form of the integral can depend on
the signs of some expressions. Rather than query the user or make
sign assumptions, Axiom returns all possible answers.
\spadpaste{integrate(1/(x**2 + a),x)}
The integrate command generally assumes that all parameters are real.
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExIntRationalWithRealParameterPatch1}
\begin{paste}{ExIntRationalWithRealParameterFull1}{ExIntRationalWithRealParameterEmpty1}
\pastebutton{ExIntRationalWithRealParameterFull1}{\hidepaste}
\tab{5}\spadcommand{integrate(1/(x**2 + a),x)}
\indentrel{3}\begin{verbatim}
              2
            (x  - a)\- a  + 2a x
        log()      x\a
                     2            atan()
                    x  + a               a
   (1)  [,]

                  2\- a              \a
              Type: Union(List Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExIntRationalWithRealParameterEmpty1}
\begin{paste}{ExIntRationalWithRealParameterEmpty1}{ExIntRationalWithRealParameterPatch1}
\pastebutton{ExIntRationalWithRealParameterEmpty1}{\showpaste}
\tab{5}\spadcommand{integrate(1/(x**2 + a),x)}
\end{paste}\end{patch}
```

———-

Integral of a Rational Function with a Complex Parameter

— **exint.ht** —

```
\begin{page}{ExIntRationalWithComplexParameter}
{Integral of a Rational Function with a Complex Parameter}
\beginscroll
If the parameter is complex instead of real, then the notion of
sign is undefined and there is a unique answer.
You can request
this answer by prepending the word ‘complex’ to the command name:
\spadpaste{complexIntegrate(1/(x**2 + a),x)}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExIntRationalWithComplexParameterPatch1}
\begin{paste}{ExIntRationalWithComplexParameterFull1}{ExIntRationalWithComplexParameterEmpty1}
\pastebutton{ExIntRationalWithComplexParameterFull1}{\hidepaste}
\tab{5}\spadcommand{complexIntegrate(1/(x**2 + a),x)}
\indentrel{3}\begin{verbatim}

           x\- a  + a        x\- a  - a
       log() - log()

              \- a              \- a
   (1)

                     2\- a
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExIntRationalWithComplexParameterEmpty1}
\begin{paste}{ExIntRationalWithComplexParameterEmpty1}{ExIntRationalWithComplexParameterPatch1}
\pastebutton{ExIntRationalWithComplexParameterEmpty1}{\showpaste}
\tab{5}\spadcommand{complexIntegrate(1/(x**2 + a),x)}
\end{paste}\end{patch}
```

———

Two Similar Integrands Producing Very Different Results

— **exint.ht** —

```
\begin{page}{ExIntTwoSimilarIntegrands}
```

```
{Two Similar Integrands Producing Very Different Results}
\beginscroll
The following two examples illustrate the limitations of table based
approaches. The two integrands are very similar, but the answer to one
of them requires the addition of two new algebraic numbers.
This one is the easy one:
\spadpaste{integrate(x**3 / (a+b*x)**(1/3),x)}
The next one looks very similar
but the answer is much more complicated. Only an algorithmic approach
is guaranteed to find what new constants must be added in order to
find a solution:
\spadpaste{integrate(1 / (x**3 * (a+b*x)**(1/3)),x)}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExIntTwoSimilarIntegrandsPatch1}
\begin{paste}{ExIntTwoSimilarIntegrandsFull1}{ExIntTwoSimilarIntegrandsEmpty1}
\pastebutton{ExIntTwoSimilarIntegrandsFull1}{\hidepaste}
\tab{5}\spadcommand{integrate(x**3 / (a+b*x)**(1/3),x)}
\indentrel{3}\begin{verbatim}
   (1)
        3 3         2 2       2          3 32
   (120b x  - 135a b x  + 162a b x - 243a )\b x + a

                              4
                          440b
                    Type: Union(Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExIntTwoSimilarIntegrandsEmpty1}
\begin{paste}{ExIntTwoSimilarIntegrandsEmpty1}{ExIntTwoSimilarIntegrandsPatch1}
\pastebutton{ExIntTwoSimilarIntegrandsEmpty1}{\showpaste}
\tab{5}\spadcommand{integrate(x**3 / (a+b*x)**(1/3),x)}
\end{paste}\end{patch}

\begin{patch}{ExIntTwoSimilarIntegrandsPatch2}
\begin{paste}{ExIntTwoSimilarIntegrandsFull2}{ExIntTwoSimilarIntegrandsEmpty2}
\pastebutton{ExIntTwoSimilarIntegrandsFull2}{\hidepaste}
\tab{5}\spadcommand{integrate(1 / (x**3 * (a+b*x)**(1/3)),x)}
\indentrel{3}\begin{verbatim}
   (2)
       -
             2 2
           2b x \3
         *
               332   32 3
           log(\a \b x + a   + \a   \b x + a + a)
     +
```

```
         2 2     32 3
       4b x \3 log(\a   \b x + a - a)
     +
                     32 3
          2 2     2\3 \a   \b x + a + a\3
       12b x atan()
                                3a
     +
                     332
       (12b x - 9a)\3 \a \b x + a
  /
        2 2 3
     18a x \3 \a
                    Type: Union(Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExIntTwoSimilarIntegrandsEmpty2}
\begin{paste}{ExIntTwoSimilarIntegrandsEmpty2}{ExIntTwoSimilarIntegrandsPatch2}
\pastebutton{ExIntTwoSimilarIntegrandsEmpty2}{\showpaste}
\tab{5}\spadcommand{integrate(1 / (x**3 * (a+b*x)**(1/3)),x)}
\end{paste}\end{patch}
```

————-

An Integral Which Does Not Exist

— exint.ht —

```
\begin{page}{ExIntNoSolution}{An Integral Which Does Not Exist}
\beginscroll
Most computer algebra systems use heuristics or table-driven
approaches to integration. When these systems cannot determine
the answer to an integration problem, they reply "I don't know".
Axiom uses a complete algorithm for integration.
It will conclusively prove that an integral
cannot be expressed in terms of elementary functions.
When Axiom returns an integral sign, it has proved
that no answer exists as an elementary function.
\spadpaste{integrate(log(1 + sqrt(a*x + b)) / x,x)}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExIntNoSolutionPatch1}
\begin{paste}{ExIntNoSolutionFull1}{ExIntNoSolutionEmpty1}
\pastebutton{ExIntNoSolutionFull1}{\hidepaste}
```

```
\tab{5}\spadcommand{integrate(log(1 + sqrt(a*x + b)) / x,x)}
\indentrel{3}\begin{verbatim}
           x
           log(\b + %T a  + 1)
   (1)       d%T
                  %T
                  Type: Union(Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExIntNoSolutionEmpty1}
\begin{paste}{ExIntNoSolutionEmpty1}{ExIntNoSolutionPatch1}
\pastebutton{ExIntNoSolutionEmpty1}{\showpaste}
\tab{5}\spadcommand{integrate(log(1 + sqrt(a*x + b)) / x,x)}
\end{paste}\end{patch}
```

———-

```
%% This example is broken
%\begin{page}{ExIntChangeOfVariables}{No Change of Variables is Required}
%\beginscroll
%Unlike computer algebra systems which rely on heuristics and
%table-lookup, the algorithmic integration facility
%of Axiom never requires you to make a change of variables
%in order to integrate a function.
%\spadpaste{integrate(sec(x)**(4/5)*csc(x)**(6/5),x)}
%\endscroll
%\autobuttons\end{page}
```

A Trigonometric Function of a Quadratic

— exint.ht —

```
\begin{page}{ExIntTrig}{A Trigonometric Function of a Quadratic}
\beginscroll
Axiom can handle complicated mixed functions way beyond what you can
find in tables:
\spadpaste{integrate((sinh(1+sqrt(x+b))+2*sqrt(x+b))/
(sqrt(x+b)*(x+cosh(1+sqrt(x+b)))),x)}
Whenever possible, Axiom tries to express the answer using the functions
present in the integrand.
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExIntTrigPatch1}
\begin{paste}{ExIntTrigFull1}{ExIntTrigEmpty1}
```

```
\pastebutton{ExIntTrigFull1}{\hidepaste}
\tab{5}\spadcommand{integrate((sinh(1+sqrt(x+b))+2*sqrt(x+b))/(sqrt(x+b)*(x+cosh(1+sqrt(x+b)))),
\indentrel{3}\begin{verbatim}
   (1)

                - 2cosh(\x + b  + 1) - 2x
     2log()

          sinh(\x + b  + 1) - cosh(\x + b  + 1)
   +

     - 2\x + b
                   Type: Union(Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExIntTrigEmpty1}
\begin{paste}{ExIntTrigEmpty1}{ExIntTrigPatch1}
\pastebutton{ExIntTrigEmpty1}{\showpaste}
\tab{5}\spadcommand{integrate((sinh(1+sqrt(x+b))+2*sqrt(x+b))/(sqrt(x+b)*(x+cosh(1+sqrt(x+b)))),
\end{paste}
\end{patch}
```

———-

Integrating a Function with a Hidden Algebraic Relation

⇒ “notitle” (ExIntAlgebraicRelationExplain) 3.30 on page 415

— **exint.ht** —

```
\begin{page}{ExIntAlgebraicRelation}
{Integrating a Function with a Hidden Algebraic Relation}
\beginscroll
A strong structure checking algorithm in Axiom finds hidden algebraic
relationships between functions.
\spadpaste{integrate(tan(atan(x)/3),x)}
The discovery of this algebraic relationship is necessary
for correctly integrating this function.
\downlink{Details.}{ExIntAlgebraicRelationExplain} \space{1}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExIntAlgebraicRelationPatch1}
\begin{paste}{ExIntAlgebraicRelationFull1}{ExIntAlgebraicRelationEmpty1}
\pastebutton{ExIntAlgebraicRelationFull1}{\hidepaste}
\tab{5}\spadcommand{integrate(tan(atan(x)/3),x)}
```

```
\indentrel{3}\begin{verbatim}
   (1)
                 atan(x) 2             atan(x) 2
       8log(3tan()  - 1) - 3tan()
                    3                     3
     +
               atan(x)
       18x tan()
                  3
  /
     18
                    Type: Union(Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExIntAlgebraicRelationEmpty1}
\begin{paste}{ExIntAlgebraicRelationEmpty1}{ExIntAlgebraicRelationPatch1}
\pastebutton{ExIntAlgebraicRelationEmpty1}{\showpaste}
\tab{5}\spadcommand{integrate(tan(atan(x)/3),x)}
\end{paste}\end{patch}
```

———–

Details for integrating a function with a Hidden Algebraic Relation

— exint.ht —

```
\begin{page}{ExIntAlgebraicRelationExplain}
{Details for integrating a function with a Hidden Algebraic Relation}
\beginscroll
Steps taken for integration of:
\centerline{{\em f := tan(atan(x)/3)}}
\beginitems
\item
1. Replace {\em f} by an equivalent algebraic function {\em g}
satisfying the algebraic relation:
\centerline{{\em g**3 - 3*x*g - 3*g + x = 0}}
\item
2. Integrate {\em g} using using this algebraic relation; this produces:
\centerline{{\em (24g**2 - 8)log(3g**2 - 1) + (81x**2 + 24)g**2 + 72xg - 27x**2 - 16}}
\centerline{{\em /   (54g**2 - 18)}}
\item
3. Rationalize the denominator, producing:
\centerline{{\em (8log(3g**2-1) - 3g**2 + 18xg + 15)/18}}
\item
4. Replace {\em g} by the initial {\em f} to produce the final result:
\centerline{{\em (8log(3tan(atan(x/3))**2-1) - 3tan(atan(x/3))**2 -
```

```
18xtan(atan(x/3) + 16)/18)}}
\enditems
\endscroll
\autobuttons\end{page}
```

———-

An Integral Involving a Root of a Transcendental Function

— exint.ht —

```
\begin{page}{ExIntRadicalOfTranscendental}
{An Integral Involving a Root of a Transcendental Function}
\beginscroll
This is an example of a mixed function where
the algebraic layer is over the transcendental one.
\spadpaste{integrate((x + 1) / (x * (x + log x)**(3/2)),x)}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExIntRadicalOfTranscendentalPatch1}
\begin{paste}{ExIntRadicalOfTranscendentalFull1}{ExIntRadicalOfTranscendentalEmpty1}
\pastebutton{ExIntRadicalOfTranscendentalFull1}{\hidepaste}
\tab{5}\spadcommand{integrate((x + 1) / (x * (x + log x)**(3/2)),x)}
\indentrel{3}\begin{verbatim}

          2\log(x) + x
   (1)  -
            log(x) + x
                    Type: Union(Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExIntRadicalOfTranscendentalEmpty1}
\begin{paste}{ExIntRadicalOfTranscendentalEmpty1}{ExIntRadicalOfTranscendentalPatch1}
\pastebutton{ExIntRadicalOfTranscendentalEmpty1}{\showpaste}
\tab{5}\spadcommand{integrate((x + 1) / (x * (x + log x)**(3/2)),x)}
\end{paste}\end{patch}
```

———-

An Integral of a Non-elementary Function

— **exint.ht** —

```
\begin{page}{ExIntNonElementary}{An Integral of a Non-elementary Function}
\beginscroll
While incomplete for non-elementary functions, Axiom can
handle some of them:
\spadpaste{integrate(exp(-x**2) * erf(x) /
(erf(x)**3 - erf(x)**2 - erf(x) + 1),x)}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExIntNonElementaryPatch1}
\begin{paste}{ExIntNonElementaryFull1}{ExIntNonElementaryEmpty1}
\pastebutton{ExIntNonElementaryFull1}{\hidepaste}
\tab{5}\spadcommand{integrate(exp(-x**2) * erf(x) / (erf(x)**3 - erf(x)**2 - erf(x) + 1),x)}
\indentrel{3}\begin{verbatim}
                         erf(x) - 1
        (erf(x) - 1)\%pi log() - 2\%pi
                              erf(x) + 1
   (1)
                        8erf(x) - 8
                    Type: Union(Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExIntNonElementaryEmpty1}
\begin{paste}{ExIntNonElementaryEmpty1}{ExIntNonElementaryPatch1}
\pastebutton{ExIntNonElementaryEmpty1}{\showpaste}
\tab{5}\spadcommand{integrate(exp(-x**2) * erf(x) / (erf(x)**3 - erf(x)**2 - erf(x) + 1),x)}
\end{paste}\end{patch}
```

————-

3.31 exlimit.ht

Computing Limits

⇒ "notitle" (ExLimitTwoSided) 3.31 on page 420
⇒ "notitle" (ExLimitOneSided) 3.31 on page 419

— **exlimit.ht** —

```
\begin{page}{ExLimitBasic}{Computing Limits}
```

```
\beginscroll
To compute a limit, you must specify a functional expression,
a variable, and a limiting value for that variable.
For example, to compute the limit of (x**2 - 3*x + 2)/(x**2 - 1)
as x approaches 1, issue the following command:
\spadpaste{limit((x**2 - 3*x + 2)/(x**2 - 1),x = 1)}
% answer := -1/2
Since you have not specified a direction, Axiom will attempt
to compute a two-sided limit. Sometimes the limit when approached from
the left is different from the limit from the right.
\downlink{Example}{ExLimitTwoSided}. In this case, you may
wish to ask for a one-sided limit.
\downlink{How.  }{ExLimitOneSided}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExLimitBasicPatch1}
\begin{paste}{ExLimitBasicFull1}{ExLimitBasicEmpty1}
\pastebutton{ExLimitBasicFull1}{\hidepaste}
\tab{5}\spadcommand{limit((x**2 - 3*x + 2)/(x**2 - 1),x = 1)}
\indentrel{3}\begin{verbatim}
          1
   (1)  -
          2
Type: Union(OrderedCompletion Fraction Polynomial Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExLimitBasicEmpty1}
\begin{paste}{ExLimitBasicEmpty1}{ExLimitBasicPatch1}
\pastebutton{ExLimitBasicEmpty1}{\showpaste}
\tab{5}\spadcommand{limit((x**2 - 3*x + 2)/(x**2 - 1),x = 1)}
\end{paste}\end{patch}
```

Limits of Functions with Parameters

— exlimit.ht —

```
\begin{page}{ExLimitParameter}{Limits of Functions with Parameters}
\beginscroll
You may also take limits of functions with parameters. The limit
will be expressed in terms of those parameters.
Here's an example:
\spadpaste{limit(sinh(a*x)/tan(b*x),x = 0)}
```

```
% answer := a/b
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExLimitParameterPatch1}
\begin{paste}{ExLimitParameterFull1}{ExLimitParameterEmpty1}
\pastebutton{ExLimitParameterFull1}{\hidepaste}
\tab{5}\spadcommand{limit(sinh(a*x)/tan(b*x),x = 0)}
\indentrel{3}\begin{verbatim}
        a
   (1)
        b
  Type: Union(OrderedCompletion Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExLimitParameterEmpty1}
\begin{paste}{ExLimitParameterEmpty1}{ExLimitParameterPatch1}
\pastebutton{ExLimitParameterEmpty1}{\showpaste}
\tab{5}\spadcommand{limit(sinh(a*x)/tan(b*x),x = 0)}
\end{paste}\end{patch}
```

———-

One-sided Limits

— exlimit.ht —

```
\begin{page}{ExLimitOneSided}{One-sided Limits}
\beginscroll
If you have a function which is only defined on one side of a particular value,
you may wish to compute a one-sided limit.
For instance, the function \spad{log(x)} is only defined to the right of zero,
i.e. for \spad{x > 0}.
Thus, when computing limits of functions involving \spad{log(x)}, you probably
will want a 'right-hand' limit.
Here's an example:
\spadpaste{limit(x * log(x),x = 0,"right")}
% answer := 0
When you do not specify \spad{right} or \spad{left} as an optional fourth
argument, the function \spadfun{limit} will try to compute a two-sided limit.
In the above case, the limit from the left does not exist, as Axiom
will indicate when you try to take a two-sided limit:
\spadpaste{limit(x * log(x),x = 0)}
% answer := [left = "failed",right = 0]
\endscroll
```

```
\autobuttons
\end{page}

\begin{patch}{ExLimitOneSidedPatch1}
\begin{paste}{ExLimitOneSidedFull1}{ExLimitOneSidedEmpty1}
\pastebutton{ExLimitOneSidedFull1}{\hidepaste}
\tab{5}\spadcommand{limit(x * log(x),x = 0,"right")}
\indentrel{3}\begin{verbatim}
   (1)  0
  Type: Union(OrderedCompletion Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExLimitOneSidedEmpty1}
\begin{paste}{ExLimitOneSidedEmpty1}{ExLimitOneSidedPatch1}
\pastebutton{ExLimitOneSidedEmpty1}{\showpaste}
\tab{5}\spadcommand{limit(x * log(x),x = 0,"right")}
\end{paste}\end{patch}

\begin{patch}{ExLimitOneSidedPatch2}
\begin{paste}{ExLimitOneSidedFull2}{ExLimitOneSidedEmpty2}
\pastebutton{ExLimitOneSidedFull2}{\hidepaste}
\tab{5}\spadcommand{limit(x * log(x),x = 0)}
\indentrel{3}\begin{verbatim}
   (2)  [leftHandLimit= "failed",rightHandLimit= 0]
Type: Union(Record(leftHandLimit: Union(OrderedCompletion Expression Integer,"failed"),rightHand
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExLimitOneSidedEmpty2}
\begin{paste}{ExLimitOneSidedEmpty2}{ExLimitOneSidedPatch2}
\pastebutton{ExLimitOneSidedEmpty2}{\showpaste}
\tab{5}\spadcommand{limit(x * log(x),x = 0)}
\end{paste}\end{patch}
```

——–

Two-sided Limits

— exlimit.ht —

```
\begin{page}{ExLimitTwoSided}{Two-sided Limits}
\beginscroll
A function may be defined on both sides of a particular value, but will
tend to different limits as its variable tends to that value from the
left and from the right.
We can construct an example of this as follows:
```

```
Since { \em sqrt(y**2)} is simply the absolute value of \spad{y},
the function \spad{sqrt(y**2)/y}
is simply the sign (+1 or -1) of the real number \spad{y}.
Therefore, \spad{sqrt(y**2)/y = -1} for \spad{y < 0} and
\spad{sqrt(y**2)/y = +1} for \spad{y > 0}.
Watch what happens when we take the limit at \spad{y = 0}.
\spadpaste{limit(sqrt(y**2)/y,y = 0)}
% answer := [left = -1,right = 1]
The answer returned by Axiom gives both a 'left-hand' and a 'right-hand'
limit.
Here's another example, this time using a more complicated function:
\spadpaste{limit(sqrt(1 - cos(t))/t,t = 0)}
% answer := [left = -sqrt(1/2),right = sqrt(1/2)]
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExLimitTwoSidedPatch1}
\begin{paste}{ExLimitTwoSidedFull1}{ExLimitTwoSidedEmpty1}
\pastebutton{ExLimitTwoSidedFull1}{\hidepaste}
\tab{5}\spadcommand{limit(sqrt(y**2)/y,y = 0)}
\indentrel{3}\begin{verbatim}
   (1)  [leftHandLimit= - 1,rightHandLimit= 1]
Type: Union(Record(leftHandLimit: Union(OrderedCompletion Expression Integer,"failed"),rightHandLimit: Unior
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExLimitTwoSidedEmpty1}
\begin{paste}{ExLimitTwoSidedEmpty1}{ExLimitTwoSidedPatch1}
\pastebutton{ExLimitTwoSidedEmpty1}{\showpaste}
\tab{5}\spadcommand{limit(sqrt(y**2)/y,y = 0)}
\end{paste}\end{patch}

\begin{patch}{ExLimitTwoSidedPatch2}
\begin{paste}{ExLimitTwoSidedFull2}{ExLimitTwoSidedEmpty2}
\pastebutton{ExLimitTwoSidedFull2}{\hidepaste}
\tab{5}\spadcommand{limit(sqrt(1 - cos(t))/t,t = 0)}
\indentrel{3}\begin{verbatim}
                           1                     1
   (2)  [leftHandLimit= - ,rightHandLimit= ]

                         \2                    \2
Type: Union(Record(leftHandLimit: Union(OrderedCompletion Expression Integer,"failed"),rightHandLimit: Unior
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExLimitTwoSidedEmpty2}
\begin{paste}{ExLimitTwoSidedEmpty2}{ExLimitTwoSidedPatch2}
\pastebutton{ExLimitTwoSidedEmpty2}{\showpaste}
\tab{5}\spadcommand{limit(sqrt(1 - cos(t))/t,t = 0)}
```

```
\end{paste}\end{patch}
```

———-

Limits at Infinity

— exlimit.ht —

```
\begin{page}{ExLimitInfinite}{Limits at Infinity}
\beginscroll
You can compute limits at infinity by passing either 'plus infinity'
or 'minus infinity' as the third argument of the function \spadfun{limit}.
To do this, use the constants \spad{\%plusInfinity} and
\spad{\%minusInfinity}. Here are two examples:
\spadpaste{limit(sqrt(3*x**2 + 1)/(5*x),x = \%plusInfinity)}
\spadpaste{limit(sqrt(3*x**2 + 1)/(5*x),x = \%minusInfinity)}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExLimitInfinitePatch1}
\begin{paste}{ExLimitInfiniteFull1}{ExLimitInfiniteEmpty1}
\pastebutton{ExLimitInfiniteFull1}{\hidepaste}
\tab{5}\spadcommand{limit(sqrt(3*x**2 + 1)/(5*x),x = \%plusInfinity)}
\indentrel{3}\begin{verbatim}

        \3
   (1)
          5
  Type: Union(OrderedCompletion Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExLimitInfiniteEmpty1}
\begin{paste}{ExLimitInfiniteEmpty1}{ExLimitInfinitePatch1}
\pastebutton{ExLimitInfiniteEmpty1}{\showpaste}
\tab{5}\spadcommand{limit(sqrt(3*x**2 + 1)/(5*x),x = \%plusInfinity)}
\end{paste}\end{patch}

\begin{patch}{ExLimitInfinitePatch2}
\begin{paste}{ExLimitInfiniteFull2}{ExLimitInfiniteEmpty2}
\pastebutton{ExLimitInfiniteFull2}{\hidepaste}
\tab{5}\spadcommand{limit(sqrt(3*x**2 + 1)/(5*x),x = \%minusInfinity)}
\indentrel{3}\begin{verbatim}

          \3
   (2)  -
```

```
           5
  Type: Union(OrderedCompletion Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExLimitInfiniteEmpty2}
\begin{paste}{ExLimitInfiniteEmpty2}{ExLimitInfinitePatch2}
\pastebutton{ExLimitInfiniteEmpty2}{\showpaste}
\tab{5}\spadcommand{limit(sqrt(3*x**2 + 1)/(5*x),x = \%minusInfinity)}
\end{paste}\end{patch}
```

———-

Real Limits vs. Complex Limits

— exlimit.ht —

```
\begin{page}{ExLimitRealComplex}{Real Limits vs. Complex Limits}
\beginscroll
When you use the function \spadfun{limit}, you will be taking the limit of a real
function of a real variable.
For example, you can compute
\spadpaste{limit(z * sin(1/z),z = 0)}
Axiom returns \spad{0} because as a function of a real variable
\spad{sin(1/z)} is always between \spad{-1} and \spad{1}, so \spad{z * sin(1/z)}
tends to \spad{0} as \spad{z} tends to \spad{0}.
However, as a function of a complex variable, \spad{sin(1/z)} is badly
behaved around \spad{0}
(one says that \spad{sin(1/z)} has an 'essential singularity' at \spad{z = 0}).
When viewed as a function of a complex variable, \spad{z * sin(1/z)}
does not approach any limit as \spad{z} tends to \spad{0} in the complex plane.
Axiom indicates this when we call the function \spadfun{complexLimit}:
\spadpaste{complexLimit(z * sin(1/z),z = 0)}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExLimitRealComplexPatch1}
\begin{paste}{ExLimitRealComplexFull1}{ExLimitRealComplexEmpty1}
\pastebutton{ExLimitRealComplexFull1}{\hidepaste}
\tab{5}\spadcommand{limit(z * sin(1/z),z = 0)}
\indentrel{3}\begin{verbatim}
   (1)  0
  Type: Union(OrderedCompletion Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ExLimitRealComplexEmpty1}
\begin{paste}{ExLimitRealComplexEmpty1}{ExLimitRealComplexPatch1}
\pastebutton{ExLimitRealComplexEmpty1}{\showpaste}
\tab{5}\spadcommand{limit(z * sin(1/z),z = 0)}
\end{paste}\end{patch}

\begin{patch}{ExLimitRealComplexPatch2}
\begin{paste}{ExLimitRealComplexFull2}{ExLimitRealComplexEmpty2}
\pastebutton{ExLimitRealComplexFull2}{\hidepaste}
\tab{5}\spadcommand{complexLimit(z * sin(1/z),z = 0)}
\indentrel{3}\begin{verbatim}
   (2)  "failed"
                              Type: Union("failed",...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExLimitRealComplexEmpty2}
\begin{paste}{ExLimitRealComplexEmpty2}{ExLimitRealComplexPatch2}
\pastebutton{ExLimitRealComplexEmpty2}{\showpaste}
\tab{5}\spadcommand{complexLimit(z * sin(1/z),z = 0)}
\end{paste}
\end{patch}
```

———-

Complex Limits at Infinity

— exlimit.ht —

```
\begin{page}{ExLimitComplexInfinite}{Complex Limits at Infinity}
\beginscroll
You may also take complex limits at infinity, i.e. limits of a function of
\spad{z} as \spad{z} approaches infinity on the Riemann sphere.
Use the symbol \spad{\%infinity} to denote ‘complex infinity’.
Also, to compute complex limits rather than real limits, use the
function \spadfun{complexLimit}.
Here is an example:
\spadpaste{complexLimit((2 + z)/(1 - z),z = \%infinity)}
In many cases, a limit of a real function of a real variable will exist
when the corresponding complex limit does not.
For example:
\spadpaste{limit(sin(x)/x,x = \%plusInfinity)}
\spadpaste{complexLimit(sin(x)/x,x = \%infinity)}
\endscroll
\autobuttons
\end{page}
```

```
\begin{patch}{ExLimitComplexInfinitePatch1}
\begin{paste}{ExLimitComplexInfiniteFull1}{ExLimitComplexInfiniteEmpty1}
\pastebutton{ExLimitComplexInfiniteFull1}{\hidepaste}
\tab{5}\spadcommand{complexLimit((2 + z)/(1 - z),z = \%infinity)}
\indentrel{3}\begin{verbatim}
   (1)  - 1
   Type: OnePointCompletion Fraction Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExLimitComplexInfiniteEmpty1}
\begin{paste}{ExLimitComplexInfiniteEmpty1}{ExLimitComplexInfinitePatch1}
\pastebutton{ExLimitComplexInfiniteEmpty1}{\showpaste}
\tab{5}\spadcommand{complexLimit((2 + z)/(1 - z),z = \%infinity)}
\end{paste}\end{patch}

\begin{patch}{ExLimitComplexInfinitePatch2}
\begin{paste}{ExLimitComplexInfiniteFull2}{ExLimitComplexInfiniteEmpty2}
\pastebutton{ExLimitComplexInfiniteFull2}{\hidepaste}
\tab{5}\spadcommand{limit(sin(x)/x,x = \%plusInfinity)}
\indentrel{3}\begin{verbatim}
   (2)  0
  Type: Union(OrderedCompletion Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExLimitComplexInfiniteEmpty2}
\begin{paste}{ExLimitComplexInfiniteEmpty2}{ExLimitComplexInfinitePatch2}
\pastebutton{ExLimitComplexInfiniteEmpty2}{\showpaste}
\tab{5}\spadcommand{limit(sin(x)/x,x = \%plusInfinity)}
\end{paste}\end{patch}

\begin{patch}{ExLimitComplexInfinitePatch3}
\begin{paste}{ExLimitComplexInfiniteFull3}{ExLimitComplexInfiniteEmpty3}
\pastebutton{ExLimitComplexInfiniteFull3}{\hidepaste}
\tab{5}\spadcommand{complexLimit(sin(x)/x,x = \%infinity)}
\indentrel{3}\begin{verbatim}
   (3)  "failed"
                              Type: Union("failed",...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExLimitComplexInfiniteEmpty3}
\begin{paste}{ExLimitComplexInfiniteEmpty3}{ExLimitComplexInfinitePatch3}
\pastebutton{ExLimitComplexInfiniteEmpty3}{\showpaste}
\tab{5}\spadcommand{complexLimit(sin(x)/x,x = \%infinity)}
\end{paste}\end{patch}
```

3.32 exmatrix.ht

Basic Arithmetic Operations on Matrices

— exmatrix.ht —

```
\begin{page}{ExMatrixBasicFunction}
{Basic Arithmetic Operations on Matrices}
\beginscroll
You can create a matrix using the function \spadfun{matrix}.
The function takes a list of lists of elements of the ring and
produces a matrix whose \spad{i}th row contains the elements of
the \spad{i}th list. For example:
\spadpaste{m1 := matrix([[1,-2,1],[4,2,-4]]) \bound{m1}}
\spadpaste{m2 := matrix([[1,0,2],[20,30,10],[0,200,100]]) \bound{m2}}
\spadpaste{m3 := matrix([[1,2,3],[2,4,6]]) \bound{m3}}
Some of the basic arithmetic operations on matrices are:
\spadpaste{m1 + m3 \free{m1} \free{m3}}
\spadpaste{100 * m1 \free{m1}}
\spadpaste{m1 * m2 \free{m1} \free{m2}}
\spadpaste{-m1 + m3 * m2 \free{m1} \free{m2} \free{m3}}
You can also multiply a matrix and a vector provided
that the matrix and vector have compatible dimensions.
\spadpaste{m3 *vector([1,0,1]) \free{m3}}
However, the dimensions of the matrices must be compatible in order for
Axiom to perform an operation - otherwise an error message will occur.
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExMatrixBasicFunctionPatch1}
\begin{paste}{ExMatrixBasicFunctionFull1}{ExMatrixBasicFunctionEmpty1}
\pastebutton{ExMatrixBasicFunctionFull1}{\hidepaste}
\tab{5}\spadcommand{m1 := matrix([[1,-2,1],[4,2,-4]])\bound{m1 }}
\indentrel{3}\begin{verbatim}
        1  - 2   1
   (1)
        4   2   - 4
                                   Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExMatrixBasicFunctionEmpty1}
\begin{paste}{ExMatrixBasicFunctionEmpty1}{ExMatrixBasicFunctionPatch1}
\pastebutton{ExMatrixBasicFunctionEmpty1}{\showpaste}
\tab{5}\spadcommand{m1 := matrix([[1,-2,1],[4,2,-4]])\bound{m1 }}
\end{paste}\end{patch}
```

```
\begin{patch}{ExMatrixBasicFunctionPatch2}
\begin{paste}{ExMatrixBasicFunctionFull2}{ExMatrixBasicFunctionEmpty2}
\pastebutton{ExMatrixBasicFunctionFull2}{\hidepaste}
\tab{5}\spadcommand{m2 := matrix([[1,0,2],[20,30,10],[0,200,100]])\bound{m2 }}
\indentrel{3}\begin{verbatim}
        1    0    2

   (2)  20  30   10

        0   200  100
                                   Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExMatrixBasicFunctionEmpty2}
\begin{paste}{ExMatrixBasicFunctionEmpty2}{ExMatrixBasicFunctionPatch2}
\pastebutton{ExMatrixBasicFunctionEmpty2}{\showpaste}
\tab{5}\spadcommand{m2 := matrix([[1,0,2],[20,30,10],[0,200,100]])\bound{m2 }}
\end{paste}\end{patch}

\begin{patch}{ExMatrixBasicFunctionPatch3}
\begin{paste}{ExMatrixBasicFunctionFull3}{ExMatrixBasicFunctionEmpty3}
\pastebutton{ExMatrixBasicFunctionFull3}{\hidepaste}
\tab{5}\spadcommand{m3 := matrix([[1,2,3],[2,4,6]])\bound{m3 }}
\indentrel{3}\begin{verbatim}
        1  2  3
   (3)
        2  4  6
                                   Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExMatrixBasicFunctionEmpty3}
\begin{paste}{ExMatrixBasicFunctionEmpty3}{ExMatrixBasicFunctionPatch3}
\pastebutton{ExMatrixBasicFunctionEmpty3}{\showpaste}
\tab{5}\spadcommand{m3 := matrix([[1,2,3],[2,4,6]])\bound{m3 }}
\end{paste}\end{patch}

\begin{patch}{ExMatrixBasicFunctionPatch4}
\begin{paste}{ExMatrixBasicFunctionFull4}{ExMatrixBasicFunctionEmpty4}
\pastebutton{ExMatrixBasicFunctionFull4}{\hidepaste}
\tab{5}\spadcommand{m1 + m3\free{m1 }\free{m3 }}
\indentrel{3}\begin{verbatim}
        2  0  4
   (4)
        6  6  2
                                   Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ExMatrixBasicFunctionEmpty4}
\begin{paste}{ExMatrixBasicFunctionEmpty4}{ExMatrixBasicFunctionPatch4}
\pastebutton{ExMatrixBasicFunctionEmpty4}{\showpaste}
\tab{5}\spadcommand{m1 + m3\free{m1 }\free{m3 }}
\end{paste}\end{patch}

\begin{patch}{ExMatrixBasicFunctionPatch5}
\begin{paste}{ExMatrixBasicFunctionFull5}{ExMatrixBasicFunctionEmpty5}
\pastebutton{ExMatrixBasicFunctionFull5}{\hidepaste}
\tab{5}\spadcommand{100 * m1\free{m1 }}
\indentrel{3}\begin{verbatim}
        100  - 200   100
   (5)
        400   200   - 400
                                   Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExMatrixBasicFunctionEmpty5}
\begin{paste}{ExMatrixBasicFunctionEmpty5}{ExMatrixBasicFunctionPatch5}
\pastebutton{ExMatrixBasicFunctionEmpty5}{\showpaste}
\tab{5}\spadcommand{100 * m1\free{m1 }}
\end{paste}\end{patch}

\begin{patch}{ExMatrixBasicFunctionPatch6}
\begin{paste}{ExMatrixBasicFunctionFull6}{ExMatrixBasicFunctionEmpty6}
\pastebutton{ExMatrixBasicFunctionFull6}{\hidepaste}
\tab{5}\spadcommand{m1 * m2\free{m1 }\free{m2 }}
\indentrel{3}\begin{verbatim}
        - 39   140    82
   (6)
         44   - 740  - 372
                                   Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExMatrixBasicFunctionEmpty6}
\begin{paste}{ExMatrixBasicFunctionEmpty6}{ExMatrixBasicFunctionPatch6}
\pastebutton{ExMatrixBasicFunctionEmpty6}{\showpaste}
\tab{5}\spadcommand{m1 * m2\free{m1 }\free{m2 }}
\end{paste}\end{patch}

\begin{patch}{ExMatrixBasicFunctionPatch7}
\begin{paste}{ExMatrixBasicFunctionFull7}{ExMatrixBasicFunctionEmpty7}
\pastebutton{ExMatrixBasicFunctionFull7}{\hidepaste}
\tab{5}\spadcommand{-m1 + m3 * m2\free{m1 }\free{m2 }\free{m3 }}
\indentrel{3}\begin{verbatim}
        40  662   321
   (7)
        78  1318  648
```

```
                                   Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExMatrixBasicFunctionEmpty7}
\begin{paste}{ExMatrixBasicFunctionEmpty7}{ExMatrixBasicFunctionPatch7}
\pastebutton{ExMatrixBasicFunctionEmpty7}{\showpaste}
\tab{5}\spadcommand{-m1 + m3 * m2\free{m1 }\free{m2 }\free{m3 }}
\end{paste}\end{patch}

\begin{patch}{ExMatrixBasicFunctionPatch8}
\begin{paste}{ExMatrixBasicFunctionFull8}{ExMatrixBasicFunctionEmpty8}
\pastebutton{ExMatrixBasicFunctionFull8}{\hidepaste}
\tab{5}\spadcommand{m3 *vector([1,0,1])\free{m3 }}
\indentrel{3}\begin{verbatim}
   (8)  [4,8]
                                   Type: Vector Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExMatrixBasicFunctionEmpty8}
\begin{paste}{ExMatrixBasicFunctionEmpty8}{ExMatrixBasicFunctionPatch8}
\pastebutton{ExMatrixBasicFunctionEmpty8}{\showpaste}
\tab{5}\spadcommand{m3 *vector([1,0,1])\free{m3 }}
\end{paste}
\end{patch}
```

———-

Constructing new Matrices

— exmatrix.ht —

```
\begin{page}{ExConstructMatrix}{Constructing new Matrices}
\beginscroll
A number of functions exist for constructing new matrices from existing ones.

If you want to create a matrix whose entries are 0 except on the main
diagonal you can use \spadfun{diagonalMatrix}.
This function takes a list of ring elements as an argument and returns a
square matrix which has these elements on the main diagonal.
Consider the following example:
\spadpaste{diagonalMatrix([1,2,3,2,1])}

The function \spadfun{subMatrix}(\spad{a},\spad{i},\spad{j},\spad{k},\spad{l})
constructs a new matrix
consisting of rows \spad{i} through \spad{j} and columns \spad{k} through
```

```
\spad{l} of \spad{a} , inclusive.
\spadpaste{subMatrix(matrix([[0,1,2,3,4],[5,6,7,8,9],[10,11,12,13,14]]),
1,3,2,4)}


The functions \spadfun{horizConcat} and \spadfun{vertConcat}
concatenate matrices
horizontally and vertically, respectively.
\spadpaste{horizConcat(matrix([[1,2,3],[6,7,8]]),
matrix([[11,12,13],[55,77,88]])) }
\spadpaste{vertConcat(matrix([[1,2,3],[6,7,8]]),
matrix([[11,12,13],[55,77,88]])) }


The function \spadfunX{setsubMatrix}(\spad{a},\spad{i},\spad{k},\spad{b})
replaces the submatrix of \spad{a}
starting at row \spad{i} and column \spad{k} with the elements of the
matrix i\spad{b}.
\spadpaste{b:=matrix([[0,1,2,3,4],[5,6,7,8,9],[10,11,12,13,14]]) \bound{b}}
\spadpaste{setsubMatrix!(b,1,1,transpose(subMatrix(b,1,3,1,3)))\free{b}}
changes the submatrix of \spad{b} consisting of the first 3 rows and columns
to its transpose.
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExConstructMatrixPatch1}
\begin{paste}{ExConstructMatrixFull1}{ExConstructMatrixEmpty1}
\pastebutton{ExConstructMatrixFull1}{\hidepaste}
\tab{5}\spadcommand{diagonalMatrix([1,2,3,2,1])}
\indentrel{3}\begin{verbatim}
        1  0  0  0  0

        0  2  0  0  0

   (1)  0  0  3  0  0

        0  0  0  2  0

        0  0  0  0  1
                                   Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExConstructMatrixEmpty1}
\begin{paste}{ExConstructMatrixEmpty1}{ExConstructMatrixPatch1}
\pastebutton{ExConstructMatrixEmpty1}{\showpaste}
\tab{5}\spadcommand{diagonalMatrix([1,2,3,2,1])}
\end{paste}\end{patch}
```

```
\begin{patch}{ExConstructMatrixPatch2}
\begin{paste}{ExConstructMatrixFull2}{ExConstructMatrixEmpty2}
\pastebutton{ExConstructMatrixFull2}{\hidepaste}
\tab{5}\spadcommand{subMatrix(matrix([[0,1,2,3,4],[5,6,7,8,9],[10,11,12,13,14]]), 1,3,2,4)}
\indentrel{3}\begin{verbatim}
       1   2   3

   (2) 6   7   8

       11  12  13
                                   Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExConstructMatrixEmpty2}
\begin{paste}{ExConstructMatrixEmpty2}{ExConstructMatrixPatch2}
\pastebutton{ExConstructMatrixEmpty2}{\showpaste}
\tab{5}\spadcommand{subMatrix(matrix([[0,1,2,3,4],[5,6,7,8,9],[10,11,12,13,14]]), 1,3,2,4)}
\end{paste}\end{patch}

\begin{patch}{ExConstructMatrixPatch3}
\begin{paste}{ExConstructMatrixFull3}{ExConstructMatrixEmpty3}
\pastebutton{ExConstructMatrixFull3}{\hidepaste}
\tab{5}\spadcommand{horizConcat(matrix([[1,2,3],[6,7,8]]),matrix([[11,12,13],[55,77,88]]))}
\indentrel{3}\begin{verbatim}
       1  2  3  11  12  13
   (3)
       6  7  8  55  77  88
                                   Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExConstructMatrixEmpty3}
\begin{paste}{ExConstructMatrixEmpty3}{ExConstructMatrixPatch3}
\pastebutton{ExConstructMatrixEmpty3}{\showpaste}
\tab{5}\spadcommand{horizConcat(matrix([[1,2,3],[6,7,8]]),matrix([[11,12,13],[55,77,88]]))}
\end{paste}\end{patch}

\begin{patch}{ExConstructMatrixPatch4}
\begin{paste}{ExConstructMatrixFull4}{ExConstructMatrixEmpty4}
\pastebutton{ExConstructMatrixFull4}{\hidepaste}
\tab{5}\spadcommand{vertConcat(matrix([[1,2,3],[6,7,8]]),matrix([[11,12,13],[55,77,88]]))}
\indentrel{3}\begin{verbatim}
       1   2   3

       6   7   8
   (4)
       11  12  13

       55  77  88
```

```
                                     Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExConstructMatrixEmpty4}
\begin{paste}{ExConstructMatrixEmpty4}{ExConstructMatrixPatch4}
\pastebutton{ExConstructMatrixEmpty4}{\showpaste}
\tab{5}\spadcommand{vertConcat(matrix([[1,2,3],[6,7,8]]),matrix([[11,12,13],[55,77,88]]))}
\end{paste}\end{patch}

\begin{patch}{ExConstructMatrixPatch5}
\begin{paste}{ExConstructMatrixFull5}{ExConstructMatrixEmpty5}
\pastebutton{ExConstructMatrixFull5}{\hidepaste}
\tab{5}\spadcommand{b:=matrix([[0,1,2,3,4],[5,6,7,8,9],[10,11,12,13,14]])\bound{b }}
\indentrel{3}\begin{verbatim}
        0   1   2   3   4

   (5)  5   6   7   8   9

        10  11  12  13  14
                                     Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExConstructMatrixEmpty5}
\begin{paste}{ExConstructMatrixEmpty5}{ExConstructMatrixPatch5}
\pastebutton{ExConstructMatrixEmpty5}{\showpaste}
\tab{5}\spadcommand{b:=matrix([[0,1,2,3,4],[5,6,7,8,9],[10,11,12,13,14]])\bound{b }}
\end{paste}\end{patch}

\begin{patch}{ExConstructMatrixPatch6}
\begin{paste}{ExConstructMatrixFull6}{ExConstructMatrixEmpty6}
\pastebutton{ExConstructMatrixFull6}{\hidepaste}
\tab{5}\spadcommand{setsubMatrix!(b,1,1,transpose(subMatrix(b,1,3,1,3)))\free{b }}
\indentrel{3}\begin{verbatim}
        0  5  10  3   4

   (6)  1  6  11  8   9

        2  7  12  13  14
                                     Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExConstructMatrixEmpty6}
\begin{paste}{ExConstructMatrixEmpty6}{ExConstructMatrixPatch6}
\pastebutton{ExConstructMatrixEmpty6}{\showpaste}
\tab{5}\spadcommand{setsubMatrix!(b,1,1,transpose(subMatrix(b,1,3,1,3)))\free{b }}
\end{paste}\end{patch}
```

———-

Trace of a Matrix

— exmatrix.ht —

```
\begin{page}{ExTraceMatrix}{Trace of a Matrix}
\beginscroll
If you have a square matrix, then you can compute its `trace'.
The function \spadfun{trace} computes the sum of all elements
on the diagonal of a matrix.
For example `trace' for a four by four Vandermonde matrix.
\spadpaste{trace( matrix([[1,x,x**2,x**3],[1,y,y**2,y**3],
[1,z,z**2,z**3],[1,u,u**2,u**3]]) )}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExTraceMatrixPatch1}
\begin{paste}{ExTraceMatrixFull1}{ExTraceMatrixEmpty1}
\pastebutton{ExTraceMatrixFull1}{\hidepaste}
\tab{5}\spadcommand{trace( matrix([[1,x,x**2,x**3],[1,y,y**2,y**3],[1,z,z**2,z**3],[1,u,u**2,u**3]]) )}
\indentrel{3}\begin{verbatim}
        2        3
   (1)  z  + y + u  + 1
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExTraceMatrixEmpty1}
\begin{paste}{ExTraceMatrixEmpty1}{ExTraceMatrixPatch1}
\pastebutton{ExTraceMatrixEmpty1}{\showpaste}
\tab{5}\spadcommand{trace( matrix([[1,x,x**2,x**3],[1,y,y**2,y**3],[1,z,z**2,z**3],[1,u,u**2,u**3]]) )}
\end{paste}\end{patch}
```

———-

Determinant of a Matrix

— exmatrix.ht —

```
\begin{page}{ExDeterminantMatrix}{Determinant of a Matrix}
\beginscroll
The function \spadfun{determinant} computes the determinant of a matrix
```

```
over a commutative ring, that is a ring whose multiplication is
commutative.
\spadpaste{determinant(matrix([[1,2,3,4],[2,3,2,5],[3,4,5,6],[4,1,6,7]]))}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExDeterminantMatrixPatch1}
\begin{paste}{ExDeterminantMatrixFull1}{ExDeterminantMatrixEmpty1}
\pastebutton{ExDeterminantMatrixFull1}{\hidepaste}
\tab{5}\spadcommand{determinant(matrix([[1,2,3,4],[2,3,2,5],[3,4,5,6],[4,1,6,7]]))}
\indentrel{3}\begin{verbatim}
   (1)  - 48
                                          Type: Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExDeterminantMatrixEmpty1}
\begin{paste}{ExDeterminantMatrixEmpty1}{ExDeterminantMatrixPatch1}
\pastebutton{ExDeterminantMatrixEmpty1}{\showpaste}
\tab{5}\spadcommand{determinant(matrix([[1,2,3,4],[2,3,2,5],[3,4,5,6],[4,1,6,7]]))}
\end{paste}
\end{patch}
```

Inverse of a Matrix

— exmatrix.ht —

```
\begin{page}{ExInverseMatrix}{Inverse of a Matrix}
\beginscroll
The function \spadfun{inverse} computes the inverse of a square matrix.
\spadpaste{inverse(matrix([[1,2,1],[-2,3,4],[-1,5,6]])) }
(If the inverse doesn't exist, then Axiom returns `failed'.)
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExInverseMatrixPatch1}
\begin{paste}{ExInverseMatrixFull1}{ExInverseMatrixEmpty1}
\pastebutton{ExInverseMatrixFull1}{\hidepaste}
\tab{5}\spadcommand{inverse(matrix([[1,2,1],[-2,3,4],[-1,5,6]]))}
\indentrel{3}\begin{verbatim}
          2        5
        -   - 1
          7        7
```

```
   (1)   8          6
             1   -
         7          7

        - 1  - 1   1
               Type: Union(Matrix Fraction Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExInverseMatrixEmpty1}
\begin{paste}{ExInverseMatrixEmpty1}{ExInverseMatrixPatch1}
\pastebutton{ExInverseMatrixEmpty1}{\showpaste}
\tab{5}\spadcommand{inverse(matrix([[1,2,1],[-2,3,4],[-1,5,6]]))}
\end{paste}\end{patch}
```

———-

Rank of a Matrix

— exmatrix.ht —

```
\begin{page}{ExRankMatrix}{Rank of a Matrix}
\beginscroll
The function \spadfun{rank} gives you the rank of a matrix:
\spadpaste{rank(matrix([[0,4,1],[5,3,-7],[-5,5,9]]))}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExRankMatrixPatch1}
\begin{paste}{ExRankMatrixFull1}{ExRankMatrixEmpty1}
\pastebutton{ExRankMatrixFull1}{\hidepaste}
\tab{5}\spadcommand{rank(matrix([[0,4,1],[5,3,-7],[-5,5,9]]))}
\indentrel{3}\begin{verbatim}
   (1)  2
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExRankMatrixEmpty1}
\begin{paste}{ExRankMatrixEmpty1}{ExRankMatrixPatch1}
\pastebutton{ExRankMatrixEmpty1}{\showpaste}
\tab{5}\spadcommand{rank(matrix([[0,4,1],[5,3,-7],[-5,5,9]]))}
\end{paste}\end{patch}
```

———-

3.33 expr.ht

Expression

⇒ "notitle" (KernelXmpPage) 3.58 on page 743
⇒ "notitle" (ugIntroCalcDerivPage) 6 on page 1580
⇒ "notitle" (ugIntroCalcLimitsPage) 6 on page 1569
⇒ "notitle" (ugIntroSeriesPage) 6 on page 1573
⇒ "notitle" (ugProblemDEQPage) 12 on page 2211
⇒ "notitle" (ugProblemIntegrationPage) 12 on page 2157
⇒ "notitle" (ugUserRulesPage) 10 on page 1949

— **expr.ht** —

```
\begin{page}{ExpressionXmpPage}{Expression}
\beginscroll

\axiomType{Expression} is a constructor that creates domains whose
objects can have very general symbolic forms.
Here are some examples:
\xtc{
This is an object of type \axiomType{Expression Integer}.
}{
\spadpaste{sin(x) + 3*cos(x)**2}
}
\xtc{
This is an object of type \axiomType{Expression Float}.
}{
\spadpaste{tan(x) - 3.45*x}
}
\xtc{
This object contains symbolic function applications, sums,
products, square roots, and a quotient.
}{
\spadpaste{(tan sqrt 7 - sin sqrt 11)**2 / (4 - cos(x - y))}
}
As you can see, \axiomType{Expression} actually takes an argument
domain.
The {\it coefficients} of the terms within the expression belong
to the argument domain.
\axiomType{Integer} and \axiomType{Float}, along with
\axiomType{Complex Integer} and \axiomType{Complex Float}
are the most common coefficient domains.
\xtc{
The choice of whether
```

```
to use a \axiomType{Complex} coefficient domain or not is
important since Axiom can perform some simplifications
on real-valued objects
}{
\spadpaste{log(exp  x)@Expression(Integer)}
}
\xtc{
... which are not valid on complex ones.
}{
\spadpaste{log(exp  x)@Expression(Complex Integer)}
}
\xtc{
Many potential coefficient domains, such as
\axiomType{AlgebraicNumber}, are not usually used because
\axiomType{Expression} can subsume them.
}{
\spadpaste{sqrt 3 + sqrt(2 + sqrt(-5)) \bound{algnum1}}
}
\xtc{
}{
\spadpaste{\% :: Expression Integer \free{algnum1}}
}
Note that we sometimes talk about ``an object of type
\axiomType{Expression}.'' This is not really correct because we
should say, for example, ``an object of type \axiomType{Expression
Integer}'' or ``an object of type \axiomType{Expression Float}.''
By a similar abuse of language, when we refer to an ``expression''
in this section we will mean an object of type
\axiomType{Expression R} for some domain {\bf R}.

The Axiom documentation contains many examples of the use
of \axiomType{Expression}.
For the rest of this section, we'll give you some pointers to those
examples plus give you some idea of how to manipulate expressions.

It is important for you to know that \axiomType{Expression}
creates domains that have category \axiomType{Field}.
Thus you can invert any non-zero expression and you shouldn't
expect an operation like \axiomFun{factor} to give you much
information.
You can imagine expressions as being represented as quotients of
``multivariate'' polynomials where the ``variables'' are kernels
(see \downlink{`Kernel'}{KernelXmpPage}\ignore{Kernel}).
A kernel can either be a symbol such as \axiom{x} or a symbolic
function application like \axiom{sin(x + 4)}.
The second example is actually a nested kernel since the argument
to \axiomFun{sin} contains the kernel \axiom{x}.
\xtc{
}{
\spadpaste{height mainKernel sin(x + 4)}
```

```
}
Actually, the argument to \axiomFun{sin} is an expression, and so
the structure of \axiomType{Expression} is recursive.
\downlink{`Kernel'}{KernelXmpPage}\ignore{Kernel}
demonstrates how to extract the kernels in an
expression.

Use the Hyperdoc Browse facility to see what operations are
applicable to expression.
At the time of this writing, there were 262 operations with 147
distinct name in \axiomType{Expression Integer}.
For example, \axiomFunFrom{numer}{Expression} and
\axiomFunFrom{denom}{Expression} extract the numerator and
denominator of an expression.
\xtc{
}{
\spadpaste{e := (sin(x) - 4)**2 / ( 1 - 2*y*sqrt(- y) ) \bound{e}}
}
\xtc{
}{
\spadpaste{numer e \free{e}}
}
\xtc{
}{
\spadpaste{denom e \free{e}}
}
\xtc{
Use \axiomFunFrom{D}{Expression} to compute partial derivatives.
}{
\spadpaste{D(e, x) \free{e}}
}
\xtc{
See \downlink{``Derivatives''}{ugIntroCalcDerivPage} in Section 1.12
\ignore{ugIntroCalcDeriv}
for more examples of expressions and derivatives.
}{
\spadpaste{D(e, [x, y], [1, 2]) \free{e}}
}
See \downlink{``Limits''}{ugIntroCalcLimitsPage}
in Section 1.10\ignore{ugIntroCalcLimits} and
\downlink{``Series''}{ugIntroSeriesPage} in Section 1.11
\ignore{ugIntroSeries}
for more examples of expressions and calculus.
Differential equations involving expressions are discussed in
\downlink{``Solution of Differential Equations''}
{ugProblemDEQPage} in Section 8.10\ignore{ugProblemDEQ}.
Chapter 8 has many advanced examples: see
\downlink{``Integration''}{ugProblemIntegrationPage}
in Section 8.8\ignore{ugProblemIntegration}
for a discussion of Axiom's integration facilities.
```

```
When an expression involves no ‘‘symbol kernels’’ (for example,
\axiom{x}), it may be possible to numerically evaluate the
expression.
\xtc{
If you suspect the evaluation will create a complex number,
use \axiomFun{complexNumeric}.
}{
\spadpaste{complexNumeric(cos(2 - 3*\%i))}
}
\xtc{
If you know it will be real, use \axiomFun{numeric}.
}{
\spadpaste{numeric(tan 3.8)}
}
The \axiomFun{numeric} operation will display an error message if
the evaluation yields a calue with an non-zero imaginary part.
Both of these operations have an optional second argument
\axiom{n} which specifies that the accuracy of the approximation
be up to \axiom{n} decimal places.

When an expression involves no ‘‘symbolic application’’ kernels,
it may be possible to convert it a polynomial or rational
function in the variables that are present.
\xtc{
}{
\spadpaste{e2 := cos(x**2 - y + 3) \bound{e2}}
}
\xtc{
}{
\spadpaste{e3 := asin(e2) - \%pi/2 \free{e2}\bound{e3}}
}
\xtc{
}{
\spadpaste{e3 :: Polynomial Integer \free{e3}}
}
\xtc{
This also works for the polynomial types where specific variables
and their ordering are given.
}{
\spadpaste{e3 :: DMP([x, y], Integer) \free{e3}}
}

Finally, a certain amount of simplication takes place as
expressions are constructed.
\xtc{
}{
\spadpaste{sin \%pi}
}
\xtc{
```

```
}{
\spadpaste{cos(\%pi / 4)}
}
\xtc{
For simplications that involve multiple terms of the expression,
use \axiomFun{simplify}.
}{
\spadpaste{tan(x)**6 + 3*tan(x)**4 + 3*tan(x)**2 + 1 \bound{tan6}}
}
\xtc{
}{
\spadpaste{simplify \% \free{tan6}}
}
See \downlink{``Rules and Pattern Matching''}{ugUserRulesPage} in
Section 6.21\ignore{ugUserRules} for examples of how to write your own
rewrite rules for expressions.

\endscroll
\autobuttons
\end{page}

\begin{patch}{ExpressionXmpPagePatch1}
\begin{paste}{ExpressionXmpPageFull1}{ExpressionXmpPageEmpty1}
\pastebutton{ExpressionXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{sin(x) + 3*cos(x)**2}
\indentrel{3}\begin{verbatim}
                        2
   (1)  sin(x) + 3cos(x)
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExpressionXmpPageEmpty1}
\begin{paste}{ExpressionXmpPageEmpty1}{ExpressionXmpPagePatch1}
\pastebutton{ExpressionXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{sin(x) + 3*cos(x)**2}
\end{paste}\end{patch}

\begin{patch}{ExpressionXmpPagePatch2}
\begin{paste}{ExpressionXmpPageFull2}{ExpressionXmpPageEmpty2}
\pastebutton{ExpressionXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{tan(x) - 3.45*x}
\indentrel{3}\begin{verbatim}
   (2)  tan(x) - 3.45 x
                                 Type: Expression Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExpressionXmpPageEmpty2}
\begin{paste}{ExpressionXmpPageEmpty2}{ExpressionXmpPagePatch2}
```

```
\pastebutton{ExpressionXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{tan(x) - 3.45*x}
\end{paste}\end{patch}

\begin{patch}{ExpressionXmpPagePatch3}
\begin{paste}{ExpressionXmpPageFull3}{ExpressionXmpPageEmpty3}
\pastebutton{ExpressionXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{(tan sqrt 7 - sin sqrt 11)**2 / (4 - cos(x - y))}
\indentrel{3}\begin{verbatim}
   (3)
          2                          2
   - tan(\7 )  + 2sin(\11 )tan(\7 ) - sin(\11 )

                    cos(y - x) - 4
                              Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExpressionXmpPageEmpty3}
\begin{paste}{ExpressionXmpPageEmpty3}{ExpressionXmpPagePatch3}
\pastebutton{ExpressionXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{(tan sqrt 7 - sin sqrt 11)**2 / (4 - cos(x - y))}
\end{paste}\end{patch}

\begin{patch}{ExpressionXmpPagePatch4}
\begin{paste}{ExpressionXmpPageFull4}{ExpressionXmpPageEmpty4}
\pastebutton{ExpressionXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{log(exp x)@Expression(Integer)}
\indentrel{3}\begin{verbatim}
   (4)  x
                              Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExpressionXmpPageEmpty4}
\begin{paste}{ExpressionXmpPageEmpty4}{ExpressionXmpPagePatch4}
\pastebutton{ExpressionXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{log(exp x)@Expression(Integer)}
\end{paste}\end{patch}

\begin{patch}{ExpressionXmpPagePatch5}
\begin{paste}{ExpressionXmpPageFull5}{ExpressionXmpPageEmpty5}
\pastebutton{ExpressionXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{log(exp x)@Expression(Complex Integer)}
\indentrel{3}\begin{verbatim}
             x
   (5)  log(%e )
                      Type: Expression Complex Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ExpressionXmpPageEmpty5}
\begin{paste}{ExpressionXmpPageEmpty5}{ExpressionXmpPagePatch5}
\pastebutton{ExpressionXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{log(exp x)@Expression(Complex Integer)}
\end{paste}\end{patch}

\begin{patch}{ExpressionXmpPagePatch6}
\begin{paste}{ExpressionXmpPageFull6}{ExpressionXmpPageEmpty6}
\pastebutton{ExpressionXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{sqrt 3 + sqrt(2 + sqrt(-5))\bound{algnum1 }}
\indentrel{3}\begin{verbatim}


   (6)  \\- 5  + 2  + \3
                                  Type: AlgebraicNumber
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExpressionXmpPageEmpty6}
\begin{paste}{ExpressionXmpPageEmpty6}{ExpressionXmpPagePatch6}
\pastebutton{ExpressionXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{sqrt 3 + sqrt(2 + sqrt(-5))\bound{algnum1 }}
\end{paste}\end{patch}

\begin{patch}{ExpressionXmpPagePatch7}
\begin{paste}{ExpressionXmpPageFull7}{ExpressionXmpPageEmpty7}
\pastebutton{ExpressionXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{\% :: Expression Integer\free{algnum1 }}
\indentrel{3}\begin{verbatim}


   (7)  \\- 5  + 2  + \3
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExpressionXmpPageEmpty7}
\begin{paste}{ExpressionXmpPageEmpty7}{ExpressionXmpPagePatch7}
\pastebutton{ExpressionXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{\% :: Expression Integer\free{algnum1 }}
\end{paste}\end{patch}

\begin{patch}{ExpressionXmpPagePatch8}
\begin{paste}{ExpressionXmpPageFull8}{ExpressionXmpPageEmpty8}
\pastebutton{ExpressionXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{height mainKernel sin(x + 4)}
\indentrel{3}\begin{verbatim}
   (8)  2
                                  Type: PositiveInteger
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExpressionXmpPageEmpty8}
\begin{paste}{ExpressionXmpPageEmpty8}{ExpressionXmpPagePatch8}
\pastebutton{ExpressionXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{height mainKernel sin(x + 4)}
\end{paste}\end{patch}

\begin{patch}{ExpressionXmpPagePatch9}
\begin{paste}{ExpressionXmpPageFull9}{ExpressionXmpPageEmpty9}
\pastebutton{ExpressionXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{e := (sin(x) - 4)**2 / ( 1 - 2*y*sqrt(- y) )\bound{e }}
\indentrel{3}\begin{verbatim}
                2
        - sin(x)  + 8sin(x) - 16
   (9)

              2y\- y  - 1
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExpressionXmpPageEmpty9}
\begin{paste}{ExpressionXmpPageEmpty9}{ExpressionXmpPagePatch9}
\pastebutton{ExpressionXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{e := (sin(x) - 4)**2 / ( 1 - 2*y*sqrt(- y) )\bound{e }}
\end{paste}\end{patch}

\begin{patch}{ExpressionXmpPagePatch10}
\begin{paste}{ExpressionXmpPageFull10}{ExpressionXmpPageEmpty10}
\pastebutton{ExpressionXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{numer e\free{e }}
\indentrel{3}\begin{verbatim}
                 2
   (10)  - sin(x)  + 8sin(x) - 16
Type: SparseMultivariatePolynomial(Integer,Kernel Expression Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExpressionXmpPageEmpty10}
\begin{paste}{ExpressionXmpPageEmpty10}{ExpressionXmpPagePatch10}
\pastebutton{ExpressionXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{numer e\free{e }}
\end{paste}\end{patch}

\begin{patch}{ExpressionXmpPagePatch11}
\begin{paste}{ExpressionXmpPageFull11}{ExpressionXmpPageEmpty11}
\pastebutton{ExpressionXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{denom e\free{e }}
```

```
\indentrel{3}\begin{verbatim}

   (11)  2y\- y  - 1
Type: SparseMultivariatePolynomial(Integer,Kernel Expression Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExpressionXmpPageEmpty11}
\begin{paste}{ExpressionXmpPageEmpty11}{ExpressionXmpPagePatch11}
\pastebutton{ExpressionXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{denom e\free{e }}
\end{paste}\end{patch}

\begin{patch}{ExpressionXmpPagePatch12}
\begin{paste}{ExpressionXmpPageFull12}{ExpressionXmpPageEmpty12}
\pastebutton{ExpressionXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{D(e, x)\free{e }}
\indentrel{3}\begin{verbatim}
   (12)

       (4y cos(x)sin(x) - 16y cos(x))\- y
     +
       - 2cos(x)sin(x) + 8cos(x)
  /
             3
     4y\- y  + 4y  - 1
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExpressionXmpPageEmpty12}
\begin{paste}{ExpressionXmpPageEmpty12}{ExpressionXmpPagePatch12}
\pastebutton{ExpressionXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{D(e, x)\free{e }}
\end{paste}\end{patch}

\begin{patch}{ExpressionXmpPagePatch13}
\begin{paste}{ExpressionXmpPageFull13}{ExpressionXmpPageEmpty13}
\pastebutton{ExpressionXmpPageFull13}{\hidepaste}
\tab{5}\spadcommand{D(e, [x, y], [1, 2])\free{e }}
\indentrel{3}\begin{verbatim}
   (13)
                 7       4
           (- 2304y  + 960y )cos(x)sin(x)
         +
               7        4
           (9216y  - 3840y )cos(x)
      *

         \- y
```

```
     +
              9        6       3
       (- 960y  + 2160y  - 180y  - 3)cos(x)sin(x)
     +
             9        6       3
       (3840y  - 8640y  + 720y  + 12)cos(x)
  /
           12        9        6       3
       (256y   - 1792y  + 1120y  - 112y  + 1)\- y
     +
              11        8       5      2
       - 1024y   + 1792y  - 448y  + 16y
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExpressionXmpPageEmpty13}
\begin{paste}{ExpressionXmpPageEmpty13}{ExpressionXmpPagePatch13}
\pastebutton{ExpressionXmpPageEmpty13}{\showpaste}
\tab{5}\spadcommand{D(e, [x, y], [1, 2])\free{e }}
\end{paste}\end{patch}

\begin{patch}{ExpressionXmpPagePatch14}
\begin{paste}{ExpressionXmpPageFull14}{ExpressionXmpPageEmpty14}
\pastebutton{ExpressionXmpPageFull14}{\hidepaste}
\tab{5}\spadcommand{complexNumeric(cos(2 - 3*\%i))}
\indentrel{3}\begin{verbatim}
   (14)
   - 4.1896256909 688072301 + 9.1092278937 55336598 %i
                                    Type: Complex Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExpressionXmpPageEmpty14}
\begin{paste}{ExpressionXmpPageEmpty14}{ExpressionXmpPagePatch14}
\pastebutton{ExpressionXmpPageEmpty14}{\showpaste}
\tab{5}\spadcommand{complexNumeric(cos(2 - 3*\%i))}
\end{paste}\end{patch}

\begin{patch}{ExpressionXmpPagePatch15}
\begin{paste}{ExpressionXmpPageFull15}{ExpressionXmpPageEmpty15}
\pastebutton{ExpressionXmpPageFull15}{\hidepaste}
\tab{5}\spadcommand{numeric(tan 3.8)}
\indentrel{3}\begin{verbatim}
   (15)  0.7735560905 0312607286
                                            Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExpressionXmpPageEmpty15}
```

```
\begin{paste}{ExpressionXmpPageEmpty15}{ExpressionXmpPagePatch15}
\pastebutton{ExpressionXmpPageEmpty15}{\showpaste}
\tab{5}\spadcommand{numeric(tan 3.8)}
\end{paste}\end{patch}

\begin{patch}{ExpressionXmpPagePatch16}
\begin{paste}{ExpressionXmpPageFull16}{ExpressionXmpPageEmpty16}
\pastebutton{ExpressionXmpPageFull16}{\hidepaste}
\tab{5}\spadcommand{e2 := cos(x**2 - y + 3)\bound{e2 }}
\indentrel{3}\begin{verbatim}
                  2
   (16)  cos(y - x  - 3)
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExpressionXmpPageEmpty16}
\begin{paste}{ExpressionXmpPageEmpty16}{ExpressionXmpPagePatch16}
\pastebutton{ExpressionXmpPageEmpty16}{\showpaste}
\tab{5}\spadcommand{e2 := cos(x**2 - y + 3)\bound{e2 }}
\end{paste}\end{patch}

\begin{patch}{ExpressionXmpPagePatch17}
\begin{paste}{ExpressionXmpPageFull17}{ExpressionXmpPageEmpty17}
\pastebutton{ExpressionXmpPageFull17}{\hidepaste}
\tab{5}\spadcommand{e3 := asin(e2) - \%pi/2\free{e2 }\bound{e3 }}
\indentrel{3}\begin{verbatim}
                2
   (17)  - y + x  + 3
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExpressionXmpPageEmpty17}
\begin{paste}{ExpressionXmpPageEmpty17}{ExpressionXmpPagePatch17}
\pastebutton{ExpressionXmpPageEmpty17}{\showpaste}
\tab{5}\spadcommand{e3 := asin(e2) - \%pi/2\free{e2 }\bound{e3 }}
\end{paste}\end{patch}

\begin{patch}{ExpressionXmpPagePatch18}
\begin{paste}{ExpressionXmpPageFull18}{ExpressionXmpPageEmpty18}
\pastebutton{ExpressionXmpPageFull18}{\hidepaste}
\tab{5}\spadcommand{e3 :: Polynomial Integer\free{e3 }}
\indentrel{3}\begin{verbatim}
                2
   (18)  - y + x  + 3
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ExpressionXmpPageEmpty18}
\begin{paste}{ExpressionXmpPageEmpty18}{ExpressionXmpPagePatch18}
\pastebutton{ExpressionXmpPageEmpty18}{\showpaste}
\tab{5}\spadcommand{e3 :: Polynomial Integer\free{e3 }}
\end{paste}\end{patch}

\begin{patch}{ExpressionXmpPagePatch19}
\begin{paste}{ExpressionXmpPageFull19}{ExpressionXmpPageEmpty19}
\pastebutton{ExpressionXmpPageFull19}{\hidepaste}
\tab{5}\spadcommand{e3 :: DMP([x, y], Integer)\free{e3 }}
\indentrel{3}\begin{verbatim}
          2
   (19)  x  - y + 3
 Type: DistributedMultivariatePolynomial([x,y],Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExpressionXmpPageEmpty19}
\begin{paste}{ExpressionXmpPageEmpty19}{ExpressionXmpPagePatch19}
\pastebutton{ExpressionXmpPageEmpty19}{\showpaste}
\tab{5}\spadcommand{e3 :: DMP([x, y], Integer)\free{e3 }}
\end{paste}\end{patch}

\begin{patch}{ExpressionXmpPagePatch20}
\begin{paste}{ExpressionXmpPageFull20}{ExpressionXmpPageEmpty20}
\pastebutton{ExpressionXmpPageFull20}{\hidepaste}
\tab{5}\spadcommand{sin \%pi}
\indentrel{3}\begin{verbatim}
   (20)  0
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExpressionXmpPageEmpty20}
\begin{paste}{ExpressionXmpPageEmpty20}{ExpressionXmpPagePatch20}
\pastebutton{ExpressionXmpPageEmpty20}{\showpaste}
\tab{5}\spadcommand{sin \%pi}
\end{paste}\end{patch}

\begin{patch}{ExpressionXmpPagePatch21}
\begin{paste}{ExpressionXmpPageFull21}{ExpressionXmpPageEmpty21}
\pastebutton{ExpressionXmpPageFull21}{\hidepaste}
\tab{5}\spadcommand{cos(\%pi / 4)}
\indentrel{3}\begin{verbatim}

         \2
   (21)
           2
                               Type: Expression Integer
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExpressionXmpPageEmpty21}
\begin{paste}{ExpressionXmpPageEmpty21}{ExpressionXmpPagePatch21}
\pastebutton{ExpressionXmpPageEmpty21}{\showpaste}
\tab{5}\spadcommand{cos(\%pi / 4)}
\end{paste}\end{patch}

\begin{patch}{ExpressionXmpPagePatch22}
\begin{paste}{ExpressionXmpPageFull22}{ExpressionXmpPageEmpty22}
\pastebutton{ExpressionXmpPageFull22}{\hidepaste}
\tab{5}\spadcommand{tan(x)**6 + 3*tan(x)**4 + 3*tan(x)**2 + 1\bound{tan6 }}
\indentrel{3}\begin{verbatim}
               6         4          2
   (22)  tan(x)  + 3tan(x)  + 3tan(x)  + 1
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExpressionXmpPageEmpty22}
\begin{paste}{ExpressionXmpPageEmpty22}{ExpressionXmpPagePatch22}
\pastebutton{ExpressionXmpPageEmpty22}{\showpaste}
\tab{5}\spadcommand{tan(x)**6 + 3*tan(x)**4 + 3*tan(x)**2 + 1\bound{tan6 }}
\end{paste}\end{patch}

\begin{patch}{ExpressionXmpPagePatch23}
\begin{paste}{ExpressionXmpPageFull23}{ExpressionXmpPageEmpty23}
\pastebutton{ExpressionXmpPageFull23}{\hidepaste}
\tab{5}\spadcommand{simplify \%\free{tan6 }}
\indentrel{3}\begin{verbatim}
            1
   (23)  -------
               6
         cos(x)
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExpressionXmpPageEmpty23}
\begin{paste}{ExpressionXmpPageEmpty23}{ExpressionXmpPagePatch23}
\pastebutton{ExpressionXmpPageEmpty23}{\showpaste}
\tab{5}\spadcommand{simplify \%\free{tan6 }}
\end{paste}\end{patch}
```

3.34 explot2d.ht

Plotting Functions of One Variable

— **explot2d.ht** —

```
\begin{page}{ExPlot2DFunctions}{Plotting Functions of One Variable}
\beginscroll
To plot a function {\em y = f(x)}, you need only specify the function and the
interval on which it is to be plotted.
\graphpaste{draw(sin(tan(x)) - tan(sin(x)),x = 0..6)}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExPlot2DFunctionsPatch1}
\begin{paste}{ExPlot2DFunctionsFull1}{ExPlot2DFunctionsEmpty1}
\pastebutton{ExPlot2DFunctionsFull1}{\hidepaste}
\tab{5}\spadgraph{draw(sin(tan(x)) - tan(sin(x)),x = 0..6)}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/explot2dfunctions1.view/image}}{viewalone\space{
\end{paste}\end{patch}

\begin{patch}{ExPlot2DFunctionsEmpty1}
\begin{paste}{ExPlot2DFunctionsEmpty1}{ExPlot2DFunctionsPatch1}
\pastebutton{ExPlot2DFunctionsEmpty1}{\showpaste}
\tab{5}\spadgraph{draw(sin(tan(x)) - tan(sin(x)),x = 0..6)}
\end{paste}\end{patch}
```

———-

Plotting Parametric Curves

— **explot2d.ht** —

```
\begin{page}{ExPlot2DParametric}{Plotting Parametric Curves}
\beginscroll
To plot a parametric curve defined by {\em x = f(t)},
{\em y = g(t)}, specify the functions
{\em f(t)} and {\em g(t)}
as arguments of the function ‘curve’, then give
the interval over which {\em t} is to range.
\graphpaste{draw(curve(9 * sin(3*t/4),8 * sin(t)),t = -4*\%pi..4*\%pi)}
\endscroll
\autobuttons
\end{page}
```

```
\begin{patch}{ExPlot2DParametricPatch1}
\begin{paste}{ExPlot2DParametricFull1}{ExPlot2DParametricEmpty1}
\pastebutton{ExPlot2DParametricFull1}{\hidepaste}
\tab{5}\spadgraph{draw(curve(9 * sin(3*t/4),8 * sin(t)),t = -4*\%pi..4*\%pi)}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/explot2dparametric1.view/image}}{view
\end{paste}\end{patch}

\begin{patch}{ExPlot2DParametricEmpty1}
\begin{paste}{ExPlot2DParametricEmpty1}{ExPlot2DParametricPatch1}
\pastebutton{ExPlot2DParametricEmpty1}{\showpaste}
\tab{5}\spadgraph{draw(curve(9 * sin(3*t/4),8 * sin(t)),t = -4*\%pi..4*\%pi)}
\end{paste}
\end{patch}
```

Plotting Using Polar Coordinates

— explot2d.ht —

```
\begin{page}{ExPlot2DPolar}{Plotting Using Polar Coordinates}
\beginscroll
To plot the function {\em r = f(theta)} in polar coordinates,
use the option {\em coordinates == polar}.
As usual,
call the function 'draw' and specify the function {\em f(theta)} and the
interval over which {\em theta} is to range.
\graphpaste{draw(sin(4*t/7),t = 0..14*\%pi,coordinates == polar)}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExPlot2DPolarPatch1}
\begin{paste}{ExPlot2DPolarFull1}{ExPlot2DPolarEmpty1}
\pastebutton{ExPlot2DPolarFull1}{\hidepaste}
\tab{5}\spadgraph{draw(sin(4*t/7),t = 0..14*\%pi,coordinates == polar)}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/explot2dpolar1.view/image}}{viewalone
\end{paste}\end{patch}

\begin{patch}{ExPlot2DPolarEmpty1}
\begin{paste}{ExPlot2DPolarEmpty1}{ExPlot2DPolarPatch1}
\pastebutton{ExPlot2DPolarEmpty1}{\showpaste}
\tab{5}\spadgraph{draw(sin(4*t/7),t = 0..14*\%pi,coordinates == polar)}
\end{paste}\end{patch}
```

———-

Plotting Plane Algebraic Curves

— explot2d.ht —

```
\begin{page}{ExPlot2DAlgebraic}{Plotting Plane Algebraic Curves}
\beginscroll
Axiom can also plot plane algebraic curves (i.e. curves defined by
an equation {\em f(x,y) = 0}) provided that the curve is non-singular
in the region to be sketched.
Here's an example:
\graphpaste{draw(y**2 + y - (x**3 - x) = 0, x, y, range == [-2..2,-2..1])}
Here the region of the sketch is {\em -2 <= x <= 2, -2 <= y <= 1}.
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExPlot2DAlgebraicPatch1}
\begin{paste}{ExPlot2DAlgebraicFull1}{ExPlot2DAlgebraicEmpty1}
\pastebutton{ExPlot2DAlgebraicFull1}{\hidepaste}
\tab{5}\spadgraph{draw(y**2 + y - (x**3 - x) = 0, x, y, range == [-2..2,-2..1])}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/explot2dalgebraic1.view/image}}{viewalone\space{
\end{paste}\end{patch}

\begin{patch}{ExPlot2DAlgebraicEmpty1}
\begin{paste}{ExPlot2DAlgebraicEmpty1}{ExPlot2DAlgebraicPatch1}
\pastebutton{ExPlot2DAlgebraicEmpty1}{\showpaste}
\tab{5}\spadgraph{draw(y**2 + y - (x**3 - x) = 0, x, y, range == [-2..2,-2..1])}
\end{paste}\end{patch}
```

———-

3.35 explot3d.ht

Plotting Functions of Two Variables

— explot3d.ht —

```
\begin{page}{ExPlot3DFunctions}{Plotting Functions of Two Variables}
\beginscroll
To plot a function {\em z = f(x,y)}, you need only specify the function
and the intervals over which the dependent variables will range.
```

```
For example, here's how you plot the function {\em z = cos(x*y)} as the
variables {\em x} and {\em y} both range between -3 and 3:
\graphpaste{draw(cos(x*y),x = -3..3,y = -3..3)}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExPlot3DFunctionsPatch1}
\begin{paste}{ExPlot3DFunctionsFull1}{ExPlot3DFunctionsEmpty1}
\pastebutton{ExPlot3DFunctionsFull1}{\hidepaste}
\tab{5}\spadgraph{draw(cos(x*y),x = -3..3,y = -3..3)}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/explot3dfunctions1.view/image}}{viewa
\end{paste}\end{patch}

\begin{patch}{ExPlot3DFunctionsEmpty1}
\begin{paste}{ExPlot3DFunctionsEmpty1}{ExPlot3DFunctionsPatch1}
\pastebutton{ExPlot3DFunctionsEmpty1}{\showpaste}
\tab{5}\spadgraph{draw(cos(x*y),x = -3..3,y = -3..3)}
\end{paste}
\end{patch}
```

Plotting Parametric Surfaces

— explot3d.ht —

```
\begin{page}{ExPlot3DParametricSurface}{Plotting Parametric Surfaces}
\beginscroll
To plot a parametric surface defined by {\em x = f(u,v)},
{\em y = g(u,v)}, {\em z = h(u,v)}, specify the functions
{\em f(u,v)}, {\em g(u,v)}, and {\em h(u,v)}
as arguments of the function `surface', then give
the intervals over which {\em u} and {\em v} are to range.
With parametric surfaces, we can create some interesting graphs.
Here's an egg:
\graphpaste{draw(surface(5*sin(u)*cos(v),4*sin(u)*sin(v),3*cos(u)),
u=0..\%pi,v=0..2*\%pi)}
Here's a cone:
\graphpaste{draw(surface(u*cos(v),u*sin(v),u),u=0..4,v=0..2*\%pi)}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExPlot3DParametricSurfacePatch1}
\begin{paste}{ExPlot3DParametricSurfaceFull1}{ExPlot3DParametricSurfaceEmpty1}
\pastebutton{ExPlot3DParametricSurfaceFull1}{\hidepaste}
```

```
\tab{5}\spadgraph{draw(surface(5*sin(u)*cos(v),4*sin(u)*sin(v),3*cos(u)),u=0..\%pi,v=0..2*\%pi)}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/explot3dparametricsurface1.view/image}}{viewalone
\end{paste}\end{patch}

\begin{patch}{ExPlot3DParametricSurfaceEmpty1}
\begin{paste}{ExPlot3DParametricSurfaceEmpty1}{ExPlot3DParametricSurfacePatch1}
\pastebutton{ExPlot3DParametricSurfaceEmpty1}{\showpaste}
\tab{5}\spadgraph{draw(surface(5*sin(u)*cos(v),4*sin(u)*sin(v),3*cos(u)),u=0..\%pi,v=0..2*\%pi)}
\end{paste}\end{patch}

\begin{patch}{ExPlot3DParametricSurfacePatch2}
\begin{paste}{ExPlot3DParametricSurfaceFull2}{ExPlot3DParametricSurfaceEmpty2}
\pastebutton{ExPlot3DParametricSurfaceFull2}{\hidepaste}
\tab{5}\spadgraph{draw(surface(u*cos(v),u*sin(v),u),u=0..4,v=0..2*\%pi)}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/explot3dparametricsurface2.view/image}}{viewalone
\end{paste}\end{patch}

\begin{patch}{ExPlot3DParametricSurfaceEmpty2}
\begin{paste}{ExPlot3DParametricSurfaceEmpty2}{ExPlot3DParametricSurfacePatch2}
\pastebutton{ExPlot3DParametricSurfaceEmpty2}{\showpaste}
\tab{5}\spadgraph{draw(surface(u*cos(v),u*sin(v),u),u=0..4,v=0..2*\%pi)}
\end{paste}\end{patch}
```

Plotting Parametric Curves

— explot3d.ht —

```
\begin{page}{ExPlot3DParametricCurve}{Plotting Parametric Curves}
\beginscroll
To plot a parametric curve defined by {\em x = f(t)},
{\em y = g(t)}, {\em z = h(t)}, specify the functions
{\em f(t)}, {\em g(t)}, and {\em h(t)}
as arguments of the function ‘curve’, then give
the interval over which {\em t} is to range.
Here is a spiral:
\graphpaste{draw(curve(cos(t),sin(t),t),t=0..6)}
Here is the {\em twisted cubic curve}:
\graphpaste{draw(curve(t,t**2,t**3),t=-3..3)}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExPlot3DParametricCurvePatch1}
\begin{paste}{ExPlot3DParametricCurveFull1}{ExPlot3DParametricCurveEmpty1}
\pastebutton{ExPlot3DParametricCurveFull1}{\hidepaste}
```

```
\tab{5}\spadgraph{draw(curve(cos(t),sin(t),t),t=0..6)}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/explot3dparametriccurve1.view/image}}
\end{paste}\end{patch}

\begin{patch}{ExPlot3DParametricCurveEmpty1}
\begin{paste}{ExPlot3DParametricCurveEmpty1}{ExPlot3DParametricCurvePatch1}
\pastebutton{ExPlot3DParametricCurveEmpty1}{\showpaste}
\tab{5}\spadgraph{draw(curve(cos(t),sin(t),t),t=0..6)}
\end{paste}\end{patch}

\begin{patch}{ExPlot3DParametricCurvePatch2}
\begin{paste}{ExPlot3DParametricCurveFull2}{ExPlot3DParametricCurveEmpty2}
\pastebutton{ExPlot3DParametricCurveFull2}{\hidepaste}
\tab{5}\spadgraph{draw(curve(t,t**2,t**3),t=-3..3)}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/explot3dparametriccurve2.view/image}}
\end{paste}\end{patch}

\begin{patch}{ExPlot3DParametricCurveEmpty2}
\begin{paste}{ExPlot3DParametricCurveEmpty2}{ExPlot3DParametricCurvePatch2}
\pastebutton{ExPlot3DParametricCurveEmpty2}{\showpaste}
\tab{5}\spadgraph{draw(curve(t,t**2,t**3),t=-3..3)}
\end{paste}
\end{patch}
```

3.36 expose.ht

Exposure

— expose.ht —

```
\begin{page}{helpExpose}{Exposure}
\beginscroll
Exposure determines what part of the Axiom library
is available to you when using Axiom.
At any time during your interactive Axiom session,
each constructor is either {\em exposed} or {\em unexposed}.
If a constructor is exposed, its operations are available to
the interpreter and therefore to you.
If a constructor is unexposed, the operations are not seen
by the interpreter and thus not available to you.
\par
If you are a beginner, you may only want
basic parts of the library exposed.
If you are an expert, you may want to have all parts of the
```

```
library exposed.
If you have an application that requires
only a segment of the library, you may want to arrange to expose
only that segment for your use.
\par
\endscroll
Additional Information:
\beginmenu
\menulink{What}{ExposureDef}\tab{8}What is an exposure group?
\menulink{System}{ExposureSystem}\tab{8}What exposure groups are
system defined?
% \menulink{User}{ExposureUser}\tab{8}How can I define my own?
\menulink{Details}{ExposureDetails}\tab{8}Some details on exposure
\endmenu
\end{page}
```

———-

System Defined Exposure Groups

— **expose.ht** —

```
\begin{page}{ExposureSystem}{System Defined Exposure Groups}
\beginscroll
Exposure is defined by {\em groups}.
Groups have names.
Seven exposure groups are system-defined:\beginmenu
\item\tab{3}{\em current}\tab{12}The currently active exposure group
\item\tab{3}{\em basic}\tab{12}The default value of {\em current}
\item\tab{3}{\em category}\tab{13}Category constructors not in {\em basic}
\item\tab{3}{\em domain}\tab{13}Domain constructors not in {\em basic}
\item\tab{3}{\em package}\tab{13}Package constructors not in {\em basic}
\item\tab{3}{\em default}\tab{13}Default constructors not in {\em basic}
\item\tab{3}{\em hidden}\tab{13}All constructors not in {\em basic}
\item\tab{3}{\em naglink}\tab{13}All constructors used in the Axiom NAG Link
\endmenu
\par
When you first use Axiom, the {\em current} exposure group is
set to {\em basic} and {\em naglink}. Using Hyperdoc or the system command
{\em expose}, you may
change the current exposure group by
adding or dropping constructors or by setting {\em current}
to an exposure group you have created.
\endscroll
Additional Information:
\beginmenu
\menulink{What}{ExposureDef}\tab{10}What is an exposure group?
```

```
%\menulink{User}{ExposureUser}\tab{10}How you can define your own exposure group
\endmenu
\end{page}
```

———-

What is an Exposure Group?

— expose.ht —

```
\begin{page}{ExposureDef}{What is an Exposure Group?}
\beginscroll
\par
An exposure group is a list of constructors to be exposed.
Those constructors on the list are exposed;
those not on the list are not exposed.
The library contains 4 kinds of constructors intuitively described as follows:
\beginmenu
\item\menuitemstyle{}{\em domain}\tab{10}Describes computational objects
and functions defined on these objects
\item\menuitemstyle{}{\em package}\tab{10}Describes functions which will
work over a variety of domains
\item\menuitemstyle{}{\em category}\tab{10}Names a set of operations
\item\menuitemstyle{}{\em default}\tab{10}Provides default functions for
a cateogry
\endmenu

An exposure group is defined by three lists:
\beginmenu
\item\menuitemstyle{}{\em groups}
\tab{13}A list of other (more basic) groups
\item\menuitemstyle{}{\em additions}
\tab{13}A list of explicit constructors to be included
\item\menuitemstyle{}{\em subtractions}
\tab{13}A list of explicit constructors to be dropped
\endmenu
You can define your own exposure groups: give them names and define the
three above lists to be anything you like.
Using Hyperdoc, you can conveniently edit your exposure groups,
install them as the {\em current} exposure, and so on.
\endscroll
\end{page}
```

———-

Details on Exposure

— **expose.ht** —

```
\begin{page}{ExposureDetails}{Details on Exposure}
\beginscroll
Exposure is generally defined by the set of domain and package
constructors you want to have available.  Category and default
constructors are generally implied.  A category constructor is exposed
if mentioned by {\em any} other constructor (including another
category).  A default constructor is exposed if its corresponding
category constructor is exposed.
\par
If you explicitly add a domain or package
constructor, its name will be put in an {\em Additions} list.
The system will also add automatically to the {\em Additions} list
any category explicity exported by that domain or package.
If that category has a corresponding default constructor, that
default constructor will also be added as well.
\par
If you like, you can explicitly drop a constructor.  Any such name is
added to the {\em Subtractions} list.  The system will drop this name
from the {\em Additions} list if it appears.
\par
If the package or domain takes arguments from an unexported
domain or declares that its arguments can come from a domain
which is a member of an unexported category, these constructors
will {\em not} be added.
\endscroll
\end{page}
```

3.37 exseries.ht

Converting Expressions to Series

— **exseries.ht** —

```
\begin{page}{ExSeriesConvert}{Converting Expressions to Series}
\beginscroll
You can convert a functional expression to a power series by using the
function 'series'.
Here's an example:
```

```
\spadpaste{series(sin(a*x),x = 0)}
This causes {\em sin(a*x)} to be expanded in powers of {\em (x - 0)},
that is, in powers of {\em x}.
You can have {\em sin(a*x)} expanded in powers of {\em (a - \%pi/4)} by
issuing the following command:
\spadpaste{series(sin(a*x),a = \%pi/4)}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExSeriesConvertPatch1}
\begin{paste}{ExSeriesConvertFull1}{ExSeriesConvertEmpty1}
\pastebutton{ExSeriesConvertFull1}{\hidepaste}
\tab{5}\spadcommand{series(sin(a*x),x = 0)}
\indentrel{3}\begin{verbatim}
   (1)
           3        5        7          9
          a   3    a   5    a    7     a     9
     a x -  x  +  x  -  x  +  x
           6      120      5040      362880
   +
          11
         a      11      12
     -  x   + O(x  )
       39916800
  Type: UnivariatePuiseuxSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExSeriesConvertEmpty1}
\begin{paste}{ExSeriesConvertEmpty1}{ExSeriesConvertPatch1}
\pastebutton{ExSeriesConvertEmpty1}{\showpaste}
\tab{5}\spadcommand{series(sin(a*x),x = 0)}
\end{paste}\end{patch}

\begin{patch}{ExSeriesConvertPatch2}
\begin{paste}{ExSeriesConvertFull2}{ExSeriesConvertEmpty2}
\pastebutton{ExSeriesConvertFull2}{\hidepaste}
\tab{5}\spadcommand{series(sin(a*x),a = \%pi/4)}
\indentrel{3}\begin{verbatim}
   (2)
        %pi x          %pi x      %pi
     sin() + x cos()(a - )
          4             4         4
   +
       2    %pi x                 3    %pi x
      x sin()              x cos()
              4         %pi 2           4         %pi 3
     -  (a - )  -  (a - )
            2           4             6            4
```

```
   +
      4    %pi x                5    %pi x
     x sin()               x cos()
           4         %pi 4           4         %pi 5
      (a - )  +  (a - )
         24            4           120           4
   +
        6    %pi x                7    %pi x
       x sin()               x cos()
             4         %pi 6           4         %pi 7
     -  (a - )  -  (a - )
           720           4          5040           4
   +
      8    %pi x                9    %pi x
     x sin()               x cos()
           4         %pi 8           4         %pi 9
      (a - )  +  (a - )
        40320          4         362880          4
   +
        10    %pi x
       x  sin()
              4         %pi 10          %pi 11
     -  (a - )   + O((a - )  )
          3628800         4               4
Type: UnivariatePuiseuxSeries(Expression Integer,a,pi/4)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExSeriesConvertEmpty2}
\begin{paste}{ExSeriesConvertEmpty2}{ExSeriesConvertPatch2}
\pastebutton{ExSeriesConvertEmpty2}{\showpaste}
\tab{5}\spadcommand{series(sin(a*x),a = \%pi/4)}
\end{paste}\end{patch}
```

———-

Manipulating Power Series

— exseries.ht —

```
\begin{page}{ExSeriesManipulate}{Manipulating Power Series}
\beginscroll
Once you have created a power series, you can perform arithmetic
operations on that series.
First compute the Taylor expansion of {\em 1/(1-x)}:
\spadpaste{f := series(1/(1-x),x = 0) \bound{f}}
Now compute the square of that series:
```

```
\spadpaste{f ** 2 \free{f}}
It's as easy as 1, 2, 3,...
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExSeriesManipulatePatch1}
\begin{paste}{ExSeriesManipulateFull1}{ExSeriesManipulateEmpty1}
\pastebutton{ExSeriesManipulateFull1}{\hidepaste}
\tab{5}\spadcommand{f := series(1/(1-x),x = 0)\bound{f }}
\indentrel{3}\begin{verbatim}
   (1)
             2    3    4    5    6    7    8    9    10
     1 + x + x  + x  + x  + x  + x  + x  + x  + x  + x
   +
        11
     O(x  )
  Type: UnivariatePuiseuxSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExSeriesManipulateEmpty1}
\begin{paste}{ExSeriesManipulateEmpty1}{ExSeriesManipulatePatch1}
\pastebutton{ExSeriesManipulateEmpty1}{\showpaste}
\tab{5}\spadcommand{f := series(1/(1-x),x = 0)\bound{f }}
\end{paste}\end{patch}

\begin{patch}{ExSeriesManipulatePatch2}
\begin{paste}{ExSeriesManipulateFull2}{ExSeriesManipulateEmpty2}
\pastebutton{ExSeriesManipulateFull2}{\hidepaste}
\tab{5}\spadcommand{f ** 2\free{f }}
\indentrel{3}\begin{verbatim}
   (2)
               2     3     4     5     6     7     8
     1 + 2x + 3x  + 4x  + 5x  + 6x  + 7x  + 8x  + 9x
   +
         9      10      11
     10x  + 11x   + O(x  )
  Type: UnivariatePuiseuxSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExSeriesManipulateEmpty2}
\begin{paste}{ExSeriesManipulateEmpty2}{ExSeriesManipulatePatch2}
\pastebutton{ExSeriesManipulateEmpty2}{\showpaste}
\tab{5}\spadcommand{f ** 2\free{f }}
\end{paste}
\end{patch}
```

———-

Functions on Power Series

— exseries.ht —

```
\begin{page}{ExSeriesFunctions}{Functions on Power Series}
\beginscroll
The usual elementary functions ({\em log}, {\em exp},
trigonometric functions, etc.)
are defined for power series.
You can create a power series:
% Warning: currently there are (interpretor) problems with converting
% rational functions and polynomials to power series.
\spadpaste{f := series(1/(1-x),x = 0) \bound{f1}}
and then apply these functions to the series:
\spadpaste{g := log(f) \free{f1} \bound{g}}
\spadpaste{exp(g) \free{g}}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExSeriesFunctionsPatch1}
\begin{paste}{ExSeriesFunctionsFull1}{ExSeriesFunctionsEmpty1}
\pastebutton{ExSeriesFunctionsFull1}{\hidepaste}
\tab{5}\spadcommand{f := series(1/(1-x),x = 0)\bound{f1 }}
\indentrel{3}\begin{verbatim}
   (1)
              2    3    4    5    6    7    8    9    10
     1 + x + x  + x  + x  + x  + x  + x  + x  + x  + x
   +
        11
     O(x  )
  Type: UnivariatePuiseuxSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExSeriesFunctionsEmpty1}
\begin{paste}{ExSeriesFunctionsEmpty1}{ExSeriesFunctionsPatch1}
\pastebutton{ExSeriesFunctionsEmpty1}{\showpaste}
\tab{5}\spadcommand{f := series(1/(1-x),x = 0)\bound{f1 }}
\end{paste}\end{patch}

\begin{patch}{ExSeriesFunctionsPatch2}
\begin{paste}{ExSeriesFunctionsFull2}{ExSeriesFunctionsEmpty2}
\pastebutton{ExSeriesFunctionsFull2}{\hidepaste}
\tab{5}\spadcommand{g := log(f)\free{f1 }\bound{g }}
\indentrel{3}\begin{verbatim}
```

```
   (2)
         1  2   1  3   1  4   1  5   1  6   1  7   1  8
     x +  x  +  x  +  x  +  x  +  x  +  x  +  x
         2      3      4      5      6      7      8
   +
     1  9    1  10    1  11      12
      x  +  x   +  x   + O(x  )
     9      10       11
  Type: UnivariatePuiseuxSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExSeriesFunctionsEmpty2}
\begin{paste}{ExSeriesFunctionsEmpty2}{ExSeriesFunctionsPatch2}
\pastebutton{ExSeriesFunctionsEmpty2}{\showpaste}
\tab{5}\spadcommand{g := log(f)\free{f1 }\bound{g }}
\end{paste}\end{patch}

\begin{patch}{ExSeriesFunctionsPatch3}
\begin{paste}{ExSeriesFunctionsFull3}{ExSeriesFunctionsEmpty3}
\pastebutton{ExSeriesFunctionsFull3}{\hidepaste}
\tab{5}\spadcommand{exp(g)\free{g }}
\indentrel{3}\begin{verbatim}
   (3)
              2    3    4    5    6    7    8    9    10
     1 + x + x  + x  + x  + x  + x  + x  + x  + x  + x
   +
        11
     O(x  )
  Type: UnivariatePuiseuxSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExSeriesFunctionsEmpty3}
\begin{paste}{ExSeriesFunctionsEmpty3}{ExSeriesFunctionsPatch3}
\pastebutton{ExSeriesFunctionsEmpty3}{\showpaste}
\tab{5}\spadcommand{exp(g)\free{g }}
\end{paste}\end{patch}
```

———-

Substituting Numerical Values in Power Series

— exseries.ht —

```
\begin{page}{ExSeriesSubstitution}
{Substituting Numerical Values in Power Series}
```

```
\beginscroll
Here's a way to obtain numerical approximations of
{\em e} from the Taylor series
expansion of {\em exp(x)}.
First you create the desired Taylor expansion:
\spadpaste{f := taylor(exp(x)) \bound{f2}}
Now you evaluate the series at the value {\em 1.0}:
% Warning: syntax for evaluating power series may change.
\spadpaste{eval(f,1.0) \free{f2}}
You get a sequence of partial sums.
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExSeriesSubstitutionPatch1}
\begin{paste}{ExSeriesSubstitutionFull1}{ExSeriesSubstitutionEmpty1}
\pastebutton{ExSeriesSubstitutionFull1}{\hidepaste}
\tab{5}\spadcommand{f := taylor(exp(x))\bound{f2 }}
\indentrel{3}\begin{verbatim}
   (1)
             1  2   1  3    1  4    1   5    1   6
     1 + x +  x  +  x  +  x  +  x  +  x
             2      6      24      120      720
   +
     1   7     1    8      1    9      1     10      11
    x  +  x  +  x  +  x   + O(x  )
   5040      40320      362880      3628800
   Type: UnivariateTaylorSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExSeriesSubstitutionEmpty1}
\begin{paste}{ExSeriesSubstitutionEmpty1}{ExSeriesSubstitutionPatch1}
\pastebutton{ExSeriesSubstitutionEmpty1}{\showpaste}
\tab{5}\spadcommand{f := taylor(exp(x))\bound{f2 }}
\end{paste}\end{patch}

\begin{patch}{ExSeriesSubstitutionPatch2}
\begin{paste}{ExSeriesSubstitutionFull2}{ExSeriesSubstitutionEmpty2}
\pastebutton{ExSeriesSubstitutionFull2}{\hidepaste}
\tab{5}\spadcommand{eval(f,1.0)\free{f2 }}
\indentrel{3}\begin{verbatim}
   (2)
   [1.0, 2.0, 2.5, 2.6666666666 666666667,
    2.7083333333 333333333, 2.7166666666 666666667,
    2.7180555555 555555556, 2.7182539682 53968254,
    2.7182787698 412698413, 2.7182815255 731922399, ...]
                          Type: Stream Expression Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ExSeriesSubstitutionEmpty2}
\begin{paste}{ExSeriesSubstitutionEmpty2}{ExSeriesSubstitutionPatch2}
\pastebutton{ExSeriesSubstitutionEmpty2}{\showpaste}
\tab{5}\spadcommand{eval(f,1.0)\free{f2 }}
\end{paste}\end{patch}
```

———-

3.38 exsum.ht

Summing the Entries of a List I

— exsum.ht —

```
\begin{page}{ExSumListEntriesI}{Summing the Entries of a List I}
\beginscroll
In Axiom, you can create lists of consecutive integers by giving the
first and last entries of the list.
Here's how you create a list of the integers between {\em 1} and {\em 15}:
\spadpaste{[i for i in 1..15]}
To sum the entries of a list, simply put {\em +/} in front of the list.
For example, the following command will sum the integers from 1 to 15:
\spadpaste{reduce(+,[i for i in 1..15])}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExSumListEntriesIPatch1}
\begin{paste}{ExSumListEntriesIFull1}{ExSumListEntriesIEmpty1}
\pastebutton{ExSumListEntriesIFull1}{\hidepaste}
\tab{5}\spadcommand{[i for i in 1..15]}
\indentrel{3}\begin{verbatim}
   (1)  [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
                              Type: List PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExSumListEntriesIEmpty1}
\begin{paste}{ExSumListEntriesIEmpty1}{ExSumListEntriesIPatch1}
\pastebutton{ExSumListEntriesIEmpty1}{\showpaste}
\tab{5}\spadcommand{[i for i in 1..15]}
\end{paste}\end{patch}

\begin{patch}{ExSumListEntriesIPatch2}
\begin{paste}{ExSumListEntriesIFull2}{ExSumListEntriesIEmpty2}
```

```
\pastebutton{ExSumListEntriesIFull2}{\hidepaste}
\tab{5}\spadcommand{reduce(+,[i for i in 1..15])}
\indentrel{3}\begin{verbatim}
   (2)  120
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExSumListEntriesIEmpty2}
\begin{paste}{ExSumListEntriesIEmpty2}{ExSumListEntriesIPatch2}
\pastebutton{ExSumListEntriesIEmpty2}{\showpaste}
\tab{5}\spadcommand{reduce(+,[i for i in 1..15])}
\end{paste}\end{patch}
```

————-

Summing the Entries of a List II

— exsum.ht —

```
\begin{page}{ExSumListEntriesII}{Summing the Entries of a List II}
\beginscroll
In Axiom, you can also create lists whose elements are some expression
{\em f(n)} as the parameter n ranges between two integers.
For example, the following command will create a list of the squares of
the integers between {\em 5} and {\em 20}:
\spadpaste{[n**2 for n in 5..20]}
You can also compute the sum of the entries of this list:
\spadpaste{reduce(+,[n**2 for n in 5..20])}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExSumListEntriesIIPatch1}
\begin{paste}{ExSumListEntriesIIFull1}{ExSumListEntriesIIEmpty1}
\pastebutton{ExSumListEntriesIIFull1}{\hidepaste}
\tab{5}\spadcommand{[n**2 for n in 5..20]}
\indentrel{3}\begin{verbatim}
   (1)
   [25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225,
    256, 289, 324, 361, 400]
                             Type: List PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExSumListEntriesIIEmpty1}
\begin{paste}{ExSumListEntriesIIEmpty1}{ExSumListEntriesIIPatch1}
```

```
\pastebutton{ExSumListEntriesIIEmpty1}{\showpaste}
\tab{5}\spadcommand{[n**2 for n in 5..20]}
\end{paste}\end{patch}

\begin{patch}{ExSumListEntriesIIPatch2}
\begin{paste}{ExSumListEntriesIIFull2}{ExSumListEntriesIIEmpty2}
\pastebutton{ExSumListEntriesIIFull2}{\hidepaste}
\tab{5}\spadcommand{reduce(+,[n**2 for n in 5..20])}
\indentrel{3}\begin{verbatim}
   (2)  2840
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExSumListEntriesIIEmpty2}
\begin{paste}{ExSumListEntriesIIEmpty2}{ExSumListEntriesIIPatch2}
\pastebutton{ExSumListEntriesIIEmpty2}{\showpaste}
\tab{5}\spadcommand{reduce(+,[n**2 for n in 5..20])}
\end{paste}\end{patch}
```

———-

Approximating e

— **exsum.ht** —

```
\begin{page}{ExSumApproximateE}{Approximating e}
\beginscroll
You can obtain a numerical approximation of the number {\em e} by summing the
entries of the following list:
\spadpaste{reduce(+,[1.0/factorial(n) for n in 0..20])}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExSumApproximateEPatch1}
\begin{paste}{ExSumApproximateEFull1}{ExSumApproximateEEmpty1}
\pastebutton{ExSumApproximateEFull1}{\hidepaste}
\tab{5}\spadcommand{reduce(+,[1.0/factorial(n) for n in 0..20])}
\indentrel{3}\begin{verbatim}
   (1)  2.7182818284 590452354
                                            Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExSumApproximateEEmpty1}
\begin{paste}{ExSumApproximateEEmpty1}{ExSumApproximateEPatch1}
```

```
\pastebutton{ExSumApproximateEEmpty1}{\showpaste}
\tab{5}\spadcommand{reduce(+,[1.0/factorial(n) for n in 0..20])}
\end{paste}\end{patch}
```

———-

Closed Form Summations

— **exsum.ht** —

```
\begin{page}{ExSumClosedForm}{Closed Form Summations}
\beginscroll
In a previous example, we found the sum of the squares of the integers
between {\em 5} and {\em 20}.
We can also use Axiom to find a formula for the sum of the squares of
the integers between {\em a} and {\em b}, where {\em a} and {\em b}
are integers which will remain
unspecified:
\spadpaste{s := sum(k**2,k = a..b) \bound{s}}
{\em sum(k**2,k = a..b)} returns the sum of {\em k**2} as the index {\em k}
runs from {\em a} to {\em b}.
Let's check our answer in one particular case by substituting specific values
for {\em a} and {\em b} in our formula:
% Warning: syntax for polynomial evaluation will probably change.
\spadpaste{eval(s,[a,b],[1,25]) \free{s}}
\spadpaste{reduce(+,[i**2 for i in 1..25])}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExSumClosedFormPatch1}
\begin{paste}{ExSumClosedFormFull1}{ExSumClosedFormEmpty1}
\pastebutton{ExSumClosedFormFull1}{\hidepaste}
\tab{5}\spadcommand{s := sum(k**2,k = a..b)\bound{s }}
\indentrel{3}\begin{verbatim}
          3     2         3     2
        2b  + 3b  + b - 2a  + 3a  - a
   (1)
                      6
                      Type: Fraction Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExSumClosedFormEmpty1}
\begin{paste}{ExSumClosedFormEmpty1}{ExSumClosedFormPatch1}
\pastebutton{ExSumClosedFormEmpty1}{\showpaste}
\tab{5}\spadcommand{s := sum(k**2,k = a..b)\bound{s }}
```

```
\end{paste}\end{patch}

\begin{patch}{ExSumClosedFormPatch2}
\begin{paste}{ExSumClosedFormFull2}{ExSumClosedFormEmpty2}
\pastebutton{ExSumClosedFormFull2}{\hidepaste}
\tab{5}\spadcommand{eval(s,[a,b],[1,25])\free{s }}
\indentrel{3}\begin{verbatim}
   (2)  5525
                     Type: Fraction Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExSumClosedFormEmpty2}
\begin{paste}{ExSumClosedFormEmpty2}{ExSumClosedFormPatch2}
\pastebutton{ExSumClosedFormEmpty2}{\showpaste}
\tab{5}\spadcommand{eval(s,[a,b],[1,25])\free{s }}
\end{paste}\end{patch}

\begin{patch}{ExSumClosedFormPatch3}
\begin{paste}{ExSumClosedFormFull3}{ExSumClosedFormEmpty3}
\pastebutton{ExSumClosedFormFull3}{\hidepaste}
\tab{5}\spadcommand{reduce(+,[i**2 for i in 1..25])}
\indentrel{3}\begin{verbatim}
   (3)  5525
                                 Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExSumClosedFormEmpty3}
\begin{paste}{ExSumClosedFormEmpty3}{ExSumClosedFormPatch3}
\pastebutton{ExSumClosedFormEmpty3}{\showpaste}
\tab{5}\spadcommand{reduce(+,[i**2 for i in 1..25])}
\end{paste}\end{patch}
```

———-

Sums of Cubes

— exsum.ht —

```
\begin{page}{ExSumCubes}{Sums of Cubes}
\beginscroll
Here's a cute example.
First compute the sum of the cubes from {\em 1} to {\em n}:
\spadpaste{sum(k**3,k = 1..n)}
Then compute the square of the sum of the integers from {\em 1} to {\em n}:
\spadpaste{sum(k,k = 1..n) ** 2}
```

```
The answers are the same.
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExSumCubesPatch1}
\begin{paste}{ExSumCubesFull1}{ExSumCubesEmpty1}
\pastebutton{ExSumCubesFull1}{\hidepaste}
\tab{5}\spadcommand{sum(k**3,k = 1..n)}
\indentrel{3}\begin{verbatim}
         4     3    2
        n  + 2n  + n
   (1)  ------------
              4
                      Type: Fraction Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExSumCubesEmpty1}
\begin{paste}{ExSumCubesEmpty1}{ExSumCubesPatch1}
\pastebutton{ExSumCubesEmpty1}{\showpaste}
\tab{5}\spadcommand{sum(k**3,k = 1..n)}
\end{paste}\end{patch}

\begin{patch}{ExSumCubesPatch2}
\begin{paste}{ExSumCubesFull2}{ExSumCubesEmpty2}
\pastebutton{ExSumCubesFull2}{\hidepaste}
\tab{5}\spadcommand{sum(k,k = 1..n) ** 2}
\indentrel{3}\begin{verbatim}
         4     3    2
        n  + 2n  + n
   (2)  ------------
              4
                      Type: Fraction Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExSumCubesEmpty2}
\begin{paste}{ExSumCubesEmpty2}{ExSumCubesPatch2}
\pastebutton{ExSumCubesEmpty2}{\showpaste}
\tab{5}\spadcommand{sum(k,k = 1..n) ** 2}
\end{paste}\end{patch}
```

Sums of Polynomials

— exsum.ht —

```
\begin{page}{ExSumPolynomial}{Sums of Polynomials}
\beginscroll
Axiom can compute {\em sum(f,k = a..b)}
when {\em f} is any polynomial in {\em k}, even
one with parameters.
\spadpaste{sum(3*k**2/(c**2 + 1) + 12*k/d,k = (3*a)..(4*b))}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExSumPolynomialPatch1}
\begin{paste}{ExSumPolynomialFull1}{ExSumPolynomialEmpty1}
\pastebutton{ExSumPolynomialFull1}{\hidepaste}
\tab{5}\spadcommand{sum(3*k**2/(c**2 + 1) + 12*k/d,k = (3*a)..(4*b))}
\indentrel{3}\begin{verbatim}
   (1)
            3      2           3      2
       (128b  + 48b  + 4b - 54a  + 27a  - 3a)d
     +
            2             2        2       2             2
       (192b  + 48b - 108a  + 36a)c  + 192b  + 48b - 108a
     +
       36a
  /
        2
     (2c  + 2)d
           Type: Union(Fraction Polynomial Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExSumPolynomialEmpty1}
\begin{paste}{ExSumPolynomialEmpty1}{ExSumPolynomialPatch1}
\pastebutton{ExSumPolynomialEmpty1}{\showpaste}
\tab{5}\spadcommand{sum(3*k**2/(c**2 + 1) + 12*k/d,k = (3*a)..(4*b))}
\end{paste}\end{patch}
```

———–

```
%\begin{page}{ExSumRationalFunction}{Sums of Rational Functions}
%\beginscroll
%Axiom can compute {\em sum(f,k = a..b)} for some rational functions (quotients
%of polynomials) in {\em k}.
%\spadpaste{sum(1/(k * (k + 2)),k = 1..n)}
%However, the method used (Gosper's method) does not guarantee an answer
```

```
%in every case.
%Here's an example of a sum that the method cannot compute:
%\spadpaste{sum(1/(k**2 + 1),k = 1..n)}
%\endscroll
%\autobuttons\end{page}
```

Sums of General Functions

— **exsum.ht** —

```
\begin{page}{ExSumGeneralFunction}{Sums of General Functions}
\beginscroll
Gosper's method can also be used to compute {\em sum(f,k = a..b)}
for some functions
f which are not rational functions in {\em k}.
Here's an example:
\spadpaste{sum(k * x**k,k = 1..n)}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExSumGeneralFunctionPatch1}
\begin{paste}{ExSumGeneralFunctionFull1}{ExSumGeneralFunctionEmpty1}
\pastebutton{ExSumGeneralFunctionFull1}{\hidepaste}
\tab{5}\spadcommand{sum(k * x**k,k = 1..n)}
\indentrel{3}\begin{verbatim}
            2               n
        (n x  + (- n - 1)x)x  + x
   (1)  
                2
               x  - 2x + 1
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExSumGeneralFunctionEmpty1}
\begin{paste}{ExSumGeneralFunctionEmpty1}{ExSumGeneralFunctionPatch1}
\pastebutton{ExSumGeneralFunctionEmpty1}{\showpaste}
\tab{5}\spadcommand{sum(k * x**k,k = 1..n)}
\end{paste}\end{patch}
```

————

Infinite Sums

Provide a package for infinite sums

— **exsum.ht** —

```
\begin{page}{ExSumInfinite}{Infinite Sums}
\beginscroll
In a few cases, we can compute infinite sums by taking limits of finite
sums.
For instance, you can compute the sum of {\em 1/(k * (k + 2))} as {\em k}
ranges from
{\em 1} to {\em infinity}.
Use {\em \%plusInfinity} to denote ‘plus infinity’.
\spadpaste{limit( sum(1/(k * (k + 2)),k = 1..n) ,n = \%plusInfinity)}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ExSumInfinitePatch1}
\begin{paste}{ExSumInfiniteFull1}{ExSumInfiniteEmpty1}
\pastebutton{ExSumInfiniteFull1}{\hidepaste}
\tab{5}\spadcommand{limit( sum(1/(k * (k + 2)),k = 1..n) ,n = \%plusInfinity)}
\indentrel{3}\begin{verbatim}
        3
   (1)
        4
Type: Union(OrderedCompletion Fraction Polynomial Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ExSumInfiniteEmpty1}
\begin{paste}{ExSumInfiniteEmpty1}{ExSumInfinitePatch1}
\pastebutton{ExSumInfiniteEmpty1}{\showpaste}
\tab{5}\spadcommand{limit( sum(1/(k * (k + 2)),k = 1..n) ,n = \%plusInfinity)}
\end{paste}\end{patch}
```

———-

3.39 farray.ht

FlexibleArray

⇒ “notitle” (OneDimensionalArrayXmpPage) 3.4 on page 138
⇒ “notitle” (VectorXmpPage) 3.114 on page 1351

— **farray.ht** —

```
\begin{page}{FlexibleArrayXmpPage}{FlexibleArray}
\beginscroll
The \spadtype{FlexibleArray} domain constructor creates
one-dimensional arrays of elements of the same type.
Flexible arrays are an attempt to provide a data type that has the
best features of both one-dimensional arrays (fast, random access
to elements) and lists (flexibility).
They are implemented by a fixed block of storage.
When necessary for expansion, a new, larger block of storage is
allocated and the elements from the old storage area are copied
into the new block.

Flexible arrays have available most of the operations provided by
\spadtype{OneDimensionalArray}
(see \downlink{`OneDimensionalArray'}{OneDimensionalArrayXmpPage}
\ignore{OneDimensionalArray}
and \downlink{`Vector'}{VectorXmpPage}\ignore{Vector}).
Since flexible arrays are also of category
\spadtype{ExtensibleLinearAggregate}, they have operations
\spadfunX{concat}, \spadfunX{delete}, \spadfunX{insert},
\spadfunX{merge}, \spadfunX{remove}, \spadfunX{removeDuplicates},
and \spadfunX{select}.
In addition, the operations \spadfun{physicalLength} and
\spadfunX{physicalLength} provide user-control over expansion and
contraction.

\xtc{
A convenient way to create a flexible array is to apply
the operation \spadfun{flexibleArray} to a list of values.
}{
\spadpaste{flexibleArray [i for i in 1..6]}
}
\xtc{
Create a flexible array of six zeroes.
}{
\spadpaste{f : FARRAY INT := new(6,0)\bound{f}}
}
\xtc{
For \texht{$i=1\ldots 6$}{i = 1..6},
set the \eth{\smath{i}} element to \smath{i}.
Display \spad{f}.
}{
\spadpaste{for i in 1..6 repeat f.i := i; f\bound{f1}\free{f}}
}
\xtc{
Initially, the physical length is the same as the number of elements.
}{
\spadpaste{physicalLength f\free{f1}}
}
\xtc{
```

```
Add an element to the end of \spad{f}.
}{
\spadpaste{concat!(f,11)\bound{f2}\free{f1}}
}
\xtc{
See that its physical length has grown.
}{
\spadpaste{physicalLength f\free{f2}}
}
\xtc{
Make \spad{f} grow to have room for \spad{15} elements.
}{
\spadpaste{physicalLength!(f,15)\bound{f3}\free{f2}}
}
\xtc{
Concatenate the elements of \spad{f} to itself.
The physical length allows room for three more values at the end.
}{
\spadpaste{concat!(f,f)\bound{f4}\free{f3}}
}
\xtc{
Use \spadfunX{insert} to add an element to the front of
a flexible array.
}{
\spadpaste{insert!(22,f,1)\bound{f5}\free{f4}}
}
\xtc{
Create a second flexible array from \spad{f} consisting of the
elements from index 10 forward.
}{
\spadpaste{g := f(10..)\bound{g}\free{f5}}
}
\xtc{
Insert this array at the front of \spad{f}.
}{
\spadpaste{insert!(g,f,1)\bound{g1}\free{g f5}}
}
\xtc{
Merge the flexible array \spad{f} into \spad{g} after sorting each in place.
}{
\spadpaste{merge!(sort! f, sort! g)\bound{f6}\free{g f5}}
}
\xtc{
Remove duplicates in place.
}{
\spadpaste{removeDuplicates! f\bound{f7}\free{f6}}
}
\xtc{
Remove all odd integers.
}{
```

```
\spadpaste{select!(i +-> even? i,f)\bound{f8}\free{f7}}
}
\xtc{
All these operations have shrunk the physical length of \spad{f}.
}{
\spadpaste{physicalLength f\free{b8}}
}
\xtc{
To force Axiom not to shrink flexible arrays call the
\spadfun{shrinkable} operation with the argument \axiom{false}.
You must package call this operation.
The previous value is returned.
}{
\spadpaste{shrinkable(false)\$FlexibleArray(Integer)}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{FlexibleArrayXmpPagePatch1}
\begin{paste}{FlexibleArrayXmpPageFull1}{FlexibleArrayXmpPageEmpty1}
\pastebutton{FlexibleArrayXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{flexibleArray [i for i in 1..6]}
\indentrel{3}\begin{verbatim}
   (1)  [1,2,3,4,5,6]
                    Type: FlexibleArray PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FlexibleArrayXmpPageEmpty1}
\begin{paste}{FlexibleArrayXmpPageEmpty1}{FlexibleArrayXmpPagePatch1}
\pastebutton{FlexibleArrayXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{flexibleArray [i for i in 1..6]}
\end{paste}\end{patch}

\begin{patch}{FlexibleArrayXmpPagePatch2}
\begin{paste}{FlexibleArrayXmpPageFull2}{FlexibleArrayXmpPageEmpty2}
\pastebutton{FlexibleArrayXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{f : FARRAY INT := new(6,0)\bound{f }}
\indentrel{3}\begin{verbatim}
   (2)  [0,0,0,0,0,0]
                            Type: FlexibleArray Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FlexibleArrayXmpPageEmpty2}
\begin{paste}{FlexibleArrayXmpPageEmpty2}{FlexibleArrayXmpPagePatch2}
\pastebutton{FlexibleArrayXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{f : FARRAY INT := new(6,0)\bound{f }}
```

```
\end{paste}\end{patch}

\begin{patch}{FlexibleArrayXmpPagePatch3}
\begin{paste}{FlexibleArrayXmpPageFull3}{FlexibleArrayXmpPageEmpty3}
\pastebutton{FlexibleArrayXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{for i in 1..6 repeat f.i := i; f\bound{f1 }\free{f }}
\indentrel{3}\begin{verbatim}
   (3)  [1,2,3,4,5,6]
                            Type: FlexibleArray Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FlexibleArrayXmpPageEmpty3}
\begin{paste}{FlexibleArrayXmpPageEmpty3}{FlexibleArrayXmpPagePatch3}
\pastebutton{FlexibleArrayXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{for i in 1..6 repeat f.i := i; f\bound{f1 }\free{f }}
\end{paste}\end{patch}

\begin{patch}{FlexibleArrayXmpPagePatch4}
\begin{paste}{FlexibleArrayXmpPageFull4}{FlexibleArrayXmpPageEmpty4}
\pastebutton{FlexibleArrayXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{physicalLength f\free{f1 }}
\indentrel{3}\begin{verbatim}
   (4)  6
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FlexibleArrayXmpPageEmpty4}
\begin{paste}{FlexibleArrayXmpPageEmpty4}{FlexibleArrayXmpPagePatch4}
\pastebutton{FlexibleArrayXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{physicalLength f\free{f1 }}
\end{paste}\end{patch}

\begin{patch}{FlexibleArrayXmpPagePatch5}
\begin{paste}{FlexibleArrayXmpPageFull5}{FlexibleArrayXmpPageEmpty5}
\pastebutton{FlexibleArrayXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{concat!(f,11)\bound{f2 }\free{f1 }}
\indentrel{3}\begin{verbatim}
   (5)  [1,2,3,4,5,6,11]
                            Type: FlexibleArray Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FlexibleArrayXmpPageEmpty5}
\begin{paste}{FlexibleArrayXmpPageEmpty5}{FlexibleArrayXmpPagePatch5}
\pastebutton{FlexibleArrayXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{concat!(f,11)\bound{f2 }\free{f1 }}
\end{paste}\end{patch}
```

```
\begin{patch}{FlexibleArrayXmpPagePatch6}
\begin{paste}{FlexibleArrayXmpPageFull6}{FlexibleArrayXmpPageEmpty6}
\pastebutton{FlexibleArrayXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{physicalLength f\free{f2 }}
\indentrel{3}\begin{verbatim}
   (6)  10
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FlexibleArrayXmpPageEmpty6}
\begin{paste}{FlexibleArrayXmpPageEmpty6}{FlexibleArrayXmpPagePatch6}
\pastebutton{FlexibleArrayXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{physicalLength f\free{f2 }}
\end{paste}\end{patch}

\begin{patch}{FlexibleArrayXmpPagePatch7}
\begin{paste}{FlexibleArrayXmpPageFull7}{FlexibleArrayXmpPageEmpty7}
\pastebutton{FlexibleArrayXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{physicalLength!(f,15)\bound{f3 }\free{f2 }}
\indentrel{3}\begin{verbatim}
   (7)  [1,2,3,4,5,6,11]
                            Type: FlexibleArray Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FlexibleArrayXmpPageEmpty7}
\begin{paste}{FlexibleArrayXmpPageEmpty7}{FlexibleArrayXmpPagePatch7}
\pastebutton{FlexibleArrayXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{physicalLength!(f,15)\bound{f3 }\free{f2 }}
\end{paste}\end{patch}

\begin{patch}{FlexibleArrayXmpPagePatch8}
\begin{paste}{FlexibleArrayXmpPageFull8}{FlexibleArrayXmpPageEmpty8}
\pastebutton{FlexibleArrayXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{concat!(f,f)\bound{f4 }\free{f3 }}
\indentrel{3}\begin{verbatim}
   (8)  [1,2,3,4,5,6,11,1,2,3,4,5,6,11]
                            Type: FlexibleArray Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FlexibleArrayXmpPageEmpty8}
\begin{paste}{FlexibleArrayXmpPageEmpty8}{FlexibleArrayXmpPagePatch8}
\pastebutton{FlexibleArrayXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{concat!(f,f)\bound{f4 }\free{f3 }}
\end{paste}\end{patch}

\begin{patch}{FlexibleArrayXmpPagePatch9}
\begin{paste}{FlexibleArrayXmpPageFull9}{FlexibleArrayXmpPageEmpty9}
```

```
\pastebutton{FlexibleArrayXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{insert!(22,f,1)\bound{f5 }\free{f4 }}
\indentrel{3}\begin{verbatim}
   (9)  [22,1,2,3,4,5,6,11,1,2,3,4,5,6,11]
                            Type: FlexibleArray Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FlexibleArrayXmpPageEmpty9}
\begin{paste}{FlexibleArrayXmpPageEmpty9}{FlexibleArrayXmpPagePatch9}
\pastebutton{FlexibleArrayXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{insert!(22,f,1)\bound{f5 }\free{f4 }}
\end{paste}\end{patch}

\begin{patch}{FlexibleArrayXmpPagePatch10}
\begin{paste}{FlexibleArrayXmpPageFull10}{FlexibleArrayXmpPageEmpty10}
\pastebutton{FlexibleArrayXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{g := f(10..)\bound{g }\free{f5 }}
\indentrel{3}\begin{verbatim}
   (10)  [2,3,4,5,6,11]
                            Type: FlexibleArray Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FlexibleArrayXmpPageEmpty10}
\begin{paste}{FlexibleArrayXmpPageEmpty10}{FlexibleArrayXmpPagePatch10}
\pastebutton{FlexibleArrayXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{g := f(10..)\bound{g }\free{f5 }}
\end{paste}\end{patch}

\begin{patch}{FlexibleArrayXmpPagePatch11}
\begin{paste}{FlexibleArrayXmpPageFull11}{FlexibleArrayXmpPageEmpty11}
\pastebutton{FlexibleArrayXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{insert!(g,f,1)\bound{g1 }\free{g f5 }}
\indentrel{3}\begin{verbatim}
   (11)  [2,3,4,5,6,11,22,1,2,3,4,5,6,11,1,2,3,4,5,6,11]
                            Type: FlexibleArray Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FlexibleArrayXmpPageEmpty11}
\begin{paste}{FlexibleArrayXmpPageEmpty11}{FlexibleArrayXmpPagePatch11}
\pastebutton{FlexibleArrayXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{insert!(g,f,1)\bound{g1 }\free{g f5 }}
\end{paste}\end{patch}

\begin{patch}{FlexibleArrayXmpPagePatch12}
\begin{paste}{FlexibleArrayXmpPageFull12}{FlexibleArrayXmpPageEmpty12}
\pastebutton{FlexibleArrayXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{merge!(sort! f, sort! g)\bound{f6 }\free{g f5 }}
```

```
\indentrel{3}\begin{verbatim}
   (12)
   [1, 1, 2, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5,
    6, 6, 6, 6, 11, 11, 11, 11, 22]
                            Type: FlexibleArray Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FlexibleArrayXmpPageEmpty12}
\begin{paste}{FlexibleArrayXmpPageEmpty12}{FlexibleArrayXmpPagePatch12}
\pastebutton{FlexibleArrayXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{merge!(sort! f, sort! g)\bound{f6 }\free{g f5 }}
\end{paste}\end{patch}

\begin{patch}{FlexibleArrayXmpPagePatch13}
\begin{paste}{FlexibleArrayXmpPageFull13}{FlexibleArrayXmpPageEmpty13}
\pastebutton{FlexibleArrayXmpPageFull13}{\hidepaste}
\tab{5}\spadcommand{removeDuplicates! f\bound{f7 }\free{f6 }}
\indentrel{3}\begin{verbatim}
   (13)  [1,2,3,4,5,6,11,22]
                            Type: FlexibleArray Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FlexibleArrayXmpPageEmpty13}
\begin{paste}{FlexibleArrayXmpPageEmpty13}{FlexibleArrayXmpPagePatch13}
\pastebutton{FlexibleArrayXmpPageEmpty13}{\showpaste}
\tab{5}\spadcommand{removeDuplicates! f\bound{f7 }\free{f6 }}
\end{paste}\end{patch}

\begin{patch}{FlexibleArrayXmpPagePatch14}
\begin{paste}{FlexibleArrayXmpPageFull14}{FlexibleArrayXmpPageEmpty14}
\pastebutton{FlexibleArrayXmpPageFull14}{\hidepaste}
\tab{5}\spadcommand{select!(i +-> even? i,f)\bound{f8 }\free{f7 }}
\indentrel{3}\begin{verbatim}
   (14)  [2,4,6,22]
                            Type: FlexibleArray Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FlexibleArrayXmpPageEmpty14}
\begin{paste}{FlexibleArrayXmpPageEmpty14}{FlexibleArrayXmpPagePatch14}
\pastebutton{FlexibleArrayXmpPageEmpty14}{\showpaste}
\tab{5}\spadcommand{select!(i +-> even? i,f)\bound{f8 }\free{f7 }}
\end{paste}\end{patch}

\begin{patch}{FlexibleArrayXmpPagePatch15}
\begin{paste}{FlexibleArrayXmpPageFull15}{FlexibleArrayXmpPageEmpty15}
\pastebutton{FlexibleArrayXmpPageFull15}{\hidepaste}
\tab{5}\spadcommand{physicalLength f\free{b8 }}
```

```
\indentrel{3}\begin{verbatim}
   (15)  8
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FlexibleArrayXmpPageEmpty15}
\begin{paste}{FlexibleArrayXmpPageEmpty15}{FlexibleArrayXmpPagePatch15}
\pastebutton{FlexibleArrayXmpPageEmpty15}{\showpaste}
\tab{5}\spadcommand{physicalLength f\free{b8 }}
\end{paste}\end{patch}

\begin{patch}{FlexibleArrayXmpPagePatch16}
\begin{paste}{FlexibleArrayXmpPageFull16}{FlexibleArrayXmpPageEmpty16}
\pastebutton{FlexibleArrayXmpPageFull16}{\hidepaste}
\tab{5}\spadcommand{shrinkable(false)$FlexibleArray(Integer)}
\indentrel{3}\begin{verbatim}
   (16)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FlexibleArrayXmpPageEmpty16}
\begin{paste}{FlexibleArrayXmpPageEmpty16}{FlexibleArrayXmpPagePatch16}
\pastebutton{FlexibleArrayXmpPageEmpty16}{\showpaste}
\tab{5}\spadcommand{shrinkable(false)$FlexibleArray(Integer)}
\end{paste}\end{patch}
```

3.40 file.ht

File

⇒ "notitle" (TextFileXmpPage) 3.107 on page 1307
⇒ "notitle" (KeyedAccessFileXmpPage) 3.57 on page 734
⇒ "notitle" (LibraryXmpPage) 3.62 on page 840
⇒ "notitle" (FileNameXmpPage) 3.42 on page 507

— **file.ht** —

```
\begin{page}{FileXmpPage}{File}
\beginscroll

The \spadtype{File(S)} domain provides a basic interface to read and
write values of type \spad{S} in files.
```

```
\xtc{
Before working with a file, it must be made accessible to Axiom with
the \spadfunFrom{open}{File} operation.
}{
\spadpaste{ifile:File List Integer:=open("/tmp/jazz1","output")
\bound{ifile}}
}
The \spadfunFrom{open}{File} function arguments are a \spadtype{FileName}
and a \spadtype{String} specifying the mode.
If a full pathname is not specified, the current default directory is
assumed.
The mode must be one of \spad{"input"} or \spad{"output"}.
If it is not specified, \spad{"input"} is assumed.
Once the file has been opened, you can read or write data.
\xtc{
The operations \spadfunFromX{read}{File} and \spadfunFromX{write}{File}
are provided.
}{
\spadpaste{write!(ifile, [-1,2,3])       \free{ifile}\bound{ifile1}}
}
\xtc{
}{
\spadpaste{write!(ifile, [10,-10,0,111])  \free{ifile1}\bound{ifile2}}
}
\xtc{
}{
\spadpaste{write!(ifile, [7])             \free{ifile2}\bound{ifile3}}
}
\xtc{
You can change from writing to reading (or vice versa)
by reopening a file.
}{
\spadpaste{reopen!(ifile, "input")        \free{ifile3}\bound{ifile4}}
}
\xtc{
}{
\spadpaste{read! ifile                    \free{ifile4}\bound{ifile5}}
}
\xtc{
}{
\spadpaste{read! ifile                    \free{ifile5}\bound{ifile6}}
}
\xtc{
The \spadfunFromX{read}{File} operation can cause an error if one tries to
read more data than is in the file.
To guard against this possibility the \spadfunFromX{readIfCan}{File}
operation should be used.
}{
\spadpaste{readIfCan! ifile  \free{ifile6}\bound{ifile7}}
}
```

```
\xtc{
}{
\spadpaste{readIfCan! ifile  \free{ifile7}\bound{ifile8}}
}
\xtc{
You can find the current mode of the file, and the file's name.
}{
\spadpaste{iomode ifile           \free{ifile}}
}
\xtc{
}{
\spadpaste{name ifile             \free{ifile}}
}
\xtc{
When you are finished with a file, you should close it.
}{
\spadpaste{close! ifile           \free{ifile}\bound{ifileA}}
}
\noOutputXtc{
}{
\spadpaste{)system rm /tmp/jazz1  \free{ifileA}}
}
%\xtc{
%}{
%\spadcommand{)clear all  \free{}\bound{}}
%}

A limitation of the underlying LISP system is that not all values can be
represented in a file.
In particular, delayed values containing compiled functions cannot be
saved.

For more information on related topics, see
\downlink{`TextFile'}{TextFileXmpPage}\ignore{TextFile},
\downlink{`KeyedAccessFile'}{KeyedAccessFileXmpPage}
\ignore{KeyedAccessFile},
\downlink{`Library'}{LibraryXmpPage}\ignore{Library}, and
\downlink{`FileName'}{FileNameXmpPage}\ignore{FileName}.
\showBlurb{File}
\endscroll
\autobuttons
\end{page}

\begin{patch}{FileXmpPagePatch1}
\begin{paste}{FileXmpPageFull1}{FileXmpPageEmpty1}
\pastebutton{FileXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{ifile:File List Integer:=open("/tmp/jazz1","output")\bound{ifile }}
\indentrel{3}\begin{verbatim}
   (1)  "/tmp/jazz1"
                                Type: File List Integer
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FileXmpPageEmpty1}
\begin{paste}{FileXmpPageEmpty1}{FileXmpPagePatch1}
\pastebutton{FileXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{ifile:File List Integer:=open("/tmp/jazz1","output")\bound{ifile }}
\end{paste}\end{patch}

\begin{patch}{FileXmpPagePatch2}
\begin{paste}{FileXmpPageFull2}{FileXmpPageEmpty2}
\pastebutton{FileXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{write!(ifile, [-1,2,3])\free{ifile }\bound{ifile1 }}
\indentrel{3}\begin{verbatim}
   (2)  [- 1,2,3]
                                    Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FileXmpPageEmpty2}
\begin{paste}{FileXmpPageEmpty2}{FileXmpPagePatch2}
\pastebutton{FileXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{write!(ifile, [-1,2,3])\free{ifile }\bound{ifile1 }}
\end{paste}\end{patch}

\begin{patch}{FileXmpPagePatch3}
\begin{paste}{FileXmpPageFull3}{FileXmpPageEmpty3}
\pastebutton{FileXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{write!(ifile, [10,-10,0,111])\free{ifile1 }\bound{ifile2 }}
\indentrel{3}\begin{verbatim}
   (3)  [10,- 10,0,111]
                                    Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FileXmpPageEmpty3}
\begin{paste}{FileXmpPageEmpty3}{FileXmpPagePatch3}
\pastebutton{FileXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{write!(ifile, [10,-10,0,111])\free{ifile1 }\bound{ifile2 }}
\end{paste}\end{patch}

\begin{patch}{FileXmpPagePatch4}
\begin{paste}{FileXmpPageFull4}{FileXmpPageEmpty4}
\pastebutton{FileXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{write!(ifile, [7])\free{ifile2 }\bound{ifile3 }}
\indentrel{3}\begin{verbatim}
   (4)  [7]
                                    Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{FileXmpPageEmpty4}
\begin{paste}{FileXmpPageEmpty4}{FileXmpPagePatch4}
\pastebutton{FileXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{write!(ifile, [7])\free{ifile2 }\bound{ifile3 }}
\end{paste}\end{patch}

\begin{patch}{FileXmpPagePatch5}
\begin{paste}{FileXmpPageFull5}{FileXmpPageEmpty5}
\pastebutton{FileXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{reopen!(ifile, "input")\free{ifile3 }\bound{ifile4 }}
\indentrel{3}\begin{verbatim}
   (5)  "/tmp/jazz1"
                                Type: File List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FileXmpPageEmpty5}
\begin{paste}{FileXmpPageEmpty5}{FileXmpPagePatch5}
\pastebutton{FileXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{reopen!(ifile, "input")\free{ifile3 }\bound{ifile4 }}
\end{paste}\end{patch}

\begin{patch}{FileXmpPagePatch6}
\begin{paste}{FileXmpPageFull6}{FileXmpPageEmpty6}
\pastebutton{FileXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{read! ifile\free{ifile4 }\bound{ifile5 }}
\indentrel{3}\begin{verbatim}
   (6)  [- 1,2,3]
                                     Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FileXmpPageEmpty6}
\begin{paste}{FileXmpPageEmpty6}{FileXmpPagePatch6}
\pastebutton{FileXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{read! ifile\free{ifile4 }\bound{ifile5 }}
\end{paste}\end{patch}

\begin{patch}{FileXmpPagePatch7}
\begin{paste}{FileXmpPageFull7}{FileXmpPageEmpty7}
\pastebutton{FileXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{read! ifile\free{ifile5 }\bound{ifile6 }}
\indentrel{3}\begin{verbatim}
   (7)  [10,- 10,0,111]
                                     Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FileXmpPageEmpty7}
```

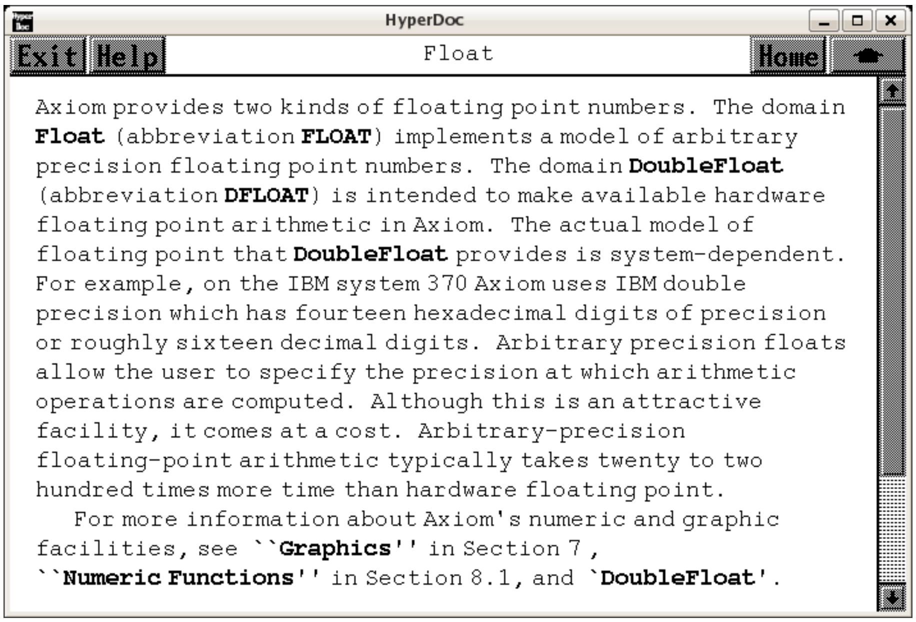
```
\begin{paste}{FileXmpPageEmpty7}{FileXmpPagePatch7}
\pastebutton{FileXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{read! ifile\free{ifile5 }\bound{ifile6 }}
\end{paste}\end{patch}

\begin{patch}{FileXmpPagePatch8}
\begin{paste}{FileXmpPageFull8}{FileXmpPageEmpty8}
\pastebutton{FileXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{readIfCan! ifile\free{ifile6 }\bound{ifile7 }}
\indentrel{3}\begin{verbatim}
   (8)  [7]
                          Type: Union(List Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FileXmpPageEmpty8}
\begin{paste}{FileXmpPageEmpty8}{FileXmpPagePatch8}
\pastebutton{FileXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{readIfCan! ifile\free{ifile6 }\bound{ifile7 }}
\end{paste}\end{patch}

\begin{patch}{FileXmpPagePatch9}
\begin{paste}{FileXmpPageFull9}{FileXmpPageEmpty9}
\pastebutton{FileXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{readIfCan! ifile\free{ifile7 }\bound{ifile8 }}
\indentrel{3}\begin{verbatim}
   (9)  "failed"
                              Type: Union("failed",...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FileXmpPageEmpty9}
\begin{paste}{FileXmpPageEmpty9}{FileXmpPagePatch9}
\pastebutton{FileXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{readIfCan! ifile\free{ifile7 }\bound{ifile8 }}
\end{paste}\end{patch}

\begin{patch}{FileXmpPagePatch10}
\begin{paste}{FileXmpPageFull10}{FileXmpPageEmpty10}
\pastebutton{FileXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{iomode ifile\free{ifile }}
\indentrel{3}\begin{verbatim}
   (10)  "input"
                                            Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FileXmpPageEmpty10}
\begin{paste}{FileXmpPageEmpty10}{FileXmpPagePatch10}
\pastebutton{FileXmpPageEmpty10}{\showpaste}
```

```
\tab{5}\spadcommand{iomode ifile\free{ifile }}
\end{paste}\end{patch}

\begin{patch}{FileXmpPagePatch11}
\begin{paste}{FileXmpPageFull11}{FileXmpPageEmpty11}
\pastebutton{FileXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{name ifile\free{ifile }}
\indentrel{3}\begin{verbatim}
   (11)  "/tmp/jazz1"
                                          Type: FileName
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FileXmpPageEmpty11}
\begin{paste}{FileXmpPageEmpty11}{FileXmpPagePatch11}
\pastebutton{FileXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{name ifile\free{ifile }}
\end{paste}\end{patch}

\begin{patch}{FileXmpPagePatch12}
\begin{paste}{FileXmpPageFull12}{FileXmpPageEmpty12}
\pastebutton{FileXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{close! ifile\free{ifile }\bound{ifileA }}
\indentrel{3}\begin{verbatim}
   (12)  "/tmp/jazz1"
                                Type: File List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FileXmpPageEmpty12}
\begin{paste}{FileXmpPageEmpty12}{FileXmpPagePatch12}
\pastebutton{FileXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{close! ifile\free{ifile }\bound{ifileA }}
\end{paste}\end{patch}

\begin{patch}{FileXmpPagePatch13}
\begin{paste}{FileXmpPageFull13}{FileXmpPageEmpty13}
\pastebutton{FileXmpPageFull13}{\hidepaste}
\tab{5}\spadcommand{)system rm /tmp/jazz1\free{ifileA }}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FileXmpPageEmpty13}
\begin{paste}{FileXmpPageEmpty13}{FileXmpPagePatch13}
\pastebutton{FileXmpPageEmpty13}{\showpaste}
\tab{5}\spadcommand{)system rm /tmp/jazz1\free{ifileA }}
\end{paste}\end{patch}
```

———-

3.41 float.ht

Float



⇐ "Axiom Number Types" (NumberPage) 3.80 on page 1021
⇒ "Graphics" (ugGraphPage) 11 on page 1967
⇒ "Numeric Functions" (ugProblemNumericPage) 12 on page 2079
⇒ "DoubleFloat" (DoubleFloatXmpPage) 3.23 on page 369
⇒ "Introduction to Float" (ugxFloatIntroPage) 3.41 on page 488
⇒ "Conversion Functions" (ugxFloatConvertPage) 3.41 on page 490
⇒ "Output Functions" (ugxFloatOutputPage) 3.41 on page 498
⇒ "Determinant of a Hilbert Matrix" (ugxFloatHilbertPage) 3.41 on page 502

— **float.ht** —

```
\begin{page}{FloatXmpPage}{Float}
\beginscroll
 Axiom provides two kinds of floating point numbers.
The domain \spadtype{Float} (abbreviation \spadtype{FLOAT})
implements a model of arbitrary
precision floating point numbers.
The domain \spadtype{DoubleFloat} (abbreviation \spadtype{DFLOAT})
is intended to make available
hardware floating point arithmetic in Axiom.
```

```
The actual model of floating point that \spadtype{DoubleFloat} provides is
system-dependent.
For example, on the IBM system 370 Axiom uses IBM double
precision which has fourteen hexadecimal digits of precision or roughly
sixteen decimal digits.
Arbitrary precision floats allow the user to specify the
precision at which arithmetic operations are computed.
Although this is an attractive facility, it comes at a cost.
Arbitrary-precision floating-point arithmetic typically takes
twenty to two hundred times more time than hardware floating point.

For more information about Axiom's numeric and graphic
facilities,  see
\downlink{``Graphics''}{ugGraphPage} in
Section 7 \ignore{ugGraph},
\downlink{``Numeric Functions''}{ugProblemNumericPage} in
Section 8.1\ignore{ugProblemNumeric}, and
\downlink{`DoubleFloat'}{DoubleFloatXmpPage}\ignore{DoubleFloat}.

\beginmenu
    \menudownlink{{9.27.1. Introduction to Float}}{ugxFloatIntroPage}
    \menudownlink{{9.27.2. Conversion Functions}}{ugxFloatConvertPage}
    \menudownlink{{9.27.3. Output Functions}}{ugxFloatOutputPage}
    \menudownlink{{9.27.4. An Example: Determinant of a Hilbert Matrix}}
{ugxFloatHilbertPage}
\endmenu
\endscroll
\autobuttons
\end{page}
```

———–

Introduction to Float

— **float.ht** —

```
\begin{page}{ugxFloatIntroPage}{Introduction to Float}
\beginscroll

Scientific notation is supported for input and output
of floating point numbers.
A floating point number is written as a string of digits containing a
decimal point optionally followed by the letter ``{\tt E}'', and then
the exponent.
\xtc{
We begin by doing some calculations using arbitrary precision floats.
The default precision is twenty decimal digits.
```

```
}{
\spadpaste{1.234}
}
\xtc{
A decimal base for the exponent is assumed, so the number
\spad{1.234E2}  denotes
\texht{$1.234 \cdot  10^2$}{\spad{1.234 * 10**2}}.
}{
\spadpaste{1.234E2}
}
\xtc{
The normal arithmetic operations are available for floating point
numbers.
}{
\spadpaste{sqrt(1.2 + 2.3 / 3.4 ** 4.5)}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxFloatIntroPagePatch1}
\begin{paste}{ugxFloatIntroPageFull1}{ugxFloatIntroPageEmpty1}
\pastebutton{ugxFloatIntroPageFull1}{\hidepaste}
\tab{5}\spadcommand{1.234}
\indentrel{3}\begin{verbatim}
   (1)  1.234
                                            Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFloatIntroPageEmpty1}
\begin{paste}{ugxFloatIntroPageEmpty1}{ugxFloatIntroPagePatch1}
\pastebutton{ugxFloatIntroPageEmpty1}{\showpaste}
\tab{5}\spadcommand{1.234}
\end{paste}\end{patch}

\begin{patch}{ugxFloatIntroPagePatch2}
\begin{paste}{ugxFloatIntroPageFull2}{ugxFloatIntroPageEmpty2}
\pastebutton{ugxFloatIntroPageFull2}{\hidepaste}
\tab{5}\spadcommand{1.234E2}
\indentrel{3}\begin{verbatim}
   (2)  123.4
                                            Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFloatIntroPageEmpty2}
\begin{paste}{ugxFloatIntroPageEmpty2}{ugxFloatIntroPagePatch2}
\pastebutton{ugxFloatIntroPageEmpty2}{\showpaste}
```

```
\tab{5}\spadcommand{1.234E2}
\end{paste}\end{patch}

\begin{patch}{ugxFloatIntroPagePatch3}
\begin{paste}{ugxFloatIntroPageFull3}{ugxFloatIntroPageEmpty3}
\pastebutton{ugxFloatIntroPageFull3}{\hidepaste}
\tab{5}\spadcommand{sqrt(1.2 + 2.3 / 3.4 ** 4.5)}
\indentrel{3}\begin{verbatim}
   (3)  1.0996972790 671286226
                                             Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFloatIntroPageEmpty3}
\begin{paste}{ugxFloatIntroPageEmpty3}{ugxFloatIntroPagePatch3}
\pastebutton{ugxFloatIntroPageEmpty3}{\showpaste}
\tab{5}\spadcommand{sqrt(1.2 + 2.3 / 3.4 ** 4.5)}
\end{paste}\end{patch}
```

———-

Conversion Functions

⇒ "notitle" (ugTypesConvertPage) 7 on page 1671

— **float.ht** —

```
\begin{page}{ugxFloatConvertPage}{Conversion Functions}
\beginscroll

\labelSpace{3pc}
\xtc{
You can use conversion
(\downlink{‘‘Conversion’’}{ugTypesConvertPage}
in Section 2.7\ignore{ugTypesConvert})
to go back and forth between \spadtype{Integer},
\spadtype{Fraction Integer} and \spadtype{Float}, as appropriate.
}{
\spadpaste{i := 3 :: Float \bound{i}}
}
\xtc{
}{
\spadpaste{i :: Integer \free{i}}
}
\xtc{
}{
\spadpaste{i :: Fraction Integer \free{i}}
}
```

```
\xtc{
Since you are explicitly asking for a conversion, you must take
responsibility for any loss of exactness.
}{
\spadpaste{r := 3/7 :: Float \bound{r}}
}
\xtc{
}{
\spadpaste{r :: Fraction Integer \free{r}}
}
\xtc{
This conversion cannot be performed: use
\spadfunFrom{truncate}{Float} or \spadfunFrom{round}{Float} if that
is what you intend.
}{
\spadpaste{r :: Integer \free{r}}
}

\xtc{
The operations \spadfunFrom{truncate}{Float} and \spadfunFrom{round}{Float}
truncate  \ldots
}{
\spadpaste{truncate 3.6}
}
\xtc{
and round
to the nearest integral \spadtype{Float} respectively.
}{
\spadpaste{round 3.6}
}
\xtc{
}{
\spadpaste{truncate(-3.6)}
}
\xtc{
}{
\spadpaste{round(-3.6)}
}
\xtc{
The operation \spadfunFrom{fractionPart}{Float} computes the fractional part of
\spad{x}, that is, \spad{x - truncate x}.
}{
\spadpaste{fractionPart 3.6}
}
\xtc{
The operation \spadfunFrom{digits}{Float} allows the user to set the
precision.
It returns the previous value it was using.
}{
\spadpaste{digits 40 \bound{d40}}
```

```
}
\xtc{
}{
\spadpaste{sqrt 0.2}
}
\xtc{
}{
\spadpaste{pi()\$Float \free{d40}}
}
\xtc{
The precision is only limited by the computer memory available.
Calculations at 500 or more digits of precision are not difficult.
}{
\spadpaste{digits 500 \bound{d1000}}
}
\xtc{
}{
\spadpaste{pi()\$Float \free{d1000}}
}
\xtc{
Reset \spadfunFrom{digits}{Float} to its default value.
}{
\spadpaste{digits 20}
}
Numbers of type \spadtype{Float} are represented as a record of two
integers, namely, the mantissa and the exponent where the base of the
exponent is binary.
That is, the floating point number \spad{(m,e)} represents the number
\texht{$m \cdot 2^e$}{\spad{m * 2**e}}.
A consequence of using a binary base is that decimal numbers can not, in
general, be represented exactly.

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxFloatConvertPagePatch1}
\begin{paste}{ugxFloatConvertPageFull1}{ugxFloatConvertPageEmpty1}
\pastebutton{ugxFloatConvertPageFull1}{\hidepaste}
\tab{5}\spadcommand{i := 3 :: Float\bound{i }}
\indentrel{3}\begin{verbatim}
   (1)  3.0
                                            Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFloatConvertPageEmpty1}
\begin{paste}{ugxFloatConvertPageEmpty1}{ugxFloatConvertPagePatch1}
\pastebutton{ugxFloatConvertPageEmpty1}{\showpaste}
\tab{5}\spadcommand{i := 3 :: Float\bound{i }}
```

```
\end{paste}\end{patch}

\begin{patch}{ugxFloatConvertPagePatch2}
\begin{paste}{ugxFloatConvertPageFull2}{ugxFloatConvertPageEmpty2}
\pastebutton{ugxFloatConvertPageFull2}{\hidepaste}
\tab{5}\spadcommand{i :: Integer\free{i }}
\indentrel{3}\begin{verbatim}
   (2)  3
                                          Type: Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFloatConvertPageEmpty2}
\begin{paste}{ugxFloatConvertPageEmpty2}{ugxFloatConvertPagePatch2}
\pastebutton{ugxFloatConvertPageEmpty2}{\showpaste}
\tab{5}\spadcommand{i :: Integer\free{i }}
\end{paste}\end{patch}

\begin{patch}{ugxFloatConvertPagePatch3}
\begin{paste}{ugxFloatConvertPageFull3}{ugxFloatConvertPageEmpty3}
\pastebutton{ugxFloatConvertPageFull3}{\hidepaste}
\tab{5}\spadcommand{i :: Fraction Integer\free{i }}
\indentrel{3}\begin{verbatim}
   (3)  3
                                 Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFloatConvertPageEmpty3}
\begin{paste}{ugxFloatConvertPageEmpty3}{ugxFloatConvertPagePatch3}
\pastebutton{ugxFloatConvertPageEmpty3}{\showpaste}
\tab{5}\spadcommand{i :: Fraction Integer\free{i }}
\end{paste}\end{patch}

\begin{patch}{ugxFloatConvertPagePatch4}
\begin{paste}{ugxFloatConvertPageFull4}{ugxFloatConvertPageEmpty4}
\pastebutton{ugxFloatConvertPageFull4}{\hidepaste}
\tab{5}\spadcommand{r := 3/7 :: Float\bound{r }}
\indentrel{3}\begin{verbatim}
   (4)  0.42857 14285 71428 57143
                                            Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFloatConvertPageEmpty4}
\begin{paste}{ugxFloatConvertPageEmpty4}{ugxFloatConvertPagePatch4}
\pastebutton{ugxFloatConvertPageEmpty4}{\showpaste}
\tab{5}\spadcommand{r := 3/7 :: Float\bound{r }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugxFloatConvertPagePatch5}
\begin{paste}{ugxFloatConvertPageFull5}{ugxFloatConvertPageEmpty5}
\pastebutton{ugxFloatConvertPageFull5}{\hidepaste}
\tab{5}\spadcommand{r :: Fraction Integer\free{r }}
\indentrel{3}\begin{verbatim}
        3
   (5)
        7
                                 Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFloatConvertPageEmpty5}
\begin{paste}{ugxFloatConvertPageEmpty5}{ugxFloatConvertPagePatch5}
\pastebutton{ugxFloatConvertPageEmpty5}{\showpaste}
\tab{5}\spadcommand{r :: Fraction Integer\free{r }}
\end{paste}\end{patch}

\begin{patch}{ugxFloatConvertPagePatch6}
\begin{paste}{ugxFloatConvertPageFull6}{ugxFloatConvertPageEmpty6}
\pastebutton{ugxFloatConvertPageFull6}{\hidepaste}
\tab{5}\spadcommand{r :: Integer\free{r }}
\indentrel{3}\begin{verbatim}
   0.42857 14285 71428 57143
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFloatConvertPageEmpty6}
\begin{paste}{ugxFloatConvertPageEmpty6}{ugxFloatConvertPagePatch6}
\pastebutton{ugxFloatConvertPageEmpty6}{\showpaste}
\tab{5}\spadcommand{r :: Integer\free{r }}
\end{paste}\end{patch}

\begin{patch}{ugxFloatConvertPagePatch7}
\begin{paste}{ugxFloatConvertPageFull7}{ugxFloatConvertPageEmpty7}
\pastebutton{ugxFloatConvertPageFull7}{\hidepaste}
\tab{5}\spadcommand{truncate 3.6}
\indentrel{3}\begin{verbatim}
   (6)  3.0
                                            Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFloatConvertPageEmpty7}
\begin{paste}{ugxFloatConvertPageEmpty7}{ugxFloatConvertPagePatch7}
\pastebutton{ugxFloatConvertPageEmpty7}{\showpaste}
\tab{5}\spadcommand{truncate 3.6}
\end{paste}\end{patch}

\begin{patch}{ugxFloatConvertPagePatch8}
```

```
\begin{paste}{ugxFloatConvertPageFull8}{ugxFloatConvertPageEmpty8}
\pastebutton{ugxFloatConvertPageFull8}{\hidepaste}
\tab{5}\spadcommand{round 3.6}
\indentrel{3}\begin{verbatim}
   (7)  4.0
                                            Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFloatConvertPageEmpty8}
\begin{paste}{ugxFloatConvertPageEmpty8}{ugxFloatConvertPagePatch8}
\pastebutton{ugxFloatConvertPageEmpty8}{\showpaste}
\tab{5}\spadcommand{round 3.6}
\end{paste}\end{patch}

\begin{patch}{ugxFloatConvertPagePatch9}
\begin{paste}{ugxFloatConvertPageFull9}{ugxFloatConvertPageEmpty9}
\pastebutton{ugxFloatConvertPageFull9}{\hidepaste}
\tab{5}\spadcommand{truncate(-3.6)}
\indentrel{3}\begin{verbatim}
   (8)  - 3.0
                                            Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFloatConvertPageEmpty9}
\begin{paste}{ugxFloatConvertPageEmpty9}{ugxFloatConvertPagePatch9}
\pastebutton{ugxFloatConvertPageEmpty9}{\showpaste}
\tab{5}\spadcommand{truncate(-3.6)}
\end{paste}\end{patch}

\begin{patch}{ugxFloatConvertPagePatch10}
\begin{paste}{ugxFloatConvertPageFull10}{ugxFloatConvertPageEmpty10}
\pastebutton{ugxFloatConvertPageFull10}{\hidepaste}
\tab{5}\spadcommand{round(-3.6)}
\indentrel{3}\begin{verbatim}
   (9)  - 4.0
                                            Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFloatConvertPageEmpty10}
\begin{paste}{ugxFloatConvertPageEmpty10}{ugxFloatConvertPagePatch10}
\pastebutton{ugxFloatConvertPageEmpty10}{\showpaste}
\tab{5}\spadcommand{round(-3.6)}
\end{paste}\end{patch}

\begin{patch}{ugxFloatConvertPagePatch11}
\begin{paste}{ugxFloatConvertPageFull11}{ugxFloatConvertPageEmpty11}
\pastebutton{ugxFloatConvertPageFull11}{\hidepaste}
```

```
\tab{5}\spadcommand{fractionPart 3.6}
\indentrel{3}\begin{verbatim}
   (10)  0.6
                                             Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFloatConvertPageEmpty11}
\begin{paste}{ugxFloatConvertPageEmpty11}{ugxFloatConvertPagePatch11}
\pastebutton{ugxFloatConvertPageEmpty11}{\showpaste}
\tab{5}\spadcommand{fractionPart 3.6}
\end{paste}\end{patch}

\begin{patch}{ugxFloatConvertPagePatch12}
\begin{paste}{ugxFloatConvertPageFull12}{ugxFloatConvertPageEmpty12}
\pastebutton{ugxFloatConvertPageFull12}{\hidepaste}
\tab{5}\spadcommand{digits 40\bound{d40 }}
\indentrel{3}\begin{verbatim}
   (11)  20
                                   Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFloatConvertPageEmpty12}
\begin{paste}{ugxFloatConvertPageEmpty12}{ugxFloatConvertPagePatch12}
\pastebutton{ugxFloatConvertPageEmpty12}{\showpaste}
\tab{5}\spadcommand{digits 40\bound{d40 }}
\end{paste}\end{patch}

\begin{patch}{ugxFloatConvertPagePatch13}
\begin{paste}{ugxFloatConvertPageFull13}{ugxFloatConvertPageEmpty13}
\pastebutton{ugxFloatConvertPageFull13}{\hidepaste}
\tab{5}\spadcommand{sqrt 0.2}
\indentrel{3}\begin{verbatim}
   (12)
   0.44721 35954 99957 93928 18347 33746 25524 70881
                                             Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFloatConvertPageEmpty13}
\begin{paste}{ugxFloatConvertPageEmpty13}{ugxFloatConvertPagePatch13}
\pastebutton{ugxFloatConvertPageEmpty13}{\showpaste}
\tab{5}\spadcommand{sqrt 0.2}
\end{paste}\end{patch}

\begin{patch}{ugxFloatConvertPagePatch14}
\begin{paste}{ugxFloatConvertPageFull14}{ugxFloatConvertPageEmpty14}
\pastebutton{ugxFloatConvertPageFull14}{\hidepaste}
\tab{5}\spadcommand{pi()$Float\free{d40 }}
```

```
\indentrel{3}\begin{verbatim}
   (13)
   3.14159 26535 89793 23846 26433 83279 50288 4197
                                              Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFloatConvertPageEmpty14}
\begin{paste}{ugxFloatConvertPageEmpty14}{ugxFloatConvertPagePatch14}
\pastebutton{ugxFloatConvertPageEmpty14}{\showpaste}
\tab{5}\spadcommand{pi()$Float\free{d40 }}
\end{paste}\end{patch}

\begin{patch}{ugxFloatConvertPagePatch15}
\begin{paste}{ugxFloatConvertPageFull15}{ugxFloatConvertPageEmpty15}
\pastebutton{ugxFloatConvertPageFull15}{\hidepaste}
\tab{5}\spadcommand{digits 500\bound{d1000 }}
\indentrel{3}\begin{verbatim}
   (14)  40
                                    Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFloatConvertPageEmpty15}
\begin{paste}{ugxFloatConvertPageEmpty15}{ugxFloatConvertPagePatch15}
\pastebutton{ugxFloatConvertPageEmpty15}{\showpaste}
\tab{5}\spadcommand{digits 500\bound{d1000 }}
\end{paste}\end{patch}

\begin{patch}{ugxFloatConvertPagePatch16}
\begin{paste}{ugxFloatConvertPageFull16}{ugxFloatConvertPageEmpty16}
\pastebutton{ugxFloatConvertPageFull16}{\hidepaste}
\tab{5}\spadcommand{pi()$Float\free{d1000 }}
\indentrel{3}\begin{verbatim}
   (15)
  3.14159 26535 89793 23846 26433 83279 50288 41971 69399
   37510 58209 74944 59230 78164 06286 20899 86280 34825
  34211 70679 82148 08651 32823 06647 09384 46095 50582 2
  3172 53594 08128 48111 74502 84102 70193 85211 05559 64
  462 29489 54930 38196 44288 10975 66593 34461 28475 648
  23 37867 83165 27120 19091 45648 56692 34603 48610 4543
  2 66482 13393 60726 02491 41273 72458 70066 06315 58817
   48815 20920 96282 92540 91715 36436 78925 90360 01133
  05305 48820 46652 13841 46951 94151 16094 33057 27036 5
  7595 91953 09218 61173 81932 61179 31051 18548 07446 23
  799 62749 56735 18857 52724 89122 79381 83011 9491
                                              Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugxFloatConvertPageEmpty16}
\begin{paste}{ugxFloatConvertPageEmpty16}{ugxFloatConvertPagePatch16}
\pastebutton{ugxFloatConvertPageEmpty16}{\showpaste}
\tab{5}\spadcommand{pi()$Float\free{d1000 }}
\end{paste}\end{patch}

\begin{patch}{ugxFloatConvertPagePatch17}
\begin{paste}{ugxFloatConvertPageFull17}{ugxFloatConvertPageEmpty17}
\pastebutton{ugxFloatConvertPageFull17}{\hidepaste}
\tab{5}\spadcommand{digits 20}
\indentrel{3}\begin{verbatim}
   (16)  500
                                   Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFloatConvertPageEmpty17}
\begin{paste}{ugxFloatConvertPageEmpty17}{ugxFloatConvertPagePatch17}
\pastebutton{ugxFloatConvertPageEmpty17}{\showpaste}
\tab{5}\spadcommand{digits 20}
\end{paste}\end{patch}
```

———-

Output Functions

— **float.ht** —

```
\begin{page}{ugxFloatOutputPage}{Output Functions}
\beginscroll

A number of operations exist for specifying how
numbers of type \spadtype{Float} are to be
displayed.
By default, spaces are inserted every ten digits in the
output for readability.\footnote{Note that you cannot include spaces
in the input form of a floating point number, though you can use
underscores.}


\xtc{
Output spacing can be modified with the \spadfunFrom{outputSpacing}{Float}
operation.
This inserts no spaces and then displays the value of \spad{x}.
}{
\spadpaste{outputSpacing 0; x := sqrt 0.2 \bound{x}\bound{os0}}
}
```

```
\xtc{
Issue this to have the spaces inserted every \spad{5} digits.
}{
\spadpaste{outputSpacing 5; x \bound{os5}\free{x}}
}
\xtc{
By default, the system displays floats in either fixed format
or scientific format, depending on the magnitude of the number.
}{
\spadpaste{y := x/10**10 \bound{y}\free{x os5}}
}
\xtc{
A particular format may be requested with the operations
\spadfunFrom{outputFloating}{Float} and \spadfunFrom{outputFixed}{Float}.
}{
\spadpaste{outputFloating(); x \bound{of} \free{os5 x}}
}
\xtc{
}{
\spadpaste{outputFixed(); y \bound{ox} \free{os5 y}}
}
\xtc{
Additionally, you can ask for \spad{n} digits to be displayed after the
decimal point.
}{
\spadpaste{outputFloating 2; y \bound{of2} \free{os5 y}}
}
\xtc{
}{
\spadpaste{outputFixed 2; x \bound{ox2} \free{os5 x}}
}
\xtc{
This resets the output printing to the default behavior.
}{
\spadpaste{outputGeneral()}
}
%

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxFloatOutputPagePatch1}
\begin{paste}{ugxFloatOutputPageFull1}{ugxFloatOutputPageEmpty1}
\pastebutton{ugxFloatOutputPageFull1}{\hidepaste}
\tab{5}\spadcommand{outputSpacing 0; x := sqrt 0.2\bound{x }\bound{os0 }}
\indentrel{3}\begin{verbatim}
   (1)  0.44721359549995793928
                                              Type: Float
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFloatOutputPageEmpty1}
\begin{paste}{ugxFloatOutputPageEmpty1}{ugxFloatOutputPagePatch1}
\pastebutton{ugxFloatOutputPageEmpty1}{\showpaste}
\tab{5}\spadcommand{outputSpacing 0; x := sqrt 0.2\bound{x }\bound{os0 }}
\end{paste}\end{patch}

\begin{patch}{ugxFloatOutputPagePatch2}
\begin{paste}{ugxFloatOutputPageFull2}{ugxFloatOutputPageEmpty2}
\pastebutton{ugxFloatOutputPageFull2}{\hidepaste}
\tab{5}\spadcommand{outputSpacing 5; x\bound{os5 }\free{x }}
\indentrel{3}\begin{verbatim}
   (2)  0.44721 35954 99957 93928
                                           Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFloatOutputPageEmpty2}
\begin{paste}{ugxFloatOutputPageEmpty2}{ugxFloatOutputPagePatch2}
\pastebutton{ugxFloatOutputPageEmpty2}{\showpaste}
\tab{5}\spadcommand{outputSpacing 5; x\bound{os5 }\free{x }}
\end{paste}\end{patch}

\begin{patch}{ugxFloatOutputPagePatch3}
\begin{paste}{ugxFloatOutputPageFull3}{ugxFloatOutputPageEmpty3}
\pastebutton{ugxFloatOutputPageFull3}{\hidepaste}
\tab{5}\spadcommand{y := x/10**10\bound{y }\free{x os5 }}
\indentrel{3}\begin{verbatim}
   (3)  0.44721 35954 99957 93928 E -10
                                           Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFloatOutputPageEmpty3}
\begin{paste}{ugxFloatOutputPageEmpty3}{ugxFloatOutputPagePatch3}
\pastebutton{ugxFloatOutputPageEmpty3}{\showpaste}
\tab{5}\spadcommand{y := x/10**10\bound{y }\free{x os5 }}
\end{paste}\end{patch}

\begin{patch}{ugxFloatOutputPagePatch4}
\begin{paste}{ugxFloatOutputPageFull4}{ugxFloatOutputPageEmpty4}
\pastebutton{ugxFloatOutputPageFull4}{\hidepaste}
\tab{5}\spadcommand{outputFloating(); x\bound{of }\free{os5 x }}
\indentrel{3}\begin{verbatim}
   (4)  0.44721 35954 99957 93928 E 0
                                           Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugxFloatOutputPageEmpty4}
\begin{paste}{ugxFloatOutputPageEmpty4}{ugxFloatOutputPagePatch4}
\pastebutton{ugxFloatOutputPageEmpty4}{\showpaste}
\tab{5}\spadcommand{outputFloating(); x\bound{of }\free{os5 x }}
\end{paste}\end{patch}

\begin{patch}{ugxFloatOutputPagePatch5}
\begin{paste}{ugxFloatOutputPageFull5}{ugxFloatOutputPageEmpty5}
\pastebutton{ugxFloatOutputPageFull5}{\hidepaste}
\tab{5}\spadcommand{outputFixed(); y\bound{ox }\free{os5 y }}
\indentrel{3}\begin{verbatim}
   (5)  0.00000 00000 44721 35954 99957 93928
                                            Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFloatOutputPageEmpty5}
\begin{paste}{ugxFloatOutputPageEmpty5}{ugxFloatOutputPagePatch5}
\pastebutton{ugxFloatOutputPageEmpty5}{\showpaste}
\tab{5}\spadcommand{outputFixed(); y\bound{ox }\free{os5 y }}
\end{paste}\end{patch}

\begin{patch}{ugxFloatOutputPagePatch6}
\begin{paste}{ugxFloatOutputPageFull6}{ugxFloatOutputPageEmpty6}
\pastebutton{ugxFloatOutputPageFull6}{\hidepaste}
\tab{5}\spadcommand{outputFloating 2; y\bound{of2 }\free{os5 y }}
\indentrel{3}\begin{verbatim}
   (6)  0.45 E -10
                                            Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFloatOutputPageEmpty6}
\begin{paste}{ugxFloatOutputPageEmpty6}{ugxFloatOutputPagePatch6}
\pastebutton{ugxFloatOutputPageEmpty6}{\showpaste}
\tab{5}\spadcommand{outputFloating 2; y\bound{of2 }\free{os5 y }}
\end{paste}\end{patch}

\begin{patch}{ugxFloatOutputPagePatch7}
\begin{paste}{ugxFloatOutputPageFull7}{ugxFloatOutputPageEmpty7}
\pastebutton{ugxFloatOutputPageFull7}{\hidepaste}
\tab{5}\spadcommand{outputFixed 2; x\bound{ox2 }\free{os5 x }}
\indentrel{3}\begin{verbatim}
   (7)  0.45
                                            Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFloatOutputPageEmpty7}
\begin{paste}{ugxFloatOutputPageEmpty7}{ugxFloatOutputPagePatch7}
```

```
\pastebutton{ugxFloatOutputPageEmpty7}{\showpaste}
\tab{5}\spadcommand{outputFixed 2; x\bound{ox2 }\free{os5 x }}
\end{paste}\end{patch}

\begin{patch}{ugxFloatOutputPagePatch8}
\begin{paste}{ugxFloatOutputPageFull8}{ugxFloatOutputPageEmpty8}
\pastebutton{ugxFloatOutputPageFull8}{\hidepaste}
\tab{5}\spadcommand{outputGeneral()}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFloatOutputPageEmpty8}
\begin{paste}{ugxFloatOutputPageEmpty8}{ugxFloatOutputPagePatch8}
\pastebutton{ugxFloatOutputPageEmpty8}{\showpaste}
\tab{5}\spadcommand{outputGeneral()}
\end{paste}\end{patch}
```

————-

An Example: Determinant of a Hilbert Matrix

— float.ht —

```
\begin{page}{ugxFloatHilbertPage}
{An Example: Determinant of a Hilbert Matrix}
\beginscroll

Consider the problem of computing the determinant of a \spad{10} by \spad{10}
Hilbert matrix.
The \eth{\spad{(i,j)}} entry of a Hilbert matrix is given by
\spad{1/(i+j+1)}.

\labelSpace{2pc}
\xtc{
First do the computation using rational numbers to obtain the
exact result.
}{
\spadpaste{a: Matrix Fraction Integer :=
matrix [[1/(i+j+1) for j in 0..9] for i in 0..9] \bound{a}}
}
\xtc{
This version of \spadfunFrom{determinant}{Matrix} uses Gaussian
elimination.
}{
\spadpaste{d:= determinant a \free{a}\bound{d}}
```

```
}
\xtc{
}{
\spadpaste{d :: Float \free{d}}
}
\xtc{
Now use hardware floats. Note that a semicolon (;) is used
to prevent the display of the matrix.
}{
\spadpaste{b: Matrix DoubleFloat :=
matrix [[1/(i+j+1\$DoubleFloat) for j in 0..9] for i in 0..9]; \bound{b}}
}
\xtc{
The result given by hardware floats is correct only to four significant
digits of precision.
In the jargon of numerical analysis, the Hilbert matrix is said to be
``ill-conditioned.''
}{
\spadpaste{determinant b \free{b}}
}
\xtc{
Now repeat the computation at a higher precision using \spadtype{Float}.
}{
\spadpaste{digits 40 \bound{d40}}
}
\xtc{
}{
\spadpaste{c: Matrix Float :=
matrix [[1/(i+j+1\$Float) for j in 0..9] for i in 0..9];
\free{d40} \bound{c}}
}
\xtc{
}{
\spadpaste{determinant c \free{c}}
}
\xtc{
Reset \spadfunFrom{digits}{Float} to its default value.
}{
\spadpaste{digits 20}
}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxFloatHilbertPagePatch1}
\begin{paste}{ugxFloatHilbertPageFull1}{ugxFloatHilbertPageEmpty1}
\pastebutton{ugxFloatHilbertPageFull1}{\hidepaste}
\tab{5}\spadcommand{a: Matrix Fraction Integer := matrix [[1/(i+j+1) for j in 0..9] for i in 0..9]\bound{a }
\indentrel{3}\begin{verbatim}
            1   1   1   1   1   1   1   1    1
```

```
        1
            2   3   4   5   6   7   8   9   10

        1   1   1   1   1   1   1   1    1   1

        2   3   4   5   6   7   8   9   10  11

        1   1   1   1   1   1   1    1   1   1

        3   4   5   6   7   8   9   10  11  12

        1   1   1   1   1   1    1   1   1   1

        4   5   6   7   8   9   10  11  12  13

        1   1   1   1   1    1   1   1   1   1

        5   6   7   8   9   10  11  12  13  14
   (1)
        1   1   1   1    1   1   1   1   1   1

        6   7   8   9   10  11  12  13  14  15

        1   1   1    1   1   1   1   1   1   1

        7   8   9   10  11  12  13  14  15  16

        1   1    1   1   1   1   1   1   1   1

        8   9   10  11  12  13  14  15  16  17

        1    1   1   1   1   1   1   1   1   1

        9   10  11  12  13  14  15  16  17  18

         1   1   1   1   1   1   1   1   1   1

        10  11  12  13  14  15  16  17  18  19
                              Type: Matrix Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFloatHilbertPageEmpty1}
\begin{paste}{ugxFloatHilbertPageEmpty1}{ugxFloatHilbertPagePatch1}
\pastebutton{ugxFloatHilbertPageEmpty1}{\showpaste}
\tab{5}\spadcommand{a: Matrix Fraction Integer := matrix [[1/(i+j+1) for j in 0..9] for i in 0..
\end{paste}\end{patch}

\begin{patch}{ugxFloatHilbertPagePatch2}
\begin{paste}{ugxFloatHilbertPageFull2}{ugxFloatHilbertPageEmpty2}
```

```
\pastebutton{ugxFloatHilbertPageFull2}{\hidepaste}
\tab{5}\spadcommand{d:= determinant a\free{a }\bound{d }}
\indentrel{3}\begin{verbatim}
   (2)
                             1

   462068939479146913162956288390362787269836800000000000
                                 Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFloatHilbertPageEmpty2}
\begin{paste}{ugxFloatHilbertPageEmpty2}{ugxFloatHilbertPagePatch2}
\pastebutton{ugxFloatHilbertPageEmpty2}{\showpaste}
\tab{5}\spadcommand{d:= determinant a\free{a }\bound{d }}
\end{paste}\end{patch}

\begin{patch}{ugxFloatHilbertPagePatch3}
\begin{paste}{ugxFloatHilbertPageFull3}{ugxFloatHilbertPageEmpty3}
\pastebutton{ugxFloatHilbertPageFull3}{\hidepaste}
\tab{5}\spadcommand{d :: Float\free{d }}
\indentrel{3}\begin{verbatim}
   (3)  0.21641 79226 43149 18691 E -52
                                            Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFloatHilbertPageEmpty3}
\begin{paste}{ugxFloatHilbertPageEmpty3}{ugxFloatHilbertPagePatch3}
\pastebutton{ugxFloatHilbertPageEmpty3}{\showpaste}
\tab{5}\spadcommand{d :: Float\free{d }}
\end{paste}\end{patch}

\begin{patch}{ugxFloatHilbertPagePatch4}
\begin{paste}{ugxFloatHilbertPageFull4}{ugxFloatHilbertPageEmpty4}
\pastebutton{ugxFloatHilbertPageFull4}{\hidepaste}
\tab{5}\spadcommand{b: Matrix DoubleFloat := matrix [[1/(i+j+1$DoubleFloat) for j in 0..9] for i in 0..9];\
\indentrel{3}\begin{verbatim}
                               Type: Matrix DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFloatHilbertPageEmpty4}
\begin{paste}{ugxFloatHilbertPageEmpty4}{ugxFloatHilbertPagePatch4}
\pastebutton{ugxFloatHilbertPageEmpty4}{\showpaste}
\tab{5}\spadcommand{b: Matrix DoubleFloat := matrix [[1/(i+j+1$DoubleFloat) for j in 0..9] for i in 0..9];\
\end{paste}\end{patch}

\begin{patch}{ugxFloatHilbertPagePatch5}
\begin{paste}{ugxFloatHilbertPageFull5}{ugxFloatHilbertPageEmpty5}
```

```
\pastebutton{ugxFloatHilbertPageFull5}{\hidepaste}
\tab{5}\spadcommand{determinant b\free{b }}
\indentrel{3}\begin{verbatim}
   (5)  2.1643677945721411e-53
                                      Type: DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFloatHilbertPageEmpty5}
\begin{paste}{ugxFloatHilbertPageEmpty5}{ugxFloatHilbertPagePatch5}
\pastebutton{ugxFloatHilbertPageEmpty5}{\showpaste}
\tab{5}\spadcommand{determinant b\free{b }}
\end{paste}\end{patch}

\begin{patch}{ugxFloatHilbertPagePatch6}
\begin{paste}{ugxFloatHilbertPageFull6}{ugxFloatHilbertPageEmpty6}
\pastebutton{ugxFloatHilbertPageFull6}{\hidepaste}
\tab{5}\spadcommand{digits 40\bound{d40 }}
\indentrel{3}\begin{verbatim}
   (6)  20
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFloatHilbertPageEmpty6}
\begin{paste}{ugxFloatHilbertPageEmpty6}{ugxFloatHilbertPagePatch6}
\pastebutton{ugxFloatHilbertPageEmpty6}{\showpaste}
\tab{5}\spadcommand{digits 40\bound{d40 }}
\end{paste}\end{patch}

\begin{patch}{ugxFloatHilbertPagePatch7}
\begin{paste}{ugxFloatHilbertPageFull7}{ugxFloatHilbertPageEmpty7}
\pastebutton{ugxFloatHilbertPageFull7}{\hidepaste}
\tab{5}\spadcommand{c: Matrix Float := matrix [[1/(i+j+1$Float) for j in 0..9] for i in 0..9];\f
\indentrel{3}\begin{verbatim}
                                     Type: Matrix Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFloatHilbertPageEmpty7}
\begin{paste}{ugxFloatHilbertPageEmpty7}{ugxFloatHilbertPagePatch7}
\pastebutton{ugxFloatHilbertPageEmpty7}{\showpaste}
\tab{5}\spadcommand{c: Matrix Float := matrix [[1/(i+j+1$Float) for j in 0..9] for i in 0..9];\f
\end{paste}\end{patch}

\begin{patch}{ugxFloatHilbertPagePatch8}
\begin{paste}{ugxFloatHilbertPageFull8}{ugxFloatHilbertPageEmpty8}
\pastebutton{ugxFloatHilbertPageFull8}{\hidepaste}
\tab{5}\spadcommand{determinant c\free{c }}
\indentrel{3}\begin{verbatim}
```

```
   (8)
   0.21641 79226 43149 18690 60594 98362 26174 36159 E -52
                                            Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFloatHilbertPageEmpty8}
\begin{paste}{ugxFloatHilbertPageEmpty8}{ugxFloatHilbertPagePatch8}
\pastebutton{ugxFloatHilbertPageEmpty8}{\showpaste}
\tab{5}\spadcommand{determinant c\free{c }}
\end{paste}\end{patch}

\begin{patch}{ugxFloatHilbertPagePatch9}
\begin{paste}{ugxFloatHilbertPageFull9}{ugxFloatHilbertPageEmpty9}
\pastebutton{ugxFloatHilbertPageFull9}{\hidepaste}
\tab{5}\spadcommand{digits 20}
\indentrel{3}\begin{verbatim}
   (9)  40
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFloatHilbertPageEmpty9}
\begin{paste}{ugxFloatHilbertPageEmpty9}{ugxFloatHilbertPagePatch9}
\pastebutton{ugxFloatHilbertPageEmpty9}{\showpaste}
\tab{5}\spadcommand{digits 20}
\end{paste}\end{patch}
```

————-

3.42 fname.ht

FileName

— **fname.ht** —

```
\begin{page}{FileNameXmpPage}{FileName}
\beginscroll

The \spadtype{FileName} domain provides an interface to the computer's
file system.
Functions are provided to manipulate file names and to test properties of
files.

The simplest way to use file names in the Axiom interpreter is to
rely on conversion to and from strings.
```

```
The syntax of these strings depends on the operating system.
\xtc{
}{
\spadpaste{fn: FileName \bound{fndecl}}
}
\xtc{
On AIX, this is a proper file syntax:
}{
\spadpaste{fn := "/spad/src/input/fname.input" \free{fndecl}\bound{fn}}
}

Although it is very convenient to be able to use string notation
for file names in the interpreter, it is desirable to have a portable
way of creating and manipulating file names from within programs.
\xtc{
A measure of portability is obtained by considering a file name
to consist of three parts: the {\it directory}, the {\it name},
and the {\it extension}.
}{
\spadpaste{directory fn \free{fn}}
}
\xtc{
}{
\spadpaste{name fn \free{fn}}
}
\xtc{
}{
\spadpaste{extension fn \free{fn}}
}
The meaning of these three parts depends on the operating system.
For example, on CMS the file \spad{"SPADPROF INPUT M"}
would have directory \spad{"M"}, name \spad{"SPADPROF"} and
extension \spad{"INPUT"}.

\xtc{
It is possible to create a filename from its parts.
}{
\spadpaste{fn := filename("/u/smwatt/work", "fname", "input")
\free{fndecl}\bound{fn1}}
}
\xtc{
When writing programs, it is helpful to refer to directories via
variables.
}{
\spadpaste{objdir := "/tmp" \bound{objdir}}
}
\xtc{
}{
\spadpaste{fn := filename(objdir, "table", "spad")
\free{fndecl,objdir}\bound{fn2}}
```

```
}
\xtc{
If the directory or the extension is given as an empty string, then
a default is used.  On AIX, the defaults are the current directory
and no extension.
}{
\spadpaste{fn := filename("", "letter", "") \free{fndecl}\bound{fn3}}
}

Three tests provide information about names in the file system.
\xtc{
The \spadfunFrom{exists?}{FileName} operation tests whether
the named file exists.
}{
\spadpaste{exists? "/etc/passwd"}
}
\xtc{
The operation \spadfunFrom{readable?}{FileName} tells whether the named
file can be read.  If the file does not exist, then it cannot be read.
}{
\spadpaste{readable? "/etc/passwd"}
}
\xtc{
}{
\spadpaste{readable? "/etc/security/passwd"}
}
\xtc{
}{
\spadpaste{readable? "/ect/passwd"}
}
\xtc{
Likewise, the operation \spadfunFrom{writable?}{FileName}
tells whether the named file
can be written.
If the file does not exist, the test is determined
by the properties of the directory.
}{
\spadpaste{writable? "/etc/passwd"}
}
\xtc{
}{
\spadpaste{writable? "/dev/null"}
}
\xtc{
}{
\spadpaste{writable? "/etc/DoesNotExist"}
}
\xtc{
}{
\spadpaste{writable? "/tmp/DoesNotExist"}
```

```
}

The \spadfunFrom{new}{FileName} operation constructs the name of a new
writable file.
The argument sequence is the same as for \spadfunFrom{filename}{FileName},
except that the name part is actually a prefix for a constructed
unique name.
\xtc{
The resulting file is in the specified directory
with the given extension, and the same defaults are used.
}{
\spadpaste{fn := new(objdir, "xxx", "yy") \free{objdir,fndecl}\bound{fn4}}
}
\endscroll
\autobuttons
\end{page}

\begin{patch}{FileNameXmpPagePatch1}
\begin{paste}{FileNameXmpPageFull1}{FileNameXmpPageEmpty1}
\pastebutton{FileNameXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{fn: FileName\bound{fndecl }}
\indentrel{3}\begin{verbatim}
                                        Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FileNameXmpPageEmpty1}
\begin{paste}{FileNameXmpPageEmpty1}{FileNameXmpPagePatch1}
\pastebutton{FileNameXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{fn: FileName\bound{fndecl }}
\end{paste}\end{patch}

\begin{patch}{FileNameXmpPagePatch2}
\begin{paste}{FileNameXmpPageFull2}{FileNameXmpPageEmpty2}
\pastebutton{FileNameXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{fn := "/spad/src/input/fname.input"\free{fndecl }\bound{fn }}
\indentrel{3}\begin{verbatim}
   (2)  "/spad/src/input/fname.input"
                                    Type: FileName
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FileNameXmpPageEmpty2}
\begin{paste}{FileNameXmpPageEmpty2}{FileNameXmpPagePatch2}
\pastebutton{FileNameXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{fn := "/spad/src/input/fname.input"\free{fndecl }\bound{fn }}
\end{paste}\end{patch}

\begin{patch}{FileNameXmpPagePatch3}
\begin{paste}{FileNameXmpPageFull3}{FileNameXmpPageEmpty3}
```

```
\pastebutton{FileNameXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{directory fn\free{fn }}
\indentrel{3}\begin{verbatim}
   (3)  "/spad/src/input"
                                          Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FileNameXmpPageEmpty3}
\begin{paste}{FileNameXmpPageEmpty3}{FileNameXmpPagePatch3}
\pastebutton{FileNameXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{directory fn\free{fn }}
\end{paste}\end{patch}

\begin{patch}{FileNameXmpPagePatch4}
\begin{paste}{FileNameXmpPageFull4}{FileNameXmpPageEmpty4}
\pastebutton{FileNameXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{name fn\free{fn }}
\indentrel{3}\begin{verbatim}
   (4)  "fname"
                                          Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FileNameXmpPageEmpty4}
\begin{paste}{FileNameXmpPageEmpty4}{FileNameXmpPagePatch4}
\pastebutton{FileNameXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{name fn\free{fn }}
\end{paste}\end{patch}

\begin{patch}{FileNameXmpPagePatch5}
\begin{paste}{FileNameXmpPageFull5}{FileNameXmpPageEmpty5}
\pastebutton{FileNameXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{extension fn\free{fn }}
\indentrel{3}\begin{verbatim}
   (5)  "input"
                                          Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FileNameXmpPageEmpty5}
\begin{paste}{FileNameXmpPageEmpty5}{FileNameXmpPagePatch5}
\pastebutton{FileNameXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{extension fn\free{fn }}
\end{paste}\end{patch}

\begin{patch}{FileNameXmpPagePatch6}
\begin{paste}{FileNameXmpPageFull6}{FileNameXmpPageEmpty6}
\pastebutton{FileNameXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{fn := filename("/u/smwatt/work", "fname", "input")\free{fndecl }\bound{fn1 }}
```

```
\indentrel{3}\begin{verbatim}
   (6)  "/u/smwatt/work/fname.input"
                                            Type: FileName
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FileNameXmpPageEmpty6}
\begin{paste}{FileNameXmpPageEmpty6}{FileNameXmpPagePatch6}
\pastebutton{FileNameXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{fn := filename("/u/smwatt/work", "fname", "input")\free{fndecl }\bound{fn1 }
\end{paste}\end{patch}

\begin{patch}{FileNameXmpPagePatch7}
\begin{paste}{FileNameXmpPageFull7}{FileNameXmpPageEmpty7}
\pastebutton{FileNameXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{objdir := "/tmp"\bound{objdir }}
\indentrel{3}\begin{verbatim}
   (7)  "/tmp"
                                              Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FileNameXmpPageEmpty7}
\begin{paste}{FileNameXmpPageEmpty7}{FileNameXmpPagePatch7}
\pastebutton{FileNameXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{objdir := "/tmp"\bound{objdir }}
\end{paste}\end{patch}

\begin{patch}{FileNameXmpPagePatch8}
\begin{paste}{FileNameXmpPageFull8}{FileNameXmpPageEmpty8}
\pastebutton{FileNameXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{fn := filename(objdir, "table", "spad")\free{fndecl objdir }\bound{fn2 }}
\indentrel{3}\begin{verbatim}
   (8)  "/tmp/table.spad"
                                            Type: FileName
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FileNameXmpPageEmpty8}
\begin{paste}{FileNameXmpPageEmpty8}{FileNameXmpPagePatch8}
\pastebutton{FileNameXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{fn := filename(objdir, "table", "spad")\free{fndecl objdir }\bound{fn2 }}
\end{paste}\end{patch}

\begin{patch}{FileNameXmpPagePatch9}
\begin{paste}{FileNameXmpPageFull9}{FileNameXmpPageEmpty9}
\pastebutton{FileNameXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{fn := filename("", "letter", "")\free{fndecl }\bound{fn3 }}
\indentrel{3}\begin{verbatim}
   (9)  "letter"
```

```
                                          Type: FileName
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FileNameXmpPageEmpty9}
\begin{paste}{FileNameXmpPageEmpty9}{FileNameXmpPagePatch9}
\pastebutton{FileNameXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{fn := filename("", "letter", "")\free{fndecl }\bound{fn3 }}
\end{paste}\end{patch}

\begin{patch}{FileNameXmpPagePatch10}
\begin{paste}{FileNameXmpPageFull10}{FileNameXmpPageEmpty10}
\pastebutton{FileNameXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{exists? "/etc/passwd"}
\indentrel{3}\begin{verbatim}
   (10)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FileNameXmpPageEmpty10}
\begin{paste}{FileNameXmpPageEmpty10}{FileNameXmpPagePatch10}
\pastebutton{FileNameXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{exists? "/etc/passwd"}
\end{paste}\end{patch}

\begin{patch}{FileNameXmpPagePatch11}
\begin{paste}{FileNameXmpPageFull11}{FileNameXmpPageEmpty11}
\pastebutton{FileNameXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{readable? "/etc/passwd"}
\indentrel{3}\begin{verbatim}
   (11)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FileNameXmpPageEmpty11}
\begin{paste}{FileNameXmpPageEmpty11}{FileNameXmpPagePatch11}
\pastebutton{FileNameXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{readable? "/etc/passwd"}
\end{paste}\end{patch}

\begin{patch}{FileNameXmpPagePatch12}
\begin{paste}{FileNameXmpPageFull12}{FileNameXmpPageEmpty12}
\pastebutton{FileNameXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{readable? "/etc/security/passwd"}
\indentrel{3}\begin{verbatim}
   (12)  false
                                          Type: Boolean
\end{verbatim}
```

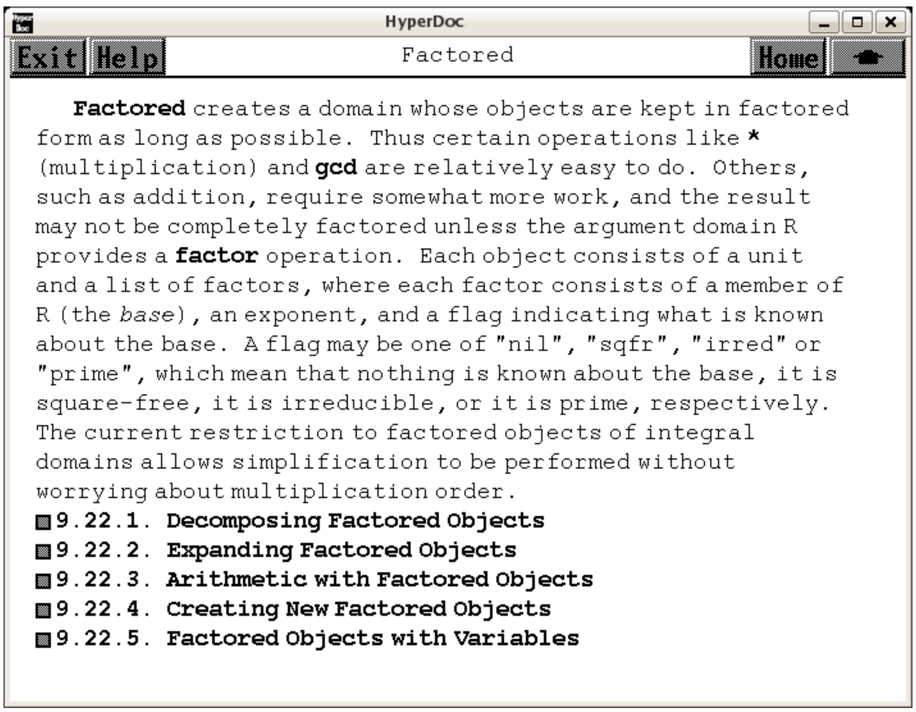
```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FileNameXmpPageEmpty12}
\begin{paste}{FileNameXmpPageEmpty12}{FileNameXmpPagePatch12}
\pastebutton{FileNameXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{readable? "/etc/security/passwd"}
\end{paste}\end{patch}

\begin{patch}{FileNameXmpPagePatch13}
\begin{paste}{FileNameXmpPageFull13}{FileNameXmpPageEmpty13}
\pastebutton{FileNameXmpPageFull13}{\hidepaste}
\tab{5}\spadcommand{readable? "/ect/passwd"}
\indentrel{3}\begin{verbatim}
   (13)  false
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FileNameXmpPageEmpty13}
\begin{paste}{FileNameXmpPageEmpty13}{FileNameXmpPagePatch13}
\pastebutton{FileNameXmpPageEmpty13}{\showpaste}
\tab{5}\spadcommand{readable? "/ect/passwd"}
\end{paste}\end{patch}

\begin{patch}{FileNameXmpPagePatch14}
\begin{paste}{FileNameXmpPageFull14}{FileNameXmpPageEmpty14}
\pastebutton{FileNameXmpPageFull14}{\hidepaste}
\tab{5}\spadcommand{writable? "/etc/passwd"}
\indentrel{3}\begin{verbatim}
   (14)  false
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FileNameXmpPageEmpty14}
\begin{paste}{FileNameXmpPageEmpty14}{FileNameXmpPagePatch14}
\pastebutton{FileNameXmpPageEmpty14}{\showpaste}
\tab{5}\spadcommand{writable? "/etc/passwd"}
\end{paste}\end{patch}

\begin{patch}{FileNameXmpPagePatch15}
\begin{paste}{FileNameXmpPageFull15}{FileNameXmpPageEmpty15}
\pastebutton{FileNameXmpPageFull15}{\hidepaste}
\tab{5}\spadcommand{writable? "/dev/null"}
\indentrel{3}\begin{verbatim}
   (15)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{FileNameXmpPageEmpty15}
\begin{paste}{FileNameXmpPageEmpty15}{FileNameXmpPagePatch15}
\pastebutton{FileNameXmpPageEmpty15}{\showpaste}
\tab{5}\spadcommand{writable? "/dev/null"}
\end{paste}\end{patch}

\begin{patch}{FileNameXmpPagePatch16}
\begin{paste}{FileNameXmpPageFull16}{FileNameXmpPageEmpty16}
\pastebutton{FileNameXmpPageFull16}{\hidepaste}
\tab{5}\spadcommand{writable? "/etc/DoesNotExist"}
\indentrel{3}\begin{verbatim}
   (16)  false
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FileNameXmpPageEmpty16}
\begin{paste}{FileNameXmpPageEmpty16}{FileNameXmpPagePatch16}
\pastebutton{FileNameXmpPageEmpty16}{\showpaste}
\tab{5}\spadcommand{writable? "/etc/DoesNotExist"}
\end{paste}\end{patch}

\begin{patch}{FileNameXmpPagePatch17}
\begin{paste}{FileNameXmpPageFull17}{FileNameXmpPageEmpty17}
\pastebutton{FileNameXmpPageFull17}{\hidepaste}
\tab{5}\spadcommand{writable? "/tmp/DoesNotExist"}
\indentrel{3}\begin{verbatim}
   (17)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FileNameXmpPageEmpty17}
\begin{paste}{FileNameXmpPageEmpty17}{FileNameXmpPagePatch17}
\pastebutton{FileNameXmpPageEmpty17}{\showpaste}
\tab{5}\spadcommand{writable? "/tmp/DoesNotExist"}
\end{paste}\end{patch}

\begin{patch}{FileNameXmpPagePatch18}
\begin{paste}{FileNameXmpPageFull18}{FileNameXmpPageEmpty18}
\pastebutton{FileNameXmpPageFull18}{\hidepaste}
\tab{5}\spadcommand{fn := new(objdir, "xxx", "yy")\free{objdir fndecl }\bound{fn4 }}
\indentrel{3}\begin{verbatim}
   (18)  "/tmp/xxx82222.yy"
                                         Type: FileName
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FileNameXmpPageEmpty18}
\begin{paste}{FileNameXmpPageEmpty18}{FileNameXmpPagePatch18}
```

```
\pastebutton{FileNameXmpPageEmpty18}{\showpaste}
\tab{5}\spadcommand{fn := new(objdir, "xxx", "yy")\free{objdir fndecl }\bound{fn4 }}
\end{paste}\end{patch}
```

3.43 fr.ht

Factored



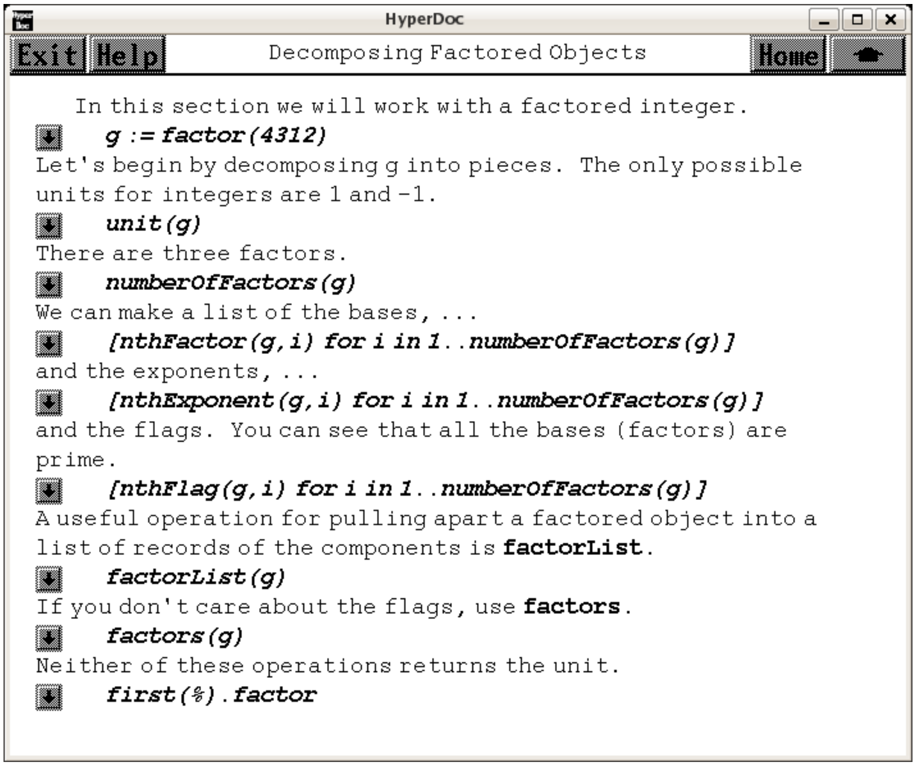
⇐ "Primes and Factorization" (ugxIntegerPrimesPage) 3.55 on page 712
⇐ "Computation of Galois Groups" (ugProblemGaloisPage) 12 on page 2306
⇐ "FactoredFunctions2" (FactoredFnsTwoXmpPage) 3.44 on page 539
⇐ "Some Examples of Domains and Packages" (ExamplesExposedPage) 3.117 on page 1369
⇒ "Decomposing Factored Objects" (ugxFactoredDecompPage) 3.43 on page 518
⇒ "Expanding Factored Objects" (ugxFactoredExpandPage) 3.43 on page 523
⇒ "Arithmetic with Factored Objects" (ugxFactoredArithPage) 3.43 on page 525
⇒ "Creating New Factored Objects" (ugxFactoredNewPage) 3.43 on page 532
⇒ "Factored Objects with Variables" (ugxFactoredVarPage) 3.43 on page 536

— **fr.ht** —

```
\begin{page}{FactoredXmpPage}{Factored}
\beginscroll

\spadtype{Factored} creates a domain whose objects are kept in factored
form as long as possible.
Thus certain operations like \spadopFrom{*}{Factored}
(multiplication) and \spadfunFrom{gcd}{Factored} are relatively
easy to do.
Others, such as addition, require somewhat more work, and
the result may not be completely factored
unless the argument domain \spad{R} provides a
\spadfunFrom{factor}{Factored} operation.
Each object consists of a unit and a list of factors, where each
factor consists of a member of \spad{R} (the {\em base}), an
exponent, and a flag indicating what is known about the base.
A flag may be one of \spad{"nil"}, \spad{"sqfr"}, \spad{"irred"}
or \spad{"prime"}, which mean that nothing is known about the
base, it is square-free, it is irreducible, or it is prime,
respectively.
The current restriction to factored objects of integral domains
allows simplification to be performed without worrying about
multiplication order.

\beginmenu
    \menudownlink{{9.22.1. Decomposing Factored Objects}}
{ugxFactoredDecompPage}
    \menudownlink{{9.22.2. Expanding Factored Objects}}
{ugxFactoredExpandPage}
    \menudownlink{{9.22.3. Arithmetic with Factored Objects}}
{ugxFactoredArithPage}
    \menudownlink{{9.22.4. Creating New Factored Objects}}
{ugxFactoredNewPage}
    \menudownlink{{9.22.5. Factored Objects with Variables}}
{ugxFactoredVarPage}
\endmenu
\endscroll
\autobuttons
\end{page}
```

———-

Decomposing Factored Objects



⇐ "Factored" (FactoredXmpPage) 3.43 on page 516

— **fr.ht** —

```
\begin{page}{ugxFactoredDecompPage}{Decomposing Factored Objects}
\beginscroll

\labelSpace{4pc}
\xtc{
In this section we will work with a factored integer.
}{
\spadpaste{g := factor(4312) \bound{g}}
}
\xtc{
Let's begin by decomposing \spad{g} into pieces.
The only possible units for integers are \spad{1} and \spad{-1}.
}{
\spadpaste{unit(g) \free{g}}
}
\xtc{
There are three factors.
```

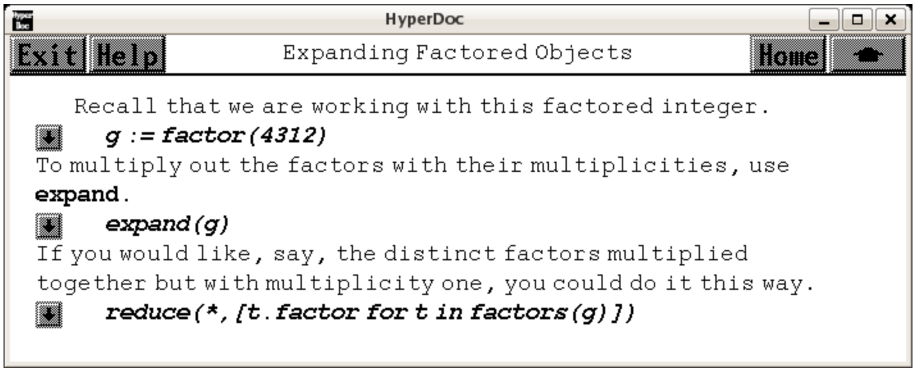
```
}{
\spadpaste{numberOfFactors(g) \free{g}}
}
\xtc{
We can make a list of the bases, \ldots
}{
\spadpaste{[nthFactor(g,i) for i in 1..numberOfFactors(g)] \free{g}}
}
\xtc{
and the exponents, \ldots
}{
\spadpaste{[nthExponent(g,i) for i in 1..numberOfFactors(g)] \free{g}}
}
\xtc{
and the flags.
You can see that all the bases (factors) are prime.
}{
\spadpaste{[nthFlag(g,i) for i in 1..numberOfFactors(g)] \free{g}}
}
\xtc{
A useful operation for pulling apart a factored object into a list
of records of the components is \spadfunFrom{factorList}{Factored}.
}{
\spadpaste{factorList(g) \free{g}}
}
\xtc{
If you don't care about the flags, use \spadfunFrom{factors}{Factored}.
}{
\spadpaste{factors(g) \free{g}\bound{prev1}}
}
\xtc{
Neither of these operations returns the unit.
}{
\spadpaste{first(\%).factor \free{prev1}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxFactoredDecompPagePatch1}
\begin{paste}{ugxFactoredDecompPageFull1}{ugxFactoredDecompPageEmpty1}
\pastebutton{ugxFactoredDecompPageFull1}{\hidepaste}
\tab{5}\spadcommand{g := factor(4312)\bound{g }}
\indentrel{3}\begin{verbatim}
         3 2
   (1)  2 7 11
                                 Type: Factored Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugxFactoredDecompPageEmpty1}
\begin{paste}{ugxFactoredDecompPageEmpty1}{ugxFactoredDecompPagePatch1}
\pastebutton{ugxFactoredDecompPageEmpty1}{\showpaste}
\tab{5}\spadcommand{g := factor(4312)\bound{g }}
\end{paste}\end{patch}

\begin{patch}{ugxFactoredDecompPagePatch2}
\begin{paste}{ugxFactoredDecompPageFull2}{ugxFactoredDecompPageEmpty2}
\pastebutton{ugxFactoredDecompPageFull2}{\hidepaste}
\tab{5}\spadcommand{unit(g)\free{g }}
\indentrel{3}\begin{verbatim}
   (2)  1
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFactoredDecompPageEmpty2}
\begin{paste}{ugxFactoredDecompPageEmpty2}{ugxFactoredDecompPagePatch2}
\pastebutton{ugxFactoredDecompPageEmpty2}{\showpaste}
\tab{5}\spadcommand{unit(g)\free{g }}
\end{paste}\end{patch}

\begin{patch}{ugxFactoredDecompPagePatch3}
\begin{paste}{ugxFactoredDecompPageFull3}{ugxFactoredDecompPageEmpty3}
\pastebutton{ugxFactoredDecompPageFull3}{\hidepaste}
\tab{5}\spadcommand{numberOfFactors(g)\free{g }}
\indentrel{3}\begin{verbatim}
   (3)  3
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFactoredDecompPageEmpty3}
\begin{paste}{ugxFactoredDecompPageEmpty3}{ugxFactoredDecompPagePatch3}
\pastebutton{ugxFactoredDecompPageEmpty3}{\showpaste}
\tab{5}\spadcommand{numberOfFactors(g)\free{g }}
\end{paste}\end{patch}

\begin{patch}{ugxFactoredDecompPagePatch4}
\begin{paste}{ugxFactoredDecompPageFull4}{ugxFactoredDecompPageEmpty4}
\pastebutton{ugxFactoredDecompPageFull4}{\hidepaste}
\tab{5}\spadcommand{[nthFactor(g,i) for i in 1..numberOfFactors(g)]\free{g }}
\indentrel{3}\begin{verbatim}
   (4)  [2,7,11]
                                     Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFactoredDecompPageEmpty4}
```

```
\begin{paste}{ugxFactoredDecompPageEmpty4}{ugxFactoredDecompPagePatch4}
\pastebutton{ugxFactoredDecompPageEmpty4}{\showpaste}
\tab{5}\spadcommand{[nthFactor(g,i) for i in 1..numberOfFactors(g)]\free{g }}
\end{paste}\end{patch}

\begin{patch}{ugxFactoredDecompPagePatch5}
\begin{paste}{ugxFactoredDecompPageFull5}{ugxFactoredDecompPageEmpty5}
\pastebutton{ugxFactoredDecompPageFull5}{\hidepaste}
\tab{5}\spadcommand{[nthExponent(g,i) for i in 1..numberOfFactors(g)]\free{g }}
\indentrel{3}\begin{verbatim}
   (5)  [3,2,1]
                                     Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFactoredDecompPageEmpty5}
\begin{paste}{ugxFactoredDecompPageEmpty5}{ugxFactoredDecompPagePatch5}
\pastebutton{ugxFactoredDecompPageEmpty5}{\showpaste}
\tab{5}\spadcommand{[nthExponent(g,i) for i in 1..numberOfFactors(g)]\free{g }}
\end{paste}\end{patch}

\begin{patch}{ugxFactoredDecompPagePatch6}
\begin{paste}{ugxFactoredDecompPageFull6}{ugxFactoredDecompPageEmpty6}
\pastebutton{ugxFactoredDecompPageFull6}{\hidepaste}
\tab{5}\spadcommand{[nthFlag(g,i) for i in 1..numberOfFactors(g)]\free{g }}
\indentrel{3}\begin{verbatim}
   (6)  ["prime","prime","prime"]
         Type: List Union("nil","sqfr","irred","prime")
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFactoredDecompPageEmpty6}
\begin{paste}{ugxFactoredDecompPageEmpty6}{ugxFactoredDecompPagePatch6}
\pastebutton{ugxFactoredDecompPageEmpty6}{\showpaste}
\tab{5}\spadcommand{[nthFlag(g,i) for i in 1..numberOfFactors(g)]\free{g }}
\end{paste}\end{patch}

\begin{patch}{ugxFactoredDecompPagePatch7}
\begin{paste}{ugxFactoredDecompPageFull7}{ugxFactoredDecompPageEmpty7}
\pastebutton{ugxFactoredDecompPageFull7}{\hidepaste}
\tab{5}\spadcommand{factorList(g)\free{g }}
\indentrel{3}\begin{verbatim}
   (7)
   [[flg= "prime",fctr= 2,xpnt= 3],
    [flg= "prime",fctr= 7,xpnt= 2],
    [flg= "prime",fctr= 11,xpnt= 1]]
Type: List Record(flg: Union("nil","sqfr","irred","prime"),fctr: Integer,xpnt: Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugxFactoredDecompPageEmpty7}
\begin{paste}{ugxFactoredDecompPageEmpty7}{ugxFactoredDecompPagePatch7}
\pastebutton{ugxFactoredDecompPageEmpty7}{\showpaste}
\tab{5}\spadcommand{factorList(g)\free{g }}
\end{paste}\end{patch}

\begin{patch}{ugxFactoredDecompPagePatch8}
\begin{paste}{ugxFactoredDecompPageFull8}{ugxFactoredDecompPageEmpty8}
\pastebutton{ugxFactoredDecompPageFull8}{\hidepaste}
\tab{5}\spadcommand{factors(g)\free{g }\bound{prev1 }}
\indentrel{3}\begin{verbatim}
   (8)
   [[factor= 2,exponent= 3], [factor= 7,exponent= 2],
    [factor= 11,exponent= 1]]
   Type: List Record(factor: Integer,exponent: Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFactoredDecompPageEmpty8}
\begin{paste}{ugxFactoredDecompPageEmpty8}{ugxFactoredDecompPagePatch8}
\pastebutton{ugxFactoredDecompPageEmpty8}{\showpaste}
\tab{5}\spadcommand{factors(g)\free{g }\bound{prev1 }}
\end{paste}\end{patch}

\begin{patch}{ugxFactoredDecompPagePatch9}
\begin{paste}{ugxFactoredDecompPageFull9}{ugxFactoredDecompPageEmpty9}
\pastebutton{ugxFactoredDecompPageFull9}{\hidepaste}
\tab{5}\spadcommand{first(\%).factor\free{prev1 }}
\indentrel{3}\begin{verbatim}
   (9)  2
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFactoredDecompPageEmpty9}
\begin{paste}{ugxFactoredDecompPageEmpty9}{ugxFactoredDecompPagePatch9}
\pastebutton{ugxFactoredDecompPageEmpty9}{\showpaste}
\tab{5}\spadcommand{first(\%).factor\free{prev1 }}
\end{paste}\end{patch}
```

Expanding Factored Objects



⇐ “Factored” (FactoredXmpPage) 3.43 on page 516

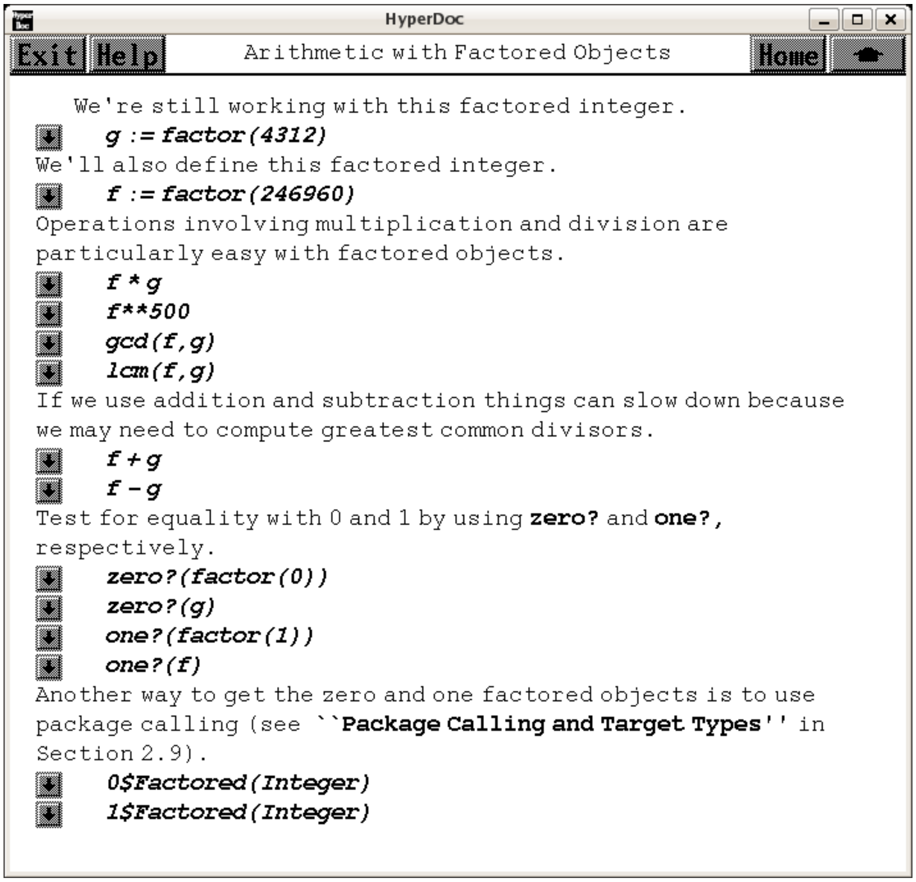
— **fr.ht** —

```
\begin{page}{ugxFactoredExpandPage}{Expanding Factored Objects}
\beginscroll

\labelSpace{4pc}
\xtc{
Recall that we are working with this factored integer.
}{
\spadpaste{g := factor(4312) \bound{g}}
}
\xtc{
To multiply out the factors with their multiplicities, use
\spadfunFrom{expand}{Factored}.
}{
\spadpaste{expand(g) \free{g}}
}
\xtc{
If you would like, say, the distinct factors multiplied together
but with multiplicity one, you could do it this way.
}{
\spadpaste{reduce(*,[t.factor for t in factors(g)]) \free{g}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxFactoredExpandPagePatch1}
\begin{paste}{ugxFactoredExpandPageFull1}{ugxFactoredExpandPageEmpty1}
\pastebutton{ugxFactoredExpandPageFull1}{\hidepaste}
\tab{5}\spadcommand{g := factor(4312)\bound{g }}
\indentrel{3}\begin{verbatim}
```

```
        3 2
   (1)  2 7 11
                                  Type: Factored Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFactoredExpandPageEmpty1}
\begin{paste}{ugxFactoredExpandPageEmpty1}{ugxFactoredExpandPagePatch1}
\pastebutton{ugxFactoredExpandPageEmpty1}{\showpaste}
\tab{5}\spadcommand{g := factor(4312)\bound{g }}
\end{paste}\end{patch}

\begin{patch}{ugxFactoredExpandPagePatch2}
\begin{paste}{ugxFactoredExpandPageFull2}{ugxFactoredExpandPageEmpty2}
\pastebutton{ugxFactoredExpandPageFull2}{\hidepaste}
\tab{5}\spadcommand{expand(g)\free{g }}
\indentrel{3}\begin{verbatim}
   (2)  4312
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFactoredExpandPageEmpty2}
\begin{paste}{ugxFactoredExpandPageEmpty2}{ugxFactoredExpandPagePatch2}
\pastebutton{ugxFactoredExpandPageEmpty2}{\showpaste}
\tab{5}\spadcommand{expand(g)\free{g }}
\end{paste}\end{patch}

\begin{patch}{ugxFactoredExpandPagePatch3}
\begin{paste}{ugxFactoredExpandPageFull3}{ugxFactoredExpandPageEmpty3}
\pastebutton{ugxFactoredExpandPageFull3}{\hidepaste}
\tab{5}\spadcommand{reduce(*,[t.factor for t in factors(g)])\free{g }}
\indentrel{3}\begin{verbatim}
   (3)  154
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFactoredExpandPageEmpty3}
\begin{paste}{ugxFactoredExpandPageEmpty3}{ugxFactoredExpandPagePatch3}
\pastebutton{ugxFactoredExpandPageEmpty3}{\showpaste}
\tab{5}\spadcommand{reduce(*,[t.factor for t in factors(g)])\free{g }}
\end{paste}\end{patch}
```

———-

Arithmetic with Factored Objects



⇐ "Factored" (FactoredXmpPage) 3.43 on page 516

— **fr.ht** —

```
\begin{page}{ugxFactoredArithPage}{Arithmetic with Factored Objects}
\beginscroll

\labelSpace{4pc}
\xtc{
We're still working with this factored integer.
}{
\spadpaste{g := factor(4312) \bound{g}}
}
\xtc{
We'll also define this factored integer.
}{
\spadpaste{f := factor(246960) \bound{f}}
```

```
}
\xtc{
Operations involving multiplication and division are particularly
easy with factored objects.
}{
\spadpaste{f * g \free{f g}}
}
\xtc{
}{
\spadpaste{f**500 \free{f}}
}
\xtc{
}{
\spadpaste{gcd(f,g) \free{f g}}
}
\xtc{
}{
\spadpaste{lcm(f,g) \free{f g}}
}
\xtc{
If we use addition and subtraction things can slow down because
we may need to compute greatest common divisors.
}{
\spadpaste{f + g \free{f g}}
}
\xtc{
}{
\spadpaste{f - g \free{f g}}
}
\xtc{
Test for equality with \spad{0} and \spad{1} by using
\spadfunFrom{zero?}{Factored} and \spadfunFrom{one?}{Factored},
respectively.
}{
\spadpaste{zero?(factor(0))}
}
\xtc{
}{
\spadpaste{zero?(g) \free{g}}
}
\xtc{
}{
\spadpaste{one?(factor(1))}
}
\xtc{
}{
\spadpaste{one?(f) \free{f}}
}
\xtc{
Another way to get the zero and one factored objects is to use
```

```
package calling
(see \downlink{``Package Calling and Target Types''}
{ugTypesPkgCallPage} in Section 2.9\ignore{ugTypesPkgCall}).
}{
\spadpaste{0\$Factored(Integer)}
}
\xtc{
}{
\spadpaste{1\$Factored(Integer)}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxFactoredArithPagePatch1}
\begin{paste}{ugxFactoredArithPageFull1}{ugxFactoredArithPageEmpty1}
\pastebutton{ugxFactoredArithPageFull1}{\hidepaste}
\tab{5}\spadcommand{g := factor(4312)\bound{g }}
\indentrel{3}\begin{verbatim}
         3 2
   (1)  2 7 11
                                  Type: Factored Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFactoredArithPageEmpty1}
\begin{paste}{ugxFactoredArithPageEmpty1}{ugxFactoredArithPagePatch1}
\pastebutton{ugxFactoredArithPageEmpty1}{\showpaste}
\tab{5}\spadcommand{g := factor(4312)\bound{g }}
\end{paste}\end{patch}

\begin{patch}{ugxFactoredArithPagePatch2}
\begin{paste}{ugxFactoredArithPageFull2}{ugxFactoredArithPageEmpty2}
\pastebutton{ugxFactoredArithPageFull2}{\hidepaste}
\tab{5}\spadcommand{f := factor(246960)\bound{f }}
\indentrel{3}\begin{verbatim}
         4 2   3
   (2)  2 3 5 7
                                  Type: Factored Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFactoredArithPageEmpty2}
\begin{paste}{ugxFactoredArithPageEmpty2}{ugxFactoredArithPagePatch2}
\pastebutton{ugxFactoredArithPageEmpty2}{\showpaste}
\tab{5}\spadcommand{f := factor(246960)\bound{f }}
\end{paste}\end{patch}

\begin{patch}{ugxFactoredArithPagePatch3}
```

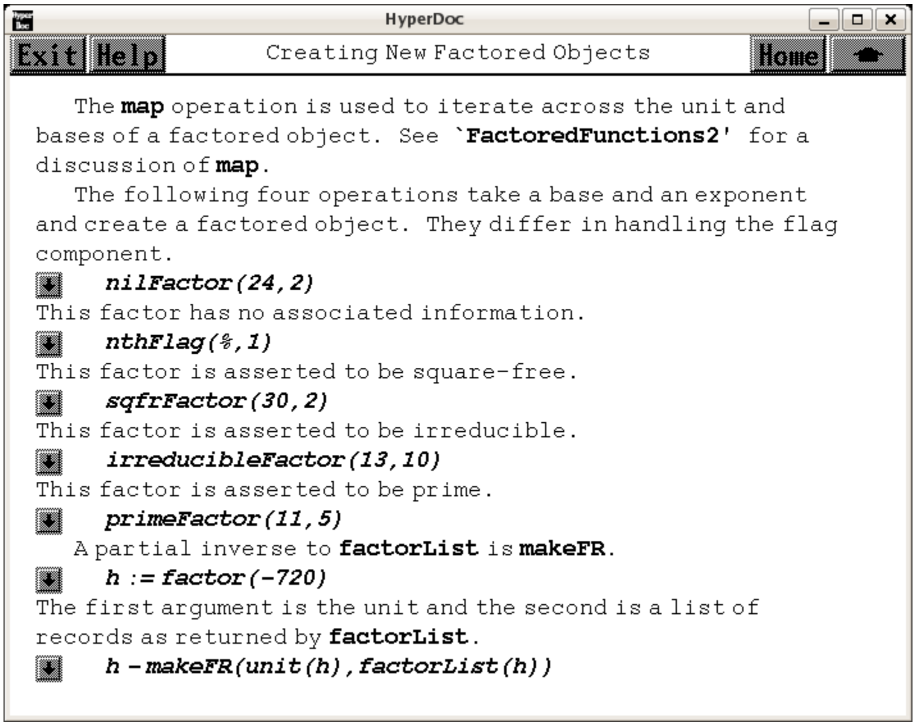
```
\begin{paste}{ugxFactoredArithPageFull3}{ugxFactoredArithPageEmpty3}
\pastebutton{ugxFactoredArithPageFull3}{\hidepaste}
\tab{5}\spadcommand{f * g\free{f g }}
\indentrel{3}\begin{verbatim}
         7 2   5
   (3)  2 3 5 7 11
                                     Type: Factored Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFactoredArithPageEmpty3}
\begin{paste}{ugxFactoredArithPageEmpty3}{ugxFactoredArithPagePatch3}
\pastebutton{ugxFactoredArithPageEmpty3}{\showpaste}
\tab{5}\spadcommand{f * g\free{f g }}
\end{paste}\end{patch}

\begin{patch}{ugxFactoredArithPagePatch4}
\begin{paste}{ugxFactoredArithPageFull4}{ugxFactoredArithPageEmpty4}
\pastebutton{ugxFactoredArithPageFull4}{\hidepaste}
\tab{5}\spadcommand{f**500\free{f }}
\indentrel{3}\begin{verbatim}
         2000 1000 500 1500
   (4)  2    3    5   7
                                     Type: Factored Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFactoredArithPageEmpty4}
\begin{paste}{ugxFactoredArithPageEmpty4}{ugxFactoredArithPagePatch4}
\pastebutton{ugxFactoredArithPageEmpty4}{\showpaste}
\tab{5}\spadcommand{f**500\free{f }}
\end{paste}\end{patch}

\begin{patch}{ugxFactoredArithPagePatch5}
\begin{paste}{ugxFactoredArithPageFull5}{ugxFactoredArithPageEmpty5}
\pastebutton{ugxFactoredArithPageFull5}{\hidepaste}
\tab{5}\spadcommand{gcd(f,g)\free{f g }}
\indentrel{3}\begin{verbatim}
         3 2
   (5)  2 7
                                     Type: Factored Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFactoredArithPageEmpty5}
\begin{paste}{ugxFactoredArithPageEmpty5}{ugxFactoredArithPagePatch5}
\pastebutton{ugxFactoredArithPageEmpty5}{\showpaste}
\tab{5}\spadcommand{gcd(f,g)\free{f g }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugxFactoredArithPagePatch6}
\begin{paste}{ugxFactoredArithPageFull6}{ugxFactoredArithPageEmpty6}
\pastebutton{ugxFactoredArithPageFull6}{\hidepaste}
\tab{5}\spadcommand{lcm(f,g)\free{f g }}
\indentrel{3}\begin{verbatim}
        4 2   3
   (6)  2 3 5 7 11
                                Type: Factored Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFactoredArithPageEmpty6}
\begin{paste}{ugxFactoredArithPageEmpty6}{ugxFactoredArithPagePatch6}
\pastebutton{ugxFactoredArithPageEmpty6}{\showpaste}
\tab{5}\spadcommand{lcm(f,g)\free{f g }}
\end{paste}\end{patch}

\begin{patch}{ugxFactoredArithPagePatch7}
\begin{paste}{ugxFactoredArithPageFull7}{ugxFactoredArithPageEmpty7}
\pastebutton{ugxFactoredArithPageFull7}{\hidepaste}
\tab{5}\spadcommand{f + g\free{f g }}
\indentrel{3}\begin{verbatim}
        3 2
   (7)  2 7 641
                                Type: Factored Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFactoredArithPageEmpty7}
\begin{paste}{ugxFactoredArithPageEmpty7}{ugxFactoredArithPagePatch7}
\pastebutton{ugxFactoredArithPageEmpty7}{\showpaste}
\tab{5}\spadcommand{f + g\free{f g }}
\end{paste}\end{patch}

\begin{patch}{ugxFactoredArithPagePatch8}
\begin{paste}{ugxFactoredArithPageFull8}{ugxFactoredArithPageEmpty8}
\pastebutton{ugxFactoredArithPageFull8}{\hidepaste}
\tab{5}\spadcommand{f - g\free{f g }}
\indentrel{3}\begin{verbatim}
        3 2
   (8)  2 7 619
                                Type: Factored Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFactoredArithPageEmpty8}
\begin{paste}{ugxFactoredArithPageEmpty8}{ugxFactoredArithPagePatch8}
\pastebutton{ugxFactoredArithPageEmpty8}{\showpaste}
\tab{5}\spadcommand{f - g\free{f g }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugxFactoredArithPagePatch9}
\begin{paste}{ugxFactoredArithPageFull9}{ugxFactoredArithPageEmpty9}
\pastebutton{ugxFactoredArithPageFull9}{\hidepaste}
\tab{5}\spadcommand{zero?(factor(0))}
\indentrel{3}\begin{verbatim}
   (9)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFactoredArithPageEmpty9}
\begin{paste}{ugxFactoredArithPageEmpty9}{ugxFactoredArithPagePatch9}
\pastebutton{ugxFactoredArithPageEmpty9}{\showpaste}
\tab{5}\spadcommand{zero?(factor(0))}
\end{paste}\end{patch}

\begin{patch}{ugxFactoredArithPagePatch10}
\begin{paste}{ugxFactoredArithPageFull10}{ugxFactoredArithPageEmpty10}
\pastebutton{ugxFactoredArithPageFull10}{\hidepaste}
\tab{5}\spadcommand{zero?(g)\free{g }}
\indentrel{3}\begin{verbatim}
   (10)  false
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFactoredArithPageEmpty10}
\begin{paste}{ugxFactoredArithPageEmpty10}{ugxFactoredArithPagePatch10}
\pastebutton{ugxFactoredArithPageEmpty10}{\showpaste}
\tab{5}\spadcommand{zero?(g)\free{g }}
\end{paste}\end{patch}

\begin{patch}{ugxFactoredArithPagePatch11}
\begin{paste}{ugxFactoredArithPageFull11}{ugxFactoredArithPageEmpty11}
\pastebutton{ugxFactoredArithPageFull11}{\hidepaste}
\tab{5}\spadcommand{one?(factor(1))}
\indentrel{3}\begin{verbatim}
   (11)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFactoredArithPageEmpty11}
\begin{paste}{ugxFactoredArithPageEmpty11}{ugxFactoredArithPagePatch11}
\pastebutton{ugxFactoredArithPageEmpty11}{\showpaste}
\tab{5}\spadcommand{one?(factor(1))}
\end{paste}\end{patch}

\begin{patch}{ugxFactoredArithPagePatch12}
```

```
\begin{paste}{ugxFactoredArithPageFull12}{ugxFactoredArithPageEmpty12}
\pastebutton{ugxFactoredArithPageFull12}{\hidepaste}
\tab{5}\spadcommand{one?(f)\free{f }}
\indentrel{3}\begin{verbatim}
   (12)  false
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFactoredArithPageEmpty12}
\begin{paste}{ugxFactoredArithPageEmpty12}{ugxFactoredArithPagePatch12}
\pastebutton{ugxFactoredArithPageEmpty12}{\showpaste}
\tab{5}\spadcommand{one?(f)\free{f }}
\end{paste}\end{patch}

\begin{patch}{ugxFactoredArithPagePatch13}
\begin{paste}{ugxFactoredArithPageFull13}{ugxFactoredArithPageEmpty13}
\pastebutton{ugxFactoredArithPageFull13}{\hidepaste}
\tab{5}\spadcommand{0$Factored(Integer)}
\indentrel{3}\begin{verbatim}
   (13)  0
                                 Type: Factored Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFactoredArithPageEmpty13}
\begin{paste}{ugxFactoredArithPageEmpty13}{ugxFactoredArithPagePatch13}
\pastebutton{ugxFactoredArithPageEmpty13}{\showpaste}
\tab{5}\spadcommand{0$Factored(Integer)}
\end{paste}\end{patch}

\begin{patch}{ugxFactoredArithPagePatch14}
\begin{paste}{ugxFactoredArithPageFull14}{ugxFactoredArithPageEmpty14}
\pastebutton{ugxFactoredArithPageFull14}{\hidepaste}
\tab{5}\spadcommand{1$Factored(Integer)}
\indentrel{3}\begin{verbatim}
   (14)  1
                                 Type: Factored Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFactoredArithPageEmpty14}
\begin{paste}{ugxFactoredArithPageEmpty14}{ugxFactoredArithPagePatch14}
\pastebutton{ugxFactoredArithPageEmpty14}{\showpaste}
\tab{5}\spadcommand{1$Factored(Integer)}
\end{paste}\end{patch}
```

Creating New Factored Objects



⇐ “Factored” (FactoredXmpPage) 3.43 on page 516
⇒ “FactoredFunctions2” (FactoredFnsTwoXmpPage) 3.44 on page 539

— **fr.ht** —

```
\begin{page}{ugxFactoredNewPage}{Creating New Factored Objects}
\beginscroll

The \spadfunFrom{map}{Factored} operation is used to iterate across the
unit and bases of a factored object.
See \downlink{‘FactoredFunctions2’}{FactoredFnsTwoXmpPage}
\ignore{FactoredFunctions2} for a discussion of
\spadfunFrom{map}{Factored}.

\labelSpace{1pc}
\xtc{
The following four operations take a base and an exponent and create
a factored object.
They differ in handling the flag component.
}{
\spadpaste{nilFactor(24,2) \bound{prev2}}
}
\xtc{
```

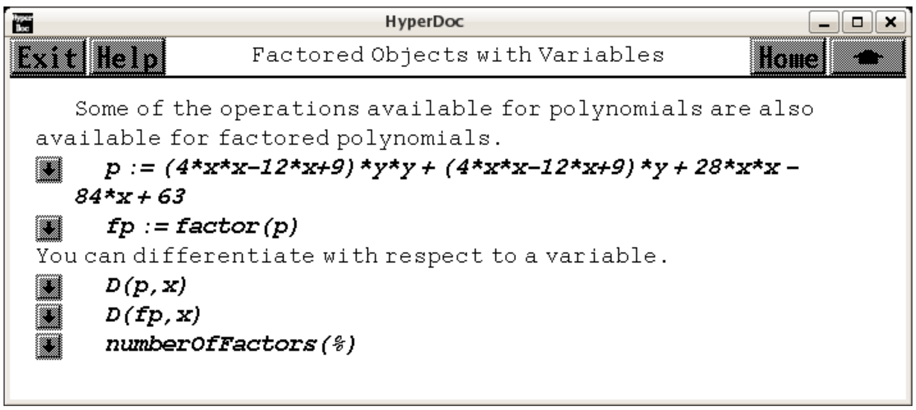
```
This factor has no associated information.
}{
\spadpaste{nthFlag(\%,1) \free{prev2}}
}
\xtc{
This factor is asserted to be square-free.
}{
\spadpaste{sqfrFactor(30,2) \bound{prev3}}
}
\xtc{
This factor is asserted to be irreducible.
}{
\spadpaste{irreducibleFactor(13,10) \bound{prev4}}
}
\xtc{
This factor is asserted to be prime.
}{
\spadpaste{primeFactor(11,5) \bound{prev5}}
}

\xtc{
A partial inverse to \spadfunFrom{factorList}{Factored} is
\spadfunFrom{makeFR}{Factored}.
}{
\spadpaste{h := factor(-720) \bound{h}}
}
\xtc{
The first argument is the unit and the second is a list of records as
returned by \spadfunFrom{factorList}{Factored}.
}{
\spadpaste{h - makeFR(unit(h),factorList(h)) \free{h}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxFactoredNewPagePatch1}
\begin{paste}{ugxFactoredNewPageFull1}{ugxFactoredNewPageEmpty1}
\pastebutton{ugxFactoredNewPageFull1}{\hidepaste}
\tab{5}\spadcommand{nilFactor(24,2)\bound{prev2 }}
\indentrel{3}\begin{verbatim}
          2
   (1)  24
                                 Type: Factored Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFactoredNewPageEmpty1}
\begin{paste}{ugxFactoredNewPageEmpty1}{ugxFactoredNewPagePatch1}
```

```
\pastebutton{ugxFactoredNewPageEmpty1}{\showpaste}
\tab{5}\spadcommand{nilFactor(24,2)\bound{prev2 }}
\end{paste}\end{patch}

\begin{patch}{ugxFactoredNewPagePatch2}
\begin{paste}{ugxFactoredNewPageFull2}{ugxFactoredNewPageEmpty2}
\pastebutton{ugxFactoredNewPageFull2}{\hidepaste}
\tab{5}\spadcommand{nthFlag(\%,1)\free{prev2 }}
\indentrel{3}\begin{verbatim}
   (2)  "nil"
                                 Type: Union("nil",...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFactoredNewPageEmpty2}
\begin{paste}{ugxFactoredNewPageEmpty2}{ugxFactoredNewPagePatch2}
\pastebutton{ugxFactoredNewPageEmpty2}{\showpaste}
\tab{5}\spadcommand{nthFlag(\%,1)\free{prev2 }}
\end{paste}\end{patch}

\begin{patch}{ugxFactoredNewPagePatch3}
\begin{paste}{ugxFactoredNewPageFull3}{ugxFactoredNewPageEmpty3}
\pastebutton{ugxFactoredNewPageFull3}{\hidepaste}
\tab{5}\spadcommand{sqfrFactor(30,2)\bound{prev3 }}
\indentrel{3}\begin{verbatim}
          2
   (3)  30
                                 Type: Factored Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFactoredNewPageEmpty3}
\begin{paste}{ugxFactoredNewPageEmpty3}{ugxFactoredNewPagePatch3}
\pastebutton{ugxFactoredNewPageEmpty3}{\showpaste}
\tab{5}\spadcommand{sqfrFactor(30,2)\bound{prev3 }}
\end{paste}\end{patch}

\begin{patch}{ugxFactoredNewPagePatch4}
\begin{paste}{ugxFactoredNewPageFull4}{ugxFactoredNewPageEmpty4}
\pastebutton{ugxFactoredNewPageFull4}{\hidepaste}
\tab{5}\spadcommand{irreducibleFactor(13,10)\bound{prev4 }}
\indentrel{3}\begin{verbatim}
          10
   (4)  13
                                 Type: Factored Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFactoredNewPageEmpty4}
\begin{paste}{ugxFactoredNewPageEmpty4}{ugxFactoredNewPagePatch4}
```

```
\pastebutton{ugxFactoredNewPageEmpty4}{\showpaste}
\tab{5}\spadcommand{irreducibleFactor(13,10)\bound{prev4 }}
\end{paste}\end{patch}

\begin{patch}{ugxFactoredNewPagePatch5}
\begin{paste}{ugxFactoredNewPageFull5}{ugxFactoredNewPageEmpty5}
\pastebutton{ugxFactoredNewPageFull5}{\hidepaste}
\tab{5}\spadcommand{primeFactor(11,5)\bound{prev5 }}
\indentrel{3}\begin{verbatim}
          5
   (5)  11
                                 Type: Factored Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFactoredNewPageEmpty5}
\begin{paste}{ugxFactoredNewPageEmpty5}{ugxFactoredNewPagePatch5}
\pastebutton{ugxFactoredNewPageEmpty5}{\showpaste}
\tab{5}\spadcommand{primeFactor(11,5)\bound{prev5 }}
\end{paste}\end{patch}

\begin{patch}{ugxFactoredNewPagePatch6}
\begin{paste}{ugxFactoredNewPageFull6}{ugxFactoredNewPageEmpty6}
\pastebutton{ugxFactoredNewPageFull6}{\hidepaste}
\tab{5}\spadcommand{h := factor(-720)\bound{h }}
\indentrel{3}\begin{verbatim}
           4 2
   (6)  - 2 3 5
                                 Type: Factored Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFactoredNewPageEmpty6}
\begin{paste}{ugxFactoredNewPageEmpty6}{ugxFactoredNewPagePatch6}
\pastebutton{ugxFactoredNewPageEmpty6}{\showpaste}
\tab{5}\spadcommand{h := factor(-720)\bound{h }}
\end{paste}\end{patch}

\begin{patch}{ugxFactoredNewPagePatch7}
\begin{paste}{ugxFactoredNewPageFull7}{ugxFactoredNewPageEmpty7}
\pastebutton{ugxFactoredNewPageFull7}{\hidepaste}
\tab{5}\spadcommand{h - makeFR(unit(h),factorList(h))\free{h }}
\indentrel{3}\begin{verbatim}
   (7)  0
                                 Type: Factored Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFactoredNewPageEmpty7}
\begin{paste}{ugxFactoredNewPageEmpty7}{ugxFactoredNewPagePatch7}
```

```
\pastebutton{ugxFactoredNewPageEmpty7}{\showpaste}
\tab{5}\spadcommand{h - makeFR(unit(h),factorList(h))\free{h }}
\end{paste}\end{patch}
```

Factored Objects with Variables



⇐ "Factored" (FactoredXmpPage) 3.43 on page 516

— **fr.ht** —

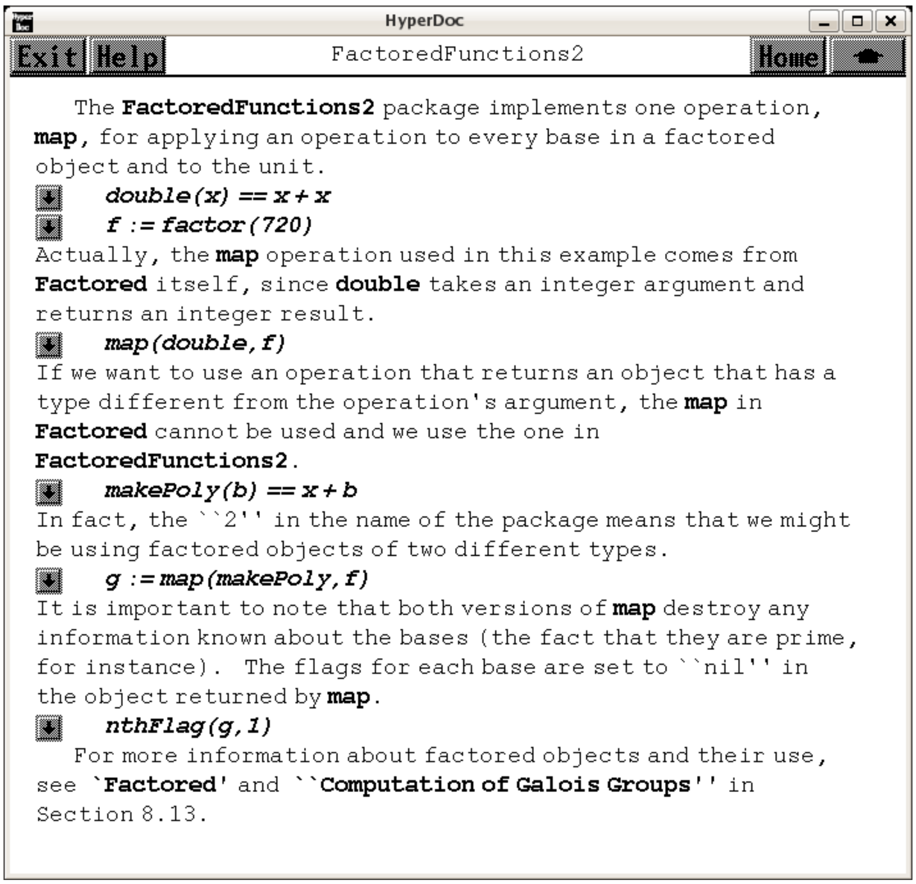
```
\begin{page}{ugxFactoredVarPage}{Factored Objects with Variables}
\beginscroll

\labelSpace{4pc}
\xtc{
Some of the operations available for polynomials are also available
for factored polynomials.
}{
\spadpaste{p := (4*x*x-12*x+9)*y*y + (4*x*x-12*x+9)*y + 28*x*x - 84*x + 63
\bound{p}}
}
\xtc{
}{
\spadpaste{fp := factor(p) \free{p}\bound{fp}}
}
\xtc{
You can differentiate with respect to a variable.
}{
\spadpaste{D(p,x) \free{p}}
}
\xtc{
}{
```

```
\spadpaste{D(fp,x) \free{fp}\bound{prev1}}
}
\xtc{
}{
\spadpaste{numberOfFactors(\%) \free{prev1}}
}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxFactoredVarPagePatch1}
\begin{paste}{ugxFactoredVarPageFull1}{ugxFactoredVarPageEmpty1}
\pastebutton{ugxFactoredVarPageFull1}{\hidepaste}
\tab{5}\spadcommand{p := (4*x*x-12*x+9)*y*y + (4*x*x-12*x+9)*y + 28*x*x - 84*x + 63\bound{p }}
\indentrel{3}\begin{verbatim}
   (1)
      2            2      2                  2
   (4x  - 12x + 9)y  + (4x  - 12x + 9)y + 28x  - 84x + 63
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFactoredVarPageEmpty1}
\begin{paste}{ugxFactoredVarPageEmpty1}{ugxFactoredVarPagePatch1}
\pastebutton{ugxFactoredVarPageEmpty1}{\showpaste}
\tab{5}\spadcommand{p := (4*x*x-12*x+9)*y*y + (4*x*x-12*x+9)*y + 28*x*x - 84*x + 63\bound{p }}
\end{paste}\end{patch}

\begin{patch}{ugxFactoredVarPagePatch2}
\begin{paste}{ugxFactoredVarPageFull2}{ugxFactoredVarPageEmpty2}
\pastebutton{ugxFactoredVarPageFull2}{\hidepaste}
\tab{5}\spadcommand{fp := factor(p)\free{p }\bound{fp }}
\indentrel{3}\begin{verbatim}
                2  2
   (2)  (2x - 3) (y  + y + 7)
                      Type: Factored Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFactoredVarPageEmpty2}
\begin{paste}{ugxFactoredVarPageEmpty2}{ugxFactoredVarPagePatch2}
\pastebutton{ugxFactoredVarPageEmpty2}{\showpaste}
\tab{5}\spadcommand{fp := factor(p)\free{p }\bound{fp }}
\end{paste}\end{patch}

\begin{patch}{ugxFactoredVarPagePatch3}
\begin{paste}{ugxFactoredVarPageFull3}{ugxFactoredVarPageEmpty3}
\pastebutton{ugxFactoredVarPageFull3}{\hidepaste}
\tab{5}\spadcommand{D(p,x)\free{p }}
\indentrel{3}\begin{verbatim}
```

```
                  2
   (3)  (8x - 12)y  + (8x - 12)y + 56x - 84
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFactoredVarPageEmpty3}
\begin{paste}{ugxFactoredVarPageEmpty3}{ugxFactoredVarPagePatch3}
\pastebutton{ugxFactoredVarPageEmpty3}{\showpaste}
\tab{5}\spadcommand{D(p,x)\free{p }}
\end{paste}\end{patch}

\begin{patch}{ugxFactoredVarPagePatch4}
\begin{paste}{ugxFactoredVarPageFull4}{ugxFactoredVarPageEmpty4}
\pastebutton{ugxFactoredVarPageFull4}{\hidepaste}
\tab{5}\spadcommand{D(fp,x)\free{fp }\bound{prev1 }}
\indentrel{3}\begin{verbatim}
                   2
   (4)  4(2x - 3)(y  + y + 7)
                      Type: Factored Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFactoredVarPageEmpty4}
\begin{paste}{ugxFactoredVarPageEmpty4}{ugxFactoredVarPagePatch4}
\pastebutton{ugxFactoredVarPageEmpty4}{\showpaste}
\tab{5}\spadcommand{D(fp,x)\free{fp }\bound{prev1 }}
\end{paste}\end{patch}

\begin{patch}{ugxFactoredVarPagePatch5}
\begin{paste}{ugxFactoredVarPageFull5}{ugxFactoredVarPageEmpty5}
\pastebutton{ugxFactoredVarPageFull5}{\hidepaste}
\tab{5}\spadcommand{numberOfFactors(\%)\free{prev1 }}
\indentrel{3}\begin{verbatim}
   (5)  3
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxFactoredVarPageEmpty5}
\begin{paste}{ugxFactoredVarPageEmpty5}{ugxFactoredVarPagePatch5}
\pastebutton{ugxFactoredVarPageEmpty5}{\showpaste}
\tab{5}\spadcommand{numberOfFactors(\%)\free{prev1 }}
\end{paste}\end{patch}
```

3.44 fr2.ht

FactoredFunctions2


HyperDoc

Exit Help FactoredFunctions2 Home

The **FactoredFunctions2** package implements one operation, **map**, for applying an operation to every base in a factored object and to the unit.

double(x) == x + x

f := factor(720)

Actually, the **map** operation used in this example comes from **Factored** itself, since **double** takes an integer argument and returns an integer result.

map(double,f)

If we want to use an operation that returns an object that has a type different from the operation's argument, the **map** in **Factored** cannot be used and we use the one in **FactoredFunctions2**.

makePoly(b) == x + b

In fact, the ``2'' in the name of the package means that we might be using factored objects of two different types.

g := map(makePoly,f)

It is important to note that both versions of **map** destroy any information known about the bases (the fact that they are prime, for instance). The flags for each base are set to ``nil'' in the object returned by **map**.

nthFlag(g,1)

For more information about factored objects and their use, see `**Factored**' and ``**Computation of Galois Groups**'' in Section 8.13.

⇐ "Creating New Factored Objects" (ugxFactoredNewPage) 3.43 on page 532
⇒ "Factored" (FactoredXmpPage) 3.43 on page 516
⇒ "Computation of Galois Groups" (ugProblemGaloisPage) 12 on page 2306

— **fr2.ht** —

```
\begin{page}{FactoredFnsTwoXmpPage}{FactoredFunctions2}
\beginscroll

The \spadtype{FactoredFunctions2} package implements one operation,
\spadfunFrom{map}{FactoredFunctions2}, for applying an operation to every
base in a factored object and to the unit.
\xtc{
}{
\spadpaste{double(x) == x + x \bound{double}}
```

```
}
\xtc{
}{
\spadpaste{f := factor(720) \bound{f}}
}
\xtc{
Actually, the \spadfunFrom{map}{FactoredFunctions2} operation used
in this example comes from \spadtype{Factored} itself, since
\userfun{double} takes an integer argument and returns an integer
result.
}{
\spadpaste{map(double,f) \free{f}\free{double}}
}
\xtc{
If we want to use an operation that returns an object that has a type
different from the operation's argument,
the \spadfunFrom{map}{FactoredFunctions2} in \spadtype{Factored}
cannot be used and we use the one in \spadtype{FactoredFunctions2}.
}{
\spadpaste{makePoly(b) == x + b \bound{makePoly}}
}
\xtc{
In fact, the ``2'' in the name of the package means that we might
be using factored objects of two different types.
}{
\spadpaste{g := map(makePoly,f) \free{f}\free{makePoly}\bound{g}}
}
It is important to note that both versions of
\spadfunFrom{map}{FactoredFunctions2}
destroy any information known about the bases (the fact that
they are prime, for instance).
\xtc{
The flags for each base are set to ``nil'' in the object returned
by \spadfunFrom{map}{FactoredFunctions2}.
}{
\spadpaste{nthFlag(g,1) \free{g}}
}

For more information about factored objects and their use, see
\downlink{`Factored'}{FactoredXmpPage}\ignore{Factored} and
\downlink{``Computation of Galois Groups''}{ugProblemGaloisPage} in
Section 8.13\ignore{ugProblemGalois}.
\endscroll
\autobuttons
\end{page}

\begin{patch}{FactoredFnsTwoXmpPagePatch1}
\begin{paste}{FactoredFnsTwoXmpPageFull1}{FactoredFnsTwoXmpPageEmpty1}
\pastebutton{FactoredFnsTwoXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{double(x) == x + x\bound{double }}
```

```
\indentrel{3}\begin{verbatim}
                                              Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FactoredFnsTwoXmpPageEmpty1}
\begin{paste}{FactoredFnsTwoXmpPageEmpty1}{FactoredFnsTwoXmpPagePatch1}
\pastebutton{FactoredFnsTwoXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{double(x) == x + x\bound{double }}
\end{paste}\end{patch}

\begin{patch}{FactoredFnsTwoXmpPagePatch2}
\begin{paste}{FactoredFnsTwoXmpPageFull2}{FactoredFnsTwoXmpPageEmpty2}
\pastebutton{FactoredFnsTwoXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{f := factor(720)\bound{f }}
\indentrel{3}\begin{verbatim}
         4 2
   (2)  2 3 5
                                  Type: Factored Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FactoredFnsTwoXmpPageEmpty2}
\begin{paste}{FactoredFnsTwoXmpPageEmpty2}{FactoredFnsTwoXmpPagePatch2}
\pastebutton{FactoredFnsTwoXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{f := factor(720)\bound{f }}
\end{paste}\end{patch}

\begin{patch}{FactoredFnsTwoXmpPagePatch3}
\begin{paste}{FactoredFnsTwoXmpPageFull3}{FactoredFnsTwoXmpPageEmpty3}
\pastebutton{FactoredFnsTwoXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{map(double,f)\free{f }\free{double }}
\indentrel{3}\begin{verbatim}
           4 2
   (3)  2 4 6 10
                                  Type: Factored Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FactoredFnsTwoXmpPageEmpty3}
\begin{paste}{FactoredFnsTwoXmpPageEmpty3}{FactoredFnsTwoXmpPagePatch3}
\pastebutton{FactoredFnsTwoXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{map(double,f)\free{f }\free{double }}
\end{paste}\end{patch}

\begin{patch}{FactoredFnsTwoXmpPagePatch4}
\begin{paste}{FactoredFnsTwoXmpPageFull4}{FactoredFnsTwoXmpPageEmpty4}
\pastebutton{FactoredFnsTwoXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{makePoly(b) == x + b\bound{makePoly }}
\indentrel{3}\begin{verbatim}
```

```
                                                Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FactoredFnsTwoXmpPageEmpty4}
\begin{paste}{FactoredFnsTwoXmpPageEmpty4}{FactoredFnsTwoXmpPagePatch4}
\pastebutton{FactoredFnsTwoXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{makePoly(b) == x + b\bound{makePoly }}
\end{paste}\end{patch}

\begin{patch}{FactoredFnsTwoXmpPagePatch5}
\begin{paste}{FactoredFnsTwoXmpPageFull5}{FactoredFnsTwoXmpPageEmpty5}
\pastebutton{FactoredFnsTwoXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{g := map(makePoly,f)\free{f }\free{makePoly }\bound{g }}
\indentrel{3}\begin{verbatim}
                      4       2
   (5)  (x + 1)(x + 2) (x + 3) (x + 5)
                      Type: Factored Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FactoredFnsTwoXmpPageEmpty5}
\begin{paste}{FactoredFnsTwoXmpPageEmpty5}{FactoredFnsTwoXmpPagePatch5}
\pastebutton{FactoredFnsTwoXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{g := map(makePoly,f)\free{f }\free{makePoly }\bound{g }}
\end{paste}\end{patch}

\begin{patch}{FactoredFnsTwoXmpPagePatch6}
\begin{paste}{FactoredFnsTwoXmpPageFull6}{FactoredFnsTwoXmpPageEmpty6}
\pastebutton{FactoredFnsTwoXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{nthFlag(g,1)\free{g }}
\indentrel{3}\begin{verbatim}
   (6)  "nil"
                                 Type: Union("nil",...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FactoredFnsTwoXmpPageEmpty6}
\begin{paste}{FactoredFnsTwoXmpPageEmpty6}{FactoredFnsTwoXmpPagePatch6}
\pastebutton{FactoredFnsTwoXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{nthFlag(g,1)\free{g }}
\end{paste}\end{patch}
```

3.45 frac.ht

Fraction

⇒ “Integer” (IntegerXmpPage) 3.55 on page 696
⇒ “notitle” (ContinuedFractionXmpPage) 3.17 on page 262
⇒ “notitle” (PartialFractionXmpPage) 3.86 on page 1088

— **frac.ht** —

```
\begin{page}{FractionXmpPage}{Fraction}
\beginscroll

The \spadtype{Fraction} domain implements quotients.
The elements must belong to a domain of category \spadtype{IntegralDomain}:
multiplication must be commutative and the product of two non-zero
elements must not be zero.
This allows you to make fractions of most things you would think of,
but don't expect to create a fraction of two matrices!
The abbreviation for \spadtype{Fraction} is \spadtype{FRAC}.

\xtc{
Use \spadopFrom{/}{Fraction} to create a fraction.
}{
\spadpaste{a := 11/12 \bound{a}}
}
\xtc{
}{
\spadpaste{b := 23/24 \bound{b}}
}
\xtc{
The standard arithmetic operations are available.
}{
\spadpaste{3 - a*b**2 + a + b/a \free{a}\free{b}}
}
\xtc{
Extract the numerator and denominator by using
\spadfunFrom{numer}{Fraction} and \spadfunFrom{denom}{Fraction},
respectively.
}{
\spadpaste{numer(a) \free{a}}
}
\xtc{
}{
\spadpaste{denom(b) \free{b}}
}
Operations like \spadfunFrom{max}{Fraction}, \spadfunFrom{min}{Fraction},
\spadfunFrom{negative?}{Fraction}, \spadfunFrom{positive?}{Fraction} and
\spadfunFrom{zero?}{Fraction} are all available if they are provided for
```

```
the numerators and denominators.
See \downlink{`Integer'}{IntegerXmpPage}\ignore{Integer} for examples.

Don't expect a useful answer from \spadfunFrom{factor}{Fraction},
\spadfunFrom{gcd}{Fraction} or \spadfunFrom{lcm}{Fraction} if you apply
them to fractions.
\xtc{
}{
\spadpaste{r := (x**2 + 2*x + 1)/(x**2 - 2*x + 1) \bound{r}}
}
\xtc{
Since all non-zero fractions are invertible, these operations have trivial
definitions.
}{
\spadpaste{factor(r) \free{r}}
}
\xtc{
Use \spadfunFrom{map}{Fraction} to apply \spadfunFrom{factor}{Fraction} to
the numerator and denominator, which is probably what you mean.
}{
\spadpaste{map(factor,r) \free{r}}
}

\xtc{
Other forms of fractions are available.
Use \spadfun{continuedFraction} to create a continued fraction.
}{
\spadpaste{continuedFraction(7/12)}
}
\xtc{
Use \spadfun{partialFraction} to create a partial fraction.
See \downlink{`ContinuedFraction'}{ContinuedFractionXmpPage}
\ignore{ContinuedFraction} and
\downlink{`PartialFraction'}{PartialFractionXmpPage}
\ignore{PartialFraction} for
additional information and examples.
}{
\spadpaste{partialFraction(7,12)}
}

\xtc{
Use conversion to create alternative views of fractions with objects
moved in and out of the numerator and denominator.
}{
\spadpaste{g := 2/3 + 4/5*\%i \bound{g}}
}
\xtc{
Conversion is discussed in detail in
\downlink{``Conversion''}{ugTypesConvertPage}
in Section 2.7\ignore{ugTypesConvert}.
```

```
}{
\spadpaste{g :: FRAC COMPLEX INT \free{g}}
}
\endscroll
\autobuttons
\end{page}

\begin{patch}{FractionXmpPagePatch1}
\begin{paste}{FractionXmpPageFull1}{FractionXmpPageEmpty1}
\pastebutton{FractionXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{a := 11/12\bound{a }}
\indentrel{3}\begin{verbatim}
        11
   (1)
        12
                                 Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FractionXmpPageEmpty1}
\begin{paste}{FractionXmpPageEmpty1}{FractionXmpPagePatch1}
\pastebutton{FractionXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{a := 11/12\bound{a }}
\end{paste}\end{patch}

\begin{patch}{FractionXmpPagePatch2}
\begin{paste}{FractionXmpPageFull2}{FractionXmpPageEmpty2}
\pastebutton{FractionXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{b := 23/24\bound{b }}
\indentrel{3}\begin{verbatim}
        23
   (2)
        24
                                 Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FractionXmpPageEmpty2}
\begin{paste}{FractionXmpPageEmpty2}{FractionXmpPagePatch2}
\pastebutton{FractionXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{b := 23/24\bound{b }}
\end{paste}\end{patch}

\begin{patch}{FractionXmpPagePatch3}
\begin{paste}{FractionXmpPageFull3}{FractionXmpPageEmpty3}
\pastebutton{FractionXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{3 - a*b**2 + a + b/a\free{a }\free{b }}
\indentrel{3}\begin{verbatim}
        313271
   (3)
```

```
         76032
                                   Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FractionXmpPageEmpty3}
\begin{paste}{FractionXmpPageEmpty3}{FractionXmpPagePatch3}
\pastebutton{FractionXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{3 - a*b**2 + a + b/a\free{a }\free{b }}
\end{paste}\end{patch}

\begin{patch}{FractionXmpPagePatch4}
\begin{paste}{FractionXmpPageFull4}{FractionXmpPageEmpty4}
\pastebutton{FractionXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{numer(a)\free{a }}
\indentrel{3}\begin{verbatim}
   (4)  11
                                    Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FractionXmpPageEmpty4}
\begin{paste}{FractionXmpPageEmpty4}{FractionXmpPagePatch4}
\pastebutton{FractionXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{numer(a)\free{a }}
\end{paste}\end{patch}

\begin{patch}{FractionXmpPagePatch5}
\begin{paste}{FractionXmpPageFull5}{FractionXmpPageEmpty5}
\pastebutton{FractionXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{denom(b)\free{b }}
\indentrel{3}\begin{verbatim}
   (5)  24
                                    Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FractionXmpPageEmpty5}
\begin{paste}{FractionXmpPageEmpty5}{FractionXmpPagePatch5}
\pastebutton{FractionXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{denom(b)\free{b }}
\end{paste}\end{patch}

\begin{patch}{FractionXmpPagePatch6}
\begin{paste}{FractionXmpPageFull6}{FractionXmpPageEmpty6}
\pastebutton{FractionXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{r := (x**2 + 2*x + 1)/(x**2 - 2*x + 1)\bound{r }}
\indentrel{3}\begin{verbatim}
         2
        x  + 2x + 1
```

```
   (6)
         2
        x  - 2x + 1
                      Type: Fraction Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FractionXmpPageEmpty6}
\begin{paste}{FractionXmpPageEmpty6}{FractionXmpPagePatch6}
\pastebutton{FractionXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{r := (x**2 + 2*x + 1)/(x**2 - 2*x + 1)\bound{r }}
\end{paste}\end{patch}

\begin{patch}{FractionXmpPagePatch7}
\begin{paste}{FractionXmpPageFull7}{FractionXmpPageEmpty7}
\pastebutton{FractionXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{factor(r)\free{r }}
\indentrel{3}\begin{verbatim}
         2
        x  + 2x + 1
   (7)
         2
        x  - 2x + 1
             Type: Factored Fraction Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FractionXmpPageEmpty7}
\begin{paste}{FractionXmpPageEmpty7}{FractionXmpPagePatch7}
\pastebutton{FractionXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{factor(r)\free{r }}
\end{paste}\end{patch}

\begin{patch}{FractionXmpPagePatch8}
\begin{paste}{FractionXmpPageFull8}{FractionXmpPageEmpty8}
\pastebutton{FractionXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{map(factor,r)\free{r }}
\indentrel{3}\begin{verbatim}
               2
        (x + 1)
   (8)
               2
        (x - 1)
             Type: Fraction Factored Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FractionXmpPageEmpty8}
\begin{paste}{FractionXmpPageEmpty8}{FractionXmpPagePatch8}
\pastebutton{FractionXmpPageEmpty8}{\showpaste}
```

```
\tab{5}\spadcommand{map(factor,r)\free{r }}
\end{paste}\end{patch}

\begin{patch}{FractionXmpPagePatch9}
\begin{paste}{FractionXmpPageFull9}{FractionXmpPageEmpty9}
\pastebutton{FractionXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{continuedFraction(7/12)}
\indentrel{3}\begin{verbatim}
          1      1      1      1
   (9)   +  +  +
         1      1      2      2
                        Type: ContinuedFraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FractionXmpPageEmpty9}
\begin{paste}{FractionXmpPageEmpty9}{FractionXmpPagePatch9}
\pastebutton{FractionXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{continuedFraction(7/12)}
\end{paste}\end{patch}

\begin{patch}{FractionXmpPagePatch10}
\begin{paste}{FractionXmpPageFull10}{FractionXmpPageEmpty10}
\pastebutton{FractionXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{partialFraction(7,12)}
\indentrel{3}\begin{verbatim}
              3   1
   (10)  1 -  +
              2   3
             2
                          Type: PartialFraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FractionXmpPageEmpty10}
\begin{paste}{FractionXmpPageEmpty10}{FractionXmpPagePatch10}
\pastebutton{FractionXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{partialFraction(7,12)}
\end{paste}\end{patch}

\begin{patch}{FractionXmpPagePatch11}
\begin{paste}{FractionXmpPageFull11}{FractionXmpPageEmpty11}
\pastebutton{FractionXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{g := 2/3 + 4/5*\%i\bound{g }}
\indentrel{3}\begin{verbatim}
         2   4
   (11)   +  %i
         3   5
                         Type: Complex Fraction Integer
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FractionXmpPageEmpty11}
\begin{paste}{FractionXmpPageEmpty11}{FractionXmpPagePatch11}
\pastebutton{FractionXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{g := 2/3 + 4/5*\%i\bound{g }}
\end{paste}\end{patch}

\begin{patch}{FractionXmpPagePatch12}
\begin{paste}{FractionXmpPageFull12}{FractionXmpPageEmpty12}
\pastebutton{FractionXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{g :: FRAC COMPLEX INT\free{g }}
\indentrel{3}\begin{verbatim}
         10 + 12%i
   (12)
             15
                         Type: Fraction Complex Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FractionXmpPageEmpty12}
\begin{paste}{FractionXmpPageEmpty12}{FractionXmpPagePatch12}
\pastebutton{FractionXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{g :: FRAC COMPLEX INT\free{g }}
\end{paste}\end{patch}
```

———

3.46 fparfrac.ht

FullPartialFracExpansion

⇒ "notitle" (PartialFractionXmpPage) 3.86 on page 1088

— **fparfrac.ht** —

```
\begin{page}{FullPartialFracExpansionXmpPage}{FullPartialFracExpansion}
\beginscroll

The domain \spadtype{FullPartialFracExpansion} implements
factor-free conversion of quotients to full partial fractions.

\xtc{
Our examples will all involve quotients of univariate polynomials
with rational number coefficients.
}{
\spadpaste{Fx := FRAC UP(x, FRAC INT) \bound{Fx}}
```

```
}
\xtc{
Here is a simple-looking rational function.
}{
\spadpaste{f : Fx := 36 / (x**5-2*x**4-2*x**3+4*x**2+x-2) \bound{f}\free{Fx}}
}
\xtc{
We use \spadfunFrom{fullPartialFraction}{FullPartialFracExpansion}
to convert it to an object of type
\spadtype{FullPartialFracExpansion}.
}{
\spadpaste{g := fullPartialFraction f \bound{g}\free{f}}
}
\xtc{
Use a coercion to change it back into a quotient.
}{
\spadpaste{g :: Fx \free{g}}
}
\xtc{
Full partial fractions differentiate faster than rational
functions.
}{
\spadpaste{g5 := D(g, 5) \free{g}\bound{g5}}
}
\xtc{
}{
\spadpaste{f5 := D(f, 5) \free{f}\bound{f5}}
}
\xtc{
We can check that the two forms represent the same function.
}{
\spadpaste{g5::Fx - f5 \free{Fx g5 f5}}
}
\xtc{
Here are some examples that are more complicated.
}{
\spadpaste{f : Fx := (x**5 * (x-1)) / ((x**2 + x + 1)**2 * (x-2)**3)
\free{Fx}\bound{f2}}
}
\xtc{
}{
\spadpaste{g := fullPartialFraction f \free{f2}\bound{g2}}
}
\xtc{
}{
\spadpaste{g :: Fx - f \free{f2 g2 Fx}}
}
\xtc{
}{
\spadpaste{f : Fx := (2*x**7-7*x**5+26*x**3+8*x) /
```

```
(x**8-5*x**6+6*x**4+4*x**2-8) \free{Fx}\bound{f3}}
}
\xtc{
}{
\spadpaste{g := fullPartialFraction f \free{f3}\bound{g3}}
}
\xtc{
}{
\spadpaste{g :: Fx - f \free{f3 g3 Fx}}
}
\xtc{
}{
\spadpaste{f:Fx := x**3 /
(x**21 + 2*x**20 + 4*x**19 + 7*x**18 + 10*x**17 + 17*x**16 +
22*x**15 + 30*x**14 + 36*x**13 + 40*x**12 + 47*x**11 + 46*x**10 +
49*x**9 + 43*x**8 + 38*x**7 + 32*x**6 + 23*x**5 + 19*x**4 +
10*x**3 + 7*x**2 + 2*x + 1)\free{Fx}\bound{f4}}
}
\xtc{
}{
\spadpaste{g := fullPartialFraction f \free{f4}\bound{g4}}
}
\xtc{
This verification takes much longer than the conversion to
partial fractions.
}{
\spadpaste{g :: Fx - f \free{f4 g4 Fx}}
}

For more information, see the paper:
Bronstein, M and Salvy, B.
``Full Partial Fraction Decomposition of Rational Functions,''
{\it Proceedings of ISSAC'93, Kiev}, ACM Press.
All see \downlink{`PartialFraction'}{PartialFractionXmpPage}
\ignore{PartialFraction} for standard partial fraction
decompositions.
\endscroll
\autobuttons
\end{page}

\begin{patch}{FullPartialFractionExpansionXmpPagePatch1}
\begin{paste}{FullPartialFractionExpansionXmpPageFull1}{FullPartialFractionExpansionXmpPageEmpty1}
\pastebutton{FullPartialFractionExpansionXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{Fx := FRAC UP(x, FRAC INT)\bound{Fx }}
\indentrel{3}\begin{verbatim}
   (1)
   Fraction UnivariatePolynomial(x,Fraction Integer)
                                        Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{FullPartialFractionExpansionXmpPageEmpty1}
\begin{paste}{FullPartialFractionExpansionXmpPageEmpty1}{FullPartialFractionExpansionXmpPagePatc
\pastebutton{FullPartialFractionExpansionXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{Fx := FRAC UP(x, FRAC INT)\bound{Fx }}
\end{paste}\end{patch}

\begin{patch}{FullPartialFractionExpansionXmpPagePatch2}
\begin{paste}{FullPartialFractionExpansionXmpPageFull2}{FullPartialFractionExpansionXmpPageEmpty
\pastebutton{FullPartialFractionExpansionXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{f : Fx := 36 / (x**5-2*x**4-2*x**3+4*x**2+x-2)\bound{f }\free{Fx }}
\indentrel{3}\begin{verbatim}
                     36
   (2)
         5     4     3     2
        x  - 2x  - 2x  + 4x  + x - 2
Type: Fraction UnivariatePolynomial(x,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FullPartialFractionExpansionXmpPageEmpty2}
\begin{paste}{FullPartialFractionExpansionXmpPageEmpty2}{FullPartialFractionExpansionXmpPagePatc
\pastebutton{FullPartialFractionExpansionXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{f : Fx := 36 / (x**5-2*x**4-2*x**3+4*x**2+x-2)\bound{f }\free{Fx }}
\end{paste}\end{patch}

\begin{patch}{FullPartialFractionExpansionXmpPagePatch3}
\begin{paste}{FullPartialFractionExpansionXmpPageFull3}{FullPartialFractionExpansionXmpPageEmpty
\pastebutton{FullPartialFractionExpansionXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{g := fullPartialFraction f\bound{g }\free{f }}
\indentrel{3}\begin{verbatim}
          4       4              - 3%A - 6
   (3)   -  +    >
        x - 2   x + 1                    2
                          2         (x - %A)
                        %A  - 1= 0
Type: FullPartialFractionExpansion(Fraction Integer,UnivariatePolynomial(x,Fraction Integer))
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FullPartialFractionExpansionXmpPageEmpty3}
\begin{paste}{FullPartialFractionExpansionXmpPageEmpty3}{FullPartialFractionExpansionXmpPagePatc
\pastebutton{FullPartialFractionExpansionXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{g := fullPartialFraction f\bound{g }\free{f }}
\end{paste}\end{patch}

\begin{patch}{FullPartialFractionExpansionXmpPagePatch4}
\begin{paste}{FullPartialFractionExpansionXmpPageFull4}{FullPartialFractionExpansionXmpPageEmpty
\pastebutton{FullPartialFractionExpansionXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{g :: Fx\free{g }}
```

```
\indentrel{3}\begin{verbatim}
                     36
   (4)
         5     4     3     2
        x  - 2x  - 2x  + 4x  + x - 2
Type: Fraction UnivariatePolynomial(x,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FullPartialFractionExpansionXmpPageEmpty4}
\begin{paste}{FullPartialFractionExpansionXmpPageEmpty4}{FullPartialFractionExpansionXmpPagePatch4}
\pastebutton{FullPartialFractionExpansionXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{g :: Fx\free{g }}
\end{paste}\end{patch}

\begin{patch}{FullPartialFractionExpansionXmpPagePatch5}
\begin{paste}{FullPartialFractionExpansionXmpPageFull5}{FullPartialFractionExpansionXmpPageEmpty5}
\pastebutton{FullPartialFractionExpansionXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{g5 := D(g, 5)\free{g }\bound{g5 }}
\indentrel{3}\begin{verbatim}
   (5)
        480        480             2160%A + 4320
   -  +  +    >
           6          6                     7
     (x - 2)    (x + 1)      2           (x - %A)
                           %A  - 1= 0
Type: FullPartialFractionExpansion(Fraction Integer,UnivariatePolynomial(x,Fraction Integer))
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FullPartialFractionExpansionXmpPageEmpty5}
\begin{paste}{FullPartialFractionExpansionXmpPageEmpty5}{FullPartialFractionExpansionXmpPagePatch5}
\pastebutton{FullPartialFractionExpansionXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{g5 := D(g, 5)\free{g }\bound{g5 }}
\end{paste}\end{patch}

\begin{patch}{FullPartialFractionExpansionXmpPagePatch6}
\begin{paste}{FullPartialFractionExpansionXmpPageFull6}{FullPartialFractionExpansionXmpPageEmpty6}
\pastebutton{FullPartialFractionExpansionXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{f5 := D(f, 5)\free{f }\bound{f5 }}
\indentrel{3}\begin{verbatim}
   (6)
                10           9            8            7
       - 544320x   + 4354560x  - 14696640x  + 28615680x
     +
                 6            5            4            3
       - 40085280x  + 46656000x  - 39411360x  + 18247680x
     +
                2
       - 5870880x  + 3317760x + 246240
```

```
  /
       20      19      18      17       16       15
      x   - 12x   + 53x   - 76x   - 159x   + 676x
    +
           14        13        12        11        10
      - 391x   - 1596x   + 2527x   + 1148x   - 4977x
    +
          9        8        7        6        5       4
      1372x  + 4907x  - 3444x  - 2381x  + 2924x  + 276x
    +
            3       2
      - 1184x  + 208x  + 192x - 64
Type: Fraction UnivariatePolynomial(x,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FullPartialFractionExpansionXmpPageEmpty6}
\begin{paste}{FullPartialFractionExpansionXmpPageEmpty6}{FullPartialFractionExpansionXmpPagePatc
\pastebutton{FullPartialFractionExpansionXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{f5 := D(f, 5)\free{f }\bound{f5 }}
\end{paste}\end{patch}

\begin{patch}{FullPartialFractionExpansionXmpPagePatch7}
\begin{paste}{FullPartialFractionExpansionXmpPageFull7}{FullPartialFractionExpansionXmpPageEmpty
\pastebutton{FullPartialFractionExpansionXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{g5::Fx - f5\free{Fx g5 f5 }}
\indentrel{3}\begin{verbatim}
   (7)  0
Type: Fraction UnivariatePolynomial(x,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FullPartialFractionExpansionXmpPageEmpty7}
\begin{paste}{FullPartialFractionExpansionXmpPageEmpty7}{FullPartialFractionExpansionXmpPagePatc
\pastebutton{FullPartialFractionExpansionXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{g5::Fx - f5\free{Fx g5 f5 }}
\end{paste}\end{patch}

\begin{patch}{FullPartialFractionExpansionXmpPagePatch8}
\begin{paste}{FullPartialFractionExpansionXmpPageFull8}{FullPartialFractionExpansionXmpPageEmpty
\pastebutton{FullPartialFractionExpansionXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{f : Fx := (x**5 * (x-1)) / ((x**2 + x + 1)**2 * (x-2)**3)\free{Fx }\bound{f2
\indentrel{3}\begin{verbatim}
                       6    5
                      x  - x
   (8)
         7     6     5     3     2
        x  - 4x  + 3x  + 9x  - 6x  - 4x - 8
Type: Fraction UnivariatePolynomial(x,Fraction Integer)
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FullPartialFractionExpansionXmpPageEmpty8}
\begin{paste}{FullPartialFractionExpansionXmpPageEmpty8}{FullPartialFractionExpansionXmpPagePatch8}
\pastebutton{FullPartialFractionExpansionXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{f : Fx := (x**5 * (x-1)) / ((x**2 + x + 1)**2 * (x-2)**3)\free{Fx }\bound{f2 }}
\end{paste}\end{patch}

\begin{patch}{FullPartialFractionExpansionXmpPagePatch9}
\begin{paste}{FullPartialFractionExpansionXmpPageFull9}{FullPartialFractionExpansionXmpPageEmpty9}
\pastebutton{FullPartialFractionExpansionXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{g := fullPartialFraction f\free{f2 }\bound{g2 }}
\indentrel{3}\begin{verbatim}
   (9)
      1952       464        32

      2401       343        49
      +  +
      x - 2          2          3
              (x - 2)    (x - 2)
   +
                          179       135
                       -  %A +
                      2401      2401
           >
                         x - %A
       2
     %A  + %A + 1= 0
   +
                        37        20
                        %A +
                    1029      1029
           >
                                2
       2                  (x - %A)
     %A  + %A + 1= 0
Type: FullPartialFractionExpansion(Fraction Integer,UnivariatePolynomial(x,Fraction Integer))
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FullPartialFractionExpansionXmpPageEmpty9}
\begin{paste}{FullPartialFractionExpansionXmpPageEmpty9}{FullPartialFractionExpansionXmpPagePatch9}
\pastebutton{FullPartialFractionExpansionXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{g := fullPartialFraction f\free{f2 }\bound{g2 }}
\end{paste}\end{patch}

\begin{patch}{FullPartialFractionExpansionXmpPagePatch10}
\begin{paste}{FullPartialFractionExpansionXmpPageFull10}{FullPartialFractionExpansionXmpPageEmpty10}
\pastebutton{FullPartialFractionExpansionXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{g :: Fx - f\free{f2 g2 Fx }}
```

```
\indentrel{3}\begin{verbatim}
   (10)  0
Type: Fraction UnivariatePolynomial(x,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FullPartialFractionExpansionXmpPageEmpty10}
\begin{paste}{FullPartialFractionExpansionXmpPageEmpty10}{FullPartialFractionExpansionXmpPagePat
\pastebutton{FullPartialFractionExpansionXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{g :: Fx - f\free{f2 g2 Fx }}
\end{paste}\end{patch}

\begin{patch}{FullPartialFractionExpansionXmpPagePatch11}
\begin{paste}{FullPartialFractionExpansionXmpPageFull11}{FullPartialFractionExpansionXmpPageEmpt
\pastebutton{FullPartialFractionExpansionXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{f : Fx := (2*x**7-7*x**5+26*x**3+8*x) / (x**8-5*x**6+6*x**4+4*x**2-8)\free{F
\indentrel{3}\begin{verbatim}
             7     5      3
           2x  - 7x  + 26x  + 8x
   (11)
          8     6     4     2
         x  - 5x  + 6x  + 4x  - 8
Type: Fraction UnivariatePolynomial(x,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FullPartialFractionExpansionXmpPageEmpty11}
\begin{paste}{FullPartialFractionExpansionXmpPageEmpty11}{FullPartialFractionExpansionXmpPagePat
\pastebutton{FullPartialFractionExpansionXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{f : Fx := (2*x**7-7*x**5+26*x**3+8*x) / (x**8-5*x**6+6*x**4+4*x**2-8)\free{F
\end{paste}\end{patch}

\begin{patch}{FullPartialFractionExpansionXmpPagePatch12}
\begin{paste}{FullPartialFractionExpansionXmpPageFull12}{FullPartialFractionExpansionXmpPageEmpt
\pastebutton{FullPartialFractionExpansionXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{g := fullPartialFraction f\free{f3 }\bound{g3 }}
\indentrel{3}\begin{verbatim}
   (12)
                    1

                 2                  1
        >         +    >
              x - %A                      3
      2                    2         (x - %A)
    %A  - 2= 0           %A  - 2= 0
   +
                    1

                 2
        >
```

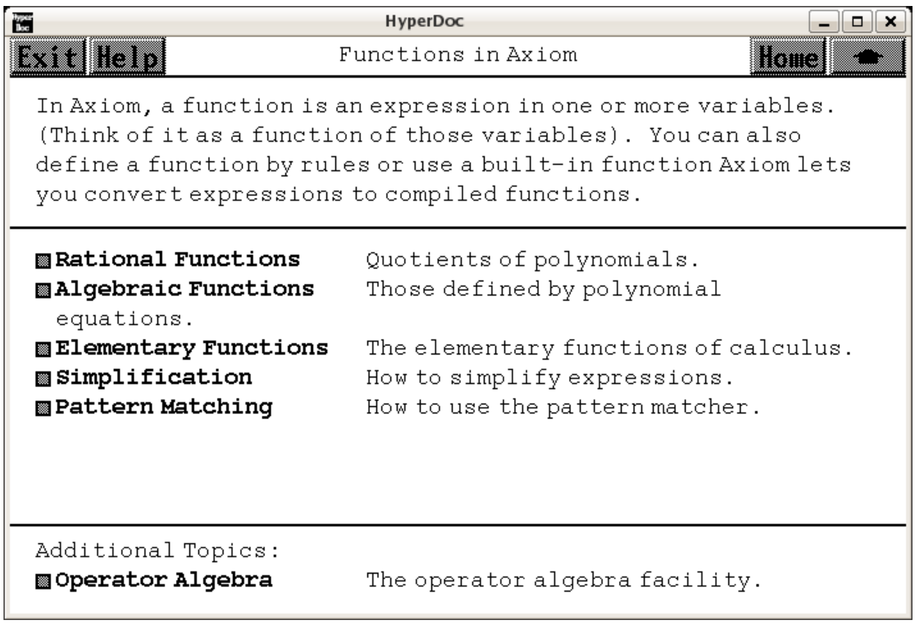
```
             x - %A
       2
     %A  + 1= 0
Type: FullPartialFractionExpansion(Fraction Integer,UnivariatePolynomial(x,Fraction Integer))
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FullPartialFractionExpansionXmpPageEmpty12}
\begin{paste}{FullPartialFractionExpansionXmpPageEmpty12}{FullPartialFractionExpansionXmpPagePatch12}
\pastebutton{FullPartialFractionExpansionXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{g := fullPartialFraction f\free{f3 }\bound{g3 }}
\end{paste}\end{patch}

\begin{patch}{FullPartialFractionExpansionXmpPagePatch13}
\begin{paste}{FullPartialFractionExpansionXmpPageFull13}{FullPartialFractionExpansionXmpPageEmpty13}
\pastebutton{FullPartialFractionExpansionXmpPageFull13}{\hidepaste}
\tab{5}\spadcommand{g :: Fx - f\free{f3 g3 Fx }}
\indentrel{3}\begin{verbatim}
   (13)  0
Type: Fraction UnivariatePolynomial(x,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FullPartialFractionExpansionXmpPageEmpty13}
\begin{paste}{FullPartialFractionExpansionXmpPageEmpty13}{FullPartialFractionExpansionXmpPagePatch13}
\pastebutton{FullPartialFractionExpansionXmpPageEmpty13}{\showpaste}
\tab{5}\spadcommand{g :: Fx - f\free{f3 g3 Fx }}
\end{paste}\end{patch}

\begin{patch}{FullPartialFractionExpansionXmpPagePatch14}
\begin{paste}{FullPartialFractionExpansionXmpPageFull14}{FullPartialFractionExpansionXmpPageEmpty14}
\pastebutton{FullPartialFractionExpansionXmpPageFull14}{\hidepaste}
\tab{5}\spadcommand{f:Fx := x**3 / (x**21 + 2*x**20 + 4*x**19 + 7*x**18 + 10*x**17 + 17*x**16 + 22*x**15 + 3
\indentrel{3}\begin{verbatim}
   (14)
      3
     x
  /
       21     20     19     18      17      16      15
      x   + 2x   + 4x   + 7x   + 10x   + 17x   + 22x
     +
         14      13      12      11      10      9      8
      30x   + 36x   + 40x   + 47x   + 46x   + 49x  + 43x
     +
         7      6      5      4      3     2
      38x  + 32x  + 23x  + 19x  + 10x  + 7x  + 2x + 1
Type: Fraction UnivariatePolynomial(x,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{FullPartialFractionExpansionXmpPageEmpty14}
\begin{paste}{FullPartialFractionExpansionXmpPageEmpty14}{FullPartialFractionExpansionXmpPagePat
\pastebutton{FullPartialFractionExpansionXmpPageEmpty14}{\showpaste}
\tab{5}\spadcommand{f:Fx := x**3 / (x**21 + 2*x**20 + 4*x**19 + 7*x**18 + 10*x**17 + 17*x**16 +
\end{paste}\end{patch}

\begin{patch}{FullPartialFractionExpansionXmpPagePatch15}
\begin{paste}{FullPartialFractionExpansionXmpPageFull15}{FullPartialFractionExpansionXmpPageEmpt
\pastebutton{FullPartialFractionExpansionXmpPageFull15}{\hidepaste}
\tab{5}\spadcommand{g := fullPartialFraction f\free{f4 }\bound{g4 }}
\indentrel{3}\begin{verbatim}
   (15)
                  1                         1      19
                   %A                       %A -
               2                      9      27
        >         +       >
              x - %A                   x - %A
      2                    2
    %A  + 1= 0           %A  + %A + 1= 0
  +
                       1       1
                       %A -
                   27      27
           >
                               2
      2                (x - %A)
    %A  + %A + 1= 0
  +
    SIGMA
         5     2
       %A  + %A  + 1= 0
   ,
               96556567040   4   420961732891   3
            -  %A  +  %A
              912390759099       912390759099
         +
               59101056149   2   373545875923
            -  %A  -  %A
              912390759099       912390759099
         +
            529673492498

            912390759099
      /
         x - %A
  +
    SIGMA
         5     2
       %A  + %A  + 1= 0
   ,
```

```
               5580868   4    2024443   3    4321919   2
           -  %A  -  %A  +  %A
             94070601       94070601       94070601
         +
               84614        5070620
           -  %A -
             1542141      94070601
      /
                    2
         (x - %A)
   +
     SIGMA
          5     2
       %A  + %A  + 1= 0
    ,
             1610957   4    2763014   3    2016775   2
             %A  +  %A  -  %A
           94070601       94070601       94070601
         +
             266953        4529359
             %A +
           94070601      94070601
      /
                    3
         (x - %A)
Type: FullPartialFractionExpansion(Fraction Integer,UnivariatePolynomial(x,Fraction Integer))
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FullPartialFractionExpansionXmpPageEmpty15}
\begin{paste}{FullPartialFractionExpansionXmpPageEmpty15}{FullPartialFractionExpansionXmpPagePatch15}
\pastebutton{FullPartialFractionExpansionXmpPageEmpty15}{\showpaste}
\tab{5}\spadcommand{g := fullPartialFraction f\free{f4 }\bound{g4 }}
\end{paste}\end{patch}

\begin{patch}{FullPartialFractionExpansionXmpPagePatch16}
\begin{paste}{FullPartialFractionExpansionXmpPageFull16}{FullPartialFractionExpansionXmpPageEmpty16}
\pastebutton{FullPartialFractionExpansionXmpPageFull16}{\hidepaste}
\tab{5}\spadcommand{g :: Fx - f\free{f4 g4 Fx }}
\indentrel{3}\begin{verbatim}
   (16)  0
Type: Fraction UnivariatePolynomial(x,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FullPartialFractionExpansionXmpPageEmpty16}
\begin{paste}{FullPartialFractionExpansionXmpPageEmpty16}{FullPartialFractionExpansionXmpPagePatch16}
\pastebutton{FullPartialFractionExpansionXmpPageEmpty16}{\showpaste}
\tab{5}\spadcommand{g :: Fx - f\free{f4 g4 Fx }}
\end{paste}\end{patch}
```

3.47 function.ht

Functions in Axiom



⇐ “Topics” (TopicPage) 3.108 on page 1313
⇒ “Rational Functions” (RationalFunctionPage) 3.47 on page 561
⇒ “Algebraic Functions” (AlgebraicFunctionPage) 3.47 on page 564
⇒ “Elementary Functions” (ElementaryFunctionPage) 3.47 on page 567
⇒ “Simplification” (FunctionSimplificationPage) 3.47 on page 568
⇒ “Pattern Matching” (ugUserRulesPage) 10 on page 1949
⇒ “Operator Algebra” (OperatorXmpPage) 3.83 on page 1071

— **function.ht** —

```
\begin{page}{FunctionPage}{Functions in Axiom}
%
In Axiom, a function is an expression in one or more variables.
(Think of it as a function of those variables).
You can also define a function by rules or use a built-in function
Axiom lets you convert expressions to compiled functions.
\beginscroll
\beginmenu
```

```
\menulink{Rational Functions}{RationalFunctionPage} \tab{22}
Quotients of polynomials.

\menulink{Algebraic Functions}{AlgebraicFunctionPage} \tab{22}
Those defined by polynomial equations.

\menulink{Elementary Functions}{ElementaryFunctionPage} \tab{22}
The elementary functions of calculus.

\menulink{Simplification}{FunctionSimplificationPage} \tab{22}
How to simplify expressions.

\menulink{Pattern Matching}{ugUserRulesPage} \tab{22}
How to use the pattern matcher.
\endmenu
\endscroll
\newline
Additional Topics:
\beginmenu

\menulink{Operator Algebra}{OperatorXmpPage}\tab{22}
The operator algebra facility.

\endmenu
\autobuttons \end{page}
```

———-

Rational Functions

— function.ht —

```
\begin{page}{RationalFunctionPage}{Rational Functions}
\beginscroll
To create a rational function, just compute the
quotient of two polynomials:
\spadpaste{f := (x - y) / (x + y)\bound{f}}
Use the functions \spadfun{numer} and \spadfun{denom}:
to recover the numerator and denominator of a fraction:
%
\spadpaste{numer f\free{f}}
\spadpaste{denom f\free{f}}
%
Since these are polynomials, you can apply all the
\downlink{polynomial operations}{PolynomialPage}
to them.
You can substitute values or
```

```
other rational functions for the variables using
the function \spadfun{eval}. The syntax for \spadfun{eval} is
similar to the one for polynomials:
\spadpaste{eval(f, x = 1/x)\free{f}}
\spadpaste{eval(f, [x = y, y = x])\free{f}}
\endscroll
\autobuttons
\end{page}

\begin{patch}{RationatFunctionPagePatch1}
\begin{paste}{RationatFunctionPageFull1}{RationatFunctionPageEmpty1}
\pastebutton{RationatFunctionPageFull1}{\hidepaste}
\tab{5}\spadcommand{f := (x - y) / (x + y)\bound{f }}
\indentrel{3}\begin{verbatim}
        - y + x
   (1)
         y + x
                      Type: Fraction Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RationatFunctionPageEmpty1}
\begin{paste}{RationatFunctionPageEmpty1}{RationatFunctionPagePatch1}
\pastebutton{RationatFunctionPageEmpty1}{\showpaste}
\tab{5}\spadcommand{f := (x - y) / (x + y)\bound{f }}
\end{paste}\end{patch}

\begin{patch}{RationatFunctionPagePatch2}
\begin{paste}{RationatFunctionPageFull2}{RationatFunctionPageEmpty2}
\pastebutton{RationatFunctionPageFull2}{\hidepaste}
\tab{5}\spadcommand{numer f\free{f }}
\indentrel{3}\begin{verbatim}
   (2)  - y + x
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RationatFunctionPageEmpty2}
\begin{paste}{RationatFunctionPageEmpty2}{RationatFunctionPagePatch2}
\pastebutton{RationatFunctionPageEmpty2}{\showpaste}
\tab{5}\spadcommand{numer f\free{f }}
\end{paste}\end{patch}

\begin{patch}{RationatFunctionPagePatch3}
\begin{paste}{RationatFunctionPageFull3}{RationatFunctionPageEmpty3}
\pastebutton{RationatFunctionPageFull3}{\hidepaste}
\tab{5}\spadcommand{denom f\free{f }}
\indentrel{3}\begin{verbatim}
   (3)  y + x
                               Type: Polynomial Integer
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RationatFunctionPageEmpty3}
\begin{paste}{RationatFunctionPageEmpty3}{RationatFunctionPagePatch3}
\pastebutton{RationatFunctionPageEmpty3}{\showpaste}
\tab{5}\spadcommand{denom f\free{f }}
\end{paste}\end{patch}

\begin{patch}{RationatFunctionPagePatch4}
\begin{paste}{RationatFunctionPageFull4}{RationatFunctionPageEmpty4}
\pastebutton{RationatFunctionPageFull4}{\hidepaste}
\tab{5}\spadcommand{eval(f, x = 1/x)\free{f }}
\indentrel{3}\begin{verbatim}
        - x y + 1
   (4)
         x y + 1
                      Type: Fraction Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RationatFunctionPageEmpty4}
\begin{paste}{RationatFunctionPageEmpty4}{RationatFunctionPagePatch4}
\pastebutton{RationatFunctionPageEmpty4}{\showpaste}
\tab{5}\spadcommand{eval(f, x = 1/x)\free{f }}
\end{paste}\end{patch}

\begin{patch}{RationatFunctionPagePatch5}
\begin{paste}{RationatFunctionPageFull5}{RationatFunctionPageEmpty5}
\pastebutton{RationatFunctionPageFull5}{\hidepaste}
\tab{5}\spadcommand{eval(f, [x = y, y = x])\free{f }}
\indentrel{3}\begin{verbatim}
        y - x
   (5)
        y + x
                      Type: Fraction Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RationatFunctionPageEmpty5}
\begin{paste}{RationatFunctionPageEmpty5}{RationatFunctionPagePatch5}
\pastebutton{RationatFunctionPageEmpty5}{\showpaste}
\tab{5}\spadcommand{eval(f, [x = y, y = x])\free{f }}
\end{paste}\end{patch}
```

Algebraic Functions

— function.ht —

```
\begin{page}{AlgebraicFunctionPage}{Algebraic Functions}
\beginscroll
Algebraic functions are functions defined by algebraic equations. There
are two ways of constructing them: using rational powers, or implicitly.
For rational powers, use \spadopFrom{**}{RadicalCategory}
(or the system functions \spadfun{sqrt} and
\spadfun{nthRoot} for square and nth roots):
\spadpaste{f := sqrt(1 + x ** (1/3))\bound{f}}
To define an algebraic function implicitly
use \spadfun{rootOf}. The following
line defines a function \spad{y} of \spad{x} satisfying the equation
\spad{y**3 = x*y - y**2 - x**3 + 1}:
\spadpaste{y := rootOf(y**3 + y**2 - x*y + x**3 - 1, y)\bound{y}}
You can manipulate, differentiate or integrate an implicitly defined
algebraic function like any other Axiom function:
\spadpaste{differentiate(y, x)\free{y}}
Higher powers of algebraic functions are automatically reduced during
calculations:
\spadpaste{(y + 1) ** 3\free{y}}
But denominators, are not automatically rationalized:
\spadpaste{g := inv f\bound{g}\free{y}}
Use \spadfun{ratDenom} to remove the algebraic quantities
from the denominator:
\spadpaste{ratDenom g\free{g}}
\endscroll
\autobuttons
\end{page}

\begin{patch}{AlgebraicFunctionPagePatch1}
\begin{paste}{AlgebraicFunctionPageFull1}{AlgebraicFunctionPageEmpty1}
\pastebutton{AlgebraicFunctionPageFull1}{\hidepaste}
\tab{5}\spadcommand{f := sqrt(1 + x ** (1/3))\bound{f }}
\indentrel{3}\begin{verbatim}

         3
   (1)  \\x  + 1
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{AlgebraicFunctionPageEmpty1}
\begin{paste}{AlgebraicFunctionPageEmpty1}{AlgebraicFunctionPagePatch1}
\pastebutton{AlgebraicFunctionPageEmpty1}{\showpaste}
\tab{5}\spadcommand{f := sqrt(1 + x ** (1/3))\bound{f }}
\end{paste}\end{patch}
```

```
\begin{patch}{AlgebraicFunctionPagePatch2}
\begin{paste}{AlgebraicFunctionPageFull2}{AlgebraicFunctionPageEmpty2}
\pastebutton{AlgebraicFunctionPageFull2}{\hidepaste}
\tab{5}\spadcommand{y := rootOf(y**3 + y**2 - x*y + x**3 - 1, y)\bound{y }}
\indentrel{3}\begin{verbatim}
   (2)  y
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{AlgebraicFunctionPageEmpty2}
\begin{paste}{AlgebraicFunctionPageEmpty2}{AlgebraicFunctionPagePatch2}
\pastebutton{AlgebraicFunctionPageEmpty2}{\showpaste}
\tab{5}\spadcommand{y := rootOf(y**3 + y**2 - x*y + x**3 - 1, y)\bound{y }}
\end{paste}\end{patch}

\begin{patch}{AlgebraicFunctionPagePatch3}
\begin{paste}{AlgebraicFunctionPageFull3}{AlgebraicFunctionPageEmpty3}
\pastebutton{AlgebraicFunctionPageFull3}{\hidepaste}
\tab{5}\spadcommand{differentiate(y, x)\free{y }}
\indentrel{3}\begin{verbatim}
                2
           y - 3x
   (3)
          2
        3y  + 2y - x
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{AlgebraicFunctionPageEmpty3}
\begin{paste}{AlgebraicFunctionPageEmpty3}{AlgebraicFunctionPagePatch3}
\pastebutton{AlgebraicFunctionPageEmpty3}{\showpaste}
\tab{5}\spadcommand{differentiate(y, x)\free{y }}
\end{paste}\end{patch}

\begin{patch}{AlgebraicFunctionPagePatch4}
\begin{paste}{AlgebraicFunctionPageFull4}{AlgebraicFunctionPageEmpty4}
\pastebutton{AlgebraicFunctionPageFull4}{\hidepaste}
\tab{5}\spadcommand{(y + 1) ** 3\free{y }}
\indentrel{3}\begin{verbatim}
          2               3
   (4)  2y  + (x + 3)y - x  + 2
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{AlgebraicFunctionPageEmpty4}
\begin{paste}{AlgebraicFunctionPageEmpty4}{AlgebraicFunctionPagePatch4}
```

```
\pastebutton{AlgebraicFunctionPageEmpty4}{\showpaste}
\tab{5}\spadcommand{(y + 1) ** 3\free{y }}
\end{paste}\end{patch}

\begin{patch}{AlgebraicFunctionPagePatch5}
\begin{paste}{AlgebraicFunctionPageFull5}{AlgebraicFunctionPageEmpty5}
\pastebutton{AlgebraicFunctionPageFull5}{\hidepaste}
\tab{5}\spadcommand{g := inv f\bound{g }\free{y }}
\indentrel{3}\begin{verbatim}
             1
   (5)


         3
        \\x  + 1
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{AlgebraicFunctionPageEmpty5}
\begin{paste}{AlgebraicFunctionPageEmpty5}{AlgebraicFunctionPagePatch5}
\pastebutton{AlgebraicFunctionPageEmpty5}{\showpaste}
\tab{5}\spadcommand{g := inv f\bound{g }\free{y }}
\end{paste}\end{patch}

\begin{patch}{AlgebraicFunctionPagePatch6}
\begin{paste}{AlgebraicFunctionPageFull6}{AlgebraicFunctionPageEmpty6}
\pastebutton{AlgebraicFunctionPageFull6}{\hidepaste}
\tab{5}\spadcommand{ratDenom g\free{g }}
\indentrel{3}\begin{verbatim}

         32   3      3
        (\x   - \x  + 1)\\x  + 1
   (6)
                    x + 1
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{AlgebraicFunctionPageEmpty6}
\begin{paste}{AlgebraicFunctionPageEmpty6}{AlgebraicFunctionPagePatch6}
\pastebutton{AlgebraicFunctionPageEmpty6}{\showpaste}
\tab{5}\spadcommand{ratDenom g\free{g }}
\end{paste}\end{patch}
```

Elementary Functions

— function.ht —

```
\begin{page}{ElementaryFunctionPage}{Elementary Functions}
\beginscroll
Axiom has most of the usual functions from calculus built-in:
\spadpaste{f := x * log y * sin(1/(x+y))\bound{f}}
You can substitute values or another elementary functions for
the variables with the function \spadfun{eval}:
\spadpaste{eval(f, [x = y, y = x])\free{f}}
As you can see, the substitutions are made 'in parallel' as in the
case of polynomials. It's also possible to substitute expressions
for kernels instead of variables:
\spadpaste{eval(f, log y = acosh(x + sqrt y))\free{f}}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ElementaryFunctionPagePatch1}
\begin{paste}{ElementaryFunctionPageFull1}{ElementaryFunctionPageEmpty1}
\pastebutton{ElementaryFunctionPageFull1}{\hidepaste}
\tab{5}\spadcommand{f := x * log y * sin(1/(x+y))\bound{f }}
\indentrel{3}\begin{verbatim}
                      1
   (1)  x log(y)sin()
                    y + x
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ElementaryFunctionPageEmpty1}
\begin{paste}{ElementaryFunctionPageEmpty1}{ElementaryFunctionPagePatch1}
\pastebutton{ElementaryFunctionPageEmpty1}{\showpaste}
\tab{5}\spadcommand{f := x * log y * sin(1/(x+y))\bound{f }}
\end{paste}\end{patch}

\begin{patch}{ElementaryFunctionPagePatch2}
\begin{paste}{ElementaryFunctionPageFull2}{ElementaryFunctionPageEmpty2}
\pastebutton{ElementaryFunctionPageFull2}{\hidepaste}
\tab{5}\spadcommand{eval(f, [x = y, y = x])\free{f }}
\indentrel{3}\begin{verbatim}
                      1
   (2)  y log(x)sin()
                    y + x
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ElementaryFunctionPageEmpty2}
\begin{paste}{ElementaryFunctionPageEmpty2}{ElementaryFunctionPagePatch2}
\pastebutton{ElementaryFunctionPageEmpty2}{\showpaste}
\tab{5}\spadcommand{eval(f, [x = y, y = x])\free{f }}
\end{paste}\end{patch}

\begin{patch}{ElementaryFunctionPagePatch3}
\begin{paste}{ElementaryFunctionPageFull3}{ElementaryFunctionPageEmpty3}
\pastebutton{ElementaryFunctionPageFull3}{\hidepaste}
\tab{5}\spadcommand{eval(f, log y = acosh(x + sqrt y))\free{f }}
\indentrel{3}\begin{verbatim}
               1
   (3)  x sin()acosh(\y  + x)
             y + x
                              Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ElementaryFunctionPageEmpty3}
\begin{paste}{ElementaryFunctionPageEmpty3}{ElementaryFunctionPagePatch3}
\pastebutton{ElementaryFunctionPageEmpty3}{\showpaste}
\tab{5}\spadcommand{eval(f, log y = acosh(x + sqrt y))\free{f }}
\end{paste}\end{patch}
```

———-

Simplification

⇒ “notitle” (ugUserRulesPage) 10 on page 1949

— **function.ht** —

```
\begin{page}{FunctionSimplificationPage}{Simplification}
\beginscroll
Simplifying an expression often means different things at
different times, so Axiom offers a large number of
‘simplification’ functions.
The most common one, which performs the usual trigonometric
simplifications is \spadfun{simplify}:
\spadpaste{f := cos(x)/sec(x) * log(sin(x)**2/(cos(x)**2+sin(x)**2))
\bound{f}}
\spadpaste{g := simplify f\bound{g}\free{f}}
If the result of \spadfun{simplify} is not satisfactory, specific
transformations are available.
For example, to rewrite \spad{g} in terms of secants and
cosecants instead of sines and cosines, issue:
%
\spadpaste{h := sin2csc cos2sec g\bound{h}\free{g}}
```

```
%
To apply the logarithm simplification rules to \spad{h}, issue:
\spadpaste{expandLog h\free{h}}
Since the square root of \spad{x**2} is the absolute value of
\spad{x} and not \spad{x} itself, algebraic radicals are not
automatically simplified, but you can specifically request it by
calling \spadfun{rootSimp}:
%
\spadpaste{f1 := sqrt((x+1)**3)\bound{f1}}
\spadpaste{rootSimp f1\free{f1}}
%
There are other transformations which are sometimes useful.
Use the functions \spadfun{complexElementary} and \spadfun{trigs}
to go back and forth between the complex exponential and
trigonometric forms of an elementary function:
%
\spadpaste{g1 := sin(x + cos x)\bound{g1}}
\spadpaste{g2 := complexElementary g1\bound{g2}\free{g1}}
\spadpaste{trigs g2\free{g2}}
%
Similarly, the functions \spadfun{realElementary} and
\spadfun{htrigs} convert hyperbolic functions in and out of their
exponential form:
%
\spadpaste{h1 := sinh(x + cosh x)\bound{h1}}
\spadpaste{h2 := realElementary h1\bound{h2}\free{h1}}
\spadpaste{htrigs h2\free{h2}}
%
Axiom has other transformations, most of which
are in the packages
\spadtype{ElementaryFunctionStructurePackage},
\spadtype{TrigonometricManipulations},
\spadtype{AlgebraicManipulations},
and \spadtype{TranscendentalManipulations}.
If you need to apply a simplification rule not built into the
system, you can use Axiom's \downlink{pattern
matcher}{ugUserRulesPage}.
\endscroll
\autobuttons
\end{page}

\begin{patch}{FunctionSimplificationPagePatch1}
\begin{paste}{FunctionSimplificationPageFull1}{FunctionSimplificationPageEmpty1}
\pastebutton{FunctionSimplificationPageFull1}{\hidepaste}
\tab{5}\spadcommand{f := cos(x)/sec(x) * log(sin(x)**2/(cos(x)**2+sin(x)**2))\bound{f }}
\indentrel{3}\begin{verbatim}
                            2
                      sin(x)
        cos(x)log()
                       2         2
```

```
                  sin(x)  + cos(x)
   (1)
                   sec(x)
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FunctionSimplificationPageEmpty1}
\begin{paste}{FunctionSimplificationPageEmpty1}{FunctionSimplificationPagePatch1}
\pastebutton{FunctionSimplificationPageEmpty1}{\showpaste}
\tab{5}\spadcommand{f := cos(x)/sec(x) * log(sin(x)**2/(cos(x)**2+sin(x)**2))\bound{f }}
\end{paste}\end{patch}

\begin{patch}{FunctionSimplificationPagePatch2}
\begin{paste}{FunctionSimplificationPageFull2}{FunctionSimplificationPageEmpty2}
\pastebutton{FunctionSimplificationPageFull2}{\hidepaste}
\tab{5}\spadcommand{g := simplify f\bound{g }\free{f }}
\indentrel{3}\begin{verbatim}
              2            2
   (2)  cos(x) log(- cos(x)  + 1)
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FunctionSimplificationPageEmpty2}
\begin{paste}{FunctionSimplificationPageEmpty2}{FunctionSimplificationPagePatch2}
\pastebutton{FunctionSimplificationPageEmpty2}{\showpaste}
\tab{5}\spadcommand{g := simplify f\bound{g }\free{f }}
\end{paste}\end{patch}

\begin{patch}{FunctionSimplificationPagePatch3}
\begin{paste}{FunctionSimplificationPageFull3}{FunctionSimplificationPageEmpty3}
\pastebutton{FunctionSimplificationPageFull3}{\hidepaste}
\tab{5}\spadcommand{h := sin2csc cos2sec g\bound{h }\free{g }}
\indentrel{3}\begin{verbatim}
                  2
            sec(x)  - 1
        log()
                    2
             sec(x)
   (3)
                   2
             sec(x)
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FunctionSimplificationPageEmpty3}
\begin{paste}{FunctionSimplificationPageEmpty3}{FunctionSimplificationPagePatch3}
\pastebutton{FunctionSimplificationPageEmpty3}{\showpaste}
```

```
\tab{5}\spadcommand{h := sin2csc cos2sec g\bound{h }\free{g }}
\end{paste}\end{patch}

\begin{patch}{FunctionSimplificationPagePatch4}
\begin{paste}{FunctionSimplificationPageFull4}{FunctionSimplificationPageEmpty4}
\pastebutton{FunctionSimplificationPageFull4}{\hidepaste}
\tab{5}\spadcommand{expandLog h\free{h }}
\indentrel{3}\begin{verbatim}
                  2
        log(sec(x)  - 1) - 2log(sec(x))
   (4)
                          2
                    sec(x)
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FunctionSimplificationPageEmpty4}
\begin{paste}{FunctionSimplificationPageEmpty4}{FunctionSimplificationPagePatch4}
\pastebutton{FunctionSimplificationPageEmpty4}{\showpaste}
\tab{5}\spadcommand{expandLog h\free{h }}
\end{paste}\end{patch}

\begin{patch}{FunctionSimplificationPagePatch5}
\begin{paste}{FunctionSimplificationPageFull5}{FunctionSimplificationPageEmpty5}
\pastebutton{FunctionSimplificationPageFull5}{\hidepaste}
\tab{5}\spadcommand{f1 := sqrt((x+1)**3)\bound{f1 }}
\indentrel{3}\begin{verbatim}

          3     2
   (5)  \x  + 3x  + 3x + 1
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FunctionSimplificationPageEmpty5}
\begin{paste}{FunctionSimplificationPageEmpty5}{FunctionSimplificationPagePatch5}
\pastebutton{FunctionSimplificationPageEmpty5}{\showpaste}
\tab{5}\spadcommand{f1 := sqrt((x+1)**3)\bound{f1 }}
\end{paste}\end{patch}

\begin{patch}{FunctionSimplificationPagePatch6}
\begin{paste}{FunctionSimplificationPageFull6}{FunctionSimplificationPageEmpty6}
\pastebutton{FunctionSimplificationPageFull6}{\hidepaste}
\tab{5}\spadcommand{rootSimp f1\free{f1 }}
\indentrel{3}\begin{verbatim}

   (6)  (x + 1)\x + 1
                               Type: Expression Integer
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FunctionSimplificationPageEmpty6}
\begin{paste}{FunctionSimplificationPageEmpty6}{FunctionSimplificationPagePatch6}
\pastebutton{FunctionSimplificationPageEmpty6}{\showpaste}
\tab{5}\spadcommand{rootSimp f1\free{f1 }}
\end{paste}\end{patch}

\begin{patch}{FunctionSimplificationPagePatch7}
\begin{paste}{FunctionSimplificationPageFull7}{FunctionSimplificationPageEmpty7}
\pastebutton{FunctionSimplificationPageFull7}{\hidepaste}
\tab{5}\spadcommand{g1 := sin(x + cos x)\bound{g1 }}
\indentrel{3}\begin{verbatim}
   (7)  sin(cos(x) + x)
                                Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FunctionSimplificationPageEmpty7}
\begin{paste}{FunctionSimplificationPageEmpty7}{FunctionSimplificationPagePatch7}
\pastebutton{FunctionSimplificationPageEmpty7}{\showpaste}
\tab{5}\spadcommand{g1 := sin(x + cos x)\bound{g1 }}
\end{paste}\end{patch}

\begin{patch}{FunctionSimplificationPagePatch8}
\begin{paste}{FunctionSimplificationPageFull8}{FunctionSimplificationPageEmpty8}
\pastebutton{FunctionSimplificationPageFull8}{\hidepaste}
\tab{5}\spadcommand{g2 := complexElementary g1\bound{g2 }\free{g1 }}
\indentrel{3}\begin{verbatim}
   (8)
       -

            \- 1
         *
                %e
             **
                                  2
                        x\- 1            x\- 1
                    \- 1 (%e       )  + 2x\- 1 %e
                  +

                    \- 1
               /

                     x\- 1
                  2%e
           **
              2
     +
```

```
       \- 1
  /
                   2
           x\- 1          x\- 1
        \- 1 (%e       )  + 2x\- 1 %e        + \- 1


                              x\- 1
                           2%e
     2%e
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FunctionSimplificationPageEmpty8}
\begin{paste}{FunctionSimplificationPageEmpty8}{FunctionSimplificationPagePatch8}
\pastebutton{FunctionSimplificationPageEmpty8}{\showpaste}
\tab{5}\spadcommand{g2 := complexElementary g1\bound{g2 }\free{g1 }}
\end{paste}\end{patch}

\begin{patch}{FunctionSimplificationPagePatch9}
\begin{paste}{FunctionSimplificationPageFull9}{FunctionSimplificationPageEmpty9}
\pastebutton{FunctionSimplificationPageFull9}{\hidepaste}
\tab{5}\spadcommand{trigs g2\free{g2 }}
\indentrel{3}\begin{verbatim}
   (9)  sin(cos(x) + x)
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FunctionSimplificationPageEmpty9}
\begin{paste}{FunctionSimplificationPageEmpty9}{FunctionSimplificationPagePatch9}
\pastebutton{FunctionSimplificationPageEmpty9}{\showpaste}
\tab{5}\spadcommand{trigs g2\free{g2 }}
\end{paste}\end{patch}

\begin{patch}{FunctionSimplificationPagePatch10}
\begin{paste}{FunctionSimplificationPageFull10}{FunctionSimplificationPageEmpty10}
\pastebutton{FunctionSimplificationPageFull10}{\hidepaste}
\tab{5}\spadcommand{h1 := sinh(x + cosh x)\bound{h1 }}
\indentrel{3}\begin{verbatim}
   (10)  sinh(cosh(x) + x)
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FunctionSimplificationPageEmpty10}
\begin{paste}{FunctionSimplificationPageEmpty10}{FunctionSimplificationPagePatch10}
\pastebutton{FunctionSimplificationPageEmpty10}{\showpaste}
\tab{5}\spadcommand{h1 := sinh(x + cosh x)\bound{h1 }}
```

```
\end{paste}\end{patch}

\begin{patch}{FunctionSimplificationPagePatch11}
\begin{paste}{FunctionSimplificationPageFull11}{FunctionSimplificationPageEmpty11}
\pastebutton{FunctionSimplificationPageFull11}{\hidepaste}
\tab{5}\spadcommand{h2 := realElementary h1\bound{h2 }\free{h1 }}
\indentrel{3}\begin{verbatim}
              x 2        x     2
           (%e )  + 2x %e  + 1

                     x
                  2%e
        (%e                  )  - 1
   (11)
                 x 2        x
              (%e )  + 2x %e  + 1

                        x
                     2%e
           2%e
                              Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FunctionSimplificationPageEmpty11}
\begin{paste}{FunctionSimplificationPageEmpty11}{FunctionSimplificationPagePatch11}
\pastebutton{FunctionSimplificationPageEmpty11}{\showpaste}
\tab{5}\spadcommand{h2 := realElementary h1\bound{h2 }\free{h1 }}
\end{paste}\end{patch}

\begin{patch}{FunctionSimplificationPagePatch12}
\begin{paste}{FunctionSimplificationPageFull12}{FunctionSimplificationPageEmpty12}
\pastebutton{FunctionSimplificationPageFull12}{\hidepaste}
\tab{5}\spadcommand{htrigs h2\free{h2 }}
\indentrel{3}\begin{verbatim}
   (12)  sinh(cosh(x) + x)
                              Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FunctionSimplificationPageEmpty12}
\begin{paste}{FunctionSimplificationPageEmpty12}{FunctionSimplificationPagePatch12}
\pastebutton{FunctionSimplificationPageEmpty12}{\showpaste}
\tab{5}\spadcommand{htrigs h2\free{h2 }}
\end{paste}
\end{patch}
```

3.48 gbf.ht

GroebnerFactorizationPkg

— gbf.ht —

```
\begin{page}{GroebnerFactorizationPkgXmpPage}
{GroebnerFactorizationPkg}
\beginscroll
%%  J. Grabmeier 26 04 91
Solving systems of polynomial equations with the
\texht{Gr\"{o}bner}{Groebner}
basis algorithm can often be very time consuming because, in general,
the algorithm has exponential run-time.
These systems, which often come from concrete applications,
frequently have symmetries which are not taken advantage of by the
algorithm.
However, it often happens in this case
that the polynomials which occur during the
\texht{Gr\"{o}bner}{Groebner} calculations are reducible.
Since Axiom has an excellent polynomial factorization algorithm, it is
very natural to combine the
\texht{Gr\"{o}bner}{Groebner} and factorization algorithms.

\spadtype{GroebnerFactorizationPkg} exports the
\axiomFunFrom{groebnerFactorize}{GroebnerFactorizationPkg}
operation which implements a modified
\texht{Gr\"{o}bner}{Groebner} basis algorithm.
In this algorithm, each polynomial that is to be put into the
partial list of the basis is first factored.
The remaining calculation is split into as many parts as there are
irreducible factors.
Call these factors \texht{$p_1, \ldots,p_n.$}{\spad{p1, ..., pn.}}
In the branches corresponding to \texht{$p_2,
\ldots,p_n,$}{\spad{p2, ..., pn,}} the factor
\texht{$p_1$}{\spad{p1}} can be divided out, and so on.
This package also contains operations that allow you to specify the
polynomials that are not zero on the common roots of the final
\texht{Gr\"{o}bner}{Groebner} basis.

Here is an example from chemistry.
In a theoretical model of the cyclohexan
\texht{${\rm C}_6{\rm H}_{12}$}{C6H12},
the six carbon atoms each sit in the center of gravity of a
tetrahedron that has two hydrogen atoms and two carbon atoms at its
corners.
We first normalize and set the length of each edge to 1.
Hence, the distances of one fixed carbon atom to each of its
```

```
immediate neighbours is 1.
We will denote the distances to the other three carbon atoms by
\smath{x}, \smath{y} and \smath{z}.

%% reference?
\texht{A.~Dress}{A. Dress} developed a theory to decide whether a set
of points and distances between them can be realized in an
\smath{n}-dimensional space. Here, of course, we have \smath{n = 3}.
\xtc{
}{
\spadpaste{mfzn : SQMATRIX(6,DMP([x,y,z],Fraction INT)) :=
[[0,1,1,1,1,1], [1,0,1,8/3,x,8/3], [1,1,0,1,8/3,y], [1,8/3,1,0,1,8/3],
[1,x,8/3,1,0,1], [1,8/3,y,8/3,1,0]] \bound{mfzn}}
}
\xtc{
For the cyclohexan, the distances have to satisfy this equation.
}{
\spadpaste{eq := determinant mfzn \free{mfzn}\bound{eq}}
}
\xtc{
They also must satisfy the equations
given by cyclic shifts of the indeterminates.
}{
\spadpaste{groebnerFactorize [eq, eval(eq, [x,y,z], [y,z,x]),
eval(eq, [x,y,z], [z,x,y])] \free{eq}}
}

The union of the solutions of this list is the
solution of our original problem.
If we impose positivity conditions, we get two relevant ideals.
One ideal is
zero-dimensional, namely \smath{x = y = z = 11/3}, and this
determines the ‘‘boat’’ form of the cyclohexan.
The other ideal is one-dimensional,
which means that we have a solution space given by one parameter.
This gives the ‘‘chair’’ form of the cyclohexan.
The parameter describes the angle of the ‘‘back of the chair.’’

\axiomFunFrom{groebnerFactorize}{GroebnerFactorizationPkg}
has an optional \axiomType{Boolean}-valued second argument.
When it is \spad{true} partial results are displayed,
since it may happen that the
calculation does not terminate in a reasonable time.
See the source code for \spadtype{GroebnerFactorizationPkg}
in {\bf groebf\spadFileExt{}} for more details about the algorithms
used.
\endscroll
\autobuttons
\end{page}
```

```
\begin{patch}{GroebnerFactorizationPackageXmpPagePatch1}
\begin{paste}{GroebnerFactorizationPackageXmpPageFull1}{GroebnerFactorizationPackageXmpPageEmpty1}
\pastebutton{GroebnerFactorizationPackageXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{mfzn : SQMATRIX(6,DMP([x,y,z],Fraction INT)) := [[0,1,1,1,1,1], [1,0,1,8/3,x,8/3], [1,1
\indentrel{3}\begin{verbatim}
        0  1  1  1  1  1

                 8     8
        1  0  1    x
                 3     3

                    8
        1  1  0  1    y
                    3

   (1)     8           8
        1    1  0  1
           3           3

              8
        1  x    1  0  1
              3

           8     8
        1    y    1  0
           3     3
Type: SquareMatrix(6,DistributedMultivariatePolynomial([x,y,z],Fraction Integer))
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{GroebnerFactorizationPackageXmpPageEmpty1}
\begin{paste}{GroebnerFactorizationPackageXmpPageEmpty1}{GroebnerFactorizationPackageXmpPagePatch1}
\pastebutton{GroebnerFactorizationPackageXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{mfzn : SQMATRIX(6,DMP([x,y,z],Fraction INT)) := [[0,1,1,1,1,1], [1,0,1,8/3,x,8/3], [1,1
\end{paste}\end{patch}

\begin{patch}{GroebnerFactorizationPackageXmpPagePatch2}
\begin{paste}{GroebnerFactorizationPackageXmpPageFull2}{GroebnerFactorizationPackageXmpPageEmpty2}
\pastebutton{GroebnerFactorizationPackageXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{eq := determinant mfzn\free{mfzn }\bound{eq }}
\indentrel{3}\begin{verbatim}
   (2)
        2 2   22  2    25  2   22    2   388       250
     - x y  +  x y -  x  +  x y  -  x y -  x
               3        9       3         9         27
   +
       25  2   250     14575
     -  y  -  y +
        9       27       81
Type: DistributedMultivariatePolynomial([x,y,z],Fraction Integer)
```

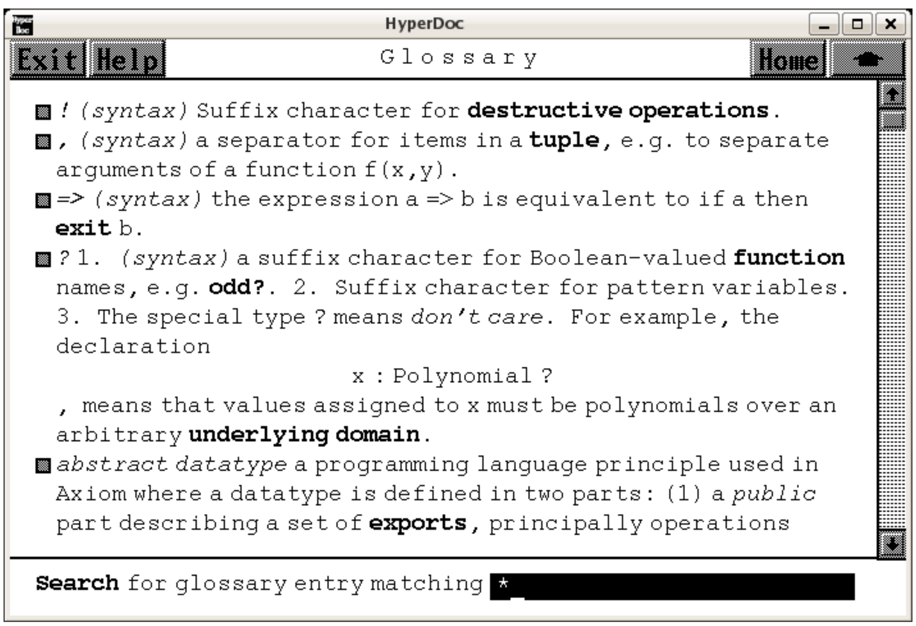
```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{GroebnerFactorizationPackageXmpPageEmpty2}
\begin{paste}{GroebnerFactorizationPackageXmpPageEmpty2}{GroebnerFactorizationPackageXmpPagePatc
\pastebutton{GroebnerFactorizationPackageXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{eq := determinant mfzn\free{mfzn }\bound{eq }}
\end{paste}\end{patch}

\begin{patch}{GroebnerFactorizationPackageXmpPagePatch3}
\begin{paste}{GroebnerFactorizationPackageXmpPageFull3}{GroebnerFactorizationPackageXmpPageEmpty
\pastebutton{GroebnerFactorizationPackageXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{groebnerFactorize [eq, eval(eq, [x,y,z], [y,z,x]), eval(eq, [x,y,z], [z,x,y]
\indentrel{3}\begin{verbatim}
   (3)
   [
                22           22     22     121
     [x y + x z -  x + y z -  y -  z + ,
                 3            3      3      3

           2   22       25        2   22       25
        x z  -  x z +  x + y z  -  y z +  y
                3       9            3        9
      +
          22  2   388     250
        -  z  +  z +
           3       9       27
      ,

         2 2   22  2    25  2   22    2   388       250
        y z  -  y z +  y  -  y z  +  y z +  y
                3       9      3        9         27
      +
        25  2   250     14575
         z  +  z -
        9       27       81
      ]
     ,
           21994  2   21994     4427     463
    [x + y -  ,y  -  y +  ,z -  ],
            5625       5625      675      87
     2   1       11     5     265        2   38     265
    [x  -  x z -  x -  z +  ,y - z,z  -  z +  ],
         2        2     6      18             3      9
         25     11     11         11     11     11
    [x -  ,y -  ,z -  ], [x -  ,y -  ,z -  ],
         9      3      3          3      3      3
         5     5     5         19     5     5
    [x +  ,y +  ,z +  ], [x -  ,y +  ,z +  ]]
         3     3     3          3     3     3
```

```
Type: List List DistributedMultivariatePolynomial([x,y,z],Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{GroebnerFactorizationPackageXmpPageEmpty3}
\begin{paste}{GroebnerFactorizationPackageXmpPageEmpty3}{GroebnerFactorizationPackageXmpPagePatch3}
\pastebutton{GroebnerFactorizationPackageXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{groebnerFactorize [eq, eval(eq, [x,y,z], [y,z,x]), eval(eq, [x,y,z], [z,x,y])]\free{eq
\end{paste}\end{patch}
```

———-

3.49 gloss.ht

Glossary



⇐ "Reference" (TopReferencePage) 3.1 on page 123
⇒ "Search" (LispFunctions) 3.71 on page 952

— **gloss.ht** —

```
\begin{page}{GlossaryPage}{G l o s s a r y}
\beginscroll\beginmenu\item\newline{\em \menuitemstyle{}}{\em !}\space{}
{\em (syntax)} Suffix character for \spadgloss{destructive operations}.
\item\newline{\em \menuitemstyle{}}{\em ,}\space{}
```

```
{\em (syntax)} a separator for items in a \spadgloss{tuple},{}
\spadignore{e.g.} to separate arguments of a function \spad{f(x,{}y)}.
\item\newline{\em \menuitemstyle{}}{\em =>}\space{}
{\em (syntax)} the expression \spad{a => b} is equivalent to
\spad{if a then} \spadgloss{exit} \spad{b}.
\item\newline{\em \menuitemstyle{}}{\em ?}\space{}
1. {\em (syntax)} a suffix character for Boolean-valued \spadfun{function}
names,{} \spadignore{e.g.} \spadfun{odd?}. 2. Suffix character for pattern
variables. 3. The special type \spad{?} means {\em don't care}. For
example,{} the declaration \newline\center{\spad{x : Polynomial ?}}\newline,
{} means that values assigned to \spad{x} must be polynomials over an
arbitrary \spadgloss{underlying domain}.
\item\newline{\em \menuitemstyle{}}{\em abstract datatype}\space{}
a programming language principle used in Axiom where a datatype
is defined in two parts: (1) a {\em public} part describing a set of
\spadgloss{exports},{} principally operations that apply to objects of
that type,{} and (2) a {\em private} part describing the implementation
of the datatype usually in terms of a \spadgloss{representation} for
objects of the type. Programs which create and otherwise manipulate
objects of the type may only do so through its exports. The representation
and other implementation information is specifically hidden.
\item\newline{\em \menuitemstyle{}}{\em abstraction}\space{}
described functionally or conceptually without regard to implementation
\item\newline{\em \menuitemstyle{}}{\em accuracy}\space{}
the degree of exactness of an approximation or measurement. In computer
algebra systems,{} computations are typically carried out with complete
accuracy using integers or rational numbers of indefinite size. Domain
\spadtype{Float} provides a function \spadfunFrom{precision}{Float} to
change the precision for floating point computations. Computations using
\spadtype{DoubleFloat} have a fixed precision but uncertain accuracy.
\item\newline{\em \menuitemstyle{}}{\em add-chain}\space{}
a hierarchy formed by \spadgloss{domain extensions}. If domain \spad{A}
extends domain \spad{B} and domain \spad{B} extends domain \spad{C},{}
then \spad{A} has {\em add-chain} \spad{B}-\spad{C}.
\item\newline{\em \menuitemstyle{}}{\em aggregate}\space{}
a data structure designed to hold multiple values. Examples of aggregates
are \spadtype{List},{} \spadtype{Set},{} \spadtype{Matrix} and \spadtype{Bits}.
\item\newline{\em \menuitemstyle{}}{\em AKCL}\space{}
Austin Kyoto Common LISP,{} a version of \spadgloss{\spad{KCL}} produced
by William Schelter,{} Austin,{} Texas.
\item\newline{\em \menuitemstyle{}}{\em algorithm}\space{}
a step-by-step procedure for a solution of a problem; a program
\item\newline{\em \menuitemstyle{}}{\em ancestor}\space{}
(of a domain) a category which is a \spadgloss{parent} of the domain,{}
or a \spadgloss{parent} of a \spadgloss{parent},{} and so on.
\item\newline{\em \menuitemstyle{}}{\em application}\space{}
{\em (syntax)} an expression denoting "application" of a function to a set
of \spadgloss{argument} parameters. Applications are written as a
\spadgloss{parameterized form}. For example,{} the form \spad{f(x,{}y)}
indicates the "application of the function \spad{f} to the tuple of
```

```
arguments \spad{x} and \spad{y}". See also \spadgloss{evaluation} and
\spadgloss{invocation}.
\item\newline{\em \menuitemstyle{}}{\em apply}\space{}
See \spadgloss{application}.
\item\newline{\em \menuitemstyle{}}{\em argument}\space{}
1. (actual argument) a value passed to a function at the time of a
\spadglossSee{function call}{application}; also called an {\em actual
parameter}. 2. (formal argument) a variable used in the definition of
a function to denote the actual argument passed when the function is called.
\item\newline{\em \menuitemstyle{}}{\em arity}\space{}
1. (function) the number of arguments. 2. (operator or operation)
corresponds to the arity of a function implementing the operator or operation.
\item\newline{\em \menuitemstyle{}}{\em assignment}\space{}
{\em (syntax)} an expression of the form \spad{x := e},{} meaning
"assign the value of \spad{e} to \spad{x"}. After \spadgloss{evaluation},{}
the \spadgloss{variable} \spad{x} \spadglossSee{points}{pointer} to an
object obtained by evaluating the expression \spad{e}. If \spad{x} has
a \spadgloss{type} as a result of a previous \spadgloss{declaration},{}
the object assigned to \spad{x} must have that type. An interpreter must
often \spadglossSee{coerce}{coercion} the value of \spad{e} to make that
happen. For example,{} in the interpreter,{} \center{\spad{x : Float := 11}}
first \spadglossSee{declares}{declaration} \spad{x} to be a float. This
declaration causes the interpreter to coerce 11 to 11.0 in order to assign
a floating point value to \spad{x}.
\item\newline{\em \menuitemstyle{}}{\em attribute}\space{}
a name or functional form denoting {\em any} useful computational property.
For example,{} \spadatt{commutative(\spad{"*"})} asserts that "\spadfun{*}
is commutative". Also,{} \spadatt{finiteAggregate} is used to assert that
an aggregate has a finite number of immediate components.
\item\newline{\em \menuitemstyle{}}{\em basis}\space{}
{\em (algebra)} \spad{S} is a basis of a module \spad{M} over a
\spadgloss{ring} if \spad{S} generates \spad{M},{} and \spad{S} is
linearly independent
\item\newline{\em \menuitemstyle{}}{\em benefactor}\space{}
(of a given domain) a domain or package that the given domain explicitly
references (for example,{} calls functions from) in its implementation
\item\newline{\em \menuitemstyle{}}{\em binary}\space{}
operation or function with \spadgloss{arity} 2
\item\newline{\em \menuitemstyle{}}{\em binding}\space{}
the association of a variable with properties such as \spadgloss{value}
and \spadgloss{type}. The top-level \spadgloss{environment} in the
interpreter consists of bindings for all user variables and functions.
Every \spadgloss{function} has an associated set of bindings,{} one for
each formal \spadgloss{argument} and \spadgloss{local variable}.
\item\newline{\em \menuitemstyle{}}{\em block}\space{}
{\em (syntax)} a control structure where expressions are sequentially
\spadglossSee{evaluated}{evaluation}.
\item\newline{\em \menuitemstyle{}}{\em body}\space{}
a \spadgloss{function body} or \spadgloss{loop body}.
\item\newline{\em \menuitemstyle{}}{\em boolean}\space{}
```

```
objects denoted by the \spadgloss{literals} \spad{true} and \spad{false};
elements of domain \spadtype{Boolean}. See also \spadtype{Bits}.
\item\newline{\em \menuitemstyle{}}{\em built-in function}\space{}
a \spadgloss{function} in the standard Axiom library. Contrast
\spadgloss{user function}.
\item\newline{\em \menuitemstyle{}}{\em cache}\space{}
1. (noun) a mechanism for immediate retrieval of previously computed data.
For example,{} a function which does a lengthy computation might store its
values in a \spadgloss{hash table} using argument as a key. The hash table
then serves as a cache for the function (see also \spadsyscom{)set function
cache}). Also,{} when \spadgloss{recurrence relations} which depend upon
\spad{n} previous values are compiled,{} the previous \spad{n} values are
normally cached (use \spadsyscom{)set functions recurrence} to change this).
2. (verb) to save values in a cache.
\item\newline{\em \menuitemstyle{}}{\em capsule}\space{}
the part of the \spadglossSee{body}{function body} of a
\spadgloss{domain constructor} that defines the functions implemented by
the constructor.
\item\newline{\em \menuitemstyle{}}{\em case}\space{}
{\em (syntax)} an operator used to conditionally evaluate code based on
the branch of a \spadgloss{Union}. For example,{} if value \spad{u} is
\spad{Union(Integer,{}"failed")},{} the conditional expression \spad{if u
case Integer then A else B} evaluate \spad{A} if \spad{u} is an integer
and \spad{B} otherwise.
\item\newline{\em \menuitemstyle{}}{\em Category}\space{}
the distinguished object denoting the type of a category; the class of
all categories.
\item\newline{\em \menuitemstyle{}}{\em category}\space{}
{\em (basic concept)} second-order types which serve to define useful
"classification worlds" for domains,{} such as algebraic constructs
(\spadignore{e.g.} groups,{} rings,{} fields),{} and data structures
(\spadignore{e.g.} homogeneous aggregates,{} collections,{} dictionaries).
Examples of categories are \spadtype{Ring} ("the class of all rings") and
\spadtype{Aggregate} ("the class of all aggregates"). The categories of a
given world are arranged in a hierarchy (formally,{} a directed acyclic
graph). Each category inherits the properties of all its ancestors. Thus,{}
for example,{} the category of ordered rings (\spadtype{OrderedRing})
inherits the properties of the category of rings (\spadtype{Ring}) and
those of the ordered sets (\spadtype{OrderedSet}). Categories provide a
database of algebraic knowledge and ensure mathematical correctness,{}
\spadignore{e.g.} that "matrices of polynomials" is correct but "polynomials
of hash tables" is not,{} that the multiply operation for "polynomials of
continued fractions" is commutative,{} but that for "matrices of power
series" is not. optionally provide "default definitions" for operations
they export. Categories are defined in Axiom by functions called
\spadgloss{category constructors}. Technically,{} a category designates
a class of domains with common \spadgloss{operations} and
\spadgloss{attributes} but usually with different \spadgloss{functions}
and \spadgloss{representations} for its constituent \spadgloss{objects}.
Categories are always defined using the Axiom library language
```

```
(see also \spadgloss{category extension}). See also file {\em catdef.spad}
for definitions of basic algebraic categories in Axiom.
\item\newline{\em \menuitemstyle{}}{\em category constructor}\space{}
a function that creates categories,{} described by an abstract datatype in
the Axiom programming language. For example,{} the category
constructor \spadtype{Module} is a function which takes a domain parameter
\spad{R} and creates the category "modules over \spad{R}".
\item\newline{\em \menuitemstyle{}}{\em category extension}\space{}
created by a category definition,{} an expression usually of the form
\spad{A == B with ...}. In English,{} this means "category A is a \spad{B}
with the new operations and attributes as given by ... . See,{} for
example,{} file {\em catdef.spad} for a definitions of the algebra
categories in Axiom,{} {\em aggcat.spad} for data structure categories.
\item\newline{\em \menuitemstyle{}}{\em category hierarchy}\space{}
hierarchy formed by category extensions. The root category is
\spadtype{Object}. A category can be defined as a \spadgloss{Join} of two
or more categories so as to have multiple \spadgloss{parents}. Categories
may also have parameterized so as to allow conditional inheritance.
\item\newline{\em \menuitemstyle{}}{\em character}\space{}
1. an element of a character set,{} as represented by a keyboard key. 2.
a component of a string. For example,{} the 0th element of the string
\spad{"hello there"} is the character {\em h}.
\item\newline{\em \menuitemstyle{}}{\em client}\space{}
(of a given domain) any domain or package that explicitly calls functions
from the given domain
\item\newline{\em \menuitemstyle{}}{\em coercion}\space{}
an automatic transformation of an object of one \spadgloss{type} to an
object of a similar or desired target type. In the interpreter,{} coercions
and \spadgloss{retractions} are done automatically by the interpreter when
a type mismatch occurs. Compare \spadgloss{conversion}.
\item\newline{\em \menuitemstyle{}}{\em comment}\space{}
textual remarks imbedded in code. Comments are preceded by a double dash
({\em --}). For Axiom library code,{} stylized comments for on-line
documentation are preceded by a two plus signs ({\em ++}).
\item\newline{\em \menuitemstyle{}}{\em Common LISP}\space{}
A version of \spadgloss{LISP} adopted as an informal standard by major
users and suppliers of LISP
\item\newline{\em \menuitemstyle{}}{\em compile-time}\space{}
the time when category or domain constructors are compiled. Contrast
\spadgloss{run-time}.
\item\newline{\em \menuitemstyle{}}{\em compiler}\space{}
a program that generates low-level code from a higher-level source
language. Axiom has three compilers. A {\em graphics compiler}
converts graphical formulas to a compiled subroutine so that points can
be rapidly produced for graphics commands. An {\em interpreter compiler}
optionally compiles \spadgloss{user functions} when first
\spadglossSee{invoked}{invocation} (use \spadsyscom{)set functions compile}
to turn this feature on). A {\em library compiler} compiles all constructors
(not available in Release 1)
\item\newline{\em \menuitemstyle{}}{\em computational object}\space{}
```

```
In Axiom,{} domains are objects. This term is used to distinquish
the objects which are members of domains rather than domains themselves.
\item\newline{\em \menuitemstyle{}}{\em conditional}\space{}
a \spadgloss{control structure} of the form \spad{if A then B else C};
The \spadgloss{evaluation} of \spad{A} produces \spad{true} or \spad{false}.
If \spad{true},{} \spad{B} evaluates to produce a value; otherwise \spad{C}
evaluates to produce a value. When the value is not used,{} \spad{else C}
part can be omitted.
\item\newline{\em \menuitemstyle{}}{\em constant}\space{}
{\em (syntax)} a reserved word used in \spadgloss{signatures} in Axiom
programming language to signify that mark an operation always returns the
same value. For example,{} the signature \spad{0: constant -> \$} in the
source code of \spadtype{AbelianMonoid} tells the Axiom compiler that
\spad{0} is a constant so that suitable optimizations might be performed.
\item\newline{\em \menuitemstyle{}}{\em constructor}\space{}
a \spadgloss{function} which creates a \spadgloss{category},{}
\spadgloss{domain},{} or \spadgloss{package}.
\item\newline{\em \menuitemstyle{}}{\em continuation}\space{}
when a line of a program is so long that it must be broken into several
lines,{} then all but the first line are called {\em continuation lines}.
If such a line is given interactively,{} then each incomplete line must
end with an underscore.
\item\newline{\em \menuitemstyle{}}{\em control structure}\space{}
program structures which can specify a departure from normal sequential
execution. Axiom has four kinds of control structures:
\spadgloss{blocks},{} \spadgloss{case} statements,{}
\spadgloss{conditionals},{} and \spadgloss{loops}.
\item\newline{\em \menuitemstyle{}}{\em conversion}\space{}
the transformation of an object on one \spadgloss{type} to one of another
type. Conversions performed automatically are called \spadgloss{coercions}.
These happen when the interpreter has a type mismatch and a similar or
declared target type is needed. In general,{} the user must use the
infix operation \spadSyntax{\spad{::}} to cause this transformation.
\item\newline{\em \menuitemstyle{}}{\em copying semantics}\space{}
the programming language semantics used in Pascal but {\em not} in
Axiom. See also \spadgloss{pointer semantics} for details.
\item\newline{\em \menuitemstyle{}}{\em data structure}\space{}
a structure for storing data in the computer. Examples are
\spadgloss{lists} and \spadgloss{hash tables}.
\item\newline{\em \menuitemstyle{}}{\em datatype}\space{}
equivalent to \spadgloss{domain} in Axiom.
\item\newline{\em \menuitemstyle{}}{\em declaration}\space{}
{\em (syntax)} an expression of the form \spad{x : T} where \spad{T} is
some \spad{type}. A declaration forces all values
\spadglossSee{assigned}{assignment} to \spad{T} to be of that type. If a
value is of a different type,{} the interpreter will try to
\spadglossSee{coerce}{coercion} the value to type \spad{T}. Declarations
are necessary in case of ambiguity or when a user wants to introduce an
an \spadglossSee{unexposed}{expose} domain.
\item\newline{\em \menuitemstyle{}}{\em default definition}\space{}
```

```
a function defined by a \spadgloss{category}. Such definitions appear
category definitions of the form \spad{C: Category == T add I} in an
optional implmentation part \spad{I} to the right of the keyword \spad{add}.
\item\newline{\em \menuitemstyle{}}{\em default package}\space{}
a optional \spadgloss{package} of \spadgloss{functions} associated with
a category. Such functions are necessarily defined in terms over other
functions exported by the category.
\item\newline{\em \menuitemstyle{}}{\em definition}\space{}
{\em (syntax)} 1. An expression of the form \spad{f(a) == b} defining
function \spad{f} with \spadgloss{formal arguments} \spad{a} and
\spadgloss{body} \spad{b}; equivalent to the statement
\spad{f == (a) +-> b}. 2. An expression of the form \spad{a == b}
where \spad{a} is a \spadgloss{symbol},{} equivalent to \spad{a() == b}.
See also \spadgloss{macro} where a similar substitution is done at
\spadgloss{parse} time.
\item\newline{\em \menuitemstyle{}}{\em delimiter}\space{}
a \spadgloss{character} which marks the beginning or end of some
syntactically correct unit in the language,{} \spadignore{e.g.}
\spadSyntax{"} for strings,{} blanks for identifiers.
\item\newline{\em \menuitemstyle{}}{\em destructive operation}\space{}
An operation which changes a component or structure of a value. In
Axiom,{} all destructive operations have names which end with an
exclamation mark ({\em !}). For example,{} domain \spadtype{List} has
two operations to reverse the elements of a list,{} one named
\spadfunFrom{reverse}{List} which returns a copy of the original list with
the elements reversed,{} another named \spadfunFrom{reverse!}{List} which
reverses the elements {\em in place} thus destructively changing the
original list.
\item\newline{\em \menuitemstyle{}}{\em documentation}\space{}
1. on-line or hard copy descriptions of Axiom; 2. text in library
code preceded by {\em ++} comments as opposed to general comments
preceded by {\em --}.
\item\newline{\em \menuitemstyle{}}{\em domain}\space{}
{\em (basic concept)} a domain corresponds to the usual notion of
abstract datatypes: that of a set of values and a set of "exported
operations" for the creation and manipulation of these values. Datatypes
are parameterized,{} dynamically constructed,{} and can combine with others
in any meaningful way,{} \spadignore{e.g.} "lists of floats"
(\spadtype{List Float}),{} "fractions of polynomials with integer
coefficients" (\spadtype{Fraction Polynomial Integer}),{} "matrices
of infinite \spadgloss{streams} of cardinal numbers"
(\spadtype{Matrix Stream CardinalNumber}). The term {\em domain} is
actually abbreviates {\em domain of computation}. Technically,{} a
domain denotes a class of objects,{} a class of \spadgloss{operations}
for creating and other manipulating these objects,{} and a class of
\spadgloss{attributes} describing computationally useful properties.
Domains also provide \spadgloss{functions} for each operation often in
terms of some \spadgloss{representation} for the objects. A domain
itself is an \spadgloss{object} created by a \spadgloss{function}
called a \spadgloss{domain constructor}.
```

```
\item\newline{\em \menuitemstyle{}}{\em domain constructor}\space{}
a function that creates domains,{} described by an abstract datatype
in the Axiom programming language. Simple domains like
\spadtype{Integer} and \spadtype{Boolean} are created by domain
constructors with no arguments. Most domain constructors take one or
more parameters,{} one usually denoting an \spadgloss{underlying domain}.
For example,{} the domain \spadtype{Matrix(R)} denotes "matrices over
\spad{R"}. Domains {\em Mapping},{} {\em Record},{} and {\em Union} are
primitive domains. All other domains are written in the Axiom
programming language and can be modified by users with access to the
library source code.
\item\newline{\em \menuitemstyle{}}{\em domain extension}\space{}
a domain constructor \spad{A} is said to {\em extend} a domain
constructor \spad{B} if \spad{A}\spad{'s} definition has the form
\spad{A == B add ...}. This intuitively means "functions not defined
by \spad{A} are assumed to come from \spad{B}". Successive domain
extensions form \spadgloss{add-chains} affecting the the
\spadglossSee{search order}{lineage} for functions not implemented
directly by the domain during \spadgloss{dynamic lookup}.
\item\newline{\em \menuitemstyle{}}{\em dot notation}\space{}
using an infix dot ({\em .}) for function application. If \spad{u} is
the list \spad{[7,{}4,{}-11]} then both \spad{u(2)} and \spad{u.2} return
4. Dot notation nests to left. Thus \spad{f . g . h} is equivalent to
\spad{(f . g) . h}.
\item\newline{\em \menuitemstyle{}}{\em dynamic}\space{}
that which is done at \spadgloss{run-time} as opposed to
\spadgloss{compile-time}. For example,{} the interpreter will build the
domain "matrices over integers" dynamically in response to user input.
However,{} the compilation of all functions for matrices and integers is
done during \spadgloss{compile-time}. Constrast \spadgloss{static}.
\item\newline{\em \menuitemstyle{}}{\em dynamic lookup}\space{}
In Axiom,{} a \spadgloss{domain} may or may not explicitly provide
\spadgloss{function} definitions for all of its exported
\spadgloss{operations}. These definitions may instead come from domains
in the \spadgloss{add-chain} or from \spadgloss{default packages}. When
a \spadglossSee{function call}{application} is made for an operation in
the domain,{} up to five steps are carried out.\begin{items} \item (1)
If the domain itself implements a function for the operation,{} that
function is returned. \item (2) Each of the domains in the
\spadgloss{add-chain} are searched for one which implements the function;
if found,{} the function is returned. \item (3) Each of the
\spadgloss{default packages} for the domain are searched in order of
the \spadgloss{lineage}. If any of the default packages implements the
function,{} the first one found is returned. \item (4) Each of the
\spadgloss{default packages} for each of the domains in the
\spadgloss{add-chain} are searched in the order of their
\spadgloss{lineage}. If any of the default packages implements the
function,{} the first one found is returned. \item (5) If all of the
above steps fail,{} an error message is reported. \end{items}
\item\newline{\em \menuitemstyle{}}{\em empty}\space{}
```

```
the unique value of objects with type \spadtype{Void}.
\item\newline{\em \menuitemstyle{}}{\em environment}\space{}
a set of \spadgloss{bindings}.
\item\newline{\em \menuitemstyle{}}{\em evaluation}\space{}
a systematic process which transforms an \spadgloss{expression} into an
object called the \spadgloss{value} of the expression. Evaluation may
produce \spadgloss{side effects}.
\item\newline{\em \menuitemstyle{}}{\em exit}\space{}
{\em (reserved word)} an \spadgloss{operator} which forces an exit from
the current block. For example,{} the \spadgloss{block} \spad{(a := 1;
if i > 0 then exit a; a := 2)} will prematurely exit at the second
statement with value 1 if the value of \spad{i} is greater than 0.
See \spadgloss{\spad{=>}} for an alternate syntax.
\item\newline{\em \menuitemstyle{}}{\em explicit export}\space{}
1. (of a domain \spad{D}) any \spadgloss{attribute},{}
\spadgloss{operation},{} or \spadgloss{category} explicitly mentioned
in the \spadgloss{type} specification part \spad{T} for the domain
constructor definition \spad{D: T == I} 2. (of a category \spad{C})
any \spadgloss{attribute},{} \spadgloss{operation},{} or
\spadgloss{category} explicitly mentioned in the \spadgloss{type}
specification part \spad{T} for the domain constructor definition
\spad{C: \spadgloss{Category} == T}
\item\newline{\em \menuitemstyle{}}{\em export}\space{}
\spadgloss{explicit export} or \spadgloss{implicit export} of a domain
or category
\item\newline{\em \menuitemstyle{}}{\em expose}\space{}
some constructors are {\em exposed},{} others {\em unexposed}. Exposed
domains and packages are recognized by the interpreter. Use \spadsys{)set
expose} to control change what is exposed. To see both exposed and
unexposed constructors,{} use \Browse{} with give the system command
\spadsyscom{)set hyperdoc browse exposure on}. Unexposed constructors
will now appear prefixed by star(\spad{*}).
\item\newline{\em \menuitemstyle{}}{\em expression}\space{}
1. any syntactically correct program fragment. 2. an element of domain
\spadtype{Expression}
\item\newline{\em \menuitemstyle{}}{\em extend}\space{}
see \spadgloss{category extension} or \spadgloss{domain extension}
\item\newline{\em \menuitemstyle{}}{\em field}\space{}
{\em (algebra)} a \spadgloss{domain} which is \spadgloss{ring} where
every non-zero element is invertible and where \spad{xy=yx}; a member
of category \spadtype{Field}. For a complete list of fields,{} click
on {\em Domains} under {\em Cross Reference} for \spadtype{Field}.
\item\newline{\em \menuitemstyle{}}{\em file}\space{}
a program or collection of data stored on disk,{} tape or other medium.
\item\newline{\em \menuitemstyle{}}{\em float}\space{}
a floating-point number with user-specified precision; an element of
domain \spadtype{Float}. Floats are \spadgloss{literals} which are written
two ways: without an exponent (\spadignore{e.g.} \spad{3.1416}),{} or with
an exponent (\spadignore{e.g.} \spad{3.12E-12}). Use function
\spadgloss{precision} to change the precision of the mantissage
```

```
(20 digits by default). See also \spadgloss{small float}.
\item\newline{\em \menuitemstyle{}}{\em formal parameter}\space{}
(of a function) an identifier \spadglossSee{bound}{binding} to the value
of an actual \spadgloss{argument} on \spadgloss{invocation}. In the
function definition \spad{f(x,{}y) == u},{} for example,{} \spad{x}
and \spad{y} are the formal parameter.
\item\newline{\em \menuitemstyle{}}{\em frame}\space{}
the basic unit of an interactive session; each frame has its own
\spadgloss{step number},{} \spadgloss{environment},{} and
\spadgloss{history}. In one interactive session,{} users can create
and drop frames,{} and have several active frames simultaneously.
\item\newline{\em \menuitemstyle{}}{\em free}\space{}
{\em (syntax)} A keyword used in user-defined functions to declare
that a variable is a \spadgloss{free variable} of that function. For
example,{} the statement \spad{free x} declares the variable \spad{x}
within the body of a function \spad{f} to be a free variable in \spad{f}.
Without such a declaration,{} any variable \spad{x} which appears on the
left hand side of an assignment is regarded as a \spadgloss{local variable}
of that function. If the intention of the assignment is to give an value
to a \spadgloss{global variable} \spad{x},{} the body of that function
must contain the statement \spad{free x}.
\item\newline{\em \menuitemstyle{}}{\em free variable}\space{}
(of a function) a variable which appears in a body of a function but is
not \spadglossSee{bound}{binding} by that function. See \spadgloss{local
variable} by default.
\item\newline{\em \menuitemstyle{}}{\em function}\space{}
implementation of \spadgloss{operation}; it takes zero or more
\spadgloss{argument} parameters and produces zero or more values.
Functions are objects which can be passed as parameters to functions
and can be returned as values of functions. Functions can also create
other functions (see also \spadtype{InputForm}). See also
\spadgloss{application} and \spadgloss{invocation}. The terms
{\em operation} and {\em function} are distinct notions in Axiom.
An operation is an abstraction of a function,{} described by declaring a
\spadgloss{signature}. A function is created by providing an implementation
of that operation by some piece of Axiom code. Consider the example
of defining a user-function \spad{fact} to compute the \spadfun{factorial}
of a nonnegative integer. The Axiom statement
\spad{fact: Integer -> Integer} describes the operation,{} whereas the
statement \spad{fact(n) = reduce(*,{}[1..n])} defines the functions. See
also \spadgloss{generic function}.
\item\newline{\em \menuitemstyle{}}{\em function body}\space{}
the part of a \spadgloss{function}\spad{'s} definition which is evaluated
when the function is called at \spadgloss{run-time}; the part of the
function definition to the right of the \spadSyntax{\spad{==}}.
\item\newline{\em \menuitemstyle{}}{\em garbage collection}\space{}
a system function that automatically recycles memory cells from the
\spadgloss{heap}. Axiom is built upon \spadgloss{Common LISP}
which provides this facility.
\item\newline{\em \menuitemstyle{}}{\em garbage collector}\space{}
```

```
a mechanism for reclaiming storage in the \spadgloss{heap}.
\item\newline{\em \menuitemstyle{}}{\em Gaussian}\space{}
a complex-valued expression,{} \spadignore{e.g.} one with both a real
and imaginary part; a member of a \spadtype{Complex} domain.
\item\newline{\em \menuitemstyle{}}{\em generic function}\space{}
the use of one function to operate on objects of different types; One
might regard Axiom as supporting generic \spadgloss{operations}
but not generic functions. One operation \spad{+: (D,{}D) -> D} exists
for adding elements in a ring; each ring however provides its own
type-specific function for implementing this operation.
\item\newline{\em \menuitemstyle{}}{\em global variable}\space{}
A variable which can be referenced freely by functions. In Axiom,{}
all top-level user-defined variables defined during an interactive user
session are global variables. Axiom does not allow {\em fluid
variables},{} that is,{} variables \spadglossSee{bound}{binding} by
functions which can be referenced by functions those functions call.
\item\newline{\em \menuitemstyle{}}{\em Groebner basis}\space{}
{\em (algebra)} a special basis for a polynomial ideal that allows a
simple test for membership. It is useful in solving systems of polynomial
equations.
\item\newline{\em \menuitemstyle{}}{\em group}\space{}
{\em (algebra)} a monoid where every element has a multiplicative inverse.
\item\newline{\em \menuitemstyle{}}{\em hash table}\space{}
A data structure that efficiency maps a given object to another. A hash
table consists of a set of {\em entries},{} each of which associates a
{\em key} with a {\em value}. Finding the object stored under a key can
be very fast even if there are a large number of entries since keys are
{\em hashed} into numerical codes for fast lookup.
\item\newline{\em \menuitemstyle{}}{\em heap}\space{}
an area of storage used by data in programs. For example,{} Axiom will
use the heap to hold the partial results of symbolic computations. When
cancellations occur,{} these results remain in the heap until
\spadglossSee{garbage collected}{garbage collector}.
\item\newline{\em \menuitemstyle{}}{\em history}\space{}
a mechanism which records the results for an interactive computation.
Using the history facility,{} users can save computations,{} review
previous steps of a computation,{} and restore a previous interactive
session at some later time. For details,{} issue the system command
{\em )history ?} to the interpreter. See also \spadgloss{frame}.
\item\newline{\em \menuitemstyle{}}{\em ideal}\space{}
{\em (algebra)} a subset of a ring that is closed under addition and
multiplication by arbitrary ring elements,{} \spadignore{i.e.}
it\spad{'s} a module over the ring.
\item\newline{\em \menuitemstyle{}}{\em identifier}\space{}
{\em (syntax)} an Axiom name; a \spadgloss{literal} of type
\spadtype{Symbol}. An identifier begins with an alphabetical character or
\% and may be followed by alphabetic characters,{} digits,{} ? or !.
Certain distinquished \spadgloss{reserved words} are not allowed as
identifiers but have special meaning in the Axiom.
\item\newline{\em \menuitemstyle{}}{\em immutable}\space{}
```

```
an object is immutable if it cannot be changed by an \spadgloss{operation};
not a \spadglossSee{mutable object}{mutable}. Algebraic objects generally
immutable: changing an algebraic expression involves copying parts of the
original object. One exception is a matrix object of type \spadtype{Matrix}.
Examples of mutable objects are data structures such as those of type
\spadtype{List}. See also \spadgloss{pointer semantics}.
\item\newline{\em \menuitemstyle{}}{\em implicit export}\space{}
(of a domain or category) any \spadgloss{attribute} or
\spadgloss{operation} which is either an explicit export or else an
explicit export of some category which an explicit category export
\spadglossSee{extends}{category extension}.
\item\newline{\em \menuitemstyle{}}{\em index}\space{}
1. a variable that counts the number of times a \spadgloss{loop} is
repeated. 2. the "address" of an element in a data structure (see also
category \spadtype{LinearAggregate}).
\item\newline{\em \menuitemstyle{}}{\em infix}\space{}
{\em (syntax)} an \spadgloss{operator} placed between two
\spadgloss{operands}; also called a {\em binary operator},{}
\spadignore{e.g.} \spad{a + b}. An infix operator may also be used as a
\spadgloss{prefix},{} \spadignore{e.g.} \spad{+(a,{}b)} is also permissable
in the Axiom language. Infix operators have a relative
\spadgloss{precedence}.
\item\newline{\em \menuitemstyle{}}{\em input area}\space{}
a rectangular area on a Hyperdoc screen into which users can enter text.
\item\newline{\em \menuitemstyle{}}{\em instantiate}\space{}
to build a \spadgloss{category},{} \spadgloss{domain},{} or
\spadgloss{package} at run-time
\item\newline{\em \menuitemstyle{}}{\em integer}\space{}
a \spadgloss{literal} object of domain \spadtype{Integer},{} the class
of integers with an unbounded number of digits. Integer literals consist
of one or more consecutive digits (0-9) with no embedded blanks.
Underscores can be used to separate digits in long integers if desirable.
\item\newline{\em \menuitemstyle{}}{\em interactive}\space{}
a system where the user interacts with the computer step-by-step
\item\newline{\em \menuitemstyle{}}{\em interpreter}\space{}
the subsysystem of Axiom responsible for handling user input during
an interactive session. The following somewhat simplified description of
the typical action of the interpreter. The interpreter parsers the
user\spad{'s} input expression to create an expression tree then does a
bottom-up traversal of the tree. Each subtree encountered which is not a
value consists of a root node denoting an operation name and one or more
leaf nodes denoting \spadgloss{operands}. The interpreter resolves type
mismatches and uses type-inferencing and a library database to determine
appropriate types of the operands and the result,{} and an operation to
be performed. The interpreter then builds a domain to perform the
indicated operation,{} then invokes a function from the domain to compute
a value. The subtree is then replaced by that value and the process
continues. Once the entire tree has been processed,{} the value replacing
the top node of the tree is displayed back to the user as the value of the
expression.
```

```
\item\newline{\em \menuitemstyle{}}{\em invocation}\space{}
(of a function) the run-time process involved in
\spadglossSee{evaluating}{evaluation} a a \spadgloss{function}
\spadgloss{application}. This process has two steps. First,{} a
local \spadgloss{environment} is created where \spadgloss{formal arguments}
are locally \spadglossSee{bound}{binding} by \spadgloss{assignment} to
their respective actual \spadgloss{argument}. Second,{} the
\spadgloss{function body} is evaluated in that local environment.
The evaluation of a function is terminated either by completely
evaluating the function body or by the evaluation of a
\spadSyntax{return} expression.
\item\newline{\em \menuitemstyle{}}{\em iteration}\space{}
repeated evaluation of an expression or a sequence of expressions.
Iterations use the reserved words \spadSyntax{for},{}
\spadSyntax{while},{} and \spadSyntax{repeat}.
\item\newline{\em \menuitemstyle{}}{\em Join}\space{}
a primitive Axiom function taking two or more categories as
arguments and producing a category containing all of the operations
and attributes from the respective categories.
\item\newline{\em \menuitemstyle{}}{\em KCL}\space{}
Kyoto Common LISP,{} a version of \spadgloss{Common LISP} which
features compilation of the compilation of LISP into the \spad{C}
Programming Language
\item\newline{\em \menuitemstyle{}}{\em library}\space{}
In Axiom,{} a collection of compiled modules respresenting
the \spadgloss{category} or \spadgloss{domain} constructor.
\item\newline{\em \menuitemstyle{}}{\em lineage}\space{}
the sequence of \spadgloss{default packages} for a given domain to be
searched during \spadgloss{dynamic lookup}. This sequence is computed
first by ordering the category \spadgloss{ancestors} of the domain
according to their {\em level number},{} an integer equal to to the
minimum distance of the domain from the category. Parents have level
1,{} parents of parents have level 2,{} and so on. Among categories
with equal level numbers,{} ones which appear in the left-most branches
of {\em Join}\spad{s} in the source code come first. See also
\spadgloss{dynamic lookup}.
\item\newline{\em \menuitemstyle{}}{\em LISP}\space{}
acronymn for List Processing Language,{} a language designed for the
manipulation of nonnumerical data. The Axiom library is
translated into LISP then compiled into machine code by an underlying LISP.
\item\newline{\em \menuitemstyle{}}{\em list}\space{}
an object of a \spadtype{List} domain.
\item\newline{\em \menuitemstyle{}}{\em literal}\space{}
an object with a special syntax in the language. In Axiom,{}
there are five types of literals: \spadgloss{booleans},{}
\spadgloss{integers},{} \spadgloss{floats},{} \spadgloss{strings},{}
and \spadgloss{symbols}.
\item\newline{\em \menuitemstyle{}}{\em local}\space{}
{\em (syntax)} A keyword used in user-defined functions to declare that
a variable is a \spadgloss{local variable} of that function. Because of
```

```
default assumptions on variables,{} such a declaration is not necessary
but is available to the user for clarity when appropriate.
\item\newline{\em \menuitemstyle{}}{\em local variable}\space{}
(of a function) a variable \spadglossSee{bound}{binding} by that function
and such that its binding is invisible to any function that function calls.
Also called a {\em lexical} variable. By default in the
interpreter:\begin{items} \item 1. any variable \spad{x} which appears
on the left hand side of an assignment is regarded a local variable of
that function. If the intention of an assignment is to change the value
of a \spadgloss{global variable} \spad{x},{} the body of the function
must then contain the statement \spad{free x}. \item 2. any other variable
is regarded as a \spadgloss{free variable}. \end{items} An optional
declaration \spad{local x} is available to explicitly declare a variable
to be a local variable. All \spadgloss{formal parameters} to the function
can be regarded as local variables to the function.
\item\newline{\em \menuitemstyle{}}{\em loop}\space{}
1. an expression containing a \spadSyntax{repeat} 2. a collection expression
having a \spadSyntax{for} or a \spadSyntax{while},{} \spadignore{e.g.}
\spad{[f(i) for i in S]}.
\item\newline{\em \menuitemstyle{}}{\em loop body}\space{}
the part of a loop following the \spadSyntax{repeat} that tells what to
do each iteration. For example,{} the body of the loop \spad{for x in S
repeat B} is \spad{B}. For a collection expression,{} the body of the loop
precedes the initial \spadSyntax{for} or \spadSyntax{while}.
\item\newline{\em \menuitemstyle{}}{\em macro}\space{}
{\em (syntax)} 1. An expression of the form \spad{macro a == b} where
\spad{a} is a \spadgloss{symbol} causes \spad{a} to be textually replaced
by the expression \spad{b} at \spadgloss{parse} time. 2. An expression of
the form \spad{macro f(a) == b} defines a parameterized macro expansion for
a parameterized form \spad{f} This macro causes a form \spad{f}(\spad{x})
to be textually replaced by the expression \spad{c} at parse time,{} where
\spad{c} is the expression obtained by replacing \spad{a} by \spad{x}
everywhere in \spad{b}. See also \spadgloss{definition} where a similar
substitution is done during \spadgloss{evaluation}.
\item\newline{\em \menuitemstyle{}}{\em mode}\space{}
a type expression containing a question-mark ({\em ?}). For example,{}
the mode {\em P ?} designates {\em the class of all polynomials over an
arbitrary ring}.
\item\newline{\em \menuitemstyle{}}{\em mutable}\space{}
objects which contain \spadgloss{pointers} to other objects and which
have operations defined on them which alter these pointers. Contrast
\spadgloss{immutable}. Axiom uses \spadgloss{pointer semantics}
as does \spadgloss{LISP} in contrast with many other languages such as
Pascal which use \spadgloss{copying semantics}. See \spadgloss{pointer
semantics} for details.
\item\newline{\em \menuitemstyle{}}{\em name}\space{}
1. a \spadgloss{symbol} denoting a \spadgloss{variable},{}
\spadignore{i.e.} the variable \spad{x}. 2. a \spadgloss{symbol}
denoting an \spadgloss{operation},{} \spadignore{i.e.} the operation
\spad{divide: (Integer,{}Integer) -> Integer}.
```

```
\item\newline{\em \menuitemstyle{}}{\em nullary}\space{}
a function with no arguments,{} \spadignore{e.g.} \spadfun{characteristic}.
\item\newline{\em \menuitemstyle{}}{\em nullary}\space{}
operation or function with \spadgloss{arity} 0
\item\newline{\em \menuitemstyle{}}{\em Object}\space{}
a category with no operations or attributes,{} from which most categories
in Axiom are \spadglossSee{extensions}{category extension}.
\item\newline{\em \menuitemstyle{}}{\em object}\space{}
a data entity created or manipulated by programs. Elements of domains,{}
functions,{} and domains themselves are objects. Whereas categories are
created by functions,{} they cannot be dynamically manipulated in the
current system and are thus not considered as objects. The most basic
objects are \spadgloss{literals}; all other objects must be created
\spadgloss{functions}. Objects can refer to other objects using
\spadgloss{pointers}. Axiom language uses
\spadgloss{pointer semantics} when dealing with \spadgloss{mutable} objects.
\item\newline{\em \menuitemstyle{}}{\em object code}\space{}
code which can be directly executed by hardware; also known as
{\em machine language}.
\item\newline{\em \menuitemstyle{}}{\em operand}\space{}
an argument of an \spadgloss{operator} (regarding an operator as a
\spadgloss{function}).
\item\newline{\em \menuitemstyle{}}{\em operation}\space{}
an abstraction of a \spadgloss{function},{} described by a
\spadgloss{signature}. For example,{}
\center{\spad{fact: NonNegativeInteger -> NonNegativeInteger}} describes
an operation for "the factorial of a (non-negative) integer".
\item\newline{\em \menuitemstyle{}}{\em operator}\space{}
special reserved words in the language such as \spadop{+} and \spadop{*};
operators can be either \spadgloss{prefix} or \spadgloss{infix} and have a
relative \spadgloss{precedence}.
\item\newline{\em \menuitemstyle{}}{\em overloading}\space{}
the use of the same name to denote distinct functions; a function is
identified by a \spadgloss{signature} identifying its name,{} the number
and types of its arguments,{} and its return types. If two functions can
have identical signatures,{} a \spadgloss{package call} must be made to
distinquish the two.
\item\newline{\em \menuitemstyle{}}{\em package}\space{}
a domain whose exported operations depend solely on the parameters and
other explicit domains,{} \spadignore{e.g.} a package for solving systems
of equations of polynomials over any field,{} \spadignore{e.g.} floats,{}
rational numbers,{} complex rational functions,{} or power series.
Facilities for integration,{} differential equations,{} solution of
linear or polynomial equations,{} and group theory are provided by
"packages". Technically,{} a package is a domain which has no
\spadgloss{signature} containing the symbol \$. While domains intuitively
provide computational objects you can compute with,{} packages intuitively
provide functions (\spadgloss{polymorphic} functions) which will work over
a variety of datatypes.
\item\newline{\em \menuitemstyle{}}{\em package call}\space{}
```

```
{\em (syntax)} an expression of the form \spad{e \$ D} where \spad{e} is
an \spadgloss{application} and \spad{D} denotes some \spadgloss{package}
(or \spadgloss{domain}).
\item\newline{\em \menuitemstyle{}}{\em package call}\space{}
{\em (syntax)} an expression of the form \spad{f(x,{}y)\$D} used to identify
that the function \spad{f} is to be one from \spad{D}.
\item\newline{\em \menuitemstyle{}}{\em package constructor}\space{}
same as \spadgloss{domain constructor}.
\item\newline{\em \menuitemstyle{}}{\em parameter}\space{}
see \spadgloss{argument}
\item\newline{\em \menuitemstyle{}}{\em parameterized datatype}\space{}
a domain that is built on another,{} for example,{} polynomials with integer
coefficients.
\item\newline{\em \menuitemstyle{}}{\em parameterized form}\space{}
a expression of the form \spad{f(x,{}y)},{} an \spadgloss{application} of a
function.
\item\newline{\em \menuitemstyle{}}{\em parent}\space{}
(of a domain) a category which is explicitly declared in the source code
definition for the domain to be an \spadgloss{export} of the domain.
\item\newline{\em \menuitemstyle{}}{\em parse}\space{}
1. (verb) to produce an internal representation of a user input string;
the resultant internal representation is then "interpreted" by Axiom
to perform some indicated action.
\item\newline{\em \menuitemstyle{}}{\em parse}\space{}
the transformation of a user input string representing a valid Axiom
expression into an internal representation as a tree-structure.
\item\newline{\em \menuitemstyle{}}{\em partially ordered set}\space{}
a set with a reflexive,{} transitive and antisymetric \spadgloss{binary}
operation.
\item\newline{\em \menuitemstyle{}}{\em pattern match}\space{}
1. (on expressions) Given a expression called a "subject" \spad{u},{} the
attempt to rewrite \spad{u} using a set of "rewrite rules". Each rule has
the form \spad{A == B} where \spad{A} indicates a expression called a
"pattern" and \spad{B} denotes a "replacement". The meaning of this rule
is "replace \spad{A} by \spad{B"}. If a given pattern \spad{A} matches a
subexpression of \spad{u},{} that subexpression is replaced by \spad{B}.
Once rewritten,{} pattern matching continues until no further changes occur.
2. (on strings) the attempt to match a string indicating a "pattern" to
another string called a "subject",{} for example,{} for the purpose of
identifying a list of names. In \Browse{},{} users may enter
\spadgloss{search strings} for the purpose of identifying constructors,{}
operations,{} and attributes.
\item\newline{\em \menuitemstyle{}}{\em pile}\space{}
alternate syntax for a block,{} using indentation and column alignment
(see also \spadgloss{block}).
\item\newline{\em \menuitemstyle{}}{\em pointer}\space{}
a reference implemented by a link directed from one object to another in
the computer memory. An object is said to {\em refer} to another if it
has a pointer to that other object. Objects can also refer to themselves
(cyclic references are legal). Also more than one object can refer to the
```

```
same object. See also \spadgloss{pointer semantics}.
\item\newline{\em \menuitemstyle{}}{\em pointer semantics}\space{}
the programming language semantics used in languages such as LISP which
allow objects to be \spadgloss{mutable}. Consider the following sequence
of Axiom statements:\begin{items} \item \spad{x : Vector Integer :=
[1,{}4,{}7]} \item \spad{y := x} \item \spad{swap!(x,{}2,{}3)} \end{items}
The function \spadfunFrom{swap!}{Vector} is used to interchange the 2nd and
3rd value in the list \spad{x} producing the value \spad{[1,{}7,{}4]}.
What value does \spad{y} have after evaluation of the third statement?
The answer is different in Axiom than it is in a language with
\spadgloss{copying semantics}. In Axiom,{} first the vector
[1,{}2,{}3] is created and the variable \spad{x} set to
\spadglossSee{point}{pointer} to this object. Let\spad{'s} call this
object \spad{V}. Now \spad{V} refers to its \spadgloss{immutable}
components 1,{}2,{} and 3. Next,{} the variable \spad{y} is made to
point to \spad{V} just as \spad{x} does. Now the third statement
interchanges the last 2 elements of \spad{V} (the {\em !} at the end
of the name \spadfunFrom{swap!}{Vector} tells you that this operation
is destructive,{} that is,{} it changes the elements {\em in place}).
Both \spad{x} and \spad{y} perceive this change to \spad{V}. Thus both
\spad{x} and \spad{y} then have the value \spad{[1,{}7,{}4]}. In Pascal,{}
the second statement causes a copy of \spad{V} to be stored under \spad{y}.
Thus the change to \spad{V} made by the third statement does not affect
\spad{y}.
\item\newline{\em \menuitemstyle{}}{\em polymorphic}\space{}
a \spadgloss{function} parameterized by one or more \spadgloss{domains};
a \spadgloss{algorithm} defined \spadglossSee{categorically}{category}.
Every function defined in a domain or package constructor with a
domain-valued parameter is polymorphic. For example,{} the same matrix
\spadSyntax{*} function is used to multiply "matrices over integers" as
"matrices over matrices over integers" Likewise,{} various
\ignore{\spad{???}} in \ignore{\spad{???}} solves polynomial equations
over any \spadgloss{field}.
\item\newline{\em \menuitemstyle{}}{\em postfix}\space{}
an \spadgloss{operator} that follows its single \spadgloss{operand}.
Postfix operators are not available in Axiom.
\item\newline{\em \menuitemstyle{}}{\em precedence}\space{}
{\em (syntax)} refers to the so-called {\em binding power} of an operator.
For example,{} \spad{*} has higher binding power than \spad{+} so that the
expression \spad{a + b * c} is equivalent to \spad{a + (b * c)}.
\item\newline{\em \menuitemstyle{}}{\em precision}\space{}
the number of digits in the specification of a number,{} \spadignore{e.g.}
as set by \spadfunFrom{precision}{Float}.
\item\newline{\em \menuitemstyle{}}{\em predicate}\space{}
1. a Boolean valued function,{} \spadignore{e.g.}
\spad{odd: Integer -> Boolean}. 2. an Boolean valued expression
\item\newline{\em \menuitemstyle{}}{\em prefix}\space{}
{\em (syntax)} an \spadgloss{operator} such as \spad{-} and \spad{not}
that is written {\em before} its single \spadgloss{operand}. Every
function of one argument can be used as a prefix operator. For example,{}
```

```
all of the following have equivalent meaning in Axiom: \spad{f(x)},{}
\spad{f x},{} and \spad{f.x}. See also \spadgloss{dot notation}.
\item\newline{\em \menuitemstyle{}}{\em quote}\space{}
the prefix \spadgloss{operator} \spadSyntax{'} meaning {\em do not evaluate}.
\item\newline{\em \menuitemstyle{}}{\em Record}\space{}
(basic domain constructor) a domain constructor used to create a
inhomogeneous aggregate composed of pairs of "selectors" and
\spadgloss{values}. A Record domain is written in the form
\spad{Record(a1:D1,{}...,{}an:Dn)} (\spad{n} > 0) where
\spad{a1},{}...,{}\spad{an} are identifiers called the
{\em selectors} of the record,{} and \spad{D1},{}...,{}\spad{Dn}
are domains indicating the type of the component stored under selector
\spad{an}.
\item\newline{\em \menuitemstyle{}}{\em recurrence relation}\space{}
A relation which can be expressed as a function \spad{f} with some
argument \spad{n} which depends on the value of \spad{f} at \spad{k}
previous values. In many cases,{} Axiom will rewrite a recurrence
relation on compilation so as to \spadgloss{cache} its previous \spad{k}
values and therefore make the computation significantly more efficient.
\item\newline{\em \menuitemstyle{}}{\em recursion}\space{}
use of a self-reference within the body of a function. Indirect recursion
is when a function uses a function below it in the call chain.
\item\newline{\em \menuitemstyle{}}{\em recursive}\space{}
1. A function that calls itself,{} either directly or indirectly through
another function. 2. self-referential. See also \spadgloss{recursive}.
\item\newline{\em \menuitemstyle{}}{\em reference}\space{}
see \spadgloss{pointer}
\item\newline{\em \menuitemstyle{}}{\em Rep}\space{}
a special identifier used as \spadgloss{local variable} of a domain
constructor body to denote the representation domain for objects of a domain.
\item\newline{\em \menuitemstyle{}}{\em representation}\space{}
a \spadgloss{domain} providing a data structure for elements of a
domain; generally denoted by the special identifier \spadgloss{Rep}
in the Axiom programming language. As domains are
\spadgloss{abstract datatypes},{} this representation is not available
to users of the domain,{} only to functions defined in the
\spadgloss{function body} for a domain constructor. Any domain can be used
as a representation.
\item\newline{\em \menuitemstyle{}}{\em reserved word}\space{}
a special sequence of non-blank characters with special meaning in the
Axiom language. Examples of reserved words are names such as
\spadSyntax{for},{} \spadSyntax{if},{} and \spadSyntax{free},{} operator
names such as \spadSyntax{+} and \spad{mod},{} special character strings
such as spadSyntax{\spad{==}} and spadSyntax{\spad{:=}}.
\item\newline{\em \menuitemstyle{}}{\em retraction}\space{}
to move an object in a parameterized domain back to the underlying
domain,{} for example to move the object \spad{7} from a "fraction of
integers" (domain \spadtype{Fraction Integer}) to "the integers"
(domain \spadtype{Integer}).
\item\newline{\em \menuitemstyle{}}{\em return}\space{}
```

```
when leaving a function,{} the value of the expression following
\spadSyntax{return} becomes the value of the function.
\item\newline{\em \menuitemstyle{}}{\em ring}\space{}
a set with a commutative addition,{} associative multiplication,{}
a unit element,{} and multiplication distributes over addition and
subtraction.
\item\newline{\em \menuitemstyle{}}{\em rule}\space{}
{\em (syntax)} 1. An expression of the form \spad{rule A == B}
indicating a "rewrite rule". 2. An expression of the form
\spad{rule (R1;...;Rn)} indicating a set of "rewrite rules"
\spad{R1},{}...,{}\spad{Rn}. See \spadgloss{pattern matching} for details.
\item\newline{\em \menuitemstyle{}}{\em run-time}\space{}
the time of doing a computation. Contrast \spadgloss{compile-time}.
 rather than prior to it; \spadgloss{dynamic} as opposed to
\spadgloss{static}. For example,{} the decision of the intepreter
to build a structure such as "matrices with power series entries"
in response to user input is made at run-time.
\item\newline{\em \menuitemstyle{}}{\em run-time check}\space{}
an error-checking which can be done only when the program receives
user input; for example,{} confirming that a value is in the proper
range for a computation.
\item\newline{\em \menuitemstyle{}}{\em search string}\space{}
a string entered into an \spadgloss{input area} on a Hyperdoc screen
\item\newline{\em \menuitemstyle{}}{\em selector}\space{}
an identifier used to address a component value of a gloss{Record} datatype.
\item\newline{\em \menuitemstyle{}}{\em semantics}\space{}
the relationships between symbols and their meanings. The rules for
obtaining the {\em meaning} of any syntactically valid expression.
\item\newline{\em \menuitemstyle{}}{\em semigroup}\space{}
{\em (algebra)} a \spadgloss{monoid} which need not have an identity;
it is closed and associative.
\item\newline{\em \menuitemstyle{}}{\em side effect}\space{}
action which changes a component or structure of a value. See
\spadgloss{destructive operation} for details.
\item\newline{\em \menuitemstyle{}}{\em signature}\space{}
{\em (syntax)} an expression describing an \spadgloss{operation}.
A signature has the form as \spad{name : source -> target},{} where
\spad{source} gives the type of the arguments of the operation,{} and
\spad{target} gives the type of the result.
\item\newline{\em \menuitemstyle{}}{\em small float}\space{}
the domain for hardware floating point arithmetic as provided by the
computer hardware.
\item\newline{\em \menuitemstyle{}}{\em small integer}\space{}
the domain for hardware integer arithmetic. as provided by the computer
hardware.
\item\newline{\em \menuitemstyle{}}{\em source}\space{}
the \spadgloss{type} of the argument of a \spadgloss{function}; the type
expression before the \spad{->} in a \spadgloss{signature}. For example,{}
the source of \spad{f : (Integer,{}Integer) -> Integer} is
\spad{(Integer,{}Integer)}.
```

```
\item\newline{\em \menuitemstyle{}}{\em sparse}\space{}
data structure whose elements are mostly identical (a sparse matrix is
one filled with mostly zeroes).
\item\newline{\em \menuitemstyle{}}{\em static}\space{}
that computation done before run-time,{} such as compilation. Contrast
\spadgloss{dynamic}.
\item\newline{\em \menuitemstyle{}}{\em step number}\space{}
the number which precedes user input lines in an interactive session;
the output of user results is also labeled by this number.
\item\newline{\em \menuitemstyle{}}{\em stream}\space{}
an object of \spadtype{Stream(R)},{} a generalization of a
\spadgloss{list} to allow an infinite number of elements. Elements of
a stream are computed "on demand". Strings are used to implement various
forms of power series (\ignore{\spad{???}}).
\item\newline{\em \menuitemstyle{}}{\em string}\space{}
an object of domain \spadtype{String}. Strings are \spadgloss{literals}
consisting of an arbitrary sequence of \spadgloss{characters} surrounded
by double-quotes (\spadSyntax{"}),{} \spadignore{e.g.} \spad{"Look here!"}.
\item\newline{\em \menuitemstyle{}}{\em subdomain}\space{}
{\em (basic concept)} a \spadgloss{domain} together with a
\spadgloss{predicate} characterizing which members of the domain belong
to the subdomain. The exports of a subdomain are usually distinct from the
domain itself. A fundamental assumption however is that values in the
subdomain are automatically \spadglossSee{coerceable}{coercion} to values
in the domain. For example,{} if \spad{n} and \spad{m} are declared to be
members of a subdomain of the integers,{} then {\em any} \spadgloss{binary}
operation from \spadtype{Integer} is available on \spad{n} and \spad{m}.
On the other hand,{} if the result of that operation is to be assigned
to,{} say,{} \spad{k},{} also declared to be of that subdomain,{} a
\spadgloss{run-time} check is generally necessary to ensure that the
result belongs to the subdomain.
\item\newline{\em \menuitemstyle{}}{\em such that clause}\space{}
the use of \spadSyntax{|} followed by an expression to filter an iteration.
\item\newline{\em \menuitemstyle{}}{\em suffix}\space{}
{\em (syntax)} an \spadgloss{operator} which placed after its operand.
Suffix operators are not allowed in the Axiom language.
\item\newline{\em \menuitemstyle{}}{\em symbol}\space{}
objects denoted by \spadgloss{identifier} \spadgloss{literals}; an
element of domain \spadtype{Symbol}. The interpreter defaultly converts
a symbol \spad{x} into \spadtype{Variable(x)}.
\item\newline{\em \menuitemstyle{}}{\em syntax}\space{}
rules of grammar,{} punctuation etc. for forming correct expressions.
\item\newline{\em \menuitemstyle{}}{\em system commands}\space{}
top-level Axiom statements that begin with {\em )}. System
commands allow users to query the database,{} read files,{} trace
functions,{} and so on.
\item\newline{\em \menuitemstyle{}}{\em tag}\space{}
an identifier used to discriminate a branch of a \spadgloss{Union} type.
\item\newline{\em \menuitemstyle{}}{\em target}\space{}
the \spadgloss{type} of the result of a \spadgloss{function}; the
```

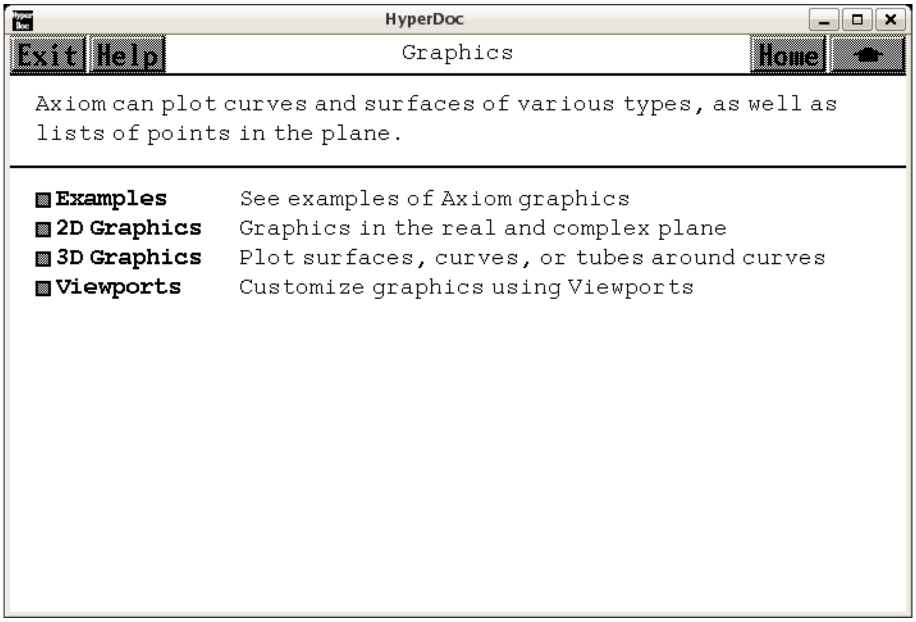
```
type expression following the \spad{->} in a \spadgloss{signature}.
\item\newline{\em \menuitemstyle{}}{\em top-level}\space{}
refers to direct user interactions with the Axiom interpreter.
\item\newline{\em \menuitemstyle{}}{\em totally ordered set}\space{}
{\em (algebra)} a partially ordered set where any two elements are
comparable.
\item\newline{\em \menuitemstyle{}}{\em trace}\space{}
use of system function \spadsys{)trace} to track the arguments passed
to a function and the values returned.
\item\newline{\em \menuitemstyle{}}{\em tuple}\space{}
an expression of two or more other expressions separated by commas,{}
\spadignore{e.g.} \spad{4,{}7,{}11}. Tuples are also used for multiple
arguments both for \spadgloss{applications} (\spadignore{e.g.}
\spad{f(x,{}y)}) and in \spadgloss{signatures} (\spadignore{e.g.}
\spad{(Integer,{}Integer) -> Integer}). A tuple is not a data structure,{}
rather a syntax mechanism for grouping expressions.
\item\newline{\em \menuitemstyle{}}{\em type}\space{}
The type of any \spadgloss{subdomain} is the unique symbol {\em Category}.
The type of a \spadgloss{domain} is any \spadgloss{category} that domain
belongs to. The type of any other object is either the (unique) domain
that object belongs to or any \spadgloss{subdomain} of that domain. The
type of objects is in general not unique.
\item\newline{\em \menuitemstyle{}}{\em type checking}\space{}
a system function which determines whether the datatype of an object is
appropriate for a given operation.
\item\newline{\em \menuitemstyle{}}{\em type constructor}\space{}
a \spadgloss{domain constructor} or \spadgloss{category constructor}.
\item\newline{\em \menuitemstyle{}}{\em type inference}\space{}
when the interpreter chooses the type for an object based on context.
For example,{} if the user interactively issues the definition
\center{\spad{f(x) == (x + \%i)**2}} then issues \spad{f(2)},{}
the interpreter will infer the type of \spad{f} to be
\spad{Integer -> Complex Integer}.
\item\newline{\em \menuitemstyle{}}{\em unary}\space{}
operation or function with \spadgloss{arity} 1
\item\newline{\em \menuitemstyle{}}{\em underlying domain}\space{}
for a \spadgloss{domain} that has a single domain-valued parameter,{}
the {\em underlying domain} refers to that parameter. For example,{}
the domain "matrices of integers" (\spadtype{Matrix Integer}) has
underlying domain \spadtype{Integer}.
\item\newline{\em \menuitemstyle{}}{\em Union}\space{}
(basic domain constructor) a domain constructor used to combine any
set of domains into a single domain. A Union domain is written in the
form \spad{Union(a1:D1,{}...,{}an:Dn)} (\spad{n} > 0) where
\spad{a1},{}...,{}\spad{an} are identifiers called the {\em tags} of
the union,{} and \spad{D1},{}...,{}\spad{Dn} are domains called the
{\em branches} of the union. The tags \spad{\spad{ai}} are optional,{}
but required when two of the \spad{\spad{Di}} are equal,{}
\spadignore{e.g.} \spad{Union(inches:Integer,{}centimeters:Integer)}.
In the interpreter,{} values of union domains are automatically coerced
```

```
to values in the branches and vice-versa as appropriate. See also
\spadgloss{case}.
\item\newline{\em \menuitemstyle{}}{\em unit}\space{}
{\em (algebra)} an invertible element.
\item\newline{\em \menuitemstyle{}}{\em user function}\space{}
a function defined by a user during an interactive session. Contrast
\spadgloss{built-in function}.
\item\newline{\em \menuitemstyle{}}{\em user variable}\space{}
a variable created by the user at top-level during an interactive session
\item\newline{\em \menuitemstyle{}}{\em value}\space{}
1. the result of \spadglossSee{evaluating}{evaluation} an expression.
2. a property associated with a \spadgloss{variable} in a \spadgloss{binding}
in an \spadgloss{environment}.
\item\newline{\em \menuitemstyle{}}{\em variable}\space{}
a means of referring to an object but itself {\spad{\bf} not} an object. A
variable has a name and an associated \spadgloss{binding} created by
\spadgloss{evaluation} of Axiom expressions such as
\spadgloss{declarations},{} \spadgloss{assignments},{} and
\spadgloss{definitions}. In the top-level \spadgloss{environment} of the
interpreter,{} variables are \spadgloss{global variables}. Such variables
can be freely referenced in user-defined functions although a
\spadgloss{free} declaration is needed to assign values to them.
See \spadgloss{local variable} for details.
\item\newline{\em \menuitemstyle{}}{\em Void}\space{}
the type given when the \spadgloss{value} and \spadgloss{type} of an
expression are not needed. Also used when there is no guarantee at
run-time that a value and predictable mode will result.
\item\newline{\em \menuitemstyle{}}{\em wild card}\space{}
a symbol which matches any substring including the empty string; for
example,{} the search string {\em *an*} matches an word containing the
consecutive letters {\em a} and {\em n}
\item\newline{\em \menuitemstyle{}}{\em workspace}\space{}
an interactive record of the user input and output held in an interactive
history file. Each user input and corresponding output expression in the
workspace has a corresponding \spadgloss{step number}. The current output
expression in the workspace is referred to as \spad{\%}. The output
expression associated with step number \spad{n} is referred to by
\spad{\%\%(n)}. The \spad{k}-th previous output expression relative
to the current step number \spad{n} is referred to by \spad{\%\%(- k)}.
Each interactive \spadgloss{frame} has its own workspace.
\endmenu\endscroll\lispdownlink{Search}{(|htGloss| "\stringvalue{pattern}")}
for glossary entry matching \inputstring{pattern}{24}{*}\end{page}
```

————

3.50 graphics.ht

Graphics



⇐ “Topics” (TopicPage) 3.108 on page 1313
⇒ “Examples” (GraphicsExamplePage) 3.50 on page 602
⇒ “2D Graphics” (TwoDimensionalGraphicsPage) 3.50 on page 643
⇒ “3D Graphics” (ThreeDimensionalGraphicsPage) 3.50 on page 628
⇒ “Viewports” (ViewportPage) 3.50 on page 662

— **graphics.ht** —

```
\begin{page}{GraphicsPage}{Graphics}
Axiom can plot curves and surfaces of
various types, as well as lists of points in the plane.
% sequences, and complex functions.
\beginscroll
\beginmenu
\menulink{Examples}{GraphicsExamplePage} \tab{13}
See examples of Axiom graphics
\menulink{2D Graphics}{TwoDimensionalGraphicsPage} \tab{13}
Graphics in the real and complex plane
\menulink{3D Graphics}{ThreeDimensionalGraphicsPage} \tab{13}
Plot surfaces, curves, or tubes around curves
\menulink{Viewports}{ViewportPage} \tab{13}
Customize graphics using Viewports
\endmenu
```

```
\endscroll
\autobuttons \end{page}
```

———-

Graphics Examples

⇒ "notitle" (AssortedGraphicsExamplePage) 3.50 on page 603
⇒ "notitle" (ThreeDimensionalGraphicsExamplePage) 3.50 on page 605
⇒ "notitle" (OneVariableGraphicsExamplePage) 3.50 on page 610
⇒ "notitle" (PolarGraphicsExamplePage) 3.50 on page 614
⇒ "notitle" (ParametricCurveGraphicsExamplePage) 3.50 on page 612
⇒ "notitle" (ImplicitCurveGraphicsExamplePage) 3.50 on page 616
⇒ "notitle" (ListPointsGraphicsExamplePage) 3.50 on page 619

— **graphics.ht** —

```
\begin{page}{GraphicsExamplePage}{Graphics Examples}
\beginscroll
Here are some examples of Axiom graphics.
Choose a specific type of graph or choose Assorted Examples.
\beginmenu
\menulink{Assorted Examples}{AssortedGraphicsExamplePage} \newline
Examples of each type of Axiom graphics.
\menulink{Three Dimensional Graphics}
{ThreeDimensionalGraphicsExamplePage} \newline
Plot parametrically defined surfaces of three functions.
\menulink{Functions of One Variable}{OneVariableGraphicsExamplePage}
\newline
Plot curves defined by an equation y = f(x).
\menulink{Parametric Curves}{ParametricCurveGraphicsExamplePage} \newline
Plot curves defined by parametric equations x = f(t), y = g(t).
\menulink{Polar Coordinates}{PolarGraphicsExamplePage} \newline
Plot curves given in polar form by an equation r = f(theta).
\menulink{Implicit Curves}{ImplicitCurveGraphicsExamplePage} \newline
Plot non-singular curves defined by a polynomial equation
\menulink{Lists of Points}{ListPointsGraphicsExamplePage} \newline
Plot lists of points in the (x,y)-plane.
% \menulink{Sequences}{SequenceGraphicsExamplePage}
% Plot a sequence a1, a2, a3,...
% \menulink{Complex Functions}{ComplexFunctionGraphicsExamplePage}
% Plot complex functions of a complex variable by means of grid plots.
\endmenu
\endscroll
\autobuttons \end{page}
```

Assorted Graphics Examples

— graphics.ht —

```
\begin{page}{AssortedGraphicsExamplePage}{Assorted Graphics Examples}
\beginscroll
Pick a specific example or choose 'All' to see all the examples.\newline
Function of two variables: z = f(x,y).
\graphpaste{draw(sin(x * y), x = -2.5..2.5, y = -2.5..2.5) \bound{example1}}
Function of one variable: y = f(x).
\graphpaste{draw(sin tan x - tan sin x,x = 0..6) \bound{example2} }
Plane parametric curve: x = f(t), y = g(t).
\graphpaste{draw(curve(sin(t)*sin(2*t), sin(3*t)*sin(4*t)), t = 0..2*\%pi)
\bound{example3}}
Space parametric curve: x = f(t), y = g(t), z = h(t).
\graphpaste{draw(curve(sin(t)*sin(2*t), sin(3*t)*sin(4*t),
sin(5*t)*sin(6*t)), t = 0..2*\%pi) \bound{example4}}
Polar coordinates: r = f(theta).
\graphpaste{draw(sin(17*t), t = 0..2*\%pi, coordinates == polar)
\bound{example5}}
Implicit curves: p(x,y) = 0.
\graphpaste{draw(y**2 + y = x**3 - x, x, y,range == \[-2..2,-2..1\])
\bound{example6}}
Run all examples.
\spadpaste{All \free{example1 example2 example3 example4 example5 example6}}
\endscroll
\autobuttons
\end{page}

\begin{patch}{AssortedGraphicsExamplePagePatch1}
\begin{paste}{AssortedGraphicsExamplePageFull1}{AssortedGraphicsExamplePageEmpty1}
\pastebutton{AssortedGraphicsExamplePageFull1}{\hidepaste}
\tab{5}\spadgraph{draw(sin(x * y), x = -2.5..2.5, y = -2.5..2.5)\bound{example1 }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/assortedgraphicsexamplepage1.view/image}}{viewalc
\end{paste}\end{patch}

\begin{patch}{AssortedGraphicsExamplePageEmpty1}
\begin{paste}{AssortedGraphicsExamplePageEmpty1}{AssortedGraphicsExamplePagePatch1}
\pastebutton{AssortedGraphicsExamplePageEmpty1}{\showpaste}
\tab{5}\spadgraph{draw(sin(x * y), x = -2.5..2.5, y = -2.5..2.5)\bound{example1 }}
\end{paste}\end{patch}

\begin{patch}{AssortedGraphicsExamplePagePatch2}
\begin{paste}{AssortedGraphicsExamplePageFull2}{AssortedGraphicsExamplePageEmpty2}
\pastebutton{AssortedGraphicsExamplePageFull2}{\hidepaste}
\tab{5}\spadgraph{draw(sin tan x - tan sin x,x = 0..6)\bound{example2 }}
```

```
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/assortedgraphicsexamplepage2.view/ima
\end{paste}\end{patch}

\begin{patch}{AssortedGraphicsExamplePageEmpty2}
\begin{paste}{AssortedGraphicsExamplePageEmpty2}{AssortedGraphicsExamplePagePatch2}
\pastebutton{AssortedGraphicsExamplePageEmpty2}{\showpaste}
\tab{5}\spadgraph{draw(sin tan x - tan sin x,x = 0..6)\bound{example2 }}
\end{paste}\end{patch}

\begin{patch}{AssortedGraphicsExamplePagePatch3}
\begin{paste}{AssortedGraphicsExamplePageFull3}{AssortedGraphicsExamplePageEmpty3}
\pastebutton{AssortedGraphicsExamplePageFull3}{\hidepaste}
\tab{5}\spadgraph{draw(curve(sin(t)*sin(2*t), sin(3*t)*sin(4*t)), t = 0..2*\%pi)\bound{example3
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/assortedgraphicsexamplepage3.view/ima
\end{paste}\end{patch}

\begin{patch}{AssortedGraphicsExamplePageEmpty3}
\begin{paste}{AssortedGraphicsExamplePageEmpty3}{AssortedGraphicsExamplePagePatch3}
\pastebutton{AssortedGraphicsExamplePageEmpty3}{\showpaste}
\tab{5}\spadgraph{draw(curve(sin(t)*sin(2*t), sin(3*t)*sin(4*t)), t = 0..2*\%pi)\bound{example3
\end{paste}\end{patch}

\begin{patch}{AssortedGraphicsExamplePagePatch4}
\begin{paste}{AssortedGraphicsExamplePageFull4}{AssortedGraphicsExamplePageEmpty4}
\pastebutton{AssortedGraphicsExamplePageFull4}{\hidepaste}
\tab{5}\spadgraph{draw(curve(sin(t)*sin(2*t), sin(3*t)*sin(4*t), sin(5*t)*sin(6*t)), t = 0..2*\%
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/assortedgraphicsexamplepage4.view/ima
\end{paste}\end{patch}

\begin{patch}{AssortedGraphicsExamplePageEmpty4}
\begin{paste}{AssortedGraphicsExamplePageEmpty4}{AssortedGraphicsExamplePagePatch4}
\pastebutton{AssortedGraphicsExamplePageEmpty4}{\showpaste}
\tab{5}\spadgraph{draw(curve(sin(t)*sin(2*t), sin(3*t)*sin(4*t), sin(5*t)*sin(6*t)), t = 0..2*\%
\end{paste}\end{patch}

\begin{patch}{AssortedGraphicsExamplePagePatch5}
\begin{paste}{AssortedGraphicsExamplePageFull5}{AssortedGraphicsExamplePageEmpty5}
\pastebutton{AssortedGraphicsExamplePageFull5}{\hidepaste}
\tab{5}\spadgraph{draw(sin(17*t), t = 0..2*\%pi, coordinates == polar)\bound{example5 }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/assortedgraphicsexamplepage5.view/ima
\end{paste}\end{patch}

\begin{patch}{AssortedGraphicsExamplePageEmpty5}
\begin{paste}{AssortedGraphicsExamplePageEmpty5}{AssortedGraphicsExamplePagePatch5}
\pastebutton{AssortedGraphicsExamplePageEmpty5}{\showpaste}
\tab{5}\spadgraph{draw(sin(17*t), t = 0..2*\%pi, coordinates == polar)\bound{example5 }}
\end{paste}\end{patch}

\begin{patch}{AssortedGraphicsExamplePagePatch6}
\begin{paste}{AssortedGraphicsExamplePageFull6}{AssortedGraphicsExamplePageEmpty6}
```

```
\pastebutton{AssortedGraphicsExamplePageFull6}{\hidepaste}
\tab{5}\spadgraph{draw(y**2 + y = x**3 - x, x, y,range == [-2..2,-2..1])\bound{example6 }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/assortedgraphicsexamplepage6.view/image}}{viewal
\end{paste}\end{patch}

\begin{patch}{AssortedGraphicsExamplePageEmpty6}
\begin{paste}{AssortedGraphicsExamplePageEmpty6}{AssortedGraphicsExamplePagePatch6}
\pastebutton{AssortedGraphicsExamplePageEmpty6}{\showpaste}
\tab{5}\spadgraph{draw(y**2 + y = x**3 - x, x, y,range == [-2..2,-2..1])\bound{example6 }}
\end{paste}\end{patch}

\begin{patch}{AssortedGraphicsExamplePagePatch7}
\begin{paste}{AssortedGraphicsExamplePageFull7}{AssortedGraphicsExamplePageEmpty7}
\pastebutton{AssortedGraphicsExamplePageFull7}{\hidepaste}
\tab{5}\spadcommand{All\free{example1 example2 example3 example4 example5 example6 }}
\indentrel{3}\begin{verbatim}
   (7)  All
                                     Type: Variable All
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{AssortedGraphicsExamplePageEmpty7}
\begin{paste}{AssortedGraphicsExamplePageEmpty7}{AssortedGraphicsExamplePagePatch7}
\pastebutton{AssortedGraphicsExamplePageEmpty7}{\showpaste}
\tab{5}\spadcommand{All\free{example1 example2 example3 example4 example5 example6 }}
\end{paste}\end{patch}
```

———-

Three Dimensional Graphics

— graphics.ht —

```
\begin{page}{ThreeDimensionalGraphicsExamplePage}
{Three Dimensional Graphics}
Plots of parametric surfaces defined by functions f(u,v), g(u,v),
and h(u,v). Choose a particular example or choose 'All' to see
all the examples.
\beginscroll
Pear Surface.
\graphpaste{draw(surface((1+exp(-100*u*u))*sin(\%pi*u)*sin(\%pi*v),
(1+exp(-100*u*u))*sin(\%pi*u)*cos(\%pi*v),
(1+exp(-100*u*u))*cos(\%pi*u)), u=0..1, v=0..2, title=="Pear")
\bound{example1}}
Trigonometric Screw.
\graphpaste{draw(surface(x*cos(y),x*sin(y),y*cos(x)),
x=-4..4, y=0..2*\%pi, var1Steps==40, var2Steps==40,
```

```
title=="Trig Screw") \bound{example2}}
Etruscan Venus. \newline
(click on the draw button to execute this example)
\spadpaste{a := 1.3 * cos(2*x) * cos(y) + sin(y) * cos(x)\bound{a}}
\newline
\spadpaste{b := 1.3 * sin(2*x) * cos(y) - sin(y) * sin(x)\bound{b}}
\newline
\spadpaste{c := 2.5 * cos(y) \bound{c}}
\newline
\graphpaste{draw(surface(a,b,c), x=0..\%pi, y=-\%pi..\%pi,
var1Steps==40, var2Steps==40, title=="Etruscan Venus")
\free{a b c} \bound{example3}}
Banchoff Klein Bottle. \newline
(click on the draw button to execute this example)
\spadpaste{f:=cos(x)*(cos(x/2)*(sqrt(2) + cos(y))+
(sin(x/2)*sin(y)*cos(y)))\bound{f}}
\newline
\spadpaste{g:=sin(x)*(cos(x/2)*(sqrt(2) + cos(y))+
(sin(x/2)*sin(y)*cos(y)))\bound{g}}
\newline
\spadpaste{h:=-sin(x/2)*(sqrt(2)+cos(y)) +
cos(x/2)*sin(y)*cos(y) \bound{h}}
\newline
\graphpaste{draw(surface(f,g,h), x=0..4*\%pi,
y=0..2*\%pi, var1Steps==50, var2Steps==50,
title=="Banchoff Klein Bottle") \free{f g h} \bound{example4}}
\newline
\spadpaste{All \free{example1 example2 example3 example4}}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ThreeDimensionalGraphicsExamplePagePatch1}
\begin{paste}{ThreeDimensionalGraphicsExamplePageFull1}{ThreeDimensionalGraphicsExamplePageEmpty
\pastebutton{ThreeDimensionalGraphicsExamplePageFull1}{\hidepaste}
\tab{5}\spadgraph{draw(surface((1+exp(-100*u*u))*sin(\%pi*u)*sin(\%pi*v), (1+exp(-100*u*u))*sin(
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/threedimensionalgraphicsexamplepage1.
\end{paste}\end{patch}

\begin{patch}{ThreeDimensionalGraphicsExamplePageEmpty1}
\begin{paste}{ThreeDimensionalGraphicsExamplePageEmpty1}{ThreeDimensionalGraphicsExamplePagePatc
\pastebutton{ThreeDimensionalGraphicsExamplePageEmpty1}{\showpaste}
\tab{5}\spadgraph{draw(surface((1+exp(-100*u*u))*sin(\%pi*u)*sin(\%pi*v), (1+exp(-100*u*u))*sin(
\end{paste}\end{patch}

\begin{patch}{ThreeDimensionalGraphicsExamplePagePatch2}
\begin{paste}{ThreeDimensionalGraphicsExamplePageFull2}{ThreeDimensionalGraphicsExamplePageEmpty
\pastebutton{ThreeDimensionalGraphicsExamplePageFull2}{\hidepaste}
\tab{5}\spadgraph{draw(surface(x*cos(y),x*sin(y),y*cos(x)), x=-4..4, y=0..2*\%pi, var1Steps==40,
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/threedimensionalgraphicsexamplepage2.
```

```
\end{paste}\end{patch}

\begin{patch}{ThreeDimensionalGraphicsExamplePageEmpty2}
\begin{paste}{ThreeDimensionalGraphicsExamplePageEmpty2}{ThreeDimensionalGraphicsExamplePagePatch2}
\pastebutton{ThreeDimensionalGraphicsExamplePageEmpty2}{\showpaste}
\tab{5}\spadgraph{draw(surface(x*cos(y),x*sin(y),y*cos(x)), x=-4..4, y=0..2*\%pi, var1Steps==40, var2Steps==
\end{paste}\end{patch}

\begin{patch}{ThreeDimensionalGraphicsExamplePagePatch3}
\begin{paste}{ThreeDimensionalGraphicsExamplePageFull3}{ThreeDimensionalGraphicsExamplePageEmpty3}
\pastebutton{ThreeDimensionalGraphicsExamplePageFull3}{\hidepaste}
\tab{5}\spadcommand{a := 1.3 * cos(2*x) * cos(y) + sin(y) * cos(x)\bound{a }}
\indentrel{3}\begin{verbatim}
   (3)  cos(x)sin(y) + 1.3 cos(2.0 x)cos(y)
                                 Type: Expression Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ThreeDimensionalGraphicsExamplePageEmpty3}
\begin{paste}{ThreeDimensionalGraphicsExamplePageEmpty3}{ThreeDimensionalGraphicsExamplePagePatch3}
\pastebutton{ThreeDimensionalGraphicsExamplePageEmpty3}{\showpaste}
\tab{5}\spadcommand{a := 1.3 * cos(2*x) * cos(y) + sin(y) * cos(x)\bound{a }}
\end{paste}\end{patch}

\begin{patch}{ThreeDimensionalGraphicsExamplePagePatch4}
\begin{paste}{ThreeDimensionalGraphicsExamplePageFull4}{ThreeDimensionalGraphicsExamplePageEmpty4}
\pastebutton{ThreeDimensionalGraphicsExamplePageFull4}{\hidepaste}
\tab{5}\spadcommand{b := 1.3 * sin(2*x) * cos(y) - sin(y) * sin(x)\bound{b }}
\indentrel{3}\begin{verbatim}
   (4)  - 1.0 sin(x)sin(y) + 1.3 cos(y)sin(2.0 x)
                                 Type: Expression Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ThreeDimensionalGraphicsExamplePageEmpty4}
\begin{paste}{ThreeDimensionalGraphicsExamplePageEmpty4}{ThreeDimensionalGraphicsExamplePagePatch4}
\pastebutton{ThreeDimensionalGraphicsExamplePageEmpty4}{\showpaste}
\tab{5}\spadcommand{b := 1.3 * sin(2*x) * cos(y) - sin(y) * sin(x)\bound{b }}
\end{paste}\end{patch}

\begin{patch}{ThreeDimensionalGraphicsExamplePagePatch5}
\begin{paste}{ThreeDimensionalGraphicsExamplePageFull5}{ThreeDimensionalGraphicsExamplePageEmpty5}
\pastebutton{ThreeDimensionalGraphicsExamplePageFull5}{\hidepaste}
\tab{5}\spadcommand{c := 2.5 * cos(y)\bound{c }}
\indentrel{3}\begin{verbatim}
   (5)  2.5 cos(y)
                                 Type: Expression Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ThreeDimensionalGraphicsExamplePageEmpty5}
\begin{paste}{ThreeDimensionalGraphicsExamplePageEmpty5}{ThreeDimensionalGraphicsExamplePagePatc
\pastebutton{ThreeDimensionalGraphicsExamplePageEmpty5}{\showpaste}
\tab{5}\spadcommand{c := 2.5 * cos(y)\bound{c }}
\end{paste}\end{patch}

\begin{patch}{ThreeDimensionalGraphicsExamplePagePatch6}
\begin{paste}{ThreeDimensionalGraphicsExamplePageFull6}{ThreeDimensionalGraphicsExamplePageEmpty
\pastebutton{ThreeDimensionalGraphicsExamplePageFull6}{\hidepaste}
\tab{5}\spadgraph{draw(surface(a,b,c), x=0..\%pi, y=-\%pi..\%pi, var1Steps==40, var2Steps==40, t
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/threedimensionalgraphicsexamplepage6.
\end{paste}\end{patch}

\begin{patch}{ThreeDimensionalGraphicsExamplePageEmpty6}
\begin{paste}{ThreeDimensionalGraphicsExamplePageEmpty6}{ThreeDimensionalGraphicsExamplePagePatc
\pastebutton{ThreeDimensionalGraphicsExamplePageEmpty6}{\showpaste}
\tab{5}\spadgraph{draw(surface(a,b,c), x=0..\%pi, y=-\%pi..\%pi, var1Steps==40, var2Steps==40, t
\end{paste}\end{patch}

\begin{patch}{ThreeDimensionalGraphicsExamplePagePatch7}
\begin{paste}{ThreeDimensionalGraphicsExamplePageFull7}{ThreeDimensionalGraphicsExamplePageEmpty
\pastebutton{ThreeDimensionalGraphicsExamplePageFull7}{\hidepaste}
\tab{5}\spadcommand{f:=cos(x)*(cos(x/2)*(sqrt(2) + cos(y))+(sin(x/2)*sin(y)*cos(y)))\bound{f }}
\indentrel{3}\begin{verbatim}
   (7)
                     x              x
     cos(x)cos(y)sin()sin(y) + cos()cos(x)cos(y)
                     2              2
   +
          x
     \2 cos()cos(x)
             2
                                 Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ThreeDimensionalGraphicsExamplePageEmpty7}
\begin{paste}{ThreeDimensionalGraphicsExamplePageEmpty7}{ThreeDimensionalGraphicsExamplePagePatc
\pastebutton{ThreeDimensionalGraphicsExamplePageEmpty7}{\showpaste}
\tab{5}\spadcommand{f:=cos(x)*(cos(x/2)*(sqrt(2) + cos(y))+(sin(x/2)*sin(y)*cos(y)))\bound{f }}
\end{paste}\end{patch}

\begin{patch}{ThreeDimensionalGraphicsExamplePagePatch8}
\begin{paste}{ThreeDimensionalGraphicsExamplePageFull8}{ThreeDimensionalGraphicsExamplePageEmpty
\pastebutton{ThreeDimensionalGraphicsExamplePageFull8}{\hidepaste}
\tab{5}\spadcommand{g:=sin(x)*(cos(x/2)*(sqrt(2) + cos(y))+(sin(x/2)*sin(y)*cos(y)))\bound{g }}
\indentrel{3}\begin{verbatim}
   (8)
                x
     cos(y)sin()sin(x)sin(y)
```

```
             2
  +
        x              x
     (cos()cos(y) + \2 cos())sin(x)
        2                 2
                              Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ThreeDimensionalGraphicsExamplePageEmpty8}
\begin{paste}{ThreeDimensionalGraphicsExamplePageEmpty8}{ThreeDimensionalGraphicsExamplePagePatch8}
\pastebutton{ThreeDimensionalGraphicsExamplePageEmpty8}{\showpaste}
\tab{5}\spadcommand{g:=sin(x)*(cos(x/2)*(sqrt(2) + cos(y))+(sin(x/2)*sin(y)*cos(y)))\bound{g }}
\end{paste}\end{patch}

\begin{patch}{ThreeDimensionalGraphicsExamplePagePatch9}
\begin{paste}{ThreeDimensionalGraphicsExamplePageFull9}{ThreeDimensionalGraphicsExamplePageEmpty9}
\pastebutton{ThreeDimensionalGraphicsExamplePageFull9}{\hidepaste}
\tab{5}\spadcommand{h:=-sin(x/2)*(sqrt(2)+cos(y)) + cos(x/2)*sin(y)*cos(y)\bound{h }}
\indentrel{3}\begin{verbatim}
          x                                x
   (9)  cos()cos(y)sin(y) + (- cos(y) - \2 )sin()
          2                                   2
                              Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ThreeDimensionalGraphicsExamplePageEmpty9}
\begin{paste}{ThreeDimensionalGraphicsExamplePageEmpty9}{ThreeDimensionalGraphicsExamplePagePatch9}
\pastebutton{ThreeDimensionalGraphicsExamplePageEmpty9}{\showpaste}
\tab{5}\spadcommand{h:=-sin(x/2)*(sqrt(2)+cos(y)) + cos(x/2)*sin(y)*cos(y)\bound{h }}
\end{paste}\end{patch}

\begin{patch}{ThreeDimensionalGraphicsExamplePagePatch10}
\begin{paste}{ThreeDimensionalGraphicsExamplePageFull10}{ThreeDimensionalGraphicsExamplePageEmpty10}
\pastebutton{ThreeDimensionalGraphicsExamplePageFull10}{\hidepaste}
\tab{5}\spadgraph{draw(surface(f,g,h), x=0..4*\%pi, y=0..2*\%pi, var1Steps==50, var2Steps==50, title=="Banc
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/threedimensionalgraphicsexamplepage10.view/image
\end{paste}\end{patch}

\begin{patch}{ThreeDimensionalGraphicsExamplePageEmpty10}
\begin{paste}{ThreeDimensionalGraphicsExamplePageEmpty10}{ThreeDimensionalGraphicsExamplePagePatch10}
\pastebutton{ThreeDimensionalGraphicsExamplePageEmpty10}{\showpaste}
\tab{5}\spadgraph{draw(surface(f,g,h), x=0..4*\%pi, y=0..2*\%pi, var1Steps==50, var2Steps==50, title=="Banc
\end{paste}\end{patch}

\begin{patch}{ThreeDimensionalGraphicsExamplePagePatch11}
\begin{paste}{ThreeDimensionalGraphicsExamplePageFull11}{ThreeDimensionalGraphicsExamplePageEmpty11}
\pastebutton{ThreeDimensionalGraphicsExamplePageFull11}{\hidepaste}
\tab{5}\spadcommand{All\free{example1 example2 example3 example4 }}
```

```
\indentrel{3}\begin{verbatim}
   (11)  All
                                     Type: Variable All
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ThreeDimensionalGraphicsExamplePageEmpty11}
\begin{paste}{ThreeDimensionalGraphicsExamplePageEmpty11}{ThreeDimensionalGraphicsExamplePagePat
\pastebutton{ThreeDimensionalGraphicsExamplePageEmpty11}{\showpaste}
\tab{5}\spadcommand{All\free{example1 example2 example3 example4 }}
\end{paste}\end{patch}
```

————-

Functions of One Variable

— graphics.ht —

```
\begin{page}{OneVariableGraphicsExamplePage}{Functions of One Variable}
\beginscroll
Plots of functions y = f(x).
Choose a particular example or choose 'All' to see all the examples.
\graphpaste{draw(sin tan x - tan sin x, x = 0..6) \bound{example1}}
\newline
\graphpaste{draw(sin x + cos x, x = 0..2*\%pi) \bound{example2}}
\newline
\graphpaste{draw(sin(1/x), x = -1..1) \bound{example3}}
\newline
\graphpaste{draw(x * sin(1/x), x = -1..1) \bound{example4}}
\newline
\spadpaste{All \free{example1 example2 example3 example4}}
\endscroll
\autobuttons
\end{page}

\begin{patch}{OneVariableGraphicsExamplePagePatch1}
\begin{paste}{OneVariableGraphicsExamplePageFull1}{OneVariableGraphicsExamplePageEmpty1}
\pastebutton{OneVariableGraphicsExamplePageFull1}{\hidepaste}
\tab{5}\spadgraph{draw(sin tan x - tan sin x, x = 0..6)\bound{example1 }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/onevariablegraphicsexamplepage1.view/
\end{paste}\end{patch}

\begin{patch}{OneVariableGraphicsExamplePageEmpty1}
\begin{paste}{OneVariableGraphicsExamplePageEmpty1}{OneVariableGraphicsExamplePagePatch1}
\pastebutton{OneVariableGraphicsExamplePageEmpty1}{\showpaste}
\tab{5}\spadgraph{draw(sin tan x - tan sin x, x = 0..6)\bound{example1 }}
\end{paste}\end{patch}
```

```
\begin{patch}{OneVariableGraphicsExamplePagePatch2}
\begin{paste}{OneVariableGraphicsExamplePageFull2}{OneVariableGraphicsExamplePageEmpty2}
\pastebutton{OneVariableGraphicsExamplePageFull2}{\hidepaste}
\tab{5}\spadgraph{draw(sin x + cos x, x = 0..2*\%pi)\bound{example2 }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/onevariablegraphicsexamplepage2.view/image}}{view
\end{paste}\end{patch}

\begin{patch}{OneVariableGraphicsExamplePageEmpty2}
\begin{paste}{OneVariableGraphicsExamplePageEmpty2}{OneVariableGraphicsExamplePagePatch2}
\pastebutton{OneVariableGraphicsExamplePageEmpty2}{\showpaste}
\tab{5}\spadgraph{draw(sin x + cos x, x = 0..2*\%pi)\bound{example2 }}
\end{paste}\end{patch}

\begin{patch}{OneVariableGraphicsExamplePagePatch3}
\begin{paste}{OneVariableGraphicsExamplePageFull3}{OneVariableGraphicsExamplePageEmpty3}
\pastebutton{OneVariableGraphicsExamplePageFull3}{\hidepaste}
\tab{5}\spadgraph{draw(sin(1/x), x = -1..1)\bound{example3 }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/onevariablegraphicsexamplepage3.view/image}}{view
\end{paste}\end{patch}

\begin{patch}{OneVariableGraphicsExamplePageEmpty3}
\begin{paste}{OneVariableGraphicsExamplePageEmpty3}{OneVariableGraphicsExamplePagePatch3}
\pastebutton{OneVariableGraphicsExamplePageEmpty3}{\showpaste}
\tab{5}\spadgraph{draw(sin(1/x), x = -1..1)\bound{example3 }}
\end{paste}\end{patch}

\begin{patch}{OneVariableGraphicsExamplePagePatch4}
\begin{paste}{OneVariableGraphicsExamplePageFull4}{OneVariableGraphicsExamplePageEmpty4}
\pastebutton{OneVariableGraphicsExamplePageFull4}{\hidepaste}
\tab{5}\spadgraph{draw(x * sin(1/x), x = -1..1)\bound{example4 }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/onevariablegraphicsexamplepage4.view/image}}{view
\end{paste}\end{patch}

\begin{patch}{OneVariableGraphicsExamplePageEmpty4}
\begin{paste}{OneVariableGraphicsExamplePageEmpty4}{OneVariableGraphicsExamplePagePatch4}
\pastebutton{OneVariableGraphicsExamplePageEmpty4}{\showpaste}
\tab{5}\spadgraph{draw(x * sin(1/x), x = -1..1)\bound{example4 }}
\end{paste}\end{patch}

\begin{patch}{OneVariableGraphicsExamplePagePatch5}
\begin{paste}{OneVariableGraphicsExamplePageFull5}{OneVariableGraphicsExamplePageEmpty5}
\pastebutton{OneVariableGraphicsExamplePageFull5}{\hidepaste}
\tab{5}\spadcommand{All\free{example1 example2 example3 example4 }}
\indentrel{3}\begin{verbatim}
   (5)  All
                                     Type: Variable All
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{OneVariableGraphicsExamplePageEmpty5}
\begin{paste}{OneVariableGraphicsExamplePageEmpty5}{OneVariableGraphicsExamplePagePatch5}
\pastebutton{OneVariableGraphicsExamplePageEmpty5}{\showpaste}
\tab{5}\spadcommand{All\free{example1 example2 example3 example4 }}
\end{paste}\end{patch}
```

———-

Parametric Curves

— graphics.ht —

```
\begin{page}{ParametricCurveGraphicsExamplePage}{Parametric Curves}
Plots of parametric curves x = f(t), y = g(t).
Pick a particular example or choose 'All' to see all the examples.
\beginscroll
The Lemniscate of Bernoulli.
\graphpaste{draw(curve(cos(t/(1+sin(t)**2)),
sin(t)*cos(t)/(1+sin(t)**2)), t = -\%pi..\%pi) \bound{example1}}
Lissajous curve.
\graphpaste{draw(curve(9*sin(3*t/4), 8*sin(t)),
t = -4*\%pi..4*\%pi) \bound{example2}}
A gnarly closed curve.
\graphpaste{draw(curve(sin(t)*sin(2*t)*sin(3*t),
sin(4*t)*sin(5*t)*sin(6*t)),t = 0..2*\%pi)
             \bound{example3}}
Another closed curve.
\graphpaste{draw(curve(cos(4*t)*cos(7*t), cos(4*t)*sin(7*t)),
t = 0..2*\%pi) \bound{example4}}
Run all examples on this page.
\spadpaste{All \free{example1 example2 example3 example4}}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ParametricCurveGraphicsExamplePagePatch1}
\begin{paste}{ParametricCurveGraphicsExamplePageFull1}{ParametricCurveGraphicsExamplePageEmpty1}
\pastebutton{ParametricCurveGraphicsExamplePageFull1}{\hidepaste}
\tab{5}\spadgraph{draw(curve(cos(t/(1+sin(t)**2)), sin(t)*cos(t)/(1+sin(t)**2)), t = -\%pi..\%pi
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/parametriccurvegraphicsexamplepage1.v
\end{paste}\end{patch}

\begin{patch}{ParametricCurveGraphicsExamplePageEmpty1}
\begin{paste}{ParametricCurveGraphicsExamplePageEmpty1}{ParametricCurveGraphicsExamplePagePatch1
\pastebutton{ParametricCurveGraphicsExamplePageEmpty1}{\showpaste}
\tab{5}\spadgraph{draw(curve(cos(t/(1+sin(t)**2)), sin(t)*cos(t)/(1+sin(t)**2)), t = -\%pi..\%pi
\end{paste}\end{patch}
```

```
\begin{patch}{ParametricCurveGraphicsExamplePagePatch2}
\begin{paste}{ParametricCurveGraphicsExamplePageFull2}{ParametricCurveGraphicsExamplePageEmpty2}
\pastebutton{ParametricCurveGraphicsExamplePageFull2}{\hidepaste}
\tab{5}\spadgraph{draw(curve(9*sin(3*t/4), 8*sin(t)), t = -4*\%pi..4*\%pi)\bound{example2 }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/parametriccurvegraphicsexamplepage2.view/image}}
\end{paste}\end{patch}

\begin{patch}{ParametricCurveGraphicsExamplePageEmpty2}
\begin{paste}{ParametricCurveGraphicsExamplePageEmpty2}{ParametricCurveGraphicsExamplePagePatch2}
\pastebutton{ParametricCurveGraphicsExamplePageEmpty2}{\showpaste}
\tab{5}\spadgraph{draw(curve(9*sin(3*t/4), 8*sin(t)), t = -4*\%pi..4*\%pi)\bound{example2 }}
\end{paste}\end{patch}

\begin{patch}{ParametricCurveGraphicsExamplePagePatch3}
\begin{paste}{ParametricCurveGraphicsExamplePageFull3}{ParametricCurveGraphicsExamplePageEmpty3}
\pastebutton{ParametricCurveGraphicsExamplePageFull3}{\hidepaste}
\tab{5}\spadgraph{draw(curve(sin(t)*sin(2*t)*sin(3*t), sin(4*t)*sin(5*t)*sin(6*t)),t = 0..2*\%pi)\bound{exam
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/parametriccurvegraphicsexamplepage3.view/image}}
\end{paste}\end{patch}

\begin{patch}{ParametricCurveGraphicsExamplePageEmpty3}
\begin{paste}{ParametricCurveGraphicsExamplePageEmpty3}{ParametricCurveGraphicsExamplePagePatch3}
\pastebutton{ParametricCurveGraphicsExamplePageEmpty3}{\showpaste}
\tab{5}\spadgraph{draw(curve(sin(t)*sin(2*t)*sin(3*t), sin(4*t)*sin(5*t)*sin(6*t)),t = 0..2*\%pi)\bound{exam
\end{paste}\end{patch}

\begin{patch}{ParametricCurveGraphicsExamplePagePatch4}
\begin{paste}{ParametricCurveGraphicsExamplePageFull4}{ParametricCurveGraphicsExamplePageEmpty4}
\pastebutton{ParametricCurveGraphicsExamplePageFull4}{\hidepaste}
\tab{5}\spadgraph{draw(curve(cos(4*t)*cos(7*t), cos(4*t)*sin(7*t)), t = 0..2*\%pi)\bound{example4 }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/parametriccurvegraphicsexamplepage4.view/image}}
\end{paste}\end{patch}

\begin{patch}{ParametricCurveGraphicsExamplePageEmpty4}
\begin{paste}{ParametricCurveGraphicsExamplePageEmpty4}{ParametricCurveGraphicsExamplePagePatch4}
\pastebutton{ParametricCurveGraphicsExamplePageEmpty4}{\showpaste}
\tab{5}\spadgraph{draw(curve(cos(4*t)*cos(7*t), cos(4*t)*sin(7*t)), t = 0..2*\%pi)\bound{example4 }}
\end{paste}\end{patch}

\begin{patch}{ParametricCurveGraphicsExamplePagePatch5}
\begin{paste}{ParametricCurveGraphicsExamplePageFull5}{ParametricCurveGraphicsExamplePageEmpty5}
\pastebutton{ParametricCurveGraphicsExamplePageFull5}{\hidepaste}
\tab{5}\spadcommand{All\free{example1 example2 example3 example4 }}
\indentrel{3}\begin{verbatim}
   (5)  All
                                    Type: Variable All
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ParametricCurveGraphicsExamplePageEmpty5}
\begin{paste}{ParametricCurveGraphicsExamplePageEmpty5}{ParametricCurveGraphicsExamplePagePatch5
\pastebutton{ParametricCurveGraphicsExamplePageEmpty5}{\showpaste}
\tab{5}\spadcommand{All\free{example1 example2 example3 example4 }}
\end{paste}\end{patch}
```

———-

Polar Coordinates

— **graphics.ht** —

```
\begin{page}{PolarGraphicsExamplePage}{Polar Coordinates}
Plots of curves given by an equation in polar coordinates, r = f(theta).
Pick a particular example or choose 'All' to see all the examples.
\beginscroll
A Circle.
\graphpaste{draw(1,t = 0..2*\%pi, coordinates == polar) \bound{example1} }
A Spiral.
\graphpaste{draw(t,t = 0..100, coordinates == polar) \bound{example2} }
A Petal Curve.
\graphpaste{draw(sin(4*t), t = 0..2*\%pi, coordinates == polar)
\bound{example3} }
A Limacon.
\graphpaste{draw(2 + 3 * sin t, t = 0..2*\%pi, coordinates == polar)
\bound{example4} }
Run all examples on this page.
\spadpaste{All \free{
%example1
example2 example3 example4}}
\endscroll
\autobuttons
\end{page}

\begin{patch}{PolarGraphicsExamplePagePatch1}
\begin{paste}{PolarGraphicsExamplePageFull1}{PolarGraphicsExamplePageEmpty1}
\pastebutton{PolarGraphicsExamplePageFull1}{\hidepaste}
\tab{5}\spadgraph{draw(1,t = 0..2*\%pi, coordinates == polar)\bound{example1 }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/polargraphicsexamplepage1.view/image}
\end{paste}\end{patch}

\begin{patch}{PolarGraphicsExamplePageEmpty1}
\begin{paste}{PolarGraphicsExamplePageEmpty1}{PolarGraphicsExamplePagePatch1}
\pastebutton{PolarGraphicsExamplePageEmpty1}{\showpaste}
\tab{5}\spadgraph{draw(1,t = 0..2*\%pi, coordinates == polar)\bound{example1 }}
\end{paste}\end{patch}
```

```
\begin{patch}{PolarGraphicsExamplePagePatch2}
\begin{paste}{PolarGraphicsExamplePageFull2}{PolarGraphicsExamplePageEmpty2}
\pastebutton{PolarGraphicsExamplePageFull2}{\hidepaste}
\tab{5}\spadgraph{draw(t,t = 0..100, coordinates == polar)\bound{example2 }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/polargraphicsexamplepage2.view/image}}{viewalone
\end{paste}\end{patch}

\begin{patch}{PolarGraphicsExamplePageEmpty2}
\begin{paste}{PolarGraphicsExamplePageEmpty2}{PolarGraphicsExamplePagePatch2}
\pastebutton{PolarGraphicsExamplePageEmpty2}{\showpaste}
\tab{5}\spadgraph{draw(t,t = 0..100, coordinates == polar)\bound{example2 }}
\end{paste}\end{patch}

\begin{patch}{PolarGraphicsExamplePagePatch3}
\begin{paste}{PolarGraphicsExamplePageFull3}{PolarGraphicsExamplePageEmpty3}
\pastebutton{PolarGraphicsExamplePageFull3}{\hidepaste}
\tab{5}\spadgraph{draw(sin(4*t), t = 0..2*\%pi, coordinates == polar)\bound{example3 }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/polargraphicsexamplepage3.view/image}}{viewalone
\end{paste}\end{patch}

\begin{patch}{PolarGraphicsExamplePageEmpty3}
\begin{paste}{PolarGraphicsExamplePageEmpty3}{PolarGraphicsExamplePagePatch3}
\pastebutton{PolarGraphicsExamplePageEmpty3}{\showpaste}
\tab{5}\spadgraph{draw(sin(4*t), t = 0..2*\%pi, coordinates == polar)\bound{example3 }}
\end{paste}\end{patch}

\begin{patch}{PolarGraphicsExamplePagePatch4}
\begin{paste}{PolarGraphicsExamplePageFull4}{PolarGraphicsExamplePageEmpty4}
\pastebutton{PolarGraphicsExamplePageFull4}{\hidepaste}
\tab{5}\spadgraph{draw(2 + 3 * sin t, t = 0..2*\%pi, coordinates == polar)\bound{example4 }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/polargraphicsexamplepage4.view/image}}{viewalone
\end{paste}\end{patch}

\begin{patch}{PolarGraphicsExamplePageEmpty4}
\begin{paste}{PolarGraphicsExamplePageEmpty4}{PolarGraphicsExamplePagePatch4}
\pastebutton{PolarGraphicsExamplePageEmpty4}{\showpaste}
\tab{5}\spadgraph{draw(2 + 3 * sin t, t = 0..2*\%pi, coordinates == polar)\bound{example4 }}
\end{paste}\end{patch}

\begin{patch}{PolarGraphicsExamplePagePatch5}
\begin{paste}{PolarGraphicsExamplePageFull5}{PolarGraphicsExamplePageEmpty5}
\pastebutton{PolarGraphicsExamplePageFull5}{\hidepaste}
\tab{5}\spadcommand{All\free{example2 example3 example4 }}
\indentrel{3}\begin{verbatim}
   (5)  All
                                     Type: Variable All
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolarGraphicsExamplePageEmpty5}
```

```
\begin{paste}{PolarGraphicsExamplePageEmpty5}{PolarGraphicsExamplePagePatch5}
\pastebutton{PolarGraphicsExamplePageEmpty5}{\showpaste}
\tab{5}\spadcommand{All\free{example2 example3 example4 }}
\end{paste}\end{patch}
```

———-

Implicit Curves

— graphics.ht —

```
\begin{page}{ImplicitCurveGraphicsExamplePage}{Implicit Curves}
Non-singular curves defined by a polynomial equation p(x,y) = 0
in a rectangular region in the plane.
Pick a particular example or choose 'All' to see all the examples.
\beginscroll
A Conic Section (Hyperbola).
\graphpaste{draw(x * y = 1, x, y, range == \[-3..3, -3..3\])
\bound{example1} }
An Elliptic Curve.
\graphpaste{draw(y**2 + y = x**3 - x, x, y,
range == \[-2..2, -2..1\]) \bound{example2} }
Cartesian Ovals.
\spadpaste{p := ((x**2 + y**2 + 1) - 8*x)**2 -
(8*(x**2 + y**2 + 1) - 4*x - 1) \bound{p} }
\graphpaste{draw(p = 0, x, y, range == \[-1..11, -7..7\],
title == "Cartesian Ovals") \free{p} \bound{example3} }
Cassinian Ovals: two loops.
\spadpaste{q := (x**2 + y**2 + 7**2)**2 - (6**4 + 4*7**2*x**2)
\bound{q} }
\graphpaste{draw(q = 0, x, y, range == \[-10..10, -4..4\],
title == "Cassinian oval with two loops") \free{q} \bound{example4} }
Run all examples on this page.
\spadpaste{All \free{example1 example2 example3 example4}}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ImplicitCurveGraphicsExamplePagePatch1}
\begin{paste}{ImplicitCurveGraphicsExamplePageFull1}{ImplicitCurveGraphicsExamplePageEmpty1}
\pastebutton{ImplicitCurveGraphicsExamplePageFull1}{\hidepaste}
\tab{5}\spadgraph{draw(x * y = 1, x, y, range == [-3..3, -3..3])\bound{example1 }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/implicitcurvegraphicsexamplepage1.vie
\end{paste}\end{patch}

\begin{patch}{ImplicitCurveGraphicsExamplePageEmpty1}
\begin{paste}{ImplicitCurveGraphicsExamplePageEmpty1}{ImplicitCurveGraphicsExamplePagePatch1}
```

```
\pastebutton{ImplicitCurveGraphicsExamplePageEmpty1}{\showpaste}
\tab{5}\spadgraph{draw(x * y = 1, x, y, range == [-3..3, -3..3])\bound{example1 }}
\end{paste}\end{patch}

\begin{patch}{ImplicitCurveGraphicsExamplePagePatch2}
\begin{paste}{ImplicitCurveGraphicsExamplePageFull2}{ImplicitCurveGraphicsExamplePageEmpty2}
\pastebutton{ImplicitCurveGraphicsExamplePageFull2}{\hidepaste}
\tab{5}\spadgraph{draw(y**2 + y = x**3 - x, x, y, range == [-2..2, -2..1])\bound{example2 }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/implicitcurvegraphicsexamplepage2.view/image}}{v
\end{paste}\end{patch}

\begin{patch}{ImplicitCurveGraphicsExamplePageEmpty2}
\begin{paste}{ImplicitCurveGraphicsExamplePageEmpty2}{ImplicitCurveGraphicsExamplePagePatch2}
\pastebutton{ImplicitCurveGraphicsExamplePageEmpty2}{\showpaste}
\tab{5}\spadgraph{draw(y**2 + y = x**3 - x, x, y, range == [-2..2, -2..1])\bound{example2 }}
\end{paste}\end{patch}

\begin{patch}{ImplicitCurveGraphicsExamplePagePatch3}
\begin{paste}{ImplicitCurveGraphicsExamplePageFull3}{ImplicitCurveGraphicsExamplePageEmpty3}
\pastebutton{ImplicitCurveGraphicsExamplePageFull3}{\hidepaste}
\tab{5}\spadcommand{p := ((x**2 + y**2 + 1) - 8*x)**2 - (8*(x**2 + y**2 + 1) - 4*x - 1)\bound{p }}
\indentrel{3}\begin{verbatim}
   (3)
    4      2            2    4      3      2
   y  + (2x  - 16x - 6)y  + x  - 16x  + 58x  - 12x - 6
                              Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ImplicitCurveGraphicsExamplePageEmpty3}
\begin{paste}{ImplicitCurveGraphicsExamplePageEmpty3}{ImplicitCurveGraphicsExamplePagePatch3}
\pastebutton{ImplicitCurveGraphicsExamplePageEmpty3}{\showpaste}
\tab{5}\spadcommand{p := ((x**2 + y**2 + 1) - 8*x)**2 - (8*(x**2 + y**2 + 1) - 4*x - 1)\bound{p }}
\end{paste}\end{patch}

\begin{patch}{ImplicitCurveGraphicsExamplePagePatch4}
\begin{paste}{ImplicitCurveGraphicsExamplePageFull4}{ImplicitCurveGraphicsExamplePageEmpty4}
\pastebutton{ImplicitCurveGraphicsExamplePageFull4}{\hidepaste}
\tab{5}\spadgraph{draw(p = 0, x, y, range == [-1..11, -7..7], title == "Cartesian Ovals")\free{p }\bound{ex
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/implicitcurvegraphicsexamplepage4.view/image}}{v
\end{paste}\end{patch}

\begin{patch}{ImplicitCurveGraphicsExamplePageEmpty4}
\begin{paste}{ImplicitCurveGraphicsExamplePageEmpty4}{ImplicitCurveGraphicsExamplePagePatch4}
\pastebutton{ImplicitCurveGraphicsExamplePageEmpty4}{\showpaste}
\tab{5}\spadgraph{draw(p = 0, x, y, range == [-1..11, -7..7], title == "Cartesian Ovals")\free{p }\bound{ex
\end{paste}\end{patch}

\begin{patch}{ImplicitCurveGraphicsExamplePagePatch5}
\begin{paste}{ImplicitCurveGraphicsExamplePageFull5}{ImplicitCurveGraphicsExamplePageEmpty5}
```

```
\pastebutton{ImplicitCurveGraphicsExamplePageFull5}{\hidepaste}
\tab{5}\spadcommand{q := (x**2 + y**2 + 7**2)**2 - (6**4 + 4*7**2*x**2)\bound{q }}
\indentrel{3}\begin{verbatim}
         4      2       2    4      2
   (5)  y  + (2x  + 98)y  + x  - 98x  + 1105
                                Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ImplicitCurveGraphicsExamplePageEmpty5}
\begin{paste}{ImplicitCurveGraphicsExamplePageEmpty5}{ImplicitCurveGraphicsExamplePagePatch5}
\pastebutton{ImplicitCurveGraphicsExamplePageEmpty5}{\showpaste}
\tab{5}\spadcommand{q := (x**2 + y**2 + 7**2)**2 - (6**4 + 4*7**2*x**2)\bound{q }}
\end{paste}\end{patch}

\begin{patch}{ImplicitCurveGraphicsExamplePagePatch6}
\begin{paste}{ImplicitCurveGraphicsExamplePageFull6}{ImplicitCurveGraphicsExamplePageEmpty6}
\pastebutton{ImplicitCurveGraphicsExamplePageFull6}{\hidepaste}
\tab{5}\spadgraph{draw(q = 0, x, y, range == [-10..10, -4..4], title == "Cassinian oval with two
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/implicitcurvegraphicsexamplepage6.vie
\end{paste}\end{patch}

\begin{patch}{ImplicitCurveGraphicsExamplePageEmpty6}
\begin{paste}{ImplicitCurveGraphicsExamplePageEmpty6}{ImplicitCurveGraphicsExamplePagePatch6}
\pastebutton{ImplicitCurveGraphicsExamplePageEmpty6}{\showpaste}
\tab{5}\spadgraph{draw(q = 0, x, y, range == [-10..10, -4..4], title == "Cassinian oval with two
\end{paste}\end{patch}

\begin{patch}{ImplicitCurveGraphicsExamplePagePatch7}
\begin{paste}{ImplicitCurveGraphicsExamplePageFull7}{ImplicitCurveGraphicsExamplePageEmpty7}
\pastebutton{ImplicitCurveGraphicsExamplePageFull7}{\hidepaste}
\tab{5}\spadcommand{All\free{example1 example2 example3 example4 }}
\indentrel{3}\begin{verbatim}
   (7)  All
                                      Type: Variable All
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ImplicitCurveGraphicsExamplePageEmpty7}
\begin{paste}{ImplicitCurveGraphicsExamplePageEmpty7}{ImplicitCurveGraphicsExamplePagePatch7}
\pastebutton{ImplicitCurveGraphicsExamplePageEmpty7}{\showpaste}
\tab{5}\spadcommand{All\free{example1 example2 example3 example4 }}
\end{paste}\end{patch}
```

Lists of Points

— graphics.ht —

```
\begin{page}{ListPointsGraphicsExamplePage}{Lists of Points}
Axiom has the ability to create lists of points in a two dimensional
graphics viewport.  This is done by utilizing the \spadtype{GraphImage}
and \spadtype{TwoDimensionalViewport} domain facilities.
\beginscroll
\indent{5}\newline
{\em NOTE:  It is only necessary to click on the makeViewport2D
command button to plot this curve example}.
\indent{0}\newline
\spadpaste{p1 := point [1::SF,1::SF]\$(Point SF) \bound{p1}}
\newline
\spadpaste{p2 := point [0::SF,1::SF]\$(Point SF) \bound{p2}}
\newline
\spadpaste{p3 := point [0::SF,0::SF]\$(Point SF) \bound{p3}}
\newline
\spadpaste{p4 := point [1::SF,0::SF]\$(Point SF) \bound{p4}}
\newline
\spadpaste{p5 := point [1::SF,.5::SF]\$(Point SF) \bound{p5}}
\newline
\spadpaste{p6 := point [.5::SF,0::SF]\$(Point SF) \bound{p6}}
\newline
\spadpaste{p7 := point [0::SF,0.5::SF]\$(Point SF) \bound{p7}}
\newline
\spadpaste{p8 := point [.5::SF,1::SF]\$(Point SF) \bound{p8}}
\newline
\spadpaste{p9 := point [.25::SF,.25::SF]\$(Point SF) \bound{p9}}
\newline
\spadpaste{p10 := point [.25::SF,.75::SF]\$(Point SF) \bound{p10}}
\newline
\spadpaste{p11 := point [.75::SF,.75::SF]\$(Point SF) \bound{p11}}
\newline
\spadpaste{p12 := point [.75::SF,.25::SF]\$(Point SF) \bound{p12}}
\newline
\spadpaste{llp := [[p1,p2],[p2,p3],[p3,p4],[p4,p1],[p5,p6],[p6,p7],
[p7,p8],[p8,p5],[p9,p10],[p10,p11],[p11,p12],[p12,p9]] \bound{llp}
\free{p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 p11 p12}}
\newline
\spadpaste{size1 := 6::PositiveInteger \bound{size1}}
\newline
\spadpaste{size2 := 8::PositiveInteger \bound{size2}}
\newline
\spadpaste{size3 := 10::PositiveInteger \bound{size3}}
\newline
\spadpaste{lsize := [size1, size1, size1, size1, size2, size2,
size2, size2, size3, size3, size3, size3] \bound{lsize}
```

```
\free{size1 size2 size3}}
\newline
\spadpaste{pc1 := pastel red() \bound{pc1}}
\newline
\spadpaste{pc2 := dim green() \bound{pc2}}
\newline
\spadpaste{pc3 := pastel yellow() \bound{pc3}}
\newline
\spadpaste{lpc := [pc1, pc1, pc1, pc1, pc2, pc2, pc2, pc2, pc3,
pc3, pc3, pc3] \bound{lpc} \free{pc1 pc2 pc3}}
\newline
\spadpaste{lc := [pastel blue(), light yellow(), dim green(),
bright red(), light green(), dim yellow(), bright blue(),
dark red(), pastel red(), light blue(), dim green(),
light yellow()] \bound{lc}}
\newline
\spadpaste{g := makeGraphImage(llp,lpc,lc,lsize)\$GRIMAGE
\bound{g} \free{llp lpc lc lsize}}
\newline
\graphpaste{makeViewport2D(g,[title("Lines")])\$VIEW2D \free{g}}
The \spadfun{makeViewport2D} command takes a list of options
as a parameter in this example.  The string "Lines" is designated
as the viewport's title.
\endscroll
\autobuttons
\end{page}

\begin{patch}{ListPointsGraphicsExamplePagePatch1}
\begin{paste}{ListPointsGraphicsExamplePageFull1}{ListPointsGraphicsExamplePageEmpty1}
\pastebutton{ListPointsGraphicsExamplePageFull1}{\hidepaste}
\tab{5}\spadcommand{p1 := point [1::SF,1::SF]$(Point SF)\bound{p1 }}
\indentrel{3}\begin{verbatim}
   (1)  [1.0,1.0]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePageEmpty1}
\begin{paste}{ListPointsGraphicsExamplePageEmpty1}{ListPointsGraphicsExamplePagePatch1}
\pastebutton{ListPointsGraphicsExamplePageEmpty1}{\showpaste}
\tab{5}\spadcommand{p1 := point [1::SF,1::SF]$(Point SF)\bound{p1 }}
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePagePatch2}
\begin{paste}{ListPointsGraphicsExamplePageFull2}{ListPointsGraphicsExamplePageEmpty2}
\pastebutton{ListPointsGraphicsExamplePageFull2}{\hidepaste}
\tab{5}\spadcommand{p2 := point [0::SF,1::SF]$(Point SF)\bound{p2 }}
\indentrel{3}\begin{verbatim}
   (2)  [0.0,1.0]
                                Type: Point DoubleFloat
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePageEmpty2}
\begin{paste}{ListPointsGraphicsExamplePageEmpty2}{ListPointsGraphicsExamplePagePatch2}
\pastebutton{ListPointsGraphicsExamplePageEmpty2}{\showpaste}
\tab{5}\spadcommand{p2 := point [0::SF,1::SF]$(Point SF)\bound{p2 }}
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePagePatch3}
\begin{paste}{ListPointsGraphicsExamplePageFull3}{ListPointsGraphicsExamplePageEmpty3}
\pastebutton{ListPointsGraphicsExamplePageFull3}{\hidepaste}
\tab{5}\spadcommand{p3 := point [0::SF,0::SF]$(Point SF)\bound{p3 }}
\indentrel{3}\begin{verbatim}
   (3)  [0.0,0.0]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePageEmpty3}
\begin{paste}{ListPointsGraphicsExamplePageEmpty3}{ListPointsGraphicsExamplePagePatch3}
\pastebutton{ListPointsGraphicsExamplePageEmpty3}{\showpaste}
\tab{5}\spadcommand{p3 := point [0::SF,0::SF]$(Point SF)\bound{p3 }}
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePagePatch4}
\begin{paste}{ListPointsGraphicsExamplePageFull4}{ListPointsGraphicsExamplePageEmpty4}
\pastebutton{ListPointsGraphicsExamplePageFull4}{\hidepaste}
\tab{5}\spadcommand{p4 := point [1::SF,0::SF]$(Point SF)\bound{p4 }}
\indentrel{3}\begin{verbatim}
   (4)  [1.0,0.0]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePageEmpty4}
\begin{paste}{ListPointsGraphicsExamplePageEmpty4}{ListPointsGraphicsExamplePagePatch4}
\pastebutton{ListPointsGraphicsExamplePageEmpty4}{\showpaste}
\tab{5}\spadcommand{p4 := point [1::SF,0::SF]$(Point SF)\bound{p4 }}
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePagePatch5}
\begin{paste}{ListPointsGraphicsExamplePageFull5}{ListPointsGraphicsExamplePageEmpty5}
\pastebutton{ListPointsGraphicsExamplePageFull5}{\hidepaste}
\tab{5}\spadcommand{p5 := point [1::SF,.5::SF]$(Point SF)\bound{p5 }}
\indentrel{3}\begin{verbatim}
   (5)  [1.0,0.5]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ListPointsGraphicsExamplePageEmpty5}
\begin{paste}{ListPointsGraphicsExamplePageEmpty5}{ListPointsGraphicsExamplePagePatch5}
\pastebutton{ListPointsGraphicsExamplePageEmpty5}{\showpaste}
\tab{5}\spadcommand{p5 := point [1::SF,.5::SF]$(Point SF)\bound{p5 }}
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePagePatch6}
\begin{paste}{ListPointsGraphicsExamplePageFull6}{ListPointsGraphicsExamplePageEmpty6}
\pastebutton{ListPointsGraphicsExamplePageFull6}{\hidepaste}
\tab{5}\spadcommand{p6 := point [.5::SF,0::SF]$(Point SF)\bound{p6 }}
\indentrel{3}\begin{verbatim}
   (6)  [0.5,0.0]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePageEmpty6}
\begin{paste}{ListPointsGraphicsExamplePageEmpty6}{ListPointsGraphicsExamplePagePatch6}
\pastebutton{ListPointsGraphicsExamplePageEmpty6}{\showpaste}
\tab{5}\spadcommand{p6 := point [.5::SF,0::SF]$(Point SF)\bound{p6 }}
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePagePatch7}
\begin{paste}{ListPointsGraphicsExamplePageFull7}{ListPointsGraphicsExamplePageEmpty7}
\pastebutton{ListPointsGraphicsExamplePageFull7}{\hidepaste}
\tab{5}\spadcommand{p7 := point [0::SF,0.5::SF]$(Point SF)\bound{p7 }}
\indentrel{3}\begin{verbatim}
   (7)  [0.0,0.5]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePageEmpty7}
\begin{paste}{ListPointsGraphicsExamplePageEmpty7}{ListPointsGraphicsExamplePagePatch7}
\pastebutton{ListPointsGraphicsExamplePageEmpty7}{\showpaste}
\tab{5}\spadcommand{p7 := point [0::SF,0.5::SF]$(Point SF)\bound{p7 }}
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePagePatch8}
\begin{paste}{ListPointsGraphicsExamplePageFull8}{ListPointsGraphicsExamplePageEmpty8}
\pastebutton{ListPointsGraphicsExamplePageFull8}{\hidepaste}
\tab{5}\spadcommand{p8 := point [.5::SF,1::SF]$(Point SF)\bound{p8 }}
\indentrel{3}\begin{verbatim}
   (8)  [0.5,1.0]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePageEmpty8}
```

```
\begin{paste}{ListPointsGraphicsExamplePageEmpty8}{ListPointsGraphicsExamplePagePatch8}
\pastebutton{ListPointsGraphicsExamplePageEmpty8}{\showpaste}
\tab{5}\spadcommand{p8 := point [.5::SF,1::SF]$(Point SF)\bound{p8 }}
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePagePatch9}
\begin{paste}{ListPointsGraphicsExamplePageFull9}{ListPointsGraphicsExamplePageEmpty9}
\pastebutton{ListPointsGraphicsExamplePageFull9}{\hidepaste}
\tab{5}\spadcommand{p9 := point [.25::SF,.25::SF]$(Point SF)\bound{p9 }}
\indentrel{3}\begin{verbatim}
   (9)  [0.25,0.25]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePageEmpty9}
\begin{paste}{ListPointsGraphicsExamplePageEmpty9}{ListPointsGraphicsExamplePagePatch9}
\pastebutton{ListPointsGraphicsExamplePageEmpty9}{\showpaste}
\tab{5}\spadcommand{p9 := point [.25::SF,.25::SF]$(Point SF)\bound{p9 }}
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePagePatch10}
\begin{paste}{ListPointsGraphicsExamplePageFull10}{ListPointsGraphicsExamplePageEmpty10}
\pastebutton{ListPointsGraphicsExamplePageFull10}{\hidepaste}
\tab{5}\spadcommand{p10 := point [.25::SF,.75::SF]$(Point SF)\bound{p10 }}
\indentrel{3}\begin{verbatim}
   (10)  [0.25,0.75]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePageEmpty10}
\begin{paste}{ListPointsGraphicsExamplePageEmpty10}{ListPointsGraphicsExamplePagePatch10}
\pastebutton{ListPointsGraphicsExamplePageEmpty10}{\showpaste}
\tab{5}\spadcommand{p10 := point [.25::SF,.75::SF]$(Point SF)\bound{p10 }}
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePagePatch11}
\begin{paste}{ListPointsGraphicsExamplePageFull11}{ListPointsGraphicsExamplePageEmpty11}
\pastebutton{ListPointsGraphicsExamplePageFull11}{\hidepaste}
\tab{5}\spadcommand{p11 := point [.75::SF,.75::SF]$(Point SF)\bound{p11 }}
\indentrel{3}\begin{verbatim}
   (11)  [0.75,0.75]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePageEmpty11}
\begin{paste}{ListPointsGraphicsExamplePageEmpty11}{ListPointsGraphicsExamplePagePatch11}
\pastebutton{ListPointsGraphicsExamplePageEmpty11}{\showpaste}
```

```
\tab{5}\spadcommand{p11 := point [.75::SF,.75::SF]$(Point SF)\bound{p11 }}
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePagePatch12}
\begin{paste}{ListPointsGraphicsExamplePageFull12}{ListPointsGraphicsExamplePageEmpty12}
\pastebutton{ListPointsGraphicsExamplePageFull12}{\hidepaste}
\tab{5}\spadcommand{p12 := point [.75::SF,.25::SF]$(Point SF)\bound{p12 }}
\indentrel{3}\begin{verbatim}
   (12)  [0.75,0.25]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePageEmpty12}
\begin{paste}{ListPointsGraphicsExamplePageEmpty12}{ListPointsGraphicsExamplePagePatch12}
\pastebutton{ListPointsGraphicsExamplePageEmpty12}{\showpaste}
\tab{5}\spadcommand{p12 := point [.75::SF,.25::SF]$(Point SF)\bound{p12 }}
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePagePatch13}
\begin{paste}{ListPointsGraphicsExamplePageFull13}{ListPointsGraphicsExamplePageEmpty13}
\pastebutton{ListPointsGraphicsExamplePageFull13}{\hidepaste}
\tab{5}\spadcommand{llp := [[p1,p2],[p2,p3],[p3,p4],[p4,p1],[p5,p6],[p6,p7],[p7,p8],[p8,p5],[p9,
\indentrel{3}\begin{verbatim}
   (13)
   [[[1.0,1.0],[0.0,1.0]], [[0.0,1.0],[0.0,0.0]],
    [[0.0,0.0],[1.0,0.0]], [[1.0,0.0],[1.0,1.0]],
    [[1.0,0.5],[0.5,0.0]], [[0.5,0.0],[0.0,0.5]],
    [[0.0,0.5],[0.5,1.0]], [[0.5,1.0],[1.0,0.5]],
    [[0.25,0.25],[0.25,0.75]], [[0.25,0.75],[0.75,0.75]],
    [[0.75,0.75],[0.75,0.25]], [[0.75,0.25],[0.25,0.25]]]
                      Type: List List Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePageEmpty13}
\begin{paste}{ListPointsGraphicsExamplePageEmpty13}{ListPointsGraphicsExamplePagePatch13}
\pastebutton{ListPointsGraphicsExamplePageEmpty13}{\showpaste}
\tab{5}\spadcommand{llp := [[p1,p2],[p2,p3],[p3,p4],[p4,p1],[p5,p6],[p6,p7],[p7,p8],[p8,p5],[p9,
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePagePatch14}
\begin{paste}{ListPointsGraphicsExamplePageFull14}{ListPointsGraphicsExamplePageEmpty14}
\pastebutton{ListPointsGraphicsExamplePageFull14}{\hidepaste}
\tab{5}\spadcommand{size1 := 6::PositiveInteger\bound{size1 }}
\indentrel{3}\begin{verbatim}
   (14)  6
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ListPointsGraphicsExamplePageEmpty14}
\begin{paste}{ListPointsGraphicsExamplePageEmpty14}{ListPointsGraphicsExamplePagePatch14}
\pastebutton{ListPointsGraphicsExamplePageEmpty14}{\showpaste}
\tab{5}\spadcommand{size1 := 6::PositiveInteger\bound{size1 }}
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePagePatch15}
\begin{paste}{ListPointsGraphicsExamplePageFull15}{ListPointsGraphicsExamplePageEmpty15}
\pastebutton{ListPointsGraphicsExamplePageFull15}{\hidepaste}
\tab{5}\spadcommand{size2 := 8::PositiveInteger\bound{size2 }}
\indentrel{3}\begin{verbatim}
   (15)  8
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePageEmpty15}
\begin{paste}{ListPointsGraphicsExamplePageEmpty15}{ListPointsGraphicsExamplePagePatch15}
\pastebutton{ListPointsGraphicsExamplePageEmpty15}{\showpaste}
\tab{5}\spadcommand{size2 := 8::PositiveInteger\bound{size2 }}
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePagePatch16}
\begin{paste}{ListPointsGraphicsExamplePageFull16}{ListPointsGraphicsExamplePageEmpty16}
\pastebutton{ListPointsGraphicsExamplePageFull16}{\hidepaste}
\tab{5}\spadcommand{size3 := 10::PositiveInteger\bound{size3 }}
\indentrel{3}\begin{verbatim}
   (16)  10
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePageEmpty16}
\begin{paste}{ListPointsGraphicsExamplePageEmpty16}{ListPointsGraphicsExamplePagePatch16}
\pastebutton{ListPointsGraphicsExamplePageEmpty16}{\showpaste}
\tab{5}\spadcommand{size3 := 10::PositiveInteger\bound{size3 }}
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePagePatch17}
\begin{paste}{ListPointsGraphicsExamplePageFull17}{ListPointsGraphicsExamplePageEmpty17}
\pastebutton{ListPointsGraphicsExamplePageFull17}{\hidepaste}
\tab{5}\spadcommand{lsize := [size1, size1, size1, size1, size2, size2, size2, size2, size3, size3, size3, s
\indentrel{3}\begin{verbatim}
   (17)  [6,6,6,6,8,8,8,8,10,10,10,10]
                             Type: List PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePageEmpty17}
```

```
\begin{paste}{ListPointsGraphicsExamplePageEmpty17}{ListPointsGraphicsExamplePagePatch17}
\pastebutton{ListPointsGraphicsExamplePageEmpty17}{\showpaste}
\tab{5}\spadcommand{lsize := [size1, size1, size1, size1, size2, size2, size2, size2, size3, siz
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePagePatch18}
\begin{paste}{ListPointsGraphicsExamplePageFull18}{ListPointsGraphicsExamplePageEmpty18}
\pastebutton{ListPointsGraphicsExamplePageFull18}{\hidepaste}
\tab{5}\spadcommand{pc1 := pastel red()\bound{pc1 }}
\indentrel{3}\begin{verbatim}
   (18)  [Hue: 1  Weight: 1.0] from the Pastel palette
                                          Type: Palette
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePageEmpty18}
\begin{paste}{ListPointsGraphicsExamplePageEmpty18}{ListPointsGraphicsExamplePagePatch18}
\pastebutton{ListPointsGraphicsExamplePageEmpty18}{\showpaste}
\tab{5}\spadcommand{pc1 := pastel red()\bound{pc1 }}
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePagePatch19}
\begin{paste}{ListPointsGraphicsExamplePageFull19}{ListPointsGraphicsExamplePageEmpty19}
\pastebutton{ListPointsGraphicsExamplePageFull19}{\hidepaste}
\tab{5}\spadcommand{pc2 := dim green()\bound{pc2 }}
\indentrel{3}\begin{verbatim}
   (19)  [Hue: 14  Weight: 1.0] from the Dim palette
                                          Type: Palette
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePageEmpty19}
\begin{paste}{ListPointsGraphicsExamplePageEmpty19}{ListPointsGraphicsExamplePagePatch19}
\pastebutton{ListPointsGraphicsExamplePageEmpty19}{\showpaste}
\tab{5}\spadcommand{pc2 := dim green()\bound{pc2 }}
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePagePatch20}
\begin{paste}{ListPointsGraphicsExamplePageFull20}{ListPointsGraphicsExamplePageEmpty20}
\pastebutton{ListPointsGraphicsExamplePageFull20}{\hidepaste}
\tab{5}\spadcommand{pc3 := pastel yellow()\bound{pc3 }}
\indentrel{3}\begin{verbatim}
   (20)  [Hue: 11  Weight: 1.0] from the Pastel palette
                                          Type: Palette
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePageEmpty20}
\begin{paste}{ListPointsGraphicsExamplePageEmpty20}{ListPointsGraphicsExamplePagePatch20}
\pastebutton{ListPointsGraphicsExamplePageEmpty20}{\showpaste}
```

```
\tab{5}\spadcommand{pc3 := pastel yellow()\bound{pc3 }}
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePagePatch21}
\begin{paste}{ListPointsGraphicsExamplePageFull21}{ListPointsGraphicsExamplePageEmpty21}
\pastebutton{ListPointsGraphicsExamplePageFull21}{\hidepaste}
\tab{5}\spadcommand{lpc := [pc1, pc1, pc1, pc1, pc2, pc2, pc2, pc2, pc3, pc3, pc3, pc3]\bound{lpc }\free{pc1
\indentrel{3}\begin{verbatim}
   (21)
   [[Hue: 1  Weight: 1.0] from the Pastel palette,
    [Hue: 1  Weight: 1.0] from the Pastel palette,
    [Hue: 1  Weight: 1.0] from the Pastel palette,
    [Hue: 1  Weight: 1.0] from the Pastel palette,
    [Hue: 14  Weight: 1.0] from the Dim palette,
    [Hue: 14  Weight: 1.0] from the Dim palette,
    [Hue: 14  Weight: 1.0] from the Dim palette,
    [Hue: 14  Weight: 1.0] from the Dim palette,
    [Hue: 11  Weight: 1.0] from the Pastel palette,
    [Hue: 11  Weight: 1.0] from the Pastel palette,
    [Hue: 11  Weight: 1.0] from the Pastel palette,
    [Hue: 11  Weight: 1.0] from the Pastel palette]
                                    Type: List Palette
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePageEmpty21}
\begin{paste}{ListPointsGraphicsExamplePageEmpty21}{ListPointsGraphicsExamplePagePatch21}
\pastebutton{ListPointsGraphicsExamplePageEmpty21}{\showpaste}
\tab{5}\spadcommand{lpc := [pc1, pc1, pc1, pc1, pc2, pc2, pc2, pc2, pc3, pc3, pc3, pc3]\bound{lpc }\free{pc1
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePagePatch22}
\begin{paste}{ListPointsGraphicsExamplePageFull22}{ListPointsGraphicsExamplePageEmpty22}
\pastebutton{ListPointsGraphicsExamplePageFull22}{\hidepaste}
\tab{5}\spadcommand{lc := [pastel blue(), light yellow(), dim green(), bright red(), light green(), dim yell
\indentrel{3}\begin{verbatim}
   (22)
   [[Hue: 22  Weight: 1.0] from the Pastel palette,
    [Hue: 11  Weight: 1.0] from the Light palette,
    [Hue: 14  Weight: 1.0] from the Dim palette,
    [Hue: 1  Weight: 1.0] from the Bright palette,
    [Hue: 14  Weight: 1.0] from the Light palette,
    [Hue: 11  Weight: 1.0] from the Dim palette,
    [Hue: 22  Weight: 1.0] from the Bright palette,
    [Hue: 1  Weight: 1.0] from the Dark palette,
    [Hue: 1  Weight: 1.0] from the Pastel palette,
    [Hue: 22  Weight: 1.0] from the Light palette,
    [Hue: 14  Weight: 1.0] from the Dim palette,
    [Hue: 11  Weight: 1.0] from the Light palette]
                                    Type: List Palette
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePageEmpty22}
\begin{paste}{ListPointsGraphicsExamplePageEmpty22}{ListPointsGraphicsExamplePagePatch22}
\pastebutton{ListPointsGraphicsExamplePageEmpty22}{\showpaste}
\tab{5}\spadcommand{lc := [pastel blue(), light yellow(), dim green(), bright red(), light green
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePagePatch23}
\begin{paste}{ListPointsGraphicsExamplePageFull23}{ListPointsGraphicsExamplePageEmpty23}
\pastebutton{ListPointsGraphicsExamplePageFull23}{\hidepaste}
\tab{5}\spadcommand{g := makeGraphImage(llp,lpc,lc,lsize)$GRIMAGE\bound{g }\free{llp lpc lc lsiz
\indentrel{3}\begin{verbatim}
   (23)  Graph with 12 point lists
                                         Type: GraphImage
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePageEmpty23}
\begin{paste}{ListPointsGraphicsExamplePageEmpty23}{ListPointsGraphicsExamplePagePatch23}
\pastebutton{ListPointsGraphicsExamplePageEmpty23}{\showpaste}
\tab{5}\spadcommand{g := makeGraphImage(llp,lpc,lc,lsize)$GRIMAGE\bound{g }\free{llp lpc lc lsiz
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePagePatch24}
\begin{paste}{ListPointsGraphicsExamplePageFull24}{ListPointsGraphicsExamplePageEmpty24}
\pastebutton{ListPointsGraphicsExamplePageFull24}{\hidepaste}
\tab{5}\spadgraph{makeViewport2D(g,[title("Lines")])$VIEW2D\free{g }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/listpointsgraphicsexamplepage24.view/
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsExamplePageEmpty24}
\begin{paste}{ListPointsGraphicsExamplePageEmpty24}{ListPointsGraphicsExamplePagePatch24}
\pastebutton{ListPointsGraphicsExamplePageEmpty24}{\showpaste}
\tab{5}\spadgraph{makeViewport2D(g,[title("Lines")])$VIEW2D\free{g }}
\end{paste}\end{patch}
```

Three Dimensional Graphing

⇒ "notitle" (TwoVariableGraphicsPage) 3.50 on page 629
⇒ "notitle" (SpaceCurveGraphicsPage) 3.50 on page 631
⇒ "notitle" (ParametricTubeGraphicsPage) 3.50 on page 633
⇒ "notitle" (ParametricSurfaceGraphicsPage) 3.50 on page 636

⇒ "notitle" (ugGraphThreeDBuildPage) 11 on page 2041

— graphics.ht —

```
\begin{page}{ThreeDimensionalGraphicsPage}{Three Dimensional Graphing}
\beginscroll
\beginmenu
\menulink{Functions of Two Variables}{TwoVariableGraphicsPage} \newline
Plot surfaces defined by an equation z = f(x,y).
\menulink{Parametric Curves}{SpaceCurveGraphicsPage} \newline
Plot curves defined by equations x = f(t), y = g(t), z = g(t).
\menulink{Parametric Tube Plots}{ParametricTubeGraphicsPage} \newline
Plot a tube around a parametric space curve.
\menulink{Parametric Surfaces}{ParametricSurfaceGraphicsPage} \newline
Plot surfaces defined by x = f(u,v), y = g(u,v), z = h(u,v).
\menulink{Building Objects}{ugGraphThreeDBuildPage} \newline
Create objects constructed from geometric primitives.
\endmenu
\endscroll
\autobuttons \end{page}
```

———-

Functions of Two Variables

— graphics.ht —

```
\begin{page}{TwoVariableGraphicsPage}{Functions of Two Variables}
\beginscroll

This page describes the plotting of surfaces defined by an equation of
two variables, z = f(x,y), for which the ranges of x and y are
explicitly defined.  The basic draw command for this function utilizes
either the uncompiled function or compiled function format.  The
general format for an uncompiled function is:
\indent{5}\newline
{\em draw(f(x,y), x = a..b, y = c..d)}
\indent{0}\newline
where a..b and c..d are segments defining the intervals [a,b] and
[c,d] over which the variables x and y span.  In this case the
function is not compiled until the draw command is executed.  Here is
an example:
\graphpaste{draw(cos(x*y),x=-3..3,y=-3..3)}
In the case of a compiled function, the function is named and compiled
independently.  This is useful if you intend to use a function often,
or if the function is long and complex.  The following line shows a
function whose parameters are of the type Small Float. The function is
```

```
compiled and stored by Axiom when it is entered.
\indent{5}\newline
{\em NOTE:  It is only necessary to click on the draw command button to
plot this example}.
\indent{0}\newline
\spadpaste{f(x:SF,y:SF):SF == sin(x)*cos(y) \bound{f}}
\newline
Once the function is compiled the draw command only needs the name
of the function to execute.  Here is a compiled function example:
\graphpaste{draw(f,-\%pi..\%pi,-\%pi..\%pi) \free{f}}
Note that the parameter ranges do not take the variable names as in the
case of uncompiled functions.  The variables are entered in the order in
which they are defined in the function specification.  In this case the
first range specifies the x-variable and the second range specifies the
y-variable.
\endscroll
\autobuttons
\end{page}

\begin{patch}{TwoVariableGraphicsPagePatch1}
\begin{paste}{TwoVariableGraphicsPageFull1}{TwoVariableGraphicsPageEmpty1}
\pastebutton{TwoVariableGraphicsPageFull1}{\hidepaste}
\tab{5}\spadgraph{draw(cos(x*y),x=-3..3,y=-3..3)}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/twovariablegraphicspage1.view/image}}
\end{paste}\end{patch}

\begin{patch}{TwoVariableGraphicsPageEmpty1}
\begin{paste}{TwoVariableGraphicsPageEmpty1}{TwoVariableGraphicsPagePatch1}
\pastebutton{TwoVariableGraphicsPageEmpty1}{\showpaste}
\tab{5}\spadgraph{draw(cos(x*y),x=-3..3,y=-3..3)}
\end{paste}\end{patch}

\begin{patch}{TwoVariableGraphicsPagePatch2}
\begin{paste}{TwoVariableGraphicsPageFull2}{TwoVariableGraphicsPageEmpty2}
\pastebutton{TwoVariableGraphicsPageFull2}{\hidepaste}
\tab{5}\spadcommand{f(x:SF,y:SF):SF == sin(x)*cos(y)\bound{f }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TwoVariableGraphicsPageEmpty2}
\begin{paste}{TwoVariableGraphicsPageEmpty2}{TwoVariableGraphicsPagePatch2}
\pastebutton{TwoVariableGraphicsPageEmpty2}{\showpaste}
\tab{5}\spadcommand{f(x:SF,y:SF):SF == sin(x)*cos(y)\bound{f }}
\end{paste}\end{patch}

\begin{patch}{TwoVariableGraphicsPagePatch3}
\begin{paste}{TwoVariableGraphicsPageFull3}{TwoVariableGraphicsPageEmpty3}
\pastebutton{TwoVariableGraphicsPageFull3}{\hidepaste}
```

```
\tab{5}\spadgraph{draw(f,-\%pi..\%pi,-\%pi..\%pi)\free{f }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/twovariablegraphicspage3.view/image}}{viewalone\s
\end{paste}\end{patch}

\begin{patch}{TwoVariableGraphicsPageEmpty3}
\begin{paste}{TwoVariableGraphicsPageEmpty3}{TwoVariableGraphicsPagePatch3}
\pastebutton{TwoVariableGraphicsPageEmpty3}{\showpaste}
\tab{5}\spadgraph{draw(f,-\%pi..\%pi,-\%pi..\%pi)\free{f }}
\end{paste}\end{patch}
```

———-

Parametric Space Curves

— graphics.ht —

```
\begin{page}{SpaceCurveGraphicsPage}{Parametric Space Curves}
\beginscroll
This page describes the plotting in three dimensional space of a curve
defined by the parametric equations x = f(t), y = g(t), z = h(t),
where f, g, and h are functions of the parameter t which ranges over a
specified interval.  The basic draw command for this function utilizes
either the uncompiled functions or compiled functions format and uses
the \spadfun{curve} command to specify the three functions for the x,
y, and z components of the curve.  The general format for uncompiled
functions is:
\indent{5}\newline
{\em draw(curve(f(t),g(t),h(t)), t = a..b)}
\indent{0}\newline
where a..b is the segment defining the interval [a,b] over which the
parameter t ranges.  In this case the functions are not compiled until
the draw command is executed.  Here is an example:
\graphpaste{draw(curve(cos(t),sin(t),t), t=-12..12)}
In the case of compiled functions, the functions are named and
compiled independently.  This is useful if you intend to use the
functions often, or if the functions are long and complex.  The
following lines show functions whose parameters are of the type Small
Float. The functions are compiled and stored by Axiom when entered.
\indent{5}\newline
{\em NOTE:  It is only necessary to click on the draw command button to
plot this example}.
\indent{0}\newline
\spadpaste{i1(t:SF):SF == sin(t)*cos(3*t/5) \bound{i1}}
\spadpaste{i2(t:SF):SF == cos(t)*cos(3*t/5) \bound{i2}}
\spadpaste{i3(t:SF):SF == cos(t)*sin(3*t/5) \bound{i3}}
Once the functions are compiled the draw command only needs the names
of the functions to execute.  Here is a compiled functions example:
```

```
\graphpaste{draw(curve(i1,i2,i3),0..15*\%pi) \free{i1 i2 i3}}
Note that the parameter range does not take the variable name as in
the case of uncompiled functions. It is understood that the indicated
range applies to the parameter of the functions, which in this case is t.
\endscroll
\autobuttons
\end{page}

\begin{patch}{SpaceCurveGraphicsPagePatch1}
\begin{paste}{SpaceCurveGraphicsPageFull1}{SpaceCurveGraphicsPageEmpty1}
\pastebutton{SpaceCurveGraphicsPageFull1}{\hidepaste}
\tab{5}\spadgraph{draw(curve(cos(t),sin(t),t), t=-12..12)}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/spacecurvegraphicspage1.view/image}}{
\end{paste}\end{patch}

\begin{patch}{SpaceCurveGraphicsPageEmpty1}
\begin{paste}{SpaceCurveGraphicsPageEmpty1}{SpaceCurveGraphicsPagePatch1}
\pastebutton{SpaceCurveGraphicsPageEmpty1}{\showpaste}
\tab{5}\spadgraph{draw(curve(cos(t),sin(t),t), t=-12..12)}
\end{paste}\end{patch}

\begin{patch}{SpaceCurveGraphicsPagePatch2}
\begin{paste}{SpaceCurveGraphicsPageFull2}{SpaceCurveGraphicsPageEmpty2}
\pastebutton{SpaceCurveGraphicsPageFull2}{\hidepaste}
\tab{5}\spadcommand{i1(t:SF):SF == sin(t)*cos(3*t/5)\bound{i1 }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SpaceCurveGraphicsPageEmpty2}
\begin{paste}{SpaceCurveGraphicsPageEmpty2}{SpaceCurveGraphicsPagePatch2}
\pastebutton{SpaceCurveGraphicsPageEmpty2}{\showpaste}
\tab{5}\spadcommand{i1(t:SF):SF == sin(t)*cos(3*t/5)\bound{i1 }}
\end{paste}\end{patch}

\begin{patch}{SpaceCurveGraphicsPagePatch3}
\begin{paste}{SpaceCurveGraphicsPageFull3}{SpaceCurveGraphicsPageEmpty3}
\pastebutton{SpaceCurveGraphicsPageFull3}{\hidepaste}
\tab{5}\spadcommand{i2(t:SF):SF == cos(t)*cos(3*t/5)\bound{i2 }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SpaceCurveGraphicsPageEmpty3}
\begin{paste}{SpaceCurveGraphicsPageEmpty3}{SpaceCurveGraphicsPagePatch3}
\pastebutton{SpaceCurveGraphicsPageEmpty3}{\showpaste}
\tab{5}\spadcommand{i2(t:SF):SF == cos(t)*cos(3*t/5)\bound{i2 }}
\end{paste}\end{patch}
```

```
\begin{patch}{SpaceCurveGraphicsPagePatch4}
\begin{paste}{SpaceCurveGraphicsPageFull4}{SpaceCurveGraphicsPageEmpty4}
\pastebutton{SpaceCurveGraphicsPageFull4}{\hidepaste}
\tab{5}\spadcommand{i3(t:SF):SF == cos(t)*sin(3*t/5)\bound{i3 }}
\indentrel{3}\begin{verbatim}
                                              Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SpaceCurveGraphicsPageEmpty4}
\begin{paste}{SpaceCurveGraphicsPageEmpty4}{SpaceCurveGraphicsPagePatch4}
\pastebutton{SpaceCurveGraphicsPageEmpty4}{\showpaste}
\tab{5}\spadcommand{i3(t:SF):SF == cos(t)*sin(3*t/5)\bound{i3 }}
\end{paste}\end{patch}

\begin{patch}{SpaceCurveGraphicsPagePatch5}
\begin{paste}{SpaceCurveGraphicsPageFull5}{SpaceCurveGraphicsPageEmpty5}
\pastebutton{SpaceCurveGraphicsPageFull5}{\hidepaste}
\tab{5}\spadgraph{draw(curve(i1,i2,i3),0..15*\%pi)\free{i1 i2 i3 }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/spacecurvegraphicspage5.view/image}}{viewalone\sp
\end{paste}\end{patch}

\begin{patch}{SpaceCurveGraphicsPageEmpty5}
\begin{paste}{SpaceCurveGraphicsPageEmpty5}{SpaceCurveGraphicsPagePatch5}
\pastebutton{SpaceCurveGraphicsPageEmpty5}{\showpaste}
\tab{5}\spadgraph{draw(curve(i1,i2,i3),0..15*\%pi)\free{i1 i2 i3 }}
\end{paste}\end{patch}
```

———-

Parametric Tube Plots

— graphics.ht —

```
\begin{page}{ParametricTubeGraphicsPage}{Parametric Tube Plots}
\beginscroll
This page describes the plotting in three dimensional space of a tube
around a parametric space curve defined by the parametric equations
x = f(t), y = g(t), z = h(t), where f, g, and h are functions of the
parameter t which ranges over a specified interval.  The basic draw
command for this function utilizes either the uncompiled functions or
compiled functions format and uses the \spadfun{curve} command to
specify the three functions for the x, y, and z components of the
curve.  This uses the same format as that for space curves except that
it requires a specification for the radius of the tube.  If the radius
of the tube is 0, then the result is the space curve itself.  The
```

```
general format for uncompiled functions is:
\indent{5}\newline
{\em draw(curve(f(t),g(t),h(t)), t = a..b, tubeRadius == r)}
\indent{0}\newline
where a..b is the segment defining the interval [a,b] over which the
parameter t ranges, and the tubeRadius is indicated by the variable r.
In this case the functions are not compiled until the draw command is
executed.  Here is an example:
\graphpaste{draw(curve(sin(t)*cos(3*t/5), cos(t)*cos(3*t/5),
cos(t)*sin(3*t/5)), t=0..15*\%pi,tubeRadius == .15)}
In the case of compiled functions, the functions are named and
compiled independently.  This is useful if you intend to use the
functions often, or if the functions are long and complex.  The
following lines show functions whose parameters are of the type Small
Float. The functions are compiled and stored by Axiom when entered.
\indent{5}\newline
{\em NOTE:  It is only necessary to click on the draw command button to
plot this example}.
\indent{0}\newline
\spadpaste{t1(t:SF):SF == 4/(2-sin(3*t))*cos(2*t) \bound{t1}}
\newline
\spadpaste{t2(t:SF):SF == 4/(2-sin(3*t))*sin(2*t) \bound{t2}}
\newline
\spadpaste{t3(t:SF):SF == 4/(2-sin(3*t))*cos(3*t) \bound{t3}}
\newline
Once the function is compiled the draw command only needs the names of
the functions to execute.  Here is a compiled functions example of a
trefoil knot:
\graphpaste{draw(curve(t1,t2,t3),0..2*\%pi,tubeRadius == .2)
\free{t1 t2 t3}}
Note that the parameter range does not take the variable name as in
the case of uncompiled functions. It is understood that the indicated
range applies to the parameter of the functions, which in this case is
t.  Typically, the radius of the tube should be set between 0 and 1.
A radius of less than 0 results in it's positive counterpart and a
radius of greater than one causes self intersection.
\endscroll
\autobuttons
\end{page}

\begin{patch}{ParametricTubeGraphicsPagePatch1}
\begin{paste}{ParametricTubeGraphicsPageFull1}{ParametricTubeGraphicsPageEmpty1}
\pastebutton{ParametricTubeGraphicsPageFull1}{\hidepaste}
\tab{5}\spadgraph{draw(curve(sin(t)*cos(3*t/5), cos(t)*cos(3*t/5), cos(t)*sin(3*t/5)), t=0..15*\
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/parametrictubegraphicspage1.view/imag
\end{paste}\end{patch}

\begin{patch}{ParametricTubeGraphicsPageEmpty1}
\begin{paste}{ParametricTubeGraphicsPageEmpty1}{ParametricTubeGraphicsPagePatch1}
\pastebutton{ParametricTubeGraphicsPageEmpty1}{\showpaste}
```

```
\tab{5}\spadgraph{draw(curve(sin(t)*cos(3*t/5), cos(t)*cos(3*t/5), cos(t)*sin(3*t/5)), t=0..15*\%pi,tubeRad
\end{paste}\end{patch}

\begin{patch}{ParametricTubeGraphicsPagePatch2}
\begin{paste}{ParametricTubeGraphicsPageFull2}{ParametricTubeGraphicsPageEmpty2}
\pastebutton{ParametricTubeGraphicsPageFull2}{\hidepaste}
\tab{5}\spadcommand{t1(t:SF):SF == 4/(2-sin(3*t))*cos(2*t)\bound{t1 }}
\indentrel{3}\begin{verbatim}
                                           Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ParametricTubeGraphicsPageEmpty2}
\begin{paste}{ParametricTubeGraphicsPageEmpty2}{ParametricTubeGraphicsPagePatch2}
\pastebutton{ParametricTubeGraphicsPageEmpty2}{\showpaste}
\tab{5}\spadcommand{t1(t:SF):SF == 4/(2-sin(3*t))*cos(2*t)\bound{t1 }}
\end{paste}\end{patch}

\begin{patch}{ParametricTubeGraphicsPagePatch3}
\begin{paste}{ParametricTubeGraphicsPageFull3}{ParametricTubeGraphicsPageEmpty3}
\pastebutton{ParametricTubeGraphicsPageFull3}{\hidepaste}
\tab{5}\spadcommand{t2(t:SF):SF == 4/(2-sin(3*t))*sin(2*t)\bound{t2 }}
\indentrel{3}\begin{verbatim}
                                           Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ParametricTubeGraphicsPageEmpty3}
\begin{paste}{ParametricTubeGraphicsPageEmpty3}{ParametricTubeGraphicsPagePatch3}
\pastebutton{ParametricTubeGraphicsPageEmpty3}{\showpaste}
\tab{5}\spadcommand{t2(t:SF):SF == 4/(2-sin(3*t))*sin(2*t)\bound{t2 }}
\end{paste}\end{patch}

\begin{patch}{ParametricTubeGraphicsPagePatch4}
\begin{paste}{ParametricTubeGraphicsPageFull4}{ParametricTubeGraphicsPageEmpty4}
\pastebutton{ParametricTubeGraphicsPageFull4}{\hidepaste}
\tab{5}\spadcommand{t3(t:SF):SF == 4/(2-sin(3*t))*cos(3*t)\bound{t3 }}
\indentrel{3}\begin{verbatim}
                                           Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ParametricTubeGraphicsPageEmpty4}
\begin{paste}{ParametricTubeGraphicsPageEmpty4}{ParametricTubeGraphicsPagePatch4}
\pastebutton{ParametricTubeGraphicsPageEmpty4}{\showpaste}
\tab{5}\spadcommand{t3(t:SF):SF == 4/(2-sin(3*t))*cos(3*t)\bound{t3 }}
\end{paste}\end{patch}

\begin{patch}{ParametricTubeGraphicsPagePatch5}
\begin{paste}{ParametricTubeGraphicsPageFull5}{ParametricTubeGraphicsPageEmpty5}
```

```
\pastebutton{ParametricTubeGraphicsPageFull5}{\hidepaste}
\tab{5}\spadgraph{draw(curve(t1,t2,t3),0..2*\%pi,tubeRadius == .2)\free{t1 t2 t3 }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/parametrictubegraphicspage5.view/imag
\end{paste}\end{patch}

\begin{patch}{ParametricTubeGraphicsPageEmpty5}
\begin{paste}{ParametricTubeGraphicsPageEmpty5}{ParametricTubeGraphicsPagePatch5}
\pastebutton{ParametricTubeGraphicsPageEmpty5}{\showpaste}
\tab{5}\spadgraph{draw(curve(t1,t2,t3),0..2*\%pi,tubeRadius == .2)\free{t1 t2 t3 }}
\end{paste}\end{patch}
```

——–

Parametric Surfaces

— graphics.ht —

```
\begin{page}{ParametricSurfaceGraphicsPage}{Parametric Surfaces}
\beginscroll
Graphing a surface defined by x = f(u,v), y = g(u,v), z = h(u,v).
\newline This page describes the plotting of surfaces defined
by the parametric equations of two variables, x = f(u,v), y = g(u,v),
and z = h(u,v), for which the ranges of u and v are explicitly
defined.  The basic draw command for this function utilizes either the
uncompiled function or compiled function format and uses the
\spadfun{surface} command to specify the three functions for the x, y
and z components of the surface.  The general format for uncompiled
functions is:
\indent{5}\newline
{\em draw(surface(f(u,v),g(u,v),h(u,v)), u = a..b, v = c..d)}
\indent{0}\newline
where a..b and c..d are segments defining the intervals [a,b] and
[c,d] over which the parameters u and v span.  In this case the
functions are not compiled until the draw command is executed.  Here
is an example of a surface plotted using the parabolic cylindrical
coordinate system option:
\graphpaste{draw(surface(u*cos(v), u*sin(v),v*cos(u)),
u=-4..4,v=0..2*\%pi, coordinates== parabolicCylindrical)}
In the case of compiled functions, the functions are named and
compiled independently.  This is useful if you intend to use the
functions often, or if the functions are long and complex.  The
following lines show functions whose parameters are of the type Small
Float. The functions are compiled and stored by Axiom when entered.
\indent{5}\newline
{\em NOTE:  It is only necessary to click on the draw command button to
plot this example}.
\indent{0}\newline
```

```
\spadpaste{n1(u:SF,v:SF):SF == u*cos(v) \bound{n1}}
\newline
\spadpaste{n2(u:SF,v:SF):SF == u*sin(v) \bound{n2}}
\newline
\spadpaste{n3(u:SF,v:SF):SF == u \bound{n3}}
Once the function is compiled the draw command only needs the names of
the functions to execute.  Here is a compiled functions example
plotted using the toroidal coordinate system option: \newline
\graphpaste{draw(surface(n1,n2,n3), 1.0..4.0, 1.0..4*\%pi,
             coordinates == toroidal(1\$SF)) \free{n1 n2 n3}}
Note that the parameter ranges do not take the variable names as in
the case of uncompiled functions.  The variables are entered in the
order in which they are defined in the function specification.  In
this case the first range specifies the u-variable and the second
range specifies the v-variable.
\endscroll
\autobuttons
\end{page}

\begin{patch}{ParametricSurfaceGraphicsPagePatch1}
\begin{paste}{ParametricSurfaceGraphicsPageFull1}{ParametricSurfaceGraphicsPageEmpty1}
\pastebutton{ParametricSurfaceGraphicsPageFull1}{\hidepaste}
\tab{5}\spadgraph{draw(surface(u*cos(v), u*sin(v),v*cos(u)),u=-4..4,v=0..2*\%pi, coordinates== parabolicCyli
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/parametricsurfacegraphicspage1.view/image}}{viewa
\end{paste}\end{patch}

\begin{patch}{ParametricSurfaceGraphicsPageEmpty1}
\begin{paste}{ParametricSurfaceGraphicsPageEmpty1}{ParametricSurfaceGraphicsPagePatch1}
\pastebutton{ParametricSurfaceGraphicsPageEmpty1}{\showpaste}
\tab{5}\spadgraph{draw(surface(u*cos(v), u*sin(v),v*cos(u)),u=-4..4,v=0..2*\%pi, coordinates== parabolicCyli
\end{paste}\end{patch}

\begin{patch}{ParametricSurfaceGraphicsPagePatch2}
\begin{paste}{ParametricSurfaceGraphicsPageFull2}{ParametricSurfaceGraphicsPageEmpty2}
\pastebutton{ParametricSurfaceGraphicsPageFull2}{\hidepaste}
\tab{5}\spadcommand{n1(u:SF,v:SF):SF == u*cos(v)\bound{n1 }}
\indentrel{3}\begin{verbatim}
                                        Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ParametricSurfaceGraphicsPageEmpty2}
\begin{paste}{ParametricSurfaceGraphicsPageEmpty2}{ParametricSurfaceGraphicsPagePatch2}
\pastebutton{ParametricSurfaceGraphicsPageEmpty2}{\showpaste}
\tab{5}\spadcommand{n1(u:SF,v:SF):SF == u*cos(v)\bound{n1 }}
\end{paste}\end{patch}

\begin{patch}{ParametricSurfaceGraphicsPagePatch3}
\begin{paste}{ParametricSurfaceGraphicsPageFull3}{ParametricSurfaceGraphicsPageEmpty3}
\pastebutton{ParametricSurfaceGraphicsPageFull3}{\hidepaste}
```

```
\tab{5}\spadcommand{n2(u:SF,v:SF):SF == u*sin(v)\bound{n2 }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ParametricSurfaceGraphicsPageEmpty3}
\begin{paste}{ParametricSurfaceGraphicsPageEmpty3}{ParametricSurfaceGraphicsPagePatch3}
\pastebutton{ParametricSurfaceGraphicsPageEmpty3}{\showpaste}
\tab{5}\spadcommand{n2(u:SF,v:SF):SF == u*sin(v)\bound{n2 }}
\end{paste}\end{patch}

\begin{patch}{ParametricSurfaceGraphicsPagePatch4}
\begin{paste}{ParametricSurfaceGraphicsPageFull4}{ParametricSurfaceGraphicsPageEmpty4}
\pastebutton{ParametricSurfaceGraphicsPageFull4}{\hidepaste}
\tab{5}\spadcommand{n3(u:SF,v:SF):SF == u\bound{n3 }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ParametricSurfaceGraphicsPageEmpty4}
\begin{paste}{ParametricSurfaceGraphicsPageEmpty4}{ParametricSurfaceGraphicsPagePatch4}
\pastebutton{ParametricSurfaceGraphicsPageEmpty4}{\showpaste}
\tab{5}\spadcommand{n3(u:SF,v:SF):SF == u\bound{n3 }}
\end{paste}\end{patch}

\begin{patch}{ParametricSurfaceGraphicsPagePatch5}
\begin{paste}{ParametricSurfaceGraphicsPageFull5}{ParametricSurfaceGraphicsPageEmpty5}
\pastebutton{ParametricSurfaceGraphicsPageFull5}{\hidepaste}
\tab{5}\spadgraph{draw(surface(n1,n2,n3), 1.0..4.0, 1.0..4*\%pi, coordinates == toroidal(1$SF))\
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/parametricsurfacegraphicspage5.view/i
\end{paste}\end{patch}

\begin{patch}{ParametricSurfaceGraphicsPageEmpty5}
\begin{paste}{ParametricSurfaceGraphicsPageEmpty5}{ParametricSurfaceGraphicsPagePatch5}
\pastebutton{ParametricSurfaceGraphicsPageEmpty5}{\showpaste}
\tab{5}\spadgraph{draw(surface(n1,n2,n3), 1.0..4.0, 1.0..4*\%pi, coordinates == toroidal(1$SF))\
\end{paste}\end{patch}
```

———-

Building 3D Objects

— graphics.ht —

```
\begin{page}{3DObjectGraphicsPage}{Building 3D Objects}
```

```
\beginscroll
This page describes the Axiom facilities for creating three
dimensional objects constructed from geometric primitives.  The Axiom
operation \spadfun{create3Space()} creates a space to which points,
curves, and polygons can be added using the operations from the
\spadtype{ThreeSpace} domain.  The contents of this space can then be
displayed in a viewport using the \spadfun{makeViewport3D()} command.
It will be necessary to have these operations exposed in order to use
them. \indent{5}\newline {\em NOTE: It is only necessary to click on
the makeViewport3D command button to plot this curve example}.
\indent{0}\newline
Initially, the space which will hold the objects must be defined and
compiled, as in the following example:
\spadpaste{space := create3Space()\$(ThreeSpace SF) \bound{space}}
Now objects can be sent to this {\em space} as per the operations
allowed by the \spadtype{ThreeSpace} domain.  The following examples
place curves into
{\em space}.
\spadpaste{curve(space,[[0,20,20],[0,20,30],[0,30,30],[0,30,100],
[0,20,100],[0,20,110],[0,50,110],[0,50,100],[0,40,100], [0,40,30],
[0,50,30],[0,50,20],[0,20,20]]) \bound{curveI}}
\newline
\spadpaste{curve(space,[[0,80,20],[0,70,20],[0,70,110],[0,110,110],
[0,120,100],[0,120,70],[0,115,65],[0,120,60],[0,120,30], [0,110,20],
[0,80,20],[0,80,30],[0,105,30],[0,110,35]]) \bound{curveB1}}
\newline
\spadpaste{curve(space,[[0,110,35],[0,110,55],[0,105,60],[0,80,60],
[0,80,70], [0,105,70],[0,110,75],[0,110,95],[0,105,100],[0,80,100],
[0,80,30]]) \bound{curveB2}}
\newline
\spadpaste{closedCurve(space,[[0,140,20],[0,140,110],[0,150,110],
[0,170,50], [0,190,110],[0,200,110],[0,200,20],[0,190,20],[0,190,75],
[0,175,35],[0,165,35],[0,150,75],[0,150,20]]) \bound{curveM}}
\spadpaste{closedCurve(space,[[200,0,20], [200,0,110], [185,0,110],
[160,0,45], [160,0,110], [150,0,110], [150,0,20], [165,0,20],
[190,0,85], [190,0,20]]) \bound{curveN}}
\spadpaste{closedCurve(space,[[140,0,20], [120,0,110], [110,0,110],
[90,0,20], [100,0,20], [108,0,50], [123,0,50], [121,0,60],
[110,0,60], [115,0,90], [130,0,20]]) \bound{curveA}}
\spadpaste{closedCurve(space,[[80,0,30], [80,0,100], [70,0,110],
[40,0,110], [30,0,100], [30,0,90], [40,0,90], [40,0,95],
[45,0,100], [65,0,100], [70,0,95], [70,0,35], [65,0,30],
[45,0,30], [40,0,35], [40,0,60], [50,0,60], [50,0,70],
[30,0,70], [30,0,30], [40,0,20], [70,0,20]]) \bound{curveG}}
Once {\em space} contains the desired elements a viewport is created
and displayed with the following command:
\graphpaste{makeViewport3D(space,[title("Curves")])\$VIEW3D
\free{space curveI curveB1 curveB2 curveM curveN curveA curveG}}
The parameters for \spadfun{makeViewport3D()} in this example are {\em
space}, which is the name of the three dimensional space that was
```

```
defined, and a string, "curve", which is the title for the viewport.
The tailing string {\em \$VIEW3D} exposes the command
\spadfun{makeViewport3D()} from the domain
\spadtype{ThreeDimensionalViewport} if these commands are unexposed.
\endscroll
\autobuttons
\end{page}

\begin{patch}{3DObjectGraphicsPagePatch1}
\begin{paste}{3DObjectGraphicsPageFull1}{3DObjectGraphicsPageEmpty1}
\pastebutton{3DObjectGraphicsPageFull1}{\hidepaste}
\tab{5}\spadcommand{space := create3Space()$(ThreeSpace SF)\bound{space }}
\indentrel{3}\begin{verbatim}
   (1)  3-Space with 0 components
                          Type: ThreeSpace DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{3DObjectGraphicsPageEmpty1}
\begin{paste}{3DObjectGraphicsPageEmpty1}{3DObjectGraphicsPagePatch1}
\pastebutton{3DObjectGraphicsPageEmpty1}{\showpaste}
\tab{5}\spadcommand{space := create3Space()$(ThreeSpace SF)\bound{space }}
\end{paste}\end{patch}

\begin{patch}{3DObjectGraphicsPagePatch2}
\begin{paste}{3DObjectGraphicsPageFull2}{3DObjectGraphicsPageEmpty2}
\pastebutton{3DObjectGraphicsPageFull2}{\hidepaste}
\tab{5}\spadcommand{curve(space,[[0,20,20],[0,20,30],[0,30,30],[0,30,100], [0,20,100],[0,20,110]
\indentrel{3}\begin{verbatim}
   (2)  3-Space with 1 component
                          Type: ThreeSpace DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{3DObjectGraphicsPageEmpty2}
\begin{paste}{3DObjectGraphicsPageEmpty2}{3DObjectGraphicsPagePatch2}
\pastebutton{3DObjectGraphicsPageEmpty2}{\showpaste}
\tab{5}\spadcommand{curve(space,[[0,20,20],[0,20,30],[0,30,30],[0,30,100], [0,20,100],[0,20,110]
\end{paste}\end{patch}

\begin{patch}{3DObjectGraphicsPagePatch3}
\begin{paste}{3DObjectGraphicsPageFull3}{3DObjectGraphicsPageEmpty3}
\pastebutton{3DObjectGraphicsPageFull3}{\hidepaste}
\tab{5}\spadcommand{curve(space,[[0,80,20],[0,70,20],[0,70,110],[0,110,110], [0,120,100],[0,120,
\indentrel{3}\begin{verbatim}
   (3)  3-Space with 2 components
                          Type: ThreeSpace DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{3DObjectGraphicsPageEmpty3}
\begin{paste}{3DObjectGraphicsPageEmpty3}{3DObjectGraphicsPagePatch3}
\pastebutton{3DObjectGraphicsPageEmpty3}{\showpaste}
\tab{5}\spadcommand{curve(space,[[0,80,20],[0,70,20],[0,70,110],[0,110,110], [0,120,100],[0,120,70],[0,115,6
\end{paste}\end{patch}

\begin{patch}{3DObjectGraphicsPagePatch4}
\begin{paste}{3DObjectGraphicsPageFull4}{3DObjectGraphicsPageEmpty4}
\pastebutton{3DObjectGraphicsPageFull4}{\hidepaste}
\tab{5}\spadcommand{curve(space,[[0,110,35],[0,110,55],[0,105,60],[0,80,60],[0,80,70], [0,105,70],[0,110,75]
\indentrel{3}\begin{verbatim}
   (4)  3-Space with 3 components
                          Type: ThreeSpace DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{3DObjectGraphicsPageEmpty4}
\begin{paste}{3DObjectGraphicsPageEmpty4}{3DObjectGraphicsPagePatch4}
\pastebutton{3DObjectGraphicsPageEmpty4}{\showpaste}
\tab{5}\spadcommand{curve(space,[[0,110,35],[0,110,55],[0,105,60],[0,80,60],[0,80,70], [0,105,70],[0,110,75]
\end{paste}\end{patch}

\begin{patch}{3DObjectGraphicsPagePatch5}
\begin{paste}{3DObjectGraphicsPageFull5}{3DObjectGraphicsPageEmpty5}
\pastebutton{3DObjectGraphicsPageFull5}{\hidepaste}
\tab{5}\spadcommand{closedCurve(space,[[0,140,20],[0,140,110],[0,150,110],[0,170,50], [0,190,110],[0,200,110
\indentrel{3}\begin{verbatim}
   (5)  3-Space with 4 components
                          Type: ThreeSpace DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{3DObjectGraphicsPageEmpty5}
\begin{paste}{3DObjectGraphicsPageEmpty5}{3DObjectGraphicsPagePatch5}
\pastebutton{3DObjectGraphicsPageEmpty5}{\showpaste}
\tab{5}\spadcommand{closedCurve(space,[[0,140,20],[0,140,110],[0,150,110],[0,170,50], [0,190,110],[0,200,110
\end{paste}\end{patch}

\begin{patch}{3DObjectGraphicsPagePatch6}
\begin{paste}{3DObjectGraphicsPageFull6}{3DObjectGraphicsPageEmpty6}
\pastebutton{3DObjectGraphicsPageFull6}{\hidepaste}
\tab{5}\spadcommand{closedCurve(space,[[200,0,20], [200,0,110], [185,0,110], [160,0,45], [160,0,110], [150,0
\indentrel{3}\begin{verbatim}
   (6)  3-Space with 5 components
                          Type: ThreeSpace DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{3DObjectGraphicsPageEmpty6}
\begin{paste}{3DObjectGraphicsPageEmpty6}{3DObjectGraphicsPagePatch6}
```

```
\pastebutton{3DObjectGraphicsPageEmpty6}{\showpaste}
\tab{5}\spadcommand{closedCurve(space,[[200,0,20], [200,0,110], [185,0,110], [160,0,45], [160,0,
\end{paste}\end{patch}

\begin{patch}{3DObjectGraphicsPagePatch7}
\begin{paste}{3DObjectGraphicsPageFull7}{3DObjectGraphicsPageEmpty7}
\pastebutton{3DObjectGraphicsPageFull7}{\hidepaste}
\tab{5}\spadcommand{closedCurve(space,[[140,0,20], [120,0,110], [110,0,110], [90,0,20], [100,0,2
\indentrel{3}\begin{verbatim}
   (7)  3-Space with 6 components
                           Type: ThreeSpace DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{3DObjectGraphicsPageEmpty7}
\begin{paste}{3DObjectGraphicsPageEmpty7}{3DObjectGraphicsPagePatch7}
\pastebutton{3DObjectGraphicsPageEmpty7}{\showpaste}
\tab{5}\spadcommand{closedCurve(space,[[140,0,20], [120,0,110], [110,0,110], [90,0,20], [100,0,2
\end{paste}\end{patch}

\begin{patch}{3DObjectGraphicsPagePatch8}
\begin{paste}{3DObjectGraphicsPageFull8}{3DObjectGraphicsPageEmpty8}
\pastebutton{3DObjectGraphicsPageFull8}{\hidepaste}
\tab{5}\spadcommand{closedCurve(space,[[80,0,30], [80,0,100], [70,0,110], [40,0,110], [30,0,100]
\indentrel{3}\begin{verbatim}
   (8)  3-Space with 7 components
                           Type: ThreeSpace DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{3DObjectGraphicsPageEmpty8}
\begin{paste}{3DObjectGraphicsPageEmpty8}{3DObjectGraphicsPagePatch8}
\pastebutton{3DObjectGraphicsPageEmpty8}{\showpaste}
\tab{5}\spadcommand{closedCurve(space,[[80,0,30], [80,0,100], [70,0,110], [40,0,110], [30,0,100]
\end{paste}\end{patch}

\begin{patch}{3DObjectGraphicsPagePatch9}
\begin{paste}{3DObjectGraphicsPageFull9}{3DObjectGraphicsPageEmpty9}
\pastebutton{3DObjectGraphicsPageFull9}{\hidepaste}
\tab{5}\spadgraph{makeViewport3D(space,[title("Curves")])$VIEW3D\free{space curveI curveB1 curve
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/3dobjectgraphicspage9.view/image}}{vi
\end{paste}\end{patch}

\begin{patch}{3DObjectGraphicsPageEmpty9}
\begin{paste}{3DObjectGraphicsPageEmpty9}{3DObjectGraphicsPagePatch9}
\pastebutton{3DObjectGraphicsPageEmpty9}{\showpaste}
\tab{5}\spadgraph{makeViewport3D(space,[title("Curves")])$VIEW3D\free{space curveI curveB1 curve
\end{paste}\end{patch}
```

———-

Two Dimensional Graphics

⇒ "notitle" (OneVariableGraphicsPage) 3.50 on page 643
⇒ "notitle" (ParametricCurveGraphicsPage) 3.50 on page 646
⇒ "notitle" (PolarGraphicsPage) 3.50 on page 648
⇒ "notitle" (ImplicitCurveGraphicsPage) 3.50 on page 650
⇒ "notitle" (ListPointsGraphicsPage) 3.50 on page 652

— **graphics.ht** —

```
\begin{page}{TwoDimensionalGraphicsPage}{Two Dimensional Graphics}
\beginscroll
\beginmenu
\menulink{Functions of One Variable}{OneVariableGraphicsPage}
\newline
Plot curves defined by an equation y = f(x).
\menulink{Parametric Curves}{ParametricCurveGraphicsPage} \newline
Plot curves defined by parametric equations x = f(t), y = g(t).
\menulink{Polar Coordinates}{PolarGraphicsPage} \newline
Plot curves given in polar form by an equation r = f(theta).
\menulink{Implicit Curves}{ImplicitCurveGraphicsPage} \newline
Plot non-singular curves defined by a polynomial equation
\menulink{Lists of Points}{ListPointsGraphicsPage} \newline
Plot lists of points in the (x,y)-plane.
% \menulink{Sequences}{SeqGraphicsPage} \newline
% Plot a sequence a1, a2, a3,...
% \menulink{Complex Functions}{CxFuncGraphicsPage} \newline
% Plot functions of a complex variable using grid plots.
\endmenu
\endscroll
\autobuttons \end{page}
```

———-

Functions of One Variable

— **graphics.ht** —

```
\begin{page}{OneVariableGraphicsPage}{Functions of One Variable}
\beginscroll
Here we wish to plot a function y = f(x) on an interval [a,b].
As an example, let's take the function y = sin(tan(x)) - tan(sin(x))
on the interval [0,6].
```

```
Here is the simplest command that will do this:
\graphpaste{draw(sin(tan(x)) - tan(sin(x)),x = 0..6)}
Notice that Axiom compiled a function before the graph was put on the
screen.  The expression sin(tan(x)) - tan(sin(x)) was converted to a
compiled function so that it's value for various values of x could be
computed quickly and efficiently.  Let's graph the same function on a
different interval and this time we'll give the graph a title.  The
title is a String, which is an optional argument of the command
'draw'.
\graphpaste{draw(sin(tan(x)) - tan(sin(x)),x = 10..16,
title == "y = sin tan x - tan sin x")}
Once again the expression sin(tan(x)) - tan(sin(x)) was
converted to a compiled function before any points were computed.
If you want to graph the same function on a number of intervals, it's
a good idea to write down a function definition so that the function
only has to be compiled once.
Here's an example:
\spadpaste{f(x) == (x-1)*(x-2)*(x-3) \bound{f}}
\newline
\graphpaste{draw(f, 0..2, title == "y = f(x) on \[0,2\]") \free{f}}
\newline
\graphpaste{draw(f, 0..4,title == "y = f(x) on \[0,4\]") \free{f}}
Notice that our titles can be whatever we want, as long as they are
enclosed by double quotes.  However, a title which is too long to fit
within the viewport title window will be clipped.
\endscroll
\autobuttons
\end{page}

\begin{patch}{OneVariableGraphicsPagePatch1}
\begin{paste}{OneVariableGraphicsPageFull1}{OneVariableGraphicsPageEmpty1}
\pastebutton{OneVariableGraphicsPageFull1}{\hidepaste}
\tab{5}\spadgraph{draw(sin(tan(x)) - tan(sin(x)),x = 0..6)}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/onevariablegraphicspage1.view/image}}
\end{paste}\end{patch}

\begin{patch}{OneVariableGraphicsPageEmpty1}
\begin{paste}{OneVariableGraphicsPageEmpty1}{OneVariableGraphicsPagePatch1}
\pastebutton{OneVariableGraphicsPageEmpty1}{\showpaste}
\tab{5}\spadgraph{draw(sin(tan(x)) - tan(sin(x)),x = 0..6)}
\end{paste}\end{patch}

\begin{patch}{OneVariableGraphicsPagePatch2}
\begin{paste}{OneVariableGraphicsPageFull2}{OneVariableGraphicsPageEmpty2}
\pastebutton{OneVariableGraphicsPageFull2}{\hidepaste}
\tab{5}\spadgraph{draw(sin(tan(x)) - tan(sin(x)),x = 10..16,title == "y = sin tan x - tan sin x"
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/onevariablegraphicspage2.view/image}}
\end{paste}\end{patch}

\begin{patch}{OneVariableGraphicsPageEmpty2}
```

```
\begin{paste}{OneVariableGraphicsPageEmpty2}{OneVariableGraphicsPagePatch2}
\pastebutton{OneVariableGraphicsPageEmpty2}{\showpaste}
\tab{5}\spadgraph{draw(sin(tan(x)) - tan(sin(x)),x = 10..16,title == "y = sin tan x - tan sin x")}
\end{paste}\end{patch}

\begin{patch}{OneVariableGraphicsPagePatch3}
\begin{paste}{OneVariableGraphicsPageFull3}{OneVariableGraphicsPageEmpty3}
\pastebutton{OneVariableGraphicsPageFull3}{\hidepaste}
\tab{5}\spadcommand{f(x) == (x-1)*(x-2)*(x-3)\bound{f }}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OneVariableGraphicsPageEmpty3}
\begin{paste}{OneVariableGraphicsPageEmpty3}{OneVariableGraphicsPagePatch3}
\pastebutton{OneVariableGraphicsPageEmpty3}{\showpaste}
\tab{5}\spadcommand{f(x) == (x-1)*(x-2)*(x-3)\bound{f }}
\end{paste}\end{patch}

\begin{patch}{OneVariableGraphicsPagePatch4}
\begin{paste}{OneVariableGraphicsPageFull4}{OneVariableGraphicsPageEmpty4}
\pastebutton{OneVariableGraphicsPageFull4}{\hidepaste}
\tab{5}\spadgraph{draw(f, 0..2, title == "y = f(x) on [0,2]")\free{f }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/OneVariableGraphicsPage4.view/image}}{viewalone\s
\end{paste}\end{patch}

\begin{patch}{OneVariableGraphicsPageEmpty4}
\begin{paste}{OneVariableGraphicsPageEmpty4}{OneVariableGraphicsPagePatch4}
\pastebutton{OneVariableGraphicsPageEmpty4}{\showpaste}
\tab{5}\spadgraph{draw(f, 0..2, title == "y = f(x) on [0,2]")\free{f }}
\end{paste}\end{patch}

\begin{patch}{OneVariableGraphicsPagePatch5}
\begin{paste}{OneVariableGraphicsPageFull5}{OneVariableGraphicsPageEmpty5}
\pastebutton{OneVariableGraphicsPageFull5}{\hidepaste}
\tab{5}\spadgraph{draw(f, 0..4,title == "y = f(x) on [0,4]")\free{f }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/OneVariableGraphicsPage5.view/image}}{viewalone\s
\end{paste}\end{patch}

\begin{patch}{OneVariableGraphicsPageEmpty5}
\begin{paste}{OneVariableGraphicsPageEmpty5}{OneVariableGraphicsPagePatch5}
\pastebutton{OneVariableGraphicsPageEmpty5}{\showpaste}
\tab{5}\spadgraph{draw(f, 0..4,title == "y = f(x) on [0,4]")\free{f }}
\end{paste}\end{patch}
```

Parametric Curves

— graphics.ht —

```
\begin{page}{ParametricCurveGraphicsPage}{Parametric Curves}
\beginscroll
One way of producing interesting curves is by using parametric
equations.  Let x = f(t) and y = g(t) for two functions f and g as the
parameter t ranges over an interval \[a,b\].
Here's an example:
\graphpaste{draw(curve(sin(t)*sin(2*t)*sin(3*t),
sin(4*t)*sin(5*t)*sin(6*t)), t = 0..2*\%pi)}
Here 0..2*\%pi represents the interval over which the variable t ranges.
In the case of parametric curves, Axiom will compile two functions,
one for each of the functions f and g.
You may also put a title on a graph.
The title may be an arbitrary string and is an optional argument
to the command 'draw'.
For example:
\graphpaste{draw(curve(cos(t), sin(t)), t = 0..2*\%pi,
title == "The Unit Circle")}
If you plan on plotting x = f(t), y = g(t) as t ranges over several
intervals, you may want to define functions f and g, so that they need
not be recompiled every time you create a new graph.  Here's an
example:
\spadpaste{f(t:SF):SF == sin(3*t/4) \bound{f}}
\newline
\spadpaste{g(t:SF):SF == sin(t) \bound{g}}
\newline
\graphpaste{draw(curve(f,g), 0..\%pi) \free{f g}}
\newline
\graphpaste{draw(curve(f,g) ,\%pi..2*\%pi) \free{f g}}
\newline
\graphpaste{draw(curve(f,g), -4*\%pi..4*\%pi) \free{f g}}
These examples show how the curve changes as the range of
parameter t varies.
\endscroll
\autobuttons
\end{page}

\begin{patch}{ParametricCurveGraphicsPagePatch1}
\begin{paste}{ParametricCurveGraphicsPageFull1}{ParametricCurveGraphicsPageEmpty1}
\pastebutton{ParametricCurveGraphicsPageFull1}{\hidepaste}
\tab{5}\spadgraph{draw(curve(sin(t)*sin(2*t)*sin(3*t), sin(4*t)*sin(5*t)*sin(6*t)), t = 0..2*\%p
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/parametriccurvegraphicspage1.view/ima
\end{paste}\end{patch}

\begin{patch}{ParametricCurveGraphicsPageEmpty1}
\begin{paste}{ParametricCurveGraphicsPageEmpty1}{ParametricCurveGraphicsPagePatch1}
```

```
\pastebutton{ParametricCurveGraphicsPageEmpty1}{\showpaste}
\tab{5}\spadgraph{draw(curve(sin(t)*sin(2*t)*sin(3*t), sin(4*t)*sin(5*t)*sin(6*t)), t = 0..2*\%pi)}
\end{paste}\end{patch}

\begin{patch}{ParametricCurveGraphicsPagePatch2}
\begin{paste}{ParametricCurveGraphicsPageFull2}{ParametricCurveGraphicsPageEmpty2}
\pastebutton{ParametricCurveGraphicsPageFull2}{\hidepaste}
\tab{5}\spadgraph{draw(curve(cos(t), sin(t)), t = 0..2*\%pi, title == "The Unit Circle")}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/parametriccurvegraphicspage2.view/image}}{viewalo
\end{paste}\end{patch}

\begin{patch}{ParametricCurveGraphicsPageEmpty2}
\begin{paste}{ParametricCurveGraphicsPageEmpty2}{ParametricCurveGraphicsPagePatch2}
\pastebutton{ParametricCurveGraphicsPageEmpty2}{\showpaste}
\tab{5}\spadgraph{draw(curve(cos(t), sin(t)), t = 0..2*\%pi, title == "The Unit Circle")}
\end{paste}\end{patch}

\begin{patch}{ParametricCurveGraphicsPagePatch3}
\begin{paste}{ParametricCurveGraphicsPageFull3}{ParametricCurveGraphicsPageEmpty3}
\pastebutton{ParametricCurveGraphicsPageFull3}{\hidepaste}
\tab{5}\spadcommand{f(t:SF):SF == sin(3*t/4)\bound{f }}
\indentrel{3}\begin{verbatim}
                                        Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ParametricCurveGraphicsPageEmpty3}
\begin{paste}{ParametricCurveGraphicsPageEmpty3}{ParametricCurveGraphicsPagePatch3}
\pastebutton{ParametricCurveGraphicsPageEmpty3}{\showpaste}
\tab{5}\spadcommand{f(t:SF):SF == sin(3*t/4)\bound{f }}
\end{paste}\end{patch}

\begin{patch}{ParametricCurveGraphicsPagePatch4}
\begin{paste}{ParametricCurveGraphicsPageFull4}{ParametricCurveGraphicsPageEmpty4}
\pastebutton{ParametricCurveGraphicsPageFull4}{\hidepaste}
\tab{5}\spadcommand{g(t:SF):SF == sin(t)\bound{g }}
\indentrel{3}\begin{verbatim}
                                        Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ParametricCurveGraphicsPageEmpty4}
\begin{paste}{ParametricCurveGraphicsPageEmpty4}{ParametricCurveGraphicsPagePatch4}
\pastebutton{ParametricCurveGraphicsPageEmpty4}{\showpaste}
\tab{5}\spadcommand{g(t:SF):SF == sin(t)\bound{g }}
\end{paste}\end{patch}

\begin{patch}{ParametricCurveGraphicsPagePatch5}
\begin{paste}{ParametricCurveGraphicsPageFull5}{ParametricCurveGraphicsPageEmpty5}
\pastebutton{ParametricCurveGraphicsPageFull5}{\hidepaste}
```

```
\tab{5}\spadgraph{draw(curve(f,g), 0..\%pi)\free{f g }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/parametriccurvegraphicspage5.view/ima
\end{paste}\end{patch}

\begin{patch}{ParametricCurveGraphicsPageEmpty5}
\begin{paste}{ParametricCurveGraphicsPageEmpty5}{ParametricCurveGraphicsPagePatch5}
\pastebutton{ParametricCurveGraphicsPageEmpty5}{\showpaste}
\tab{5}\spadgraph{draw(curve(f,g), 0..\%pi)\free{f g }}
\end{paste}\end{patch}

\begin{patch}{ParametricCurveGraphicsPagePatch6}
\begin{paste}{ParametricCurveGraphicsPageFull6}{ParametricCurveGraphicsPageEmpty6}
\pastebutton{ParametricCurveGraphicsPageFull6}{\hidepaste}
\tab{5}\spadgraph{draw(curve(f,g) ,\%pi..2*\%pi)\free{f g }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/parametriccurvegraphicspage6.view/ima
\end{paste}\end{patch}

\begin{patch}{ParametricCurveGraphicsPageEmpty6}
\begin{paste}{ParametricCurveGraphicsPageEmpty6}{ParametricCurveGraphicsPagePatch6}
\pastebutton{ParametricCurveGraphicsPageEmpty6}{\showpaste}
\tab{5}\spadgraph{draw(curve(f,g) ,\%pi..2*\%pi)\free{f g }}
\end{paste}\end{patch}

\begin{patch}{ParametricCurveGraphicsPagePatch7}
\begin{paste}{ParametricCurveGraphicsPageFull7}{ParametricCurveGraphicsPageEmpty7}
\pastebutton{ParametricCurveGraphicsPageFull7}{\hidepaste}
\tab{5}\spadgraph{draw(curve(f,g), -4*\%pi..4*\%pi)\free{f g }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/parametriccurvegraphicspage7.view/ima
\end{paste}\end{patch}

\begin{patch}{ParametricCurveGraphicsPageEmpty7}
\begin{paste}{ParametricCurveGraphicsPageEmpty7}{ParametricCurveGraphicsPagePatch7}
\pastebutton{ParametricCurveGraphicsPageEmpty7}{\showpaste}
\tab{5}\spadgraph{draw(curve(f,g), -4*\%pi..4*\%pi)\free{f g }}
\end{paste}\end{patch}
```

———-

Polar Coordinates

— graphics.ht —

```
\begin{page}{PolarGraphicsPage}{Polar Coordinates}
\beginscroll
Graphs in polar coordinates are given by an equation r = f(theta) as
theta ranges over an interval.
This is equivalent to the parametric curve x = f(theta) * cos(theta),
```

```
y = f(theta) * sin(theta) as theta ranges over the same interval.
You may create such curves using the command 'draw', with the optional
argument 'coordinates == polar'.
Here are some examples:
\graphpaste{draw(1,t = 0..2*\%pi,coordinates == polar,
title == "The Unit Circle")}
\newline
\graphpaste{draw(sin(17*t), t = 0..2*\%pi, coordinates == polar,
title == "A Petal Curve")}
%When you don't specify an interval, Axiom will assume that you
%mean 0..2*\%pi.
You may also define your own functions, when you plan on plotting the
same curve as theta varies over several intervals.
\spadpaste{f(t) == cos(4*t/7) \bound{f}}
\newline
\graphpaste{draw(f, 0..2*\%pi, coordinates == polar) \free{f}}
\newline
\graphpaste{draw(f, 0..14*\%pi, coordinates == polar) \free{f}}
For information on plotting graphs in other coordinate systems see the
pages for the \spadtype{CoordinateSystems} domain.
\endscroll
\autobuttons
\end{page}

\begin{patch}{PolarGraphicsPagePatch1}
\begin{paste}{PolarGraphicsPageFull1}{PolarGraphicsPageEmpty1}
\pastebutton{PolarGraphicsPageFull1}{\hidepaste}
\tab{5}\spadgraph{draw(1,t = 0..2*\%pi,coordinates == polar, title == "The Unit Circle")}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/polargraphicspage1.view/image}}{viewalone\space{
\end{paste}\end{patch}

\begin{patch}{PolarGraphicsPageEmpty1}
\begin{paste}{PolarGraphicsPageEmpty1}{PolarGraphicsPagePatch1}
\pastebutton{PolarGraphicsPageEmpty1}{\showpaste}
\tab{5}\spadgraph{draw(1,t = 0..2*\%pi,coordinates == polar, title == "The Unit Circle")}
\end{paste}\end{patch}

\begin{patch}{PolarGraphicsPagePatch2}
\begin{paste}{PolarGraphicsPageFull2}{PolarGraphicsPageEmpty2}
\pastebutton{PolarGraphicsPageFull2}{\hidepaste}
\tab{5}\spadgraph{draw(sin(17*t), t = 0..2*\%pi, coordinates == polar, title == "A Petal Curve")}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/polargraphicspage2.view/image}}{viewalone\space{
\end{paste}\end{patch}

\begin{patch}{PolarGraphicsPageEmpty2}
\begin{paste}{PolarGraphicsPageEmpty2}{PolarGraphicsPagePatch2}
\pastebutton{PolarGraphicsPageEmpty2}{\showpaste}
\tab{5}\spadgraph{draw(sin(17*t), t = 0..2*\%pi, coordinates == polar, title == "A Petal Curve")}
\end{paste}\end{patch}
```

```
\begin{patch}{PolarGraphicsPagePatch3}
\begin{paste}{PolarGraphicsPageFull3}{PolarGraphicsPageEmpty3}
\pastebutton{PolarGraphicsPageFull3}{\hidepaste}
\tab{5}\spadcommand{f(t) == cos(4*t/7)\bound{f }}
\indentrel{3}\begin{verbatim}
                                          Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolarGraphicsPageEmpty3}
\begin{paste}{PolarGraphicsPageEmpty3}{PolarGraphicsPagePatch3}
\pastebutton{PolarGraphicsPageEmpty3}{\showpaste}
\tab{5}\spadcommand{f(t) == cos(4*t/7)\bound{f }}
\end{paste}\end{patch}

\begin{patch}{PolarGraphicsPagePatch4}
\begin{paste}{PolarGraphicsPageFull4}{PolarGraphicsPageEmpty4}
\pastebutton{PolarGraphicsPageFull4}{\hidepaste}
\tab{5}\spadgraph{draw(f, 0..2*\%pi, coordinates == polar)\free{f }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/polargraphicspage4.view/image}}{viewa
\end{paste}\end{patch}

\begin{patch}{PolarGraphicsPageEmpty4}
\begin{paste}{PolarGraphicsPageEmpty4}{PolarGraphicsPagePatch4}
\pastebutton{PolarGraphicsPageEmpty4}{\showpaste}
\tab{5}\spadgraph{draw(f, 0..2*\%pi, coordinates == polar)\free{f }}
\end{paste}\end{patch}

\begin{patch}{PolarGraphicsPagePatch5}
\begin{paste}{PolarGraphicsPageFull5}{PolarGraphicsPageEmpty5}
\pastebutton{PolarGraphicsPageFull5}{\hidepaste}
\tab{5}\spadgraph{draw(f, 0..14*\%pi, coordinates == polar)\free{f }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/polargraphicspage5.view/image}}{viewa
\end{paste}\end{patch}

\begin{patch}{PolarGraphicsPageEmpty5}
\begin{paste}{PolarGraphicsPageEmpty5}{PolarGraphicsPagePatch5}
\pastebutton{PolarGraphicsPageEmpty5}{\showpaste}
\tab{5}\spadgraph{draw(f, 0..14*\%pi, coordinates == polar)\free{f }}
\end{paste}\end{patch}
```

———-

Implicit Curves

— graphics.ht —

```
\begin{page}{ImplicitCurveGraphicsPage}{Implicit Curves}
\beginscroll
Axiom has facilities for graphing a non-singular algebraic curve
in a rectangular region of the plane.
An algebraic curve is a curve defined by a polynomial equation
p(x,y) = 0.
Non-singular means that the curve is "smooth" in that it does not
cross itself or come to a point (cusp).
Algebraically, this means that for any point (a,b) on the curve
(i.e. a point such that p(a,b) = 0), the partial derivatives
dp/dx(a,b) and dp/dy(a,b) are not both zero.
We require that the polynomial have rational or integral coefficients.
Here is a Cartesian ovals algebraic curve example:
(click on the draw button to execute this example)
\spadpaste{p := ((x**2 + y**2 + 1) - 8*x)**2 -
(8*(x**2 + y**2 + 1) - 4*x - 1) \bound{p} }
\graphpaste{draw(p = 0, x, y, range == \[-1..11, -7..7\],
title == "Cartesian Ovals") \free{p}}
{\em A range must be declared for each variable specified in the
algebraic curve equation}.
\endscroll
\autobuttons
\end{page}

\begin{patch}{ImplicitCurveGraphicsPagePatch1}
\begin{paste}{ImplicitCurveGraphicsPageFull1}{ImplicitCurveGraphicsPageEmpty1}
\pastebutton{ImplicitCurveGraphicsPageFull1}{\hidepaste}
\tab{5}\spadcommand{p := ((x**2 + y**2 + 1) - 8*x)**2 - (8*(x**2 + y**2 + 1) - 4*x - 1)\bound{p }}
\indentrel{3}\begin{verbatim}
   (1)
    4      2            2    4      3      2
   y  + (2x  - 16x - 6)y  + x  - 16x  + 58x  - 12x - 6
                                 Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ImplicitCurveGraphicsPageEmpty1}
\begin{paste}{ImplicitCurveGraphicsPageEmpty1}{ImplicitCurveGraphicsPagePatch1}
\pastebutton{ImplicitCurveGraphicsPageEmpty1}{\showpaste}
\tab{5}\spadcommand{p := ((x**2 + y**2 + 1) - 8*x)**2 - (8*(x**2 + y**2 + 1) - 4*x - 1)\bound{p }}
\end{paste}\end{patch}

\begin{patch}{ImplicitCurveGraphicsPagePatch2}
\begin{paste}{ImplicitCurveGraphicsPageFull2}{ImplicitCurveGraphicsPageEmpty2}
\pastebutton{ImplicitCurveGraphicsPageFull2}{\hidepaste}
\tab{5}\spadgraph{draw(p = 0, x, y, range == [-1..11, -7..7], title == "Cartesian Ovals")\free{p }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/implicitcurvegraphicspage2.view/image}}{viewalone
\end{paste}\end{patch}

\begin{patch}{ImplicitCurveGraphicsPageEmpty2}
```

```
\begin{paste}{ImplicitCurveGraphicsPageEmpty2}{ImplicitCurveGraphicsPagePatch2}
\pastebutton{ImplicitCurveGraphicsPageEmpty2}{\showpaste}
\tab{5}\spadgraph{draw(p = 0, x, y, range == [-1..11, -7..7], title == "Cartesian Ovals")\free{p
\end{paste}\end{patch}
```

———-

Lists of Points

— graphics.ht —

```
\begin{page}{ListPointsGraphicsPage}{Lists of Points}
\beginscroll
Axiom has the ability to create lists of points in a two dimensional
graphics viewport.  This is done by utilizing the \spadtype{GraphImage}
and \spadtype{TwoDimensionalViewport} domain facilities.
\indent{5}\newline
{\em NOTE:  It is only necessary to click on the makeViewport2D command
button to plot this curve example}.
\indent{0}\newline
In this example, the \spadfun{makeGraphImage} command takes a list of
lists of points parameter, a list of colors for each point in the
graph, a list of colors for each line in the graph, and a list of
numbers which indicate the size of each point in the graph.  The
following lines create list of lists of points which can be read be
made into two dimensional graph images.
\spadpaste{p1 := point [1::SF,1::SF]\$(Point SF) \bound{p1}}
\newline
\spadpaste{p2 := point [0::SF,1::SF]\$(Point SF) \bound{p2}}
\newline
\spadpaste{p3 := point [0::SF,0::SF]\$(Point SF) \bound{p3}}
\newline
\spadpaste{p4 := point [1::SF,0::SF]\$(Point SF) \bound{p4}}
\newline
\spadpaste{p5 := point [1::SF,.5::SF]\$(Point SF) \bound{p5}}
\newline
\spadpaste{p6 := point [.5::SF,0::SF]\$(Point SF) \bound{p6}}
\newline
\spadpaste{p7 := point [0::SF,0.5::SF]\$(Point SF) \bound{p7}}
\newline
\spadpaste{p8 := point [.5::SF,1::SF]\$(Point SF) \bound{p8}}
\newline
\spadpaste{p9 := point [.25::SF,.25::SF]\$(Point SF) \bound{p9}}
\newline
\spadpaste{p10 := point [.25::SF,.75::SF]\$(Point SF) \bound{p10}}
\newline
\spadpaste{p11 := point [.75::SF,.75::SF]\$(Point SF) \bound{p11}}
```

```
\newline
\spadpaste{p12 := point [.75::SF,.25::SF]\$(Point SF) \bound{p12}}
\newline
\spadpaste{llp := [[p1,p2],[p2,p3],[p3,p4],[p4,p1],[p5,p6],[p6,p7],
[p7,p8],[p8,p5],[p9,p10],[p10,p11],[p11,p12],[p12,p9]] \bound{llp}
\free{p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 p11 p12}}
\newline
These lines set the point color and size, and the line color for
all components of the graph.
\spadpaste{size1 := 6::PositiveInteger \bound{size1}}
\newline
\spadpaste{size2 := 8::PositiveInteger \bound{size2}}
\newline
\spadpaste{size3 := 10::PositiveInteger \bound{size3}}
\newline
\spadpaste{lsize := [size1, size1, size1, size1, size2, size2, size2,
size2, size3, size3, size3, size3] \bound{lsize}
\free{size1 size2 size3}}
\newline
\spadpaste{pc1 := pastel red() \bound{pc1}}
\newline
\spadpaste{pc2 := dim green() \bound{pc2}}
\newline
\spadpaste{pc3 := pastel yellow() \bound{pc3}}
\newline
\spadpaste{lpc := [pc1, pc1, pc1, pc1, pc2, pc2, pc2, pc2, pc3, pc3,
pc3, pc3] \bound{lpc} \free{pc1 pc2 pc3}}
\newline
\spadpaste{lc := [pastel blue(), light yellow(), dim green(),
bright red(), light green(), dim yellow(), bright blue(),
dark red(), pastel red(), light blue(), dim green(),
light yellow()] \bound{lc}}
\newline
Now the graph image is created and named according to the component
specifications indicated above.  The \spadfun{makeViewport2D} command
then creates a two dimensional viewport for this graph according to
the list of options specified within the brackets.
\spadpaste{g := makeGraphImage(llp,lpc,lc,lsize)\$GRIMAGE \bound{g}
\free{llp lpc lc lsize}}
\newline
\graphpaste{makeViewport2D(g,[title("Lines")])\$VIEW2D \free{g}}
The \spadfun{makeViewport2D} command takes a list of options as a
parameter.  In this example the string "Lines" is designated as the
viewport's title.
\endscroll
\autobuttons
\end{page}

\begin{patch}{ListPointsGraphicsPagePatch1}
\begin{paste}{ListPointsGraphicsPageFull1}{ListPointsGraphicsPageEmpty1}
```

```
\pastebutton{ListPointsGraphicsPageFull1}{\hidepaste}
\tab{5}\spadcommand{p1 := point [1::SF,1::SF]$(Point SF)\bound{p1 }}
\indentrel{3}\begin{verbatim}
   (1)  [1.0,1.0]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPageEmpty1}
\begin{paste}{ListPointsGraphicsPageEmpty1}{ListPointsGraphicsPagePatch1}
\pastebutton{ListPointsGraphicsPageEmpty1}{\showpaste}
\tab{5}\spadcommand{p1 := point [1::SF,1::SF]$(Point SF)\bound{p1 }}
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPagePatch2}
\begin{paste}{ListPointsGraphicsPageFull2}{ListPointsGraphicsPageEmpty2}
\pastebutton{ListPointsGraphicsPageFull2}{\hidepaste}
\tab{5}\spadcommand{p2 := point [0::SF,1::SF]$(Point SF)\bound{p2 }}
\indentrel{3}\begin{verbatim}
   (2)  [0.0,1.0]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPageEmpty2}
\begin{paste}{ListPointsGraphicsPageEmpty2}{ListPointsGraphicsPagePatch2}
\pastebutton{ListPointsGraphicsPageEmpty2}{\showpaste}
\tab{5}\spadcommand{p2 := point [0::SF,1::SF]$(Point SF)\bound{p2 }}
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPagePatch3}
\begin{paste}{ListPointsGraphicsPageFull3}{ListPointsGraphicsPageEmpty3}
\pastebutton{ListPointsGraphicsPageFull3}{\hidepaste}
\tab{5}\spadcommand{p3 := point [0::SF,0::SF]$(Point SF)\bound{p3 }}
\indentrel{3}\begin{verbatim}
   (3)  [0.0,0.0]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPageEmpty3}
\begin{paste}{ListPointsGraphicsPageEmpty3}{ListPointsGraphicsPagePatch3}
\pastebutton{ListPointsGraphicsPageEmpty3}{\showpaste}
\tab{5}\spadcommand{p3 := point [0::SF,0::SF]$(Point SF)\bound{p3 }}
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPagePatch4}
\begin{paste}{ListPointsGraphicsPageFull4}{ListPointsGraphicsPageEmpty4}
\pastebutton{ListPointsGraphicsPageFull4}{\hidepaste}
\tab{5}\spadcommand{p4 := point [1::SF,0::SF]$(Point SF)\bound{p4 }}
```

```
\indentrel{3}\begin{verbatim}
   (4)  [1.0,0.0]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPageEmpty4}
\begin{paste}{ListPointsGraphicsPageEmpty4}{ListPointsGraphicsPagePatch4}
\pastebutton{ListPointsGraphicsPageEmpty4}{\showpaste}
\tab{5}\spadcommand{p4 := point [1::SF,0::SF]$(Point SF)\bound{p4 }}
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPagePatch5}
\begin{paste}{ListPointsGraphicsPageFull5}{ListPointsGraphicsPageEmpty5}
\pastebutton{ListPointsGraphicsPageFull5}{\hidepaste}
\tab{5}\spadcommand{p5 := point [1::SF,.5::SF]$(Point SF)\bound{p5 }}
\indentrel{3}\begin{verbatim}
   (5)  [1.0,0.5]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPageEmpty5}
\begin{paste}{ListPointsGraphicsPageEmpty5}{ListPointsGraphicsPagePatch5}
\pastebutton{ListPointsGraphicsPageEmpty5}{\showpaste}
\tab{5}\spadcommand{p5 := point [1::SF,.5::SF]$(Point SF)\bound{p5 }}
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPagePatch6}
\begin{paste}{ListPointsGraphicsPageFull6}{ListPointsGraphicsPageEmpty6}
\pastebutton{ListPointsGraphicsPageFull6}{\hidepaste}
\tab{5}\spadcommand{p6 := point [.5::SF,0::SF]$(Point SF)\bound{p6 }}
\indentrel{3}\begin{verbatim}
   (6)  [0.5,0.0]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPageEmpty6}
\begin{paste}{ListPointsGraphicsPageEmpty6}{ListPointsGraphicsPagePatch6}
\pastebutton{ListPointsGraphicsPageEmpty6}{\showpaste}
\tab{5}\spadcommand{p6 := point [.5::SF,0::SF]$(Point SF)\bound{p6 }}
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPagePatch7}
\begin{paste}{ListPointsGraphicsPageFull7}{ListPointsGraphicsPageEmpty7}
\pastebutton{ListPointsGraphicsPageFull7}{\hidepaste}
\tab{5}\spadcommand{p7 := point [0::SF,0.5::SF]$(Point SF)\bound{p7 }}
\indentrel{3}\begin{verbatim}
   (7)  [0.0,0.5]
```

```
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPageEmpty7}
\begin{paste}{ListPointsGraphicsPageEmpty7}{ListPointsGraphicsPagePatch7}
\pastebutton{ListPointsGraphicsPageEmpty7}{\showpaste}
\tab{5}\spadcommand{p7 := point [0::SF,0.5::SF]$(Point SF)\bound{p7 }}
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPagePatch8}
\begin{paste}{ListPointsGraphicsPageFull8}{ListPointsGraphicsPageEmpty8}
\pastebutton{ListPointsGraphicsPageFull8}{\hidepaste}
\tab{5}\spadcommand{p8 := point [.5::SF,1::SF]$(Point SF)\bound{p8 }}
\indentrel{3}\begin{verbatim}
   (8)  [0.5,1.0]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPageEmpty8}
\begin{paste}{ListPointsGraphicsPageEmpty8}{ListPointsGraphicsPagePatch8}
\pastebutton{ListPointsGraphicsPageEmpty8}{\showpaste}
\tab{5}\spadcommand{p8 := point [.5::SF,1::SF]$(Point SF)\bound{p8 }}
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPagePatch9}
\begin{paste}{ListPointsGraphicsPageFull9}{ListPointsGraphicsPageEmpty9}
\pastebutton{ListPointsGraphicsPageFull9}{\hidepaste}
\tab{5}\spadcommand{p9 := point [.25::SF,.25::SF]$(Point SF)\bound{p9 }}
\indentrel{3}\begin{verbatim}
   (9)  [0.25,0.25]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPageEmpty9}
\begin{paste}{ListPointsGraphicsPageEmpty9}{ListPointsGraphicsPagePatch9}
\pastebutton{ListPointsGraphicsPageEmpty9}{\showpaste}
\tab{5}\spadcommand{p9 := point [.25::SF,.25::SF]$(Point SF)\bound{p9 }}
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPagePatch10}
\begin{paste}{ListPointsGraphicsPageFull10}{ListPointsGraphicsPageEmpty10}
\pastebutton{ListPointsGraphicsPageFull10}{\hidepaste}
\tab{5}\spadcommand{p10 := point [.25::SF,.75::SF]$(Point SF)\bound{p10 }}
\indentrel{3}\begin{verbatim}
   (10)  [0.25,0.75]
                                Type: Point DoubleFloat
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPageEmpty10}
\begin{paste}{ListPointsGraphicsPageEmpty10}{ListPointsGraphicsPagePatch10}
\pastebutton{ListPointsGraphicsPageEmpty10}{\showpaste}
\tab{5}\spadcommand{p10 := point [.25::SF,.75::SF]$(Point SF)\bound{p10 }}
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPagePatch11}
\begin{paste}{ListPointsGraphicsPageFull11}{ListPointsGraphicsPageEmpty11}
\pastebutton{ListPointsGraphicsPageFull11}{\hidepaste}
\tab{5}\spadcommand{p11 := point [.75::SF,.75::SF]$(Point SF)\bound{p11 }}
\indentrel{3}\begin{verbatim}
   (11)  [0.75,0.75]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPageEmpty11}
\begin{paste}{ListPointsGraphicsPageEmpty11}{ListPointsGraphicsPagePatch11}
\pastebutton{ListPointsGraphicsPageEmpty11}{\showpaste}
\tab{5}\spadcommand{p11 := point [.75::SF,.75::SF]$(Point SF)\bound{p11 }}
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPagePatch12}
\begin{paste}{ListPointsGraphicsPageFull12}{ListPointsGraphicsPageEmpty12}
\pastebutton{ListPointsGraphicsPageFull12}{\hidepaste}
\tab{5}\spadcommand{p12 := point [.75::SF,.25::SF]$(Point SF)\bound{p12 }}
\indentrel{3}\begin{verbatim}
   (12)  [0.75,0.25]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPageEmpty12}
\begin{paste}{ListPointsGraphicsPageEmpty12}{ListPointsGraphicsPagePatch12}
\pastebutton{ListPointsGraphicsPageEmpty12}{\showpaste}
\tab{5}\spadcommand{p12 := point [.75::SF,.25::SF]$(Point SF)\bound{p12 }}
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPagePatch13}
\begin{paste}{ListPointsGraphicsPageFull13}{ListPointsGraphicsPageEmpty13}
\pastebutton{ListPointsGraphicsPageFull13}{\hidepaste}
\tab{5}\spadcommand{llp := [[p1,p2],[p2,p3],[p3,p4],[p4,p1],[p5,p6],[p6,p7],[p7,p8],[p8,p5],[p9,p10],[p10,p
\indentrel{3}\begin{verbatim}
   (13)
   [[[1.0,1.0],[0.0,1.0]], [[0.0,1.0],[0.0,0.0]],
    [[0.0,0.0],[1.0,0.0]], [[1.0,0.0],[1.0,1.0]],
    [[1.0,0.5],[0.5,0.0]], [[0.5,0.0],[0.0,0.5]],
    [[0.0,0.5],[0.5,1.0]], [[0.5,1.0],[1.0,0.5]],
```

```
    [[0.25,0.25],[0.25,0.75]], [[0.25,0.75],[0.75,0.75]],
    [[0.75,0.75],[0.75,0.25]], [[0.75,0.25],[0.25,0.25]]]
                      Type: List List Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPageEmpty13}
\begin{paste}{ListPointsGraphicsPageEmpty13}{ListPointsGraphicsPagePatch13}
\pastebutton{ListPointsGraphicsPageEmpty13}{\showpaste}
\tab{5}\spadcommand{llp := [[p1,p2],[p2,p3],[p3,p4],[p4,p1],[p5,p6],[p6,p7],[p7,p8],[p8,p5],[p9,
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPagePatch14}
\begin{paste}{ListPointsGraphicsPageFull14}{ListPointsGraphicsPageEmpty14}
\pastebutton{ListPointsGraphicsPageFull14}{\hidepaste}
\tab{5}\spadcommand{size1 := 6::PositiveInteger\bound{size1 }}
\indentrel{3}\begin{verbatim}
   (14)  6
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPageEmpty14}
\begin{paste}{ListPointsGraphicsPageEmpty14}{ListPointsGraphicsPagePatch14}
\pastebutton{ListPointsGraphicsPageEmpty14}{\showpaste}
\tab{5}\spadcommand{size1 := 6::PositiveInteger\bound{size1 }}
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPagePatch15}
\begin{paste}{ListPointsGraphicsPageFull15}{ListPointsGraphicsPageEmpty15}
\pastebutton{ListPointsGraphicsPageFull15}{\hidepaste}
\tab{5}\spadcommand{size2 := 8::PositiveInteger\bound{size2 }}
\indentrel{3}\begin{verbatim}
   (15)  8
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPageEmpty15}
\begin{paste}{ListPointsGraphicsPageEmpty15}{ListPointsGraphicsPagePatch15}
\pastebutton{ListPointsGraphicsPageEmpty15}{\showpaste}
\tab{5}\spadcommand{size2 := 8::PositiveInteger\bound{size2 }}
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPagePatch16}
\begin{paste}{ListPointsGraphicsPageFull16}{ListPointsGraphicsPageEmpty16}
\pastebutton{ListPointsGraphicsPageFull16}{\hidepaste}
\tab{5}\spadcommand{size3 := 10::PositiveInteger\bound{size3 }}
\indentrel{3}\begin{verbatim}
   (16)  10
```

```
                                 Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPageEmpty16}
\begin{paste}{ListPointsGraphicsPageEmpty16}{ListPointsGraphicsPagePatch16}
\pastebutton{ListPointsGraphicsPageEmpty16}{\showpaste}
\tab{5}\spadcommand{size3 := 10::PositiveInteger\bound{size3 }}
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPagePatch17}
\begin{paste}{ListPointsGraphicsPageFull17}{ListPointsGraphicsPageEmpty17}
\pastebutton{ListPointsGraphicsPageFull17}{\hidepaste}
\tab{5}\spadcommand{lsize := [size1, size1, size1, size1, size2, size2, size2, size2, size3, size3, size3, s
\indentrel{3}\begin{verbatim}
   (17)  [6,6,6,6,8,8,8,8,10,10,10,10]
                            Type: List PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPageEmpty17}
\begin{paste}{ListPointsGraphicsPageEmpty17}{ListPointsGraphicsPagePatch17}
\pastebutton{ListPointsGraphicsPageEmpty17}{\showpaste}
\tab{5}\spadcommand{lsize := [size1, size1, size1, size1, size2, size2, size2, size2, size3, size3, size3, s
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPagePatch18}
\begin{paste}{ListPointsGraphicsPageFull18}{ListPointsGraphicsPageEmpty18}
\pastebutton{ListPointsGraphicsPageFull18}{\hidepaste}
\tab{5}\spadcommand{pc1 := pastel red()\bound{pc1 }}
\indentrel{3}\begin{verbatim}
   (18)  [Hue: 1  Weight: 1.0] from the Pastel palette
                                         Type: Palette
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPageEmpty18}
\begin{paste}{ListPointsGraphicsPageEmpty18}{ListPointsGraphicsPagePatch18}
\pastebutton{ListPointsGraphicsPageEmpty18}{\showpaste}
\tab{5}\spadcommand{pc1 := pastel red()\bound{pc1 }}
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPagePatch19}
\begin{paste}{ListPointsGraphicsPageFull19}{ListPointsGraphicsPageEmpty19}
\pastebutton{ListPointsGraphicsPageFull19}{\hidepaste}
\tab{5}\spadcommand{pc2 := dim green()\bound{pc2 }}
\indentrel{3}\begin{verbatim}
   (19)  [Hue: 14  Weight: 1.0] from the Dim palette
                                         Type: Palette
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPageEmpty19}
\begin{paste}{ListPointsGraphicsPageEmpty19}{ListPointsGraphicsPagePatch19}
\pastebutton{ListPointsGraphicsPageEmpty19}{\showpaste}
\tab{5}\spadcommand{pc2 := dim green()\bound{pc2 }}
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPagePatch20}
\begin{paste}{ListPointsGraphicsPageFull20}{ListPointsGraphicsPageEmpty20}
\pastebutton{ListPointsGraphicsPageFull20}{\hidepaste}
\tab{5}\spadcommand{pc3 := pastel yellow()\bound{pc3 }}
\indentrel{3}\begin{verbatim}
   (20)  [Hue: 11  Weight: 1.0] from the Pastel palette
                                          Type: Palette
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPageEmpty20}
\begin{paste}{ListPointsGraphicsPageEmpty20}{ListPointsGraphicsPagePatch20}
\pastebutton{ListPointsGraphicsPageEmpty20}{\showpaste}
\tab{5}\spadcommand{pc3 := pastel yellow()\bound{pc3 }}
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPagePatch21}
\begin{paste}{ListPointsGraphicsPageFull21}{ListPointsGraphicsPageEmpty21}
\pastebutton{ListPointsGraphicsPageFull21}{\hidepaste}
\tab{5}\spadcommand{lpc := [pc1, pc1, pc1, pc1, pc2, pc2, pc2, pc2, pc3, pc3, pc3, pc3]\bound{lp
\indentrel{3}\begin{verbatim}
   (21)
   [[Hue: 1  Weight: 1.0] from the Pastel palette,
    [Hue: 1  Weight: 1.0] from the Pastel palette,
    [Hue: 1  Weight: 1.0] from the Pastel palette,
    [Hue: 1  Weight: 1.0] from the Pastel palette,
    [Hue: 14  Weight: 1.0] from the Dim palette,
    [Hue: 14  Weight: 1.0] from the Dim palette,
    [Hue: 14  Weight: 1.0] from the Dim palette,
    [Hue: 14  Weight: 1.0] from the Dim palette,
    [Hue: 11  Weight: 1.0] from the Pastel palette,
    [Hue: 11  Weight: 1.0] from the Pastel palette,
    [Hue: 11  Weight: 1.0] from the Pastel palette,
    [Hue: 11  Weight: 1.0] from the Pastel palette]
                                     Type: List Palette
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPageEmpty21}
\begin{paste}{ListPointsGraphicsPageEmpty21}{ListPointsGraphicsPagePatch21}
\pastebutton{ListPointsGraphicsPageEmpty21}{\showpaste}
\tab{5}\spadcommand{lpc := [pc1, pc1, pc1, pc1, pc2, pc2, pc2, pc2, pc3, pc3, pc3, pc3]\bound{lp
```

```
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPagePatch22}
\begin{paste}{ListPointsGraphicsPageFull22}{ListPointsGraphicsPageEmpty22}
\pastebutton{ListPointsGraphicsPageFull22}{\hidepaste}
\tab{5}\spadcommand{lc := [pastel blue(), light yellow(), dim green(), bright red(), light green(), dim yel
\indentrel{3}\begin{verbatim}
   (22)
   [[Hue: 22  Weight: 1.0] from the Pastel palette,
    [Hue: 11  Weight: 1.0] from the Light palette,
    [Hue: 14  Weight: 1.0] from the Dim palette,
    [Hue: 1  Weight: 1.0] from the Bright palette,
    [Hue: 14  Weight: 1.0] from the Light palette,
    [Hue: 11  Weight: 1.0] from the Dim palette,
    [Hue: 22  Weight: 1.0] from the Bright palette,
    [Hue: 1  Weight: 1.0] from the Dark palette,
    [Hue: 1  Weight: 1.0] from the Pastel palette,
    [Hue: 22  Weight: 1.0] from the Light palette,
    [Hue: 14  Weight: 1.0] from the Dim palette,
    [Hue: 11  Weight: 1.0] from the Light palette]
                                      Type: List Palette
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPageEmpty22}
\begin{paste}{ListPointsGraphicsPageEmpty22}{ListPointsGraphicsPagePatch22}
\pastebutton{ListPointsGraphicsPageEmpty22}{\showpaste}
\tab{5}\spadcommand{lc := [pastel blue(), light yellow(), dim green(), bright red(), light green(), dim yel
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPagePatch23}
\begin{paste}{ListPointsGraphicsPageFull23}{ListPointsGraphicsPageEmpty23}
\pastebutton{ListPointsGraphicsPageFull23}{\hidepaste}
\tab{5}\spadcommand{g := makeGraphImage(llp,lpc,lc,lsize)$GRIMAGE\bound{g }\free{llp lpc lc lsize }}
\indentrel{3}\begin{verbatim}
   (23)  Graph with 12 point lists
                                        Type: GraphImage
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPageEmpty23}
\begin{paste}{ListPointsGraphicsPageEmpty23}{ListPointsGraphicsPagePatch23}
\pastebutton{ListPointsGraphicsPageEmpty23}{\showpaste}
\tab{5}\spadcommand{g := makeGraphImage(llp,lpc,lc,lsize)$GRIMAGE\bound{g }\free{llp lpc lc lsize }}
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPagePatch24}
\begin{paste}{ListPointsGraphicsPageFull24}{ListPointsGraphicsPageEmpty24}
\pastebutton{ListPointsGraphicsPageFull24}{\hidepaste}
\tab{5}\spadgraph{makeViewport2D(g,[title("Lines")])$VIEW2D\free{g }}
```

```
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/listpointsgraphicspage24.view/image}}
\end{paste}\end{patch}

\begin{patch}{ListPointsGraphicsPageEmpty24}
\begin{paste}{ListPointsGraphicsPageEmpty24}{ListPointsGraphicsPagePatch24}
\pastebutton{ListPointsGraphicsPageEmpty24}{\showpaste}
\tab{5}\spadgraph{makeViewport2D(g,[title("Lines")])$VIEW2D\free{g }}
\end{paste}\end{patch}
```

———-

Stand-alone Viewport

— graphics.ht —

```
\begin{page}{ViewportPage}{Stand-alone Viewport}
\beginscroll
To get a viewport on a Hyperdoc page, you first need to
create one in Axiom and write it out to a
file that Hyperdoc can call up. \newline
For example, here we draw a saddle function and assign
the result to the variable \spad{v}:
\newline
\graphpaste{v := draw(x*x-y*y,x=-1..1,y=-1..1) \bound{v}}
\newline
Now that we've created the viewport, we want to write
the data out to a file. \newline
To do this, we use the \spadfunFrom{write}{ThreeDimensionalViewport}
command which takes the following arguments: the viewport to write
out, the title of the file to write it out to, and an optional
argument telling the write command what type (or types) of data you
want to write in additional to the one Axiom will always write
out. The optional argument could be a string, like "pixmap", or a list
of strings, like \["postscript","pixmap"\].  Hyperdoc needs a
"pixmap" data type to include a graph in a page so in this case, we
write the viewport and tell it to also write a "pixmap" file, as well:
\newline
\spadpaste{write(v,"saddle","pixmap") \free{v}}
\newline
Now we want to put this viewport into a Hyperdoc page.
Say you've created a viewport and written it out
to a file called "/tmp/mobius". (Axiom actually
tags a ".view" at the end of a viewport data file to
make it easier to spot when you're rummaging through
your file system, but you needn't worry about that here
since Axiom will always automatically add on a
".view" for you.)  \newline
```

```
{\bf Including Viewports} \newline
To put a viewport in a
Hyperdoc page, include the following line in your Hyperdoc
source code: \newline
\space{5}\\viewport\{/tmp/mobius\} \newline
You will get this on your page: \newline
%%%\space{4}\viewport{/tmp/mobius}\newline
%\space{4}\spadviewport{mobius}\newline
\centerline{\spadviewport{mobius}}\newline

{\bf Creating Viewport Buttons} \newline
To make an active button that would make this viewport
come to life, include the following: \newline
\space{5}\\viewportbutton\{ViewButton\}\{/tmp/mobius\} \newline
this creates this button...\newline
%%%\centerline{\viewportbutton{ViewButton}{/tmp/mobius}}\newline
\centerline{\spadviewportbutton{ViewButton}{mobius}}\newline

{\bf Creating Active Viewports} \newline
To merge the two things descibed above, namely, getting a picture of a
viewport and creating a button to invoke a live viewport, you can do
the following: \newline

%\space{5}\\viewportasbutton\{/tmp/mobius\} \newline
\centerline{\\viewportasbutton\{/tmp/mobius\}}\newline

This would create a picture of a viewport that is an active
button as well. Try it:

%%%\space{5}\viewportasbutton{/tmp/mobius} \newline
%\space{5}\spadviewportasbutton{mobius} \newline
\centerline{\spadviewportasbutton{mobius}}\newline

{\bf Including Viewports Distributed with Axiom} \newline
All the above commands have counterparts that allow you to
access viewports that are already packaged with Axiom.
To include those viewports, just add on an "axiom" prefix
to the above commands: \newline

\centerline{\\axiomviewport\{vA\} for \\viewport\{vA\}}
\centerline{\\axiomviewportbutton\{vB\} for \\viewportbutton\{vB\}}
\centerline{\\axiomviewportasbutton\{vC\} for \\viewportasbutton\{vC\}}
\newline

All these macros really do is include some path that
indicates where Axiom stores the viewports.
\newline
\endscroll
\autobuttons
```

```
\end{page}

\begin{patch}{ViewportPagePatch1}
\begin{paste}{ViewportPageFull1}{ViewportPageEmpty1}
\pastebutton{ViewportPageFull1}{\hidepaste}
\tab{5}\spadgraph{v := draw(x*x-y*y,x=-1..1,y=-1..1)\bound{v }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/viewportpage1.view/image}}{viewalone\
\end{paste}\end{patch}

\begin{patch}{ViewportPageEmpty1}
\begin{paste}{ViewportPageEmpty1}{ViewportPagePatch1}
\pastebutton{ViewportPageEmpty1}{\showpaste}
\tab{5}\spadgraph{v := draw(x*x-y*y,x=-1..1,y=-1..1)\bound{v }}
\end{paste}\end{patch}

\begin{patch}{ViewportPagePatch2}
\begin{paste}{ViewportPageFull2}{ViewportPageEmpty2}
\pastebutton{ViewportPageFull2}{\hidepaste}
\tab{5}\spadcommand{write(v,"saddle","pixmap")\free{v }}
\indentrel{3}\begin{verbatim}
   (2)  "NIL"
                                            Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ViewportPageEmpty2}
\begin{paste}{ViewportPageEmpty2}{ViewportPagePatch2}
\pastebutton{ViewportPageEmpty2}{\showpaste}
\tab{5}\spadcommand{write(v,"saddle","pixmap")\free{v }}
\end{paste}\end{patch}
```

———-

3.51 grpthry.ht

Group Theory

⇒ "notitle" (InfoGroupTheoryPage) 3.51 on page 685
⇒ "notitle" (InfoRepTheoryPage) 3.51 on page 684
⇒ "notitle" (RepA6Page) 3.51 on page 665

— **grpthry.ht** —

```
\begin{page}{GroupTheoryPage}{Group Theory}
% authors: H. Gollan, J. Grabmeier, August 1989
\beginscroll
```

```
Axiom can work with individual permutations, permutation
groups and do representation theory.
\horizontalline
\newline
\beginmenu
\menulink{Info on Group Theory}{InfoGroupTheoryPage}


%\menulink{Permutation}{PermutationXmpPage}
%Calculating within symmetric groups.

%\menulink{Permutation Groups}{PermutationGroupXmpPage}
%Working with subgroups of a symmetric group.

%\menulink{Permutation Group Examples}{PermutationGroupExampleXmpPage}
%Working with permutation groups, predefined in the system as Rubik's
%group.

\menulink{Info on Representation Theory}{InfoRepTheoryPage}

%\menulink{Irreducible Representations of Symmetric Groups}
%{IrrRepSymNatXmpPage}
%Alfred Young's natural form for these representations.

%\menulink{Representations of Higher Degree}
%{RepresentationPackage1XmpPage}
%Constructing new representations by symmetric and antisymmetric
%tensors.

%\menulink{Decomposing Representations}
%{RepresentationPackage2XmpPage}
%Parker's `Meat-Axe', working in prime characteristics.

\menulink{Representations of \texht{$A_6$}{A6}}{RepA6Page}
The irreducible representations of the alternating group
\texht{$A_6$}{A6} over fields of characteristic 2.
\endmenu
\endscroll
\autobuttons \end{page}
```

———-

Representations of A_6 A6

— grpthry.ht —

```
\begin{page}{RepA6Page}{Representations of \texht{$A_6$}{A6}}
```

```
% author: J. Grabmeier, 08/08/89
\beginscroll
In what follows you'll see how to use Axiom to get all the irreducible
representations of the alternating group \texht{$A_6$}{A6} over the
field with two elements (GF 2). First, we generate \texht{$A_6$}{A6}
by a three-cycle: x = (1,2,3) and a 5-cycle: y = (2,3,4,5,6). Next we
have Axiom calculate the permutation representation over the integers
and over GF 2:
\spadpaste{genA6 : LIST PERM INT := [cycle [1,2,3],cycle [2,3,4,5,6]]
\bound{genA6}}
\spadpaste{pRA6 := permutationRepresentation (genA6, 6) \bound{pRA6}
\free{genA6}
}
Now we apply Parker's 'Meat-Axe' and split it:
\spadpaste{sp0 := meatAxe (pRA6::(LIST MATRIX PF 2)) \free{pRA6}
\bound{sp0}}
We have found the trivial module as a quotient module
and a 5-dimensional sub-module.
Try to split again:
\spadpaste{sp1 := meatAxe sp0.1 \bound{sp1}}
and we find a 4-dimensional sub-module and the trivial one again.
Now we can test if this representaton is absolutely irreducible:
\spadpaste{isAbsolutelyIrreducible? sp1.2 }
and we see that this 4-dimensional representation is absolutely
irreducible. So, we have found a second irreducible representation.
Now, we construct a representation by reducing an irreducible one of
the symmetric group S_6 over the integers mod 2. We take the one
labelled by the partition [2,2,1,1] and restrict it to
\texht{$A_6$}{A6}:
\spadpaste{d2211 := irreducibleRepresentation ([2,2,1,1],genA6)
\bound{d2211} }
Now split it:
\spadpaste{d2211m2 := d2211:: (LIST MATRIX PF 2);
sp2 := meatAxe d2211m2 \free{d2211}
\bound{sp2}}
This gave both a five and a four dimensional representation.
Now we take the 4-dimensional one
and we shall see that it is absolutely irreducible:
\spadpaste{isAbsolutelyIrreducible? sp2.1}
The two 4-dimensional representations are not equivalent:
\spadpaste{areEquivalent? (sp1.2, sp2.1)}
So we have found a third irreducible representation.
Now we construct a new representation using the tensor product
and try to split it:
\spadpaste{dA6d16 := tensorProduct(sp1.2,sp2.1);
meatAxe dA6d16 \bound{dA6d16}}
The representation is irreducible, but it may be not
absolutely irreducible.
\spadpaste{isAbsolutelyIrreducible? dA6d16}
So let's try the same procedure over the field with 4 elements:
```

```
\spadpaste{sp3 := meatAxe (dA6d16 :: (LIST MATRIX FF(2,2))) \bound{sp3}}
Now we find two 8-dimensional representations, dA6d8a and dA6d8b.
Both are absolutely irreducible...
\spadpaste{isAbsolutelyIrreducible?  sp3.1}
\spadpaste{isAbsolutelyIrreducible?  sp3.2}
and they are not equivalent:
\spadpaste{areEquivalent? (sp3.1,sp3.2)}
So we have found five absolutely irreducible representations
of \texht{$A_6$}{A6}
in characteristic 2.
General theory now tells us that there are no more irreducible ones.
Here, for future reference are all the absolutely irreducible 2-modular
representations of \texht{$A_6$}{A6}
\spadpaste{sp0.2 \free{sp0}}
\spadpaste{sp1.2 \free{sp1}}
\spadpaste{sp2.1 \free{sp2}}
\spadpaste{sp3.1 \free{sp3}}
\spadpaste{sp3.2 \free{sp3}}
And here again is the irreducible, but not absolutely irreducible
representations of \texht{$A_6$}{A6} over GF 2
\spadpaste{dA6d16 \free{dA6d16}}
\endscroll
\autobuttons
\end{page}

\begin{patch}{RepA6PagePatch1}
\begin{paste}{RepA6PageFull1}{RepA6PageEmpty1}
\pastebutton{RepA6PageFull1}{\hidepaste}
\tab{5}\spadcommand{genA6 : LIST PERM INT := [cycle [1,2,3],cycle [2,3,4,5,6]]\bound{genA6 }}
\indentrel{3}\begin{verbatim}
   (1)  [(1 2 3),(2 3 4 5 6)]
                         Type: List Permutation Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RepA6PageEmpty1}
\begin{paste}{RepA6PageEmpty1}{RepA6PagePatch1}
\pastebutton{RepA6PageEmpty1}{\showpaste}
\tab{5}\spadcommand{genA6 : LIST PERM INT := [cycle [1,2,3],cycle [2,3,4,5,6]]\bound{genA6 }}
\end{paste}\end{patch}

\begin{patch}{RepA6PagePatch2}
\begin{paste}{RepA6PageFull2}{RepA6PageEmpty2}
\pastebutton{RepA6PageFull2}{\hidepaste}
\tab{5}\spadcommand{pRA6 := permutationRepresentation (genA6, 6)\bound{pRA6 }\free{genA6 }}
\indentrel{3}\begin{verbatim}
         0  0  1  0  0  0 1  0  0  0  0  0

         1  0  0  0  0  0 0  0  0  0  0  1
```

```
        0  1  0  0  0  0 0  1  0  0  0  0
   (2)  [                   ,               ]
        0  0  0  1  0  0 0  0  1  0  0  0

        0  0  0  0  1  0 0  0  0  1  0  0

        0  0  0  0  0  1 0  0  0  0  1  0
                              Type: List Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RepA6PageEmpty2}
\begin{paste}{RepA6PageEmpty2}{RepA6PagePatch2}
\pastebutton{RepA6PageEmpty2}{\showpaste}
\tab{5}\spadcommand{pRA6 := permutationRepresentation (genA6, 6)\bound{pRA6 }\free{genA6 }}
\end{paste}\end{patch}

\begin{patch}{RepA6PagePatch3}
\begin{paste}{RepA6PageFull3}{RepA6PageEmpty3}
\pastebutton{RepA6PageFull3}{\hidepaste}
\tab{5}\spadcommand{sp0 := meatAxe (pRA6::(LIST MATRIX PF 2))\free{pRA6 }\bound{sp0 }}
\indentrel{3}\begin{verbatim}
   Fingerprint element in generated algebra is singular
     A proper cyclic submodule is found.
     Transition matrix computed
     The inverse of the transition matrix computed
     Now transform the matrices
         0  0  1  0  0 1  0  0  0  0

         1  0  0  0  0 1  1  1  1  1

   (3)  [[0  1  0  0  0,0  1  0  0  0],[[1],[1]]]

         0  0  0  1  0 0  0  1  0  0

         0  0  0  0  1 0  0  0  1  0
                    Type: List List Matrix PrimeField 2
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RepA6PageEmpty3}
\begin{paste}{RepA6PageEmpty3}{RepA6PagePatch3}
\pastebutton{RepA6PageEmpty3}{\showpaste}
\tab{5}\spadcommand{sp0 := meatAxe (pRA6::(LIST MATRIX PF 2))\free{pRA6 }\bound{sp0 }}
\end{paste}\end{patch}

\begin{patch}{RepA6PagePatch4}
\begin{paste}{RepA6PageFull4}{RepA6PageEmpty4}
\pastebutton{RepA6PageFull4}{\hidepaste}
\tab{5}\spadcommand{sp1 := meatAxe sp0.1\bound{sp1 }}
```

```
\indentrel{3}\begin{verbatim}
   Fingerprint element in generated algebra is singular
     The generated cyclic submodule was not proper
     The generated cyclic submodule was not proper
     The generated cyclic submodule was not proper
    We know that all the cyclic submodules generated by a
  ll
      non-trivial element of the singular matrix under vi
  ew are
      not proper, hence Norton's irreducibility test can
  be done:
     A proper cyclic submodule is found.
     Transition matrix computed
     The inverse of the transition matrix computed
     Now transform the matrices
    Representation is not irreducible and it will be spli
  t:
                    0  1  0  0 0  1  1  1

                    0  0  1  0 1  1  0  1
   (4)  [[[1],[1]],[          ,          ]]
                    1  0  0  0 1  1  1  0

                    0  0  0  1 1  1  1  1
                    Type: List List Matrix PrimeField 2
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RepA6PageEmpty4}
\begin{paste}{RepA6PageEmpty4}{RepA6PagePatch4}
\pastebutton{RepA6PageEmpty4}{\showpaste}
\tab{5}\spadcommand{sp1 := meatAxe sp0.1\bound{sp1 }}
\end{paste}\end{patch}

\begin{patch}{RepA6PagePatch5}
\begin{paste}{RepA6PageFull5}{RepA6PageEmpty5}
\pastebutton{RepA6PageFull5}{\hidepaste}
\tab{5}\spadcommand{isAbsolutelyIrreducible? sp1.2}
\indentrel{3}\begin{verbatim}
   Random element in generated algebra does
     not have a one-dimensional kernel
   Random element in generated algebra does
     not have a one-dimensional kernel
   Random element in generated algebra has
     one-dimensional kernel
    We know that all the cyclic submodules generated by a
  ll
      non-trivial element of the singular matrix under vi
  ew are
      not proper, hence Norton's irreducibility test can
```

```
  be done:
     The generated cyclic submodule was not proper
     Representation is absolutely irreducible
   (5)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RepA6PageEmpty5}
\begin{paste}{RepA6PageEmpty5}{RepA6PagePatch5}
\pastebutton{RepA6PageEmpty5}{\showpaste}
\tab{5}\spadcommand{isAbsolutelyIrreducible? sp1.2}
\end{paste}\end{patch}

\begin{patch}{RepA6PagePatch6}
\begin{paste}{RepA6PageFull6}{RepA6PageEmpty6}
\pastebutton{RepA6PageFull6}{\hidepaste}
\tab{5}\spadcommand{d2211 := irreducibleRepresentation ([2,2,1,1],genA6)\bound{d2211 }}
\indentrel{3}\begin{verbatim}
   (6)
    1  0  0  - 1   1    0    0    0    0

    0  1  0   1    0    1    0    0    0

    0  0  1   0    1  - 1    0    0    0

    0  0  0  - 1   0    0  - 1    0    0

   [0  0  0   0  - 1    0    0  - 1    0 ,

    0  0  0   0    0  - 1    0    0  - 1

    0  0  0   1    0    0    0    0    0

    0  0  0   0    1    0    0    0    0

    0  0  0   0    0    1    0    0    0
     0    0   1   0   0  0   1   0  0

     0    0   0   0   1  0 - 1   0  0

     0    0   0   0   0  1   1   0  0

     0    0   0   0   0  0   1   1  0

     0    0   0   0   0  0 - 1   0  1]

     0    0   0   0   0  0   1   0  0

   - 1    0   0   0   0  0 - 1   0  0
```

```
     0   - 1  0   0   0  0   1   0  0

     0    0   0  - 1  0  0  - 1  0  0
                              Type: List Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RepA6PageEmpty6}
\begin{paste}{RepA6PageEmpty6}{RepA6PagePatch6}
\pastebutton{RepA6PageEmpty6}{\showpaste}
\tab{5}\spadcommand{d2211 := irreducibleRepresentation ([2,2,1,1],genA6)\bound{d2211 }}
\end{paste}\end{patch}

\begin{patch}{RepA6PagePatch7}
\begin{paste}{RepA6PageFull7}{RepA6PageEmpty7}
\pastebutton{RepA6PageFull7}{\hidepaste}
\tab{5}\spadcommand{d2211m2 := d2211:: (LIST MATRIX PF 2); sp2 := meatAxe d2211m2\free{d2211 }\bound{sp2 }}
\indentrel{3}\begin{verbatim}
   Fingerprint element in generated algebra is singular
     A proper cyclic submodule is found.
     Transition matrix computed
     The inverse of the transition matrix computed
     Now transform the matrices
   (7)
     1  0  1  1 0  0  1  0

     0  1  0  1 1  1  1  1
   [[          ,          ],
     1  1  0  0 1  0  1  1

     0  1  0  0 0  1  0  1
     1  0  0  0  0 1  1  1  0  0

     0  1  1  1  1 0  0  1  1  1

    [0  1  1  0  0,1  0  0  1  0]]

     0  1  0  1  0 0  0  1  0  1

     0  1  1  1  0 1  0  0  1  1
                    Type: List List Matrix PrimeField 2
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RepA6PageEmpty7}
\begin{paste}{RepA6PageEmpty7}{RepA6PagePatch7}
\pastebutton{RepA6PageEmpty7}{\showpaste}
\tab{5}\spadcommand{d2211m2 := d2211:: (LIST MATRIX PF 2); sp2 := meatAxe d2211m2\free{d2211 }\bound{sp2 }}
\end{paste}\end{patch}
```

```
\begin{patch}{RepA6PagePatch8}
\begin{paste}{RepA6PageFull8}{RepA6PageEmpty8}
\pastebutton{RepA6PageFull8}{\hidepaste}
\tab{5}\spadcommand{isAbsolutelyIrreducible? sp2.1}
\indentrel{3}\begin{verbatim}
   Random element in generated algebra does
     not have a one-dimensional kernel
   Random element in generated algebra has
     one-dimensional kernel
    We know that all the cyclic submodules generated by a
  ll
     non-trivial element of the singular matrix under vi
  ew are
     not proper, hence Norton's irreducibility test can
  be done:
     The generated cyclic submodule was not proper
     Representation is absolutely irreducible
   (8)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RepA6PageEmpty8}
\begin{paste}{RepA6PageEmpty8}{RepA6PagePatch8}
\pastebutton{RepA6PageEmpty8}{\showpaste}
\tab{5}\spadcommand{isAbsolutelyIrreducible? sp2.1}
\end{paste}\end{patch}

\begin{patch}{RepA6PagePatch9}
\begin{paste}{RepA6PageFull9}{RepA6PageEmpty9}
\pastebutton{RepA6PageFull9}{\hidepaste}
\tab{5}\spadcommand{areEquivalent? (sp1.2, sp2.1)}
\indentrel{3}\begin{verbatim}
   Random element in generated algebra does
     not have a one-dimensional kernel
   Random element in generated algebra does
     not have a one-dimensional kernel
   Random element in generated algebra has
     one-dimensional kernel
     There is no isomorphism, as the only possible one
       fails to do the necessary base change

   Representations are not equivalent.
   (9)  [0]
                              Type: Matrix PrimeField 2
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RepA6PageEmpty9}
```

```
\begin{paste}{RepA6PageEmpty9}{RepA6PagePatch9}
\pastebutton{RepA6PageEmpty9}{\showpaste}
\tab{5}\spadcommand{areEquivalent? (sp1.2, sp2.1)}
\end{paste}\end{patch}

\begin{patch}{RepA6PagePatch10}
\begin{paste}{RepA6PageFull10}{RepA6PageEmpty10}
\pastebutton{RepA6PageFull10}{\hidepaste}
\tab{5}\spadcommand{dA6d16 := tensorProduct(sp1.2,sp2.1); meatAxe dA6d16\bound{dA6d16 }}
\indentrel{3}\begin{verbatim}
  Fingerprint element in generated algebra is non-singula
  r
   Fingerprint element in generated algebra is singular
     The generated cyclic submodule was not proper
     The generated cyclic submodule was not proper
     The generated cyclic submodule was not proper
     The generated cyclic submodule was not proper
     The generated cyclic submodule was not proper
     The generated cyclic submodule was not proper
     The generated cyclic submodule was not proper
  Fingerprint element in generated algebra is non-singula
  r
   Fingerprint element in generated algebra is singular
     The generated cyclic submodule was not proper
     The generated cyclic submodule was not proper
     The generated cyclic submodule was not proper
     The generated cyclic submodule was not proper
     The generated cyclic submodule was not proper
     The generated cyclic submodule was not proper
     The generated cyclic submodule was not proper
   Fingerprint element in generated algebra is singular
     The generated cyclic submodule was not proper
     The generated cyclic submodule was not proper
     The generated cyclic submodule was not proper
    We know that all the cyclic submodules generated by a
  ll
      non-trivial element of the singular matrix under vi
  ew are
      not proper, hence Norton's irreducibility test can
  be done:
     The generated cyclic submodule was not proper
     Representation is irreducible, but we don't know
       whether it is absolutely irreducible
   (10)
   [
      0  0  0  0  0  0  0  0  1  0  1  0  0  0  0  0

      0  0  0  0  0  0  0  0  0  1  1  1  0  0  0  0

      0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0
```

```
 0  0  0  0  0  0  0  0  1  1  0  0  0  0  0  0

 1  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0

 0  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0

 1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0

 1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0
[                                               ,
 0  0  0  0  1  0  1  0  0  0  0  0  0  0  0  0

 0  0  0  0  0  1  1  1  0  0  0  0  0  0  0  0

 0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0

 0  0  0  0  1  1  0  0  0  0  0  0  0  0  0  0

 0  0  0  0  0  0  0  0  0  0  0  0  1  0  1  0

 0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  1

 0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0

 0  0  0  0  0  0  0  0  0  0  0  0  1  1  0  0
 0  0  0  0  0  1  1  0  0  1  1  0  0  1  1  0

 0  0  0  0  0  1  0  1  0  1  0  1  0  1  0  1

 0  0  0  0  1  1  1  0  1  1  1  0  1  1  1  0

 0  0  0  0  0  1  1  1  0  1  1  1  0  1  1  1

 0  1  1  0  0  1  1  0  0  1  1  0  0  1  1  0

 0  1  0  1  0  1  0  1  0  1  0  1  0  1  0  1

 1  1  1  0  1  1  1  0  1  1  1  0  1  1  1  0

 0  1  1  1  0  1  1  1  0  1  1  1  0  1  1  1
                                                ]
 0  1  1  0  0  0  0  0  0  1  1  0  0  1  1  0

 0  1  0  1  0  0  0  0  0  1  0  1  0  1  0  1

 1  1  1  0  0  0  0  0  1  1  1  0  1  1  1  0

 0  1  1  1  0  0  0  0  0  1  1  1  0  1  1  1
```

```
      0  1  1  0  0  1  1  0  0  0  0  0  0  1  1  0

      0  1  0  1  0  1  0  1  0  0  0  0  0  1  0  1

      1  1  1  0  1  1  1  0  0  0  0  0  1  1  1  0

      0  1  1  1  0  1  1  1  0  0  0  0  0  1  1  1
     ]
                    Type: List List Matrix PrimeField 2
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RepA6PageEmpty10}
\begin{paste}{RepA6PageEmpty10}{RepA6PagePatch10}
\pastebutton{RepA6PageEmpty10}{\showpaste}
\tab{5}\spadcommand{dA6d16 := tensorProduct(sp1.2,sp2.1); meatAxe dA6d16\bound{dA6d16 }}
\end{paste}\end{patch}

\begin{patch}{RepA6PagePatch11}
\begin{paste}{RepA6PageFull11}{RepA6PageEmpty11}
\pastebutton{RepA6PageFull11}{\hidepaste}
\tab{5}\spadcommand{isAbsolutelyIrreducible? dA6d16}
\indentrel{3}\begin{verbatim}
   Random element in generated algebra does
     not have a one-dimensional kernel
   Random element in generated algebra does
     not have a one-dimensional kernel
   Random element in generated algebra does
     not have a one-dimensional kernel
   Random element in generated algebra does
     not have a one-dimensional kernel
   Random element in generated algebra does
     not have a one-dimensional kernel
   Random element in generated algebra does
     not have a one-dimensional kernel
   Random element in generated algebra does
     not have a one-dimensional kernel
   Random element in generated algebra does
     not have a one-dimensional kernel
   Random element in generated algebra does
     not have a one-dimensional kernel
   Random element in generated algebra does
     not have a one-dimensional kernel
   Random element in generated algebra does
     not have a one-dimensional kernel
   Random element in generated algebra does
     not have a one-dimensional kernel
   Random element in generated algebra does
     not have a one-dimensional kernel
   Random element in generated algebra does
```

```
     not have a one-dimensional kernel
   Random element in generated algebra does
     not have a one-dimensional kernel
   Random element in generated algebra does
     not have a one-dimensional kernel
   Random element in generated algebra does
     not have a one-dimensional kernel
   Random element in generated algebra does
     not have a one-dimensional kernel
   Random element in generated algebra does
     not have a one-dimensional kernel
   Random element in generated algebra does
     not have a one-dimensional kernel
   Random element in generated algebra does
     not have a one-dimensional kernel
   Random element in generated algebra does
     not have a one-dimensional kernel
   Random element in generated algebra does
     not have a one-dimensional kernel
   Random element in generated algebra does
     not have a one-dimensional kernel
   Random element in generated algebra does
     not have a one-dimensional kernel
   We have not found a one-dimensional kernel so far,
     as we do a random search you could try again
   (11)  false
                                           Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RepA6PageEmpty11}
\begin{paste}{RepA6PageEmpty11}{RepA6PagePatch11}
\pastebutton{RepA6PageEmpty11}{\showpaste}
\tab{5}\spadcommand{isAbsolutelyIrreducible? dA6d16}
\end{paste}\end{patch}

\begin{patch}{RepA6PagePatch12}
\begin{paste}{RepA6PageFull12}{RepA6PageEmpty12}
\pastebutton{RepA6PageFull12}{\hidepaste}
\tab{5}\spadcommand{sp3 := meatAxe (dA6d16 :: (LIST MATRIX FF(2,2)))\bound{sp3 }}
\indentrel{3}\begin{verbatim}
  Fingerprint element in generated algebra is non-singula
  r
   Fingerprint element in generated algebra is singular
     The generated cyclic submodule was not proper
     The generated cyclic submodule was not proper
     The generated cyclic submodule was not proper
     The generated cyclic submodule was not proper
     The generated cyclic submodule was not proper
     The generated cyclic submodule was not proper
```

```
   The generated cyclic submodule was not proper
Fingerprint element in generated algebra is non-singula
r
 Fingerprint element in generated algebra is singular
   The generated cyclic submodule was not proper
   The generated cyclic submodule was not proper
   The generated cyclic submodule was not proper
   The generated cyclic submodule was not proper
   The generated cyclic submodule was not proper
   The generated cyclic submodule was not proper
   The generated cyclic submodule was not proper
 Fingerprint element in generated algebra is singular
   The generated cyclic submodule was not proper
   The generated cyclic submodule was not proper
   A proper cyclic submodule is found.
   Transition matrix computed
   The inverse of the transition matrix computed
   Now transform the matrices
 (12)
 [
   [
 [[%A,%A + 1,0,%A,1,%A + 1,0,0],
  [0,0,%A,%A + 1,%A,%A,0,0],
  [%A,%A + 1,%A,1,%A + 1,0,0,0],
  [%A,%A + 1,%A,1,%A,0,0,0],
  [%A + 1,1,1,1,0,0,%A + 1,%A],
  [0,0,%A + 1,1,0,0,%A,0], [1,0,1,1,0,0,0,0],
  [1,1,0,0,0,0,0,0]]
     ,

 [[1,0,%A,0,1,1,%A,%A + 1],
  [1,%A + 1,0,0,0,%A + 1,1,%A + 1],
  [%A,1,%A + 1,%A + 1,%A + 1,1,%A,0],
  [%A + 1,%A + 1,0,0,1,%A + 1,1,1],
  [1,0,%A + 1,0,1,1,%A,%A],
  [0,0,%A + 1,%A + 1,%A + 1,1,1,%A], [0,0,1,0,0,1,0,1],
  [0,%A,0,%A,1,%A + 1,%A + 1,%A]]
     ]
   ,

    0     1       1     %A + 1  0  0  0  0

    1     1     %A + 1    0     0  0  0  0

    %A    0       0       0     0  0  0  0

    1     %A      0       0     0  0  0  0
   [                                       ,
    %A  %A + 1    1       1     1  0  1  1
```

```
      0     0       %A      1     0  1  0  1

      %A    1       0       1     1  1  0  0

      1     %A    %A + 1    %A    0  1  0  0

   [[%A + 1,1,%A,0,0,%A + 1,0,1],
    [0,%A,1,1,1,0,%A + 1,%A],
    [0,%A + 1,0,%A + 1,%A + 1,1,%A + 1,%A],
    [1,%A + 1,1,%A + 1,0,0,%A + 1,1],
    [0,%A,0,%A + 1,%A + 1,0,0,%A + 1],
    [%A + 1,0,%A + 1,%A,0,%A + 1,0,%A + 1],
    [0,1,0,1,%A + 1,0,%A + 1,%A + 1],
    [%A,%A,%A,1,%A,%A,1,%A + 1]]
       ]
     ]
                Type: List List Matrix FiniteField(2,2)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RepA6PageEmpty12}
\begin{paste}{RepA6PageEmpty12}{RepA6PagePatch12}
\pastebutton{RepA6PageEmpty12}{\showpaste}
\tab{5}\spadcommand{sp3 := meatAxe (dA6d16 :: (LIST MATRIX FF(2,2)))\bound{sp3 }}
\end{paste}\end{patch}

\begin{patch}{RepA6PagePatch13}
\begin{paste}{RepA6PageFull13}{RepA6PageEmpty13}
\pastebutton{RepA6PageFull13}{\hidepaste}
\tab{5}\spadcommand{isAbsolutelyIrreducible? sp3.1}
\indentrel{3}\begin{verbatim}
   Random element in generated algebra does
     not have a one-dimensional kernel
   Random element in generated algebra does
     not have a one-dimensional kernel
   Random element in generated algebra does
     not have a one-dimensional kernel
   Random element in generated algebra does
     not have a one-dimensional kernel
   Random element in generated algebra does
     not have a one-dimensional kernel
   Random element in generated algebra does
     not have a one-dimensional kernel
   Random element in generated algebra does
     not have a one-dimensional kernel
   Random element in generated algebra has
     one-dimensional kernel
    We know that all the cyclic submodules generated by a
  ll
      non-trivial element of the singular matrix under vi
```

```
  ew are
      not proper, hence Norton's irreducibility test can
  be done:
     The generated cyclic submodule was not proper
     Representation is absolutely irreducible
   (13)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RepA6PageEmpty13}
\begin{paste}{RepA6PageEmpty13}{RepA6PagePatch13}
\pastebutton{RepA6PageEmpty13}{\showpaste}
\tab{5}\spadcommand{isAbsolutelyIrreducible? sp3.1}
\end{paste}\end{patch}

\begin{patch}{RepA6PagePatch14}
\begin{paste}{RepA6PageFull14}{RepA6PageEmpty14}
\pastebutton{RepA6PageFull14}{\hidepaste}
\tab{5}\spadcommand{isAbsolutelyIrreducible? sp3.2}
\indentrel{3}\begin{verbatim}
   Random element in generated algebra does
     not have a one-dimensional kernel
   Random element in generated algebra does
     not have a one-dimensional kernel
   Random element in generated algebra does
     not have a one-dimensional kernel
   Random element in generated algebra has
     one-dimensional kernel
    We know that all the cyclic submodules generated by a
  ll
      non-trivial element of the singular matrix under vi
  ew are
      not proper, hence Norton's irreducibility test can
  be done:
     The generated cyclic submodule was not proper
     Representation is absolutely irreducible
   (14)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RepA6PageEmpty14}
\begin{paste}{RepA6PageEmpty14}{RepA6PagePatch14}
\pastebutton{RepA6PageEmpty14}{\showpaste}
\tab{5}\spadcommand{isAbsolutelyIrreducible? sp3.2}
\end{paste}\end{patch}

\begin{patch}{RepA6PagePatch15}
\begin{paste}{RepA6PageFull15}{RepA6PageEmpty15}
```

```
\pastebutton{RepA6PageFull15}{\hidepaste}
\tab{5}\spadcommand{areEquivalent? (sp3.1,sp3.2)}
\indentrel{3}\begin{verbatim}
   Random element in generated algebra does
     not have a one-dimensional kernel
   Random element in generated algebra does
     not have a one-dimensional kernel
   Random element in generated algebra does
     not have a one-dimensional kernel
   Random element in generated algebra does
     not have a one-dimensional kernel
   Random element in generated algebra has
     one-dimensional kernel
     There is no isomorphism, as the only possible one
       fails to do the necessary base change

   Representations are not equivalent.
   (15)  [0]
                          Type: Matrix FiniteField(2,2)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RepA6PageEmpty15}
\begin{paste}{RepA6PageEmpty15}{RepA6PagePatch15}
\pastebutton{RepA6PageEmpty15}{\showpaste}
\tab{5}\spadcommand{areEquivalent? (sp3.1,sp3.2)}
\end{paste}\end{patch}

\begin{patch}{RepA6PagePatch16}
\begin{paste}{RepA6PageFull16}{RepA6PageEmpty16}
\pastebutton{RepA6PageFull16}{\hidepaste}
\tab{5}\spadcommand{sp0.2\free{sp0 }}
\indentrel{3}\begin{verbatim}
   (16)  [[1],[1]]
                         Type: List Matrix PrimeField 2
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RepA6PageEmpty16}
\begin{paste}{RepA6PageEmpty16}{RepA6PagePatch16}
\pastebutton{RepA6PageEmpty16}{\showpaste}
\tab{5}\spadcommand{sp0.2\free{sp0 }}
\end{paste}\end{patch}

\begin{patch}{RepA6PagePatch17}
\begin{paste}{RepA6PageFull17}{RepA6PageEmpty17}
\pastebutton{RepA6PageFull17}{\hidepaste}
\tab{5}\spadcommand{sp1.2\free{sp1 }}
\indentrel{3}\begin{verbatim}
          0  1  0  0 0  1  1  1
```

```
         0  0  1  0 1  1  0  1
   (17)  [          ,          ]
         1  0  0  0 1  1  1  0

         0  0  0  1 1  1  1  1
                         Type: List Matrix PrimeField 2
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RepA6PageEmpty17}
\begin{paste}{RepA6PageEmpty17}{RepA6PagePatch17}
\pastebutton{RepA6PageEmpty17}{\showpaste}
\tab{5}\spadcommand{sp1.2\free{sp1 }}
\end{paste}\end{patch}

\begin{patch}{RepA6PagePatch18}
\begin{paste}{RepA6PageFull18}{RepA6PageEmpty18}
\pastebutton{RepA6PageFull18}{\hidepaste}
\tab{5}\spadcommand{sp2.1\free{sp2 }}
\indentrel{3}\begin{verbatim}
         1  0  1  1 0  0  1  0

         0  1  0  1 1  1  1  1
   (18)  [          ,          ]
         1  1  0  0 1  0  1  1

         0  1  0  0 0  1  0  1
                         Type: List Matrix PrimeField 2
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RepA6PageEmpty18}
\begin{paste}{RepA6PageEmpty18}{RepA6PagePatch18}
\pastebutton{RepA6PageEmpty18}{\showpaste}
\tab{5}\spadcommand{sp2.1\free{sp2 }}
\end{paste}\end{patch}

\begin{patch}{RepA6PagePatch19}
\begin{paste}{RepA6PageFull19}{RepA6PageEmpty19}
\pastebutton{RepA6PageFull19}{\hidepaste}
\tab{5}\spadcommand{sp3.1\free{sp3 }}
\indentrel{3}\begin{verbatim}
   (19)
   [
   [[%A,%A + 1,0,%A,1,%A + 1,0,0],
    [0,0,%A,%A + 1,%A,%A,0,0],
    [%A,%A + 1,%A,1,%A + 1,0,0,0],
    [%A,%A + 1,%A,1,%A,0,0,0],
    [%A + 1,1,1,1,0,0,%A + 1,%A],
```

```
    [0,0,%A + 1,1,0,0,%A,0], [1,0,1,1,0,0,0,0],
    [1,1,0,0,0,0,0,0]]
     ,

   [[1,0,%A,0,1,1,%A,%A + 1],
    [1,%A + 1,0,0,0,%A + 1,1,%A + 1],
    [%A,1,%A + 1,%A + 1,%A + 1,1,%A,0],
    [%A + 1,%A + 1,0,0,1,%A + 1,1,1],
    [1,0,%A + 1,0,1,1,%A,%A],
    [0,0,%A + 1,%A + 1,%A + 1,1,1,%A], [0,0,1,0,0,1,0,1],
    [0,%A,0,%A,1,%A + 1,%A + 1,%A]]
     ]
                 Type: List Matrix FiniteField(2,2)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RepA6PageEmpty19}
\begin{paste}{RepA6PageEmpty19}{RepA6PagePatch19}
\pastebutton{RepA6PageEmpty19}{\showpaste}
\tab{5}\spadcommand{sp3.1\free{sp3 }}
\end{paste}\end{patch}

\begin{patch}{RepA6PagePatch20}
\begin{paste}{RepA6PageFull20}{RepA6PageEmpty20}
\pastebutton{RepA6PageFull20}{\hidepaste}
\tab{5}\spadcommand{sp3.2\free{sp3 }}
\indentrel{3}\begin{verbatim}
   (20)
    0     1       1     %A + 1  0  0  0  0

    1     1     %A + 1    0     0  0  0  0

    %A    0       0       0     0  0  0  0

    1     %A      0       0     0  0  0  0
   [                                       ,
    %A  %A + 1    1       1     1  0  1  1

    0     0       %A      1     0  1  0  1

    %A    1       0       1     1  1  0  0

    1     %A    %A + 1    %A    0  1  0  0

   [[%A + 1,1,%A,0,0,%A + 1,0,1],
    [0,%A,1,1,1,0,%A + 1,%A],
    [0,%A + 1,0,%A + 1,%A + 1,1,%A + 1,%A],
    [1,%A + 1,1,%A + 1,0,0,%A + 1,1],
    [0,%A,0,%A + 1,%A + 1,0,0,%A + 1],
    [%A + 1,0,%A + 1,%A,0,%A + 1,0,%A + 1],
```

```
    [0,1,0,1,%A + 1,0,%A + 1,%A + 1],
    [%A,%A,%A,1,%A,%A,1,%A + 1]]
     ]
                     Type: List Matrix FiniteField(2,2)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RepA6PageEmpty20}
\begin{paste}{RepA6PageEmpty20}{RepA6PagePatch20}
\pastebutton{RepA6PageEmpty20}{\showpaste}
\tab{5}\spadcommand{sp3.2\free{sp3 }}
\end{paste}\end{patch}

\begin{patch}{RepA6PagePatch21}
\begin{paste}{RepA6PageFull21}{RepA6PageEmpty21}
\pastebutton{RepA6PageFull21}{\hidepaste}
\tab{5}\spadcommand{dA6d16\free{dA6d16 }}
\indentrel{3}\begin{verbatim}
   (21)
    0  0  0  0  1  0  1  1  0  0  0  0  0  0  0  0

    0  0  0  0  0  1  0  1  0  0  0  0  0  0  0  0

    0  0  0  0  1  1  0  0  0  0  0  0  0  0  0  0

    0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0

    0  0  0  0  0  0  0  0  1  0  1  1  0  0  0  0

    0  0  0  0  0  0  0  0  0  1  0  1  0  0  0  0

    0  0  0  0  0  0  0  0  1  1  0  0  0  0  0  0

    0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0
   [                                               ,
    1  0  1  1  0  0  0  0  0  0  0  0  0  0  0  0

    0  1  0  1  0  0  0  0  0  0  0  0  0  0  0  0

    1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0

    0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0

    0  0  0  0  0  0  0  0  0  0  0  0  1  0  1  1

    0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  1

    0  0  0  0  0  0  0  0  0  0  0  0  1  1  0  0

    0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0
```

```
    0  0  0  0  0  0  1  0  0  0  1  0  0  0  1  0

    0  0  0  0  1  1  1  1  1  1  1  1  1  1  1  1

    0  0  0  0  1  0  1  1  1  0  1  1  1  0  1  1

    0  0  0  0  0  1  0  1  0  1  0  1  0  1  0  1

    0  0  1  0  0  0  1  0  0  0  0  0  0  0  1  0

    1  1  1  1  1  1  1  1  0  0  0  0  1  1  1  1

    1  0  1  1  1  0  1  1  0  0  0  0  1  0  1  1

    0  1  0  1  0  1  0  1  0  0  0  0  0  1  0  1
                                                   ]
    0  0  1  0  0  0  1  0  0  0  1  0  0  0  0  0

    1  1  1  1  1  1  1  1  1  1  1  1  0  0  0  0

    1  0  1  1  1  0  1  1  1  0  1  1  0  0  0  0

    0  1  0  1  0  1  0  1  0  1  0  1  0  0  0  0

    0  0  1  0  0  0  1  0  0  0  1  0  0  0  1  0

    1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1

    1  0  1  1  1  0  1  1  1  0  1  1  1  0  1  1

    0  1  0  1  0  1  0  1  0  1  0  1  0  1  0  1
                              Type: List Matrix PrimeField 2
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RepA6PageEmpty21}
\begin{paste}{RepA6PageEmpty21}{RepA6PagePatch21}
\pastebutton{RepA6PageEmpty21}{\showpaste}
\tab{5}\spadcommand{dA6d16\free{dA6d16 }}
\end{paste}\end{patch}
```

———-

Representation Theory

— grpthry.ht —

```
\begin{page}{InfoRepTheoryPage}{Representation Theory}
\beginscroll
\horizontalline
Representation theory for finite groups studies finite groups by
embedding them in a general linear group over a field or an integral
domain.  Hence, we are representing each element of the group by an
invertible matrix.  Two matrix representations of a given group are
equivalent, if, by changing the basis of the underlying space, you can
go from one to the other.  When you change bases, you transform the
matrices that are the images of elements by conjugating them by an
invertible matrix.
\newline
\newline
If we can find a subspace which is fixed under the image of the group,
then there exists a `base change' after which all the representing
matrices are in upper triangular block form. The block matrices on
the main diagonal give a new representation of the group of lower
degree.  Such a representation is said to be `reducible'.
\newline
\beginmenu
%\menulink{Irreducible Representations of Symmetric Groups}
%{IrrRepSymNatXmpPage}

%Alfred Young's natural form for these representations.

%\menulink{Representations of Higher Degree}
%{RepresentationPackage1XmpPage}
%Constructing new representations by symmetric and antisymmetric
%tensors.

%\menulink{Decomposing Representations}
%{RepresentationPackage2XmpPage}
%Parker's `Meat-Axe', working in prime characteristics.

\menulink{Representations of \texht{$A_6$}{A6}}{RepA6Page}
The irreducible representations of the alternating group
\texht{$A_6$}{A6} over fields of characteristic 2.
\endmenu
\endscroll
\autobuttons \end{page}
```

———-

Group Theory

— grpthry.ht —

```
\begin{page}{InfoGroupTheoryPage}{Group Theory}
%%
%% Johannes Grabmeier 03/02/90
%%
\beginscroll
A {\it group} is a set G together with an associative operation *
satisfying the axioms of existence of a unit element and an inverse of
every element of the group.  The Axiom category \spadtype{Group}
represents this setting.  Many data structures in Axiom are groups and
therefore there is a large variety of examples as fields and
polynomials, although the main interest there is not the group
structure.

To work with and in groups in a concrete manner some way of
representing groups has to be chosen. A group can be given as a list
of generators and a set of relations. If there are no relations, then
we have a {\it free group}, realized in the domain
\spadtype{FreeMonoid} which won't be discussed here.  We consider {\it
permutation groups}, where a group is realized as a subgroup of the
symmetric group of a set, i.e.  the group of all bijections of a set,
the operation being the composition of maps.  Indeed, every group can
be realized this way, although this may not be practical.

Furthermore group elements can be given as invertible matrices.  The
group operation is reflected by matrix multiplication.  More precise
in representation theory group homomophisms from a group to general
linear groups are contructed.  Some algorithms are implemented in
Axiom.
\newline
%\beginmenu
%\menulink{Permutation}{PermutationXmpPage}
%Calculating within symmetric groups.

%\menulink{Permutation Groups}{PermutationGroupXmpPage}
%Working with subgroups of a symmetric group.

%\menulink{Permutation Group Examples}
%{PermutationGroupExampleXmpPage}
%Working with permutation groups, predefined
in the system as Rubik's group.
%\endmenu
\endscroll
\autobuttons \end{page}
```

———-

3.52 gstbl.ht

GeneralSparseTable

⇒ "notitle" (TableXmpPage) 3.106 on page 1298

— **gstbl.ht** —

```
\begin{page}{GeneralSparseTableXmpPage}{GeneralSparseTable}
\beginscroll

Sometimes when working with tables there is a natural value to use as
the entry in all but a few cases.  The \spadtype{GeneralSparseTable}
constructor can be used to provide any table type with a default value
for entries.  See \downlink{`Table'}{TableXmpPage}\ignore{Table} for
general information about tables.
\showBlurb{GeneralSparseTable}

Suppose we launched a fund-raising campaign to raise fifty thousand
dollars.  To record the contributions, we want a table with strings as
keys (for the names) and integer entries (for the amount).  In a data
base of cash contributions, unless someone has been explicitly
entered, it is reasonable to assume they have made a zero dollar
contribution.
\xtc{
This creates a keyed access file with default entry \spad{0}.
}{
\spadpaste{patrons: GeneralSparseTable(String, Integer,
KeyedAccessFile(Integer), 0) := table() ; \bound{patrons}}
}
\xtc{
Now \spad{patrons} can be used just as any other table.
Here we record two gifts.
}{
\spadpaste{patrons."Smith" := 10500 \free{patrons}\bound{smith}}
}
\xtc{
}{
\spadpaste{patrons."Jones" := 22000 \free{smith}\bound{jones}}
}
\xtc{
Now let us look up the size of the contributions from Jones and Stingy.
}{
\spadpaste{patrons."Jones"  \free{jones}}
}
\xtc{
}{
\spadpaste{patrons."Stingy" \free{jones}}
}
```

```
\xtc{
Have we met our seventy thousand dollar goal?
}{
\spadpaste{reduce(+, entries patrons) \free{jones}}
}
\noOutputXtc{
So the project is cancelled and we can delete the data base:
}{
\spadpaste{)system rm -r kaf*.sdata \free{patrons}}
}
\endscroll
\autobuttons
\end{page}

\begin{patch}{GeneralSparseTableXmpPagePatch1}
\begin{paste}{GeneralSparseTableXmpPageFull1}{GeneralSparseTableXmpPageEmpty1}
\pastebutton{GeneralSparseTableXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{patrons: GeneralSparseTable(String, Integer, KeyedAccessFile(Integer), 0) :=
\indentrel{3}\begin{verbatim}
Type: GeneralSparseTable(String,Integer,KeyedAccessFile Integer,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{GeneralSparseTableXmpPageEmpty1}
\begin{paste}{GeneralSparseTableXmpPageEmpty1}{GeneralSparseTableXmpPagePatch1}
\pastebutton{GeneralSparseTableXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{patrons: GeneralSparseTable(String, Integer, KeyedAccessFile(Integer), 0) :=
\end{paste}\end{patch}

\begin{patch}{GeneralSparseTableXmpPagePatch2}
\begin{paste}{GeneralSparseTableXmpPageFull2}{GeneralSparseTableXmpPageEmpty2}
\pastebutton{GeneralSparseTableXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{patrons."Smith" := 10500\free{patrons }\bound{smith }}
\indentrel{3}\begin{verbatim}
   (2)  10500
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{GeneralSparseTableXmpPageEmpty2}
\begin{paste}{GeneralSparseTableXmpPageEmpty2}{GeneralSparseTableXmpPagePatch2}
\pastebutton{GeneralSparseTableXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{patrons."Smith" := 10500\free{patrons }\bound{smith }}
\end{paste}\end{patch}

\begin{patch}{GeneralSparseTableXmpPagePatch3}
\begin{paste}{GeneralSparseTableXmpPageFull3}{GeneralSparseTableXmpPageEmpty3}
\pastebutton{GeneralSparseTableXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{patrons."Jones" := 22000\free{smith }\bound{jones }}
\indentrel{3}\begin{verbatim}
```

```
   (3)  22000
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{GeneralSparseTableXmpPageEmpty3}
\begin{paste}{GeneralSparseTableXmpPageEmpty3}{GeneralSparseTableXmpPagePatch3}
\pastebutton{GeneralSparseTableXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{patrons."Jones" := 22000\free{smith }\bound{jones }}
\end{paste}\end{patch}

\begin{patch}{GeneralSparseTableXmpPagePatch4}
\begin{paste}{GeneralSparseTableXmpPageFull4}{GeneralSparseTableXmpPageEmpty4}
\pastebutton{GeneralSparseTableXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{patrons."Jones"\free{jones }}
\indentrel{3}\begin{verbatim}
   (4)  22000
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{GeneralSparseTableXmpPageEmpty4}
\begin{paste}{GeneralSparseTableXmpPageEmpty4}{GeneralSparseTableXmpPagePatch4}
\pastebutton{GeneralSparseTableXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{patrons."Jones"\free{jones }}
\end{paste}\end{patch}

\begin{patch}{GeneralSparseTableXmpPagePatch5}
\begin{paste}{GeneralSparseTableXmpPageFull5}{GeneralSparseTableXmpPageEmpty5}
\pastebutton{GeneralSparseTableXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{patrons."Stingy"\free{jones }}
\indentrel{3}\begin{verbatim}
   (5)  0
                               Type: NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{GeneralSparseTableXmpPageEmpty5}
\begin{paste}{GeneralSparseTableXmpPageEmpty5}{GeneralSparseTableXmpPagePatch5}
\pastebutton{GeneralSparseTableXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{patrons."Stingy"\free{jones }}
\end{paste}\end{patch}

\begin{patch}{GeneralSparseTableXmpPagePatch6}
\begin{paste}{GeneralSparseTableXmpPageFull6}{GeneralSparseTableXmpPageEmpty6}
\pastebutton{GeneralSparseTableXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{reduce(+, entries patrons)\free{jones }}
\indentrel{3}\begin{verbatim}
   (6)  32500
                                  Type: PositiveInteger
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{GeneralSparseTableXmpPageEmpty6}
\begin{paste}{GeneralSparseTableXmpPageEmpty6}{GeneralSparseTableXmpPagePatch6}
\pastebutton{GeneralSparseTableXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{reduce(+, entries patrons)\free{jones }}
\end{paste}\end{patch}

\begin{patch}{GeneralSparseTableXmpPagePatch7}
\begin{paste}{GeneralSparseTableXmpPageFull7}{GeneralSparseTableXmpPageEmpty7}
\pastebutton{GeneralSparseTableXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{)system rm -r kaf*.sdata\free{patrons }}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{GeneralSparseTableXmpPageEmpty7}
\begin{paste}{GeneralSparseTableXmpPageEmpty7}{GeneralSparseTableXmpPagePatch7}
\pastebutton{GeneralSparseTableXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{)system rm -r kaf*.sdata\free{patrons }}
\end{paste}\end{patch}
```

———-

3.53 heap.ht

Heap

⇒ "notitle" (FlexibleArrayXmpPage) 3.39 on page 472

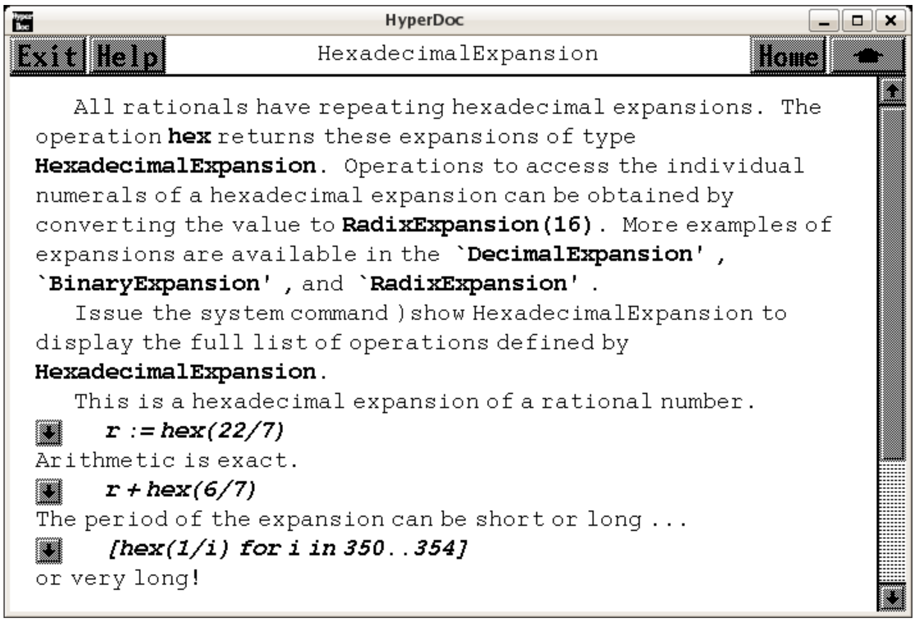
— **heap.ht** —

```
\begin{page}{HeapXmpPage}{Heap}
\beginscroll
The domain \spadtype{Heap(S)} implements a priority queue of objects
of type \spad{S} such that the operation \spadfunX{extract} removes
and returns the maximum element.  The implementation represents heaps
as flexible arrays (see \downlink{`FlexibleArray'}{FlexibleArrayXmpPage}
\ignore{FlexibleArray}).  The representation and algorithms give
complexity of \texht{$O(\log(n))$}{O(log n)} for insertion and
extractions, and \texht{$O(n)$}{O(n)} for construction.
\xtc{
Create a heap of six elements.
}{
\spadpaste{h := heap [-4,9,11,2,7,-7]\bound{h}}
}
```

```
\xtc{
Use \spadfunX{insert} to add an element.
}{
\spadpaste{insert!(3,h)\bound{h1}\free{h}}
}
\xtc{
The operation \spadfunX{extract} removes and returns
the maximum element.
}{
\spadpaste{extract! h\bound{h2}\free{h1}}
}
\xtc{
The internal structure of \spad{h} has been
appropriately adjusted.
}{
\spadpaste{h\free{h2}}
}
\xtc{
Now \spadfunX{extract} elements repeatedly
until none are left, collecting the elements in a list.
}{
\spadpaste{[extract!(h) while not empty?(h)]\bound{h2}}
}
\xtc{
Another way to produce the same result is by defining
a \userfun{heapsort} function.
}{
\spadpaste{heapsort(x) == (empty? x => []; cons(extract!(x),heapsort x))
\bound{f}}
}
\xtc{
Create another sample heap.
}{
\spadpaste{h1 := heap [17,-4,9,-11,2,7,-7]\bound{h1}}
}
\xtc{
Apply \spadfun{heapsort} to present elements in order.
}{
\spadpaste{heapsort h1\free{f}}
}
\endscroll
\autobuttons
\end{page}
```

3.54 hexadec.ht

HexadecimalExpansion



⇐ "Axiom Number Types" (NumberPage) 3.80 on page 1021
⇐ "Integer" (IntegerXmpPage) 3.55 on page 696
⇒ "DecimalExpansion" (DecimalExpansionXmpPage) 3.21 on page 348
⇒ "BinaryExpansion" (BinaryExpansionXmpPage) 3.8 on page 163
⇒ "RadixExpansion" (RadixExpansionXmpPage) 3.90 on page 1140

— **hexadec.ht** —

```
\begin{page}{HexExpansionXmpPage}{HexadecimalExpansion}
\beginscroll
All rationals have repeating hexadecimal expansions.  The operation
\spadfunFrom{hex}{HexadecimalExpansion} returns these expansions of
type \spadtype{HexadecimalExpansion}.  Operations to access the
individual numerals of a hexadecimal expansion can be obtained by
converting the value to \spadtype{RadixExpansion(16)}.  More examples
of expansions are available in the
\downlink{`DecimalExpansion'}{DecimalExpansionXmpPage}
\ignore{DecimalExpansion},
\downlink{`BinaryExpansion'}{BinaryExpansionXmpPage}
\ignore{BinaryExpansion}, and
\downlink{`RadixExpansion'}{RadixExpansionXmpPage}
\ignore{RadixExpansion}.
```

```
\showBlurb{HexadecimalExpansion}

\xtc{
This is a hexadecimal expansion of a rational number.
}{
\spadpaste{r := hex(22/7) \bound{r}}
}
\xtc{
Arithmetic is exact.
}{
\spadpaste{r + hex(6/7) \free{r}}
}
\xtc{
The period of the expansion can be short or long \ldots
}{
\spadpaste{[hex(1/i) for i in 350..354] }
}
\xtc{
or very long!
}{
\spadpaste{hex(1/1007) }
}
\xtc{
These numbers are bona fide algebraic objects.
}{
\spadpaste{p := hex(1/4)*x**2 + hex(2/3)*x + hex(4/9)  \bound{p}}
}
\xtc{
}{
\spadpaste{q := D(p, x) \free{p}\bound{q}}
}
\xtc{
}{
\spadpaste{g := gcd(p, q)            \free{p}\free{q}}
}
\endscroll
\autobuttons
\end{page}

\begin{patch}{HexExpansionXmpPagePatch1}
\begin{paste}{HexExpansionXmpPageFull1}{HexExpansionXmpPageEmpty1}
\pastebutton{HexExpansionXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{r := hex(22/7)\bound{r }}
\indentrel{3}\begin{verbatim}
          ---
   (1)  3.249
                            Type: HexadecimalExpansion
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{HexExpansionXmpPageEmpty1}
\begin{paste}{HexExpansionXmpPageEmpty1}{HexExpansionXmpPagePatch1}
\pastebutton{HexExpansionXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{r := hex(22/7)\bound{r }}
\end{paste}\end{patch}

\begin{patch}{HexExpansionXmpPagePatch2}
\begin{paste}{HexExpansionXmpPageFull2}{HexExpansionXmpPageEmpty2}
\pastebutton{HexExpansionXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{r + hex(6/7)\free{r }}
\indentrel{3}\begin{verbatim}
   (2)  4
                             Type: HexadecimalExpansion
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{HexExpansionXmpPageEmpty2}
\begin{paste}{HexExpansionXmpPageEmpty2}{HexExpansionXmpPagePatch2}
\pastebutton{HexExpansionXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{r + hex(6/7)\free{r }}
\end{paste}\end{patch}

\begin{patch}{HexExpansionXmpPagePatch3}
\begin{paste}{HexExpansionXmpPageFull3}{HexExpansionXmpPageEmpty3}
\pastebutton{HexExpansionXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{[hex(1/i) for i in 350..354]}
\indentrel{3}\begin{verbatim}
   (3)
       ---------------    ---------      -----
   [0.00BB3EE721A54D88, 0.00BAB6561, 0.00BA2E8,
      ----------------------
    0.00B9A7862A0FF465879D5F,
       -----------------------------
    0.00B92143FA36F5E02E4850FE8DBD78]
                        Type: List HexadecimalExpansion
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{HexExpansionXmpPageEmpty3}
\begin{paste}{HexExpansionXmpPageEmpty3}{HexExpansionXmpPagePatch3}
\pastebutton{HexExpansionXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{[hex(1/i) for i in 350..354]}
\end{paste}\end{patch}

\begin{patch}{HexExpansionXmpPagePatch4}
\begin{paste}{HexExpansionXmpPageFull4}{HexExpansionXmpPageEmpty4}
\pastebutton{HexExpansionXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{hex(1/1007)}
\indentrel{3}\begin{verbatim}
   (4)
```

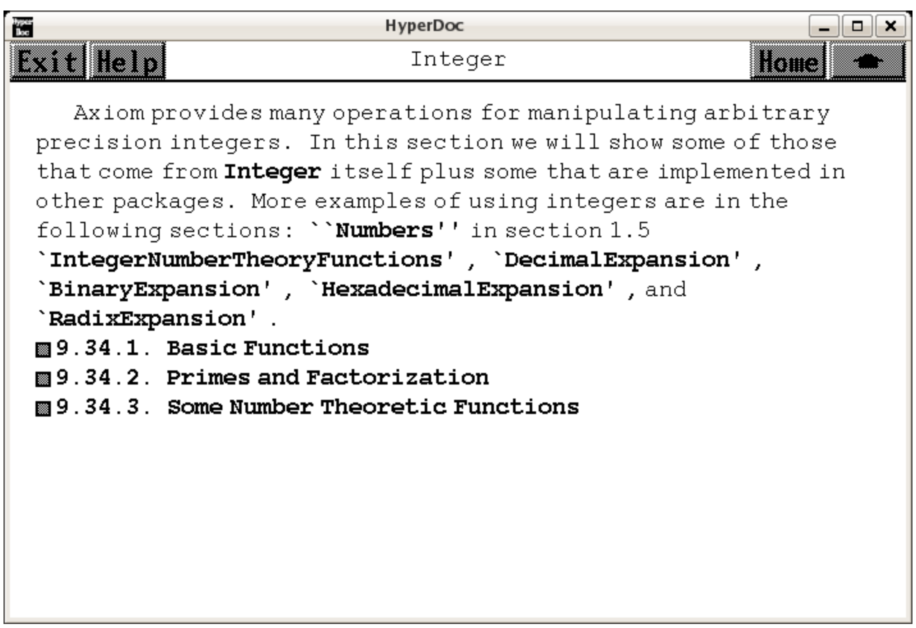
```
   0.
     OVERBAR
        0041149783F0BF2C7D13933192AF6980619EE345E91EC2BB9
          D5CCA5C071E40926E54E8DDAE24196C0B2F8A0AAD60DBA5
          7F5D4C8536262210C74F1
                              Type: HexadecimalExpansion
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{HexExpansionXmpPageEmpty4}
\begin{paste}{HexExpansionXmpPageEmpty4}{HexExpansionXmpPagePatch4}
\pastebutton{HexExpansionXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{hex(1/1007)}
\end{paste}\end{patch}

\begin{patch}{HexExpansionXmpPagePatch5}
\begin{paste}{HexExpansionXmpPageFull5}{HexExpansionXmpPageEmpty5}
\pastebutton{HexExpansionXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{p := hex(1/4)*x**2 + hex(2/3)*x + hex(4/9)\bound{p }}
\indentrel{3}\begin{verbatim}
           2     _      ___
   (5)  0.4x  + 0.Ax + 0.71C
                  Type: Polynomial HexadecimalExpansion
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{HexExpansionXmpPageEmpty5}
\begin{paste}{HexExpansionXmpPageEmpty5}{HexExpansionXmpPagePatch5}
\pastebutton{HexExpansionXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{p := hex(1/4)*x**2 + hex(2/3)*x + hex(4/9)\bound{p }}
\end{paste}\end{patch}

\begin{patch}{HexExpansionXmpPagePatch6}
\begin{paste}{HexExpansionXmpPageFull6}{HexExpansionXmpPageEmpty6}
\pastebutton{HexExpansionXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{q := D(p, x)\free{p }\bound{q }}
\indentrel{3}\begin{verbatim}
                  _
   (6)  0.8x + 0.A
                  Type: Polynomial HexadecimalExpansion
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{HexExpansionXmpPageEmpty6}
\begin{paste}{HexExpansionXmpPageEmpty6}{HexExpansionXmpPagePatch6}
\pastebutton{HexExpansionXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{q := D(p, x)\free{p }\bound{q }}
\end{paste}\end{patch}

\begin{patch}{HexExpansionXmpPagePatch7}
```

```
\begin{paste}{HexExpansionXmpPageFull7}{HexExpansionXmpPageEmpty7}
\pastebutton{HexExpansionXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{g := gcd(p, q)\free{p }\free{q }}
\indentrel{3}\begin{verbatim}
                 _
   (7)  x + 1.5
                      Type: Polynomial HexadecimalExpansion
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{HexExpansionXmpPageEmpty7}
\begin{paste}{HexExpansionXmpPageEmpty7}{HexExpansionXmpPagePatch7}
\pastebutton{HexExpansionXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{g := gcd(p, q)\free{p }\free{q }}
\end{paste}\end{patch}
```

———–

3.55 int.ht

Integer



⇐ "Integers" (IntegerPage) 3.80 on page 1029
⇒ "Numbers" (ugIntroNumbersPage) 6 on page 1513

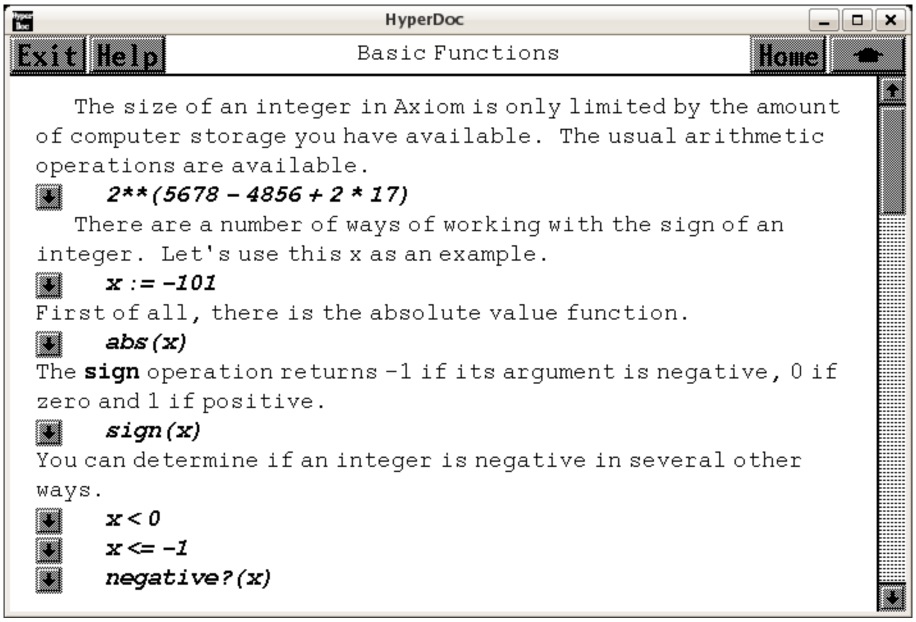
⇒ “IntegerNumberTheoryFunctions” (IntNumberTheoryFnsXmpPage) 3.56 on page 722
⇒ “DecimalExpansion” (DecimalExpansionXmpPage) 3.21 on page 348
⇒ “BinaryExpansion” (BinaryExpansionXmpPage) 3.8 on page 163
⇒ “HexadecimalExpansion” (HexExpansionXmpPage) 3.54 on page 692
⇒ “RadixExpansion” (RadixExpansionXmpPage) 3.90 on page 1140
⇒ “Basic Functions” (ugxIntegerBasicPage) 3.55 on page 698
⇒ “Primes and Factorization” (ugxIntegerPrimesPage) 3.55 on page 712
⇒ “Some Number Theoretic Functions” (ugxIntegerNTPage) 3.55 on page 716

— **int.ht** —

```
\begin{page}{IntegerXmpPage}{Integer}
\beginscroll

Axiom provides many operations for manipulating arbitrary precision
integers.  In this section we will show some of those that come from
\spadtype{Integer} itself plus some that are implemented in other
packages.  More examples of using integers are in the following
sections:
\downlink{``Numbers''}{ugIntroNumbersPage} in section 1.5
\downlink{`IntegerNumberTheoryFunctions'}
{IntNumberTheoryFnsXmpPage}
\ignore{IntegerNumberTheoryFunctions},
\downlink{`DecimalExpansion'}{DecimalExpansionXmpPage}
\ignore{DecimalExpansion},
\downlink{`BinaryExpansion'}{BinaryExpansionXmpPage}
\ignore{BinaryExpansion},
\downlink{`HexadecimalExpansion'}{HexExpansionXmpPage}
\ignore{HexadecimalExpansion},
and
\downlink{`RadixExpansion'}{RadixExpansionXmpPage}
\ignore{RadixExpansion}.

\beginmenu
    \menudownlink{{9.34.1. Basic Functions}}{ugxIntegerBasicPage}
    \menudownlink{{9.34.2. Primes and Factorization}}
                   {ugxIntegerPrimesPage}
    \menudownlink{{9.34.3. Some Number Theoretic Functions}}
                  {ugxIntegerNTPage}
\endmenu
\endscroll
\autobuttons
\end{page}
```

————

Basic Functions



⇐ "Integer" (IntegerXmpPage) 3.55 on page 696
⇒ "Fraction" (FractionXmpPage) 3.45 on page 543
⇒ "Unions" (ugTypesUnionsPage) 7 on page 1656
⇒ "Records" (ugTypesRecordsPage) 7 on page 1647

— **int.ht** —

```
\begin{page}{ugxIntegerBasicPage}{Basic Functions}
\beginscroll

\labelSpace{3pc}
\xtc{
The size of an integer in Axiom is only limited by the amount of
computer storage you have available.
The usual arithmetic operations are available.
}{
\spadpaste{2**(5678 - 4856 + 2 * 17)}
}

\xtc{
There are a number of ways of working with the sign of an integer.
Let's use this \spad{x} as an example.
}{
\spadpaste{x := -101 \bound{x}}
}
\xtc{
```

```
First of all, there is the absolute value function.
}{
\spadpaste{abs(x) \free{x}}
}
\xtc{
The \spadfunFrom{sign}{Integer} operation returns \spad{-1} if its
argument is negative, \spad{0} if zero and \spad{1} if positive.
}{
\spadpaste{sign(x) \free{x}}
}
%
\xtc{
You can determine if an integer is negative in several other ways.
}{
\spadpaste{x < 0 \free{x}}
}
\xtc{
}{
\spadpaste{x <= -1 \free{x}}
}
\xtc{
}{
\spadpaste{negative?(x) \free{x}}
}
%
\xtc{
Similarly, you can find out if it is positive.
}{
\spadpaste{x > 0 \free{x}}
}
\xtc{
}{
\spadpaste{x >= 1 \free{x}}
}
\xtc{
}{
\spadpaste{positive?(x) \free{x}}
}
\xtc{
This is the recommended way of determining whether an integer is zero.
}{
\spadpaste{zero?(x) \free{x}}
}

\beginImportant
Use the \spadfunFrom{zero?}{Integer} operation whenever you are
testing any mathematical object for equality with zero.  This is
usually more efficient that using \spadop{=} (think of matrices: it is
easier to tell if a matrix is zero by just checking term by term than
constructing another ‘‘zero’’ matrix and comparing the two matrices
```

```
term by term) and also avoids the problem that \spadop{=} is usually
used for creating equations.
\endImportant

\xtc{
This is the recommended way of determining
whether an integer is equal to one.
}{
\spadpaste{one?(x) \free{x}}
}

\xtc{
This syntax is used to test equality using \spadopFrom{=}{Integer}.
It says that you want a \spadtype{Boolean} (\spad{true} or
\spad{false}) answer rather than an equation.
}{
\spadpaste{(x = -101)@Boolean \free{x}}
}

\xtc{
The operations \spadfunFrom{odd?}{Integer} and
\spadfunFrom{even?}{Integer} determine whether an integer is odd or
even, respectively.  They each return a \spadtype{Boolean} object.
}{
\spadpaste{odd?(x) \free{x}}
}
\xtc{
}{
\spadpaste{even?(x) \free{x}}
}
\xtc{
The operation \spadfunFrom{gcd}{Integer} computes the greatest common
divisor of two integers.
}{
\spadpaste{gcd(56788,43688)}
}
\xtc{
The operation
\spadfunFrom{lcm}{Integer} computes their least common multiple.
}{
\spadpaste{lcm(56788,43688)}
}
\xtc{
To determine the maximum of two integers, use \spadfunFrom{max}{Integer}.
}{
\spadpaste{max(678,567)}
}
\xtc{
To determine the minimum, use \spadfunFrom{min}{Integer}.
}{
```

```
\spadpaste{min(678,567)}
}

\xtc{
The \spadfun{reduce} operation is used to extend
binary operations to more than two arguments.
For example, you can use \spadfun{reduce} to find the maximum integer in
a list or compute the least common multiple of all integers in the list.
}{
\spadpaste{reduce(max,[2,45,-89,78,100,-45])}
}
\xtc{
}{
\spadpaste{reduce(min,[2,45,-89,78,100,-45])}
}
\xtc{
}{
\spadpaste{reduce(gcd,[2,45,-89,78,100,-45])}
}
\xtc{
}{
\spadpaste{reduce(lcm,[2,45,-89,78,100,-45])}
}

\xtc{
The infix operator ``/'' is {\it not} used to compute the quotient
of integers.
Rather, it is used to create rational numbers as described in
\downlink{`Fraction'}{FractionXmpPage}\ignore{Fraction}.
}{
\spadpaste{13 / 4}
}
\xtc{
The infix operation \spadfunFrom{quo}{Integer} computes the integer
quotient.
}{
\spadpaste{13 quo 4}
}
\xtc{
The infix operation \spadfunFrom{rem}{Integer} computes the integer
remainder.
}{
\spadpaste{13 rem 4}
}
\xtc{
One integer is evenly divisible by another if the remainder is zero.
The operation \spadfunFrom{exquo}{Integer} can also be used.
See \downlink{``Unions''}{ugTypesUnionsPage} in Section 2.5
\ignore{ugTypesUnions} for an example.
}{
```

```
\spadpaste{zero?(167604736446952 rem 2003644)}
}

\xtc{
The operation \spadfunFrom{divide}{Integer} returns a record of the
quotient and remainder and thus is more efficient when both are needed.
}{
\spadpaste{d := divide(13,4) \bound{d}}
}
\xtc{
}{
\spadpaste{d.quotient \free{d}}
}
\xtc{
Records are discussed in detail in
\downlink{``Records''}{ugTypesRecordsPage} in Section 2.4
\ignore{ugTypesRecords}.
}{
\spadpaste{d.remainder \free{d}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxIntegerBasicPagePatch1}
\begin{paste}{ugxIntegerBasicPageFull1}{ugxIntegerBasicPageEmpty1}
\pastebutton{ugxIntegerBasicPageFull1}{\hidepaste}
\tab{5}\spadcommand{2**(5678 - 4856 + 2 * 17)}
\indentrel{3}\begin{verbatim}
   (1)
  480481077043500814718154092512592439123952613987168226_
   347385561008808420007630829308634252709141208374307 45_
   722782114960762769220264334356875273349802495393024 25_
   425230458177649495442143929053063884787051467457680 73_
   8771416988598154956329352887833342506287 75936
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPageEmpty1}
\begin{paste}{ugxIntegerBasicPageEmpty1}{ugxIntegerBasicPagePatch1}
\pastebutton{ugxIntegerBasicPageEmpty1}{\showpaste}
\tab{5}\spadcommand{2**(5678 - 4856 + 2 * 17)}
\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPagePatch2}
\begin{paste}{ugxIntegerBasicPageFull2}{ugxIntegerBasicPageEmpty2}
\pastebutton{ugxIntegerBasicPageFull2}{\hidepaste}
\tab{5}\spadcommand{x := -101\bound{x }}
```

```
\indentrel{3}\begin{verbatim}
   (2)  - 101
                                          Type: Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPageEmpty2}
\begin{paste}{ugxIntegerBasicPageEmpty2}{ugxIntegerBasicPagePatch2}
\pastebutton{ugxIntegerBasicPageEmpty2}{\showpaste}
\tab{5}\spadcommand{x := -101\bound{x }}
\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPagePatch3}
\begin{paste}{ugxIntegerBasicPageFull3}{ugxIntegerBasicPageEmpty3}
\pastebutton{ugxIntegerBasicPageFull3}{\hidepaste}
\tab{5}\spadcommand{abs(x)\free{x }}
\indentrel{3}\begin{verbatim}
   (3)  101
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPageEmpty3}
\begin{paste}{ugxIntegerBasicPageEmpty3}{ugxIntegerBasicPagePatch3}
\pastebutton{ugxIntegerBasicPageEmpty3}{\showpaste}
\tab{5}\spadcommand{abs(x)\free{x }}
\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPagePatch4}
\begin{paste}{ugxIntegerBasicPageFull4}{ugxIntegerBasicPageEmpty4}
\pastebutton{ugxIntegerBasicPageFull4}{\hidepaste}
\tab{5}\spadcommand{sign(x)\free{x }}
\indentrel{3}\begin{verbatim}
   (4)  - 1
                                          Type: Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPageEmpty4}
\begin{paste}{ugxIntegerBasicPageEmpty4}{ugxIntegerBasicPagePatch4}
\pastebutton{ugxIntegerBasicPageEmpty4}{\showpaste}
\tab{5}\spadcommand{sign(x)\free{x }}
\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPagePatch5}
\begin{paste}{ugxIntegerBasicPageFull5}{ugxIntegerBasicPageEmpty5}
\pastebutton{ugxIntegerBasicPageFull5}{\hidepaste}
\tab{5}\spadcommand{x < 0\free{x }}
\indentrel{3}\begin{verbatim}
   (5)  true
```

```
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPageEmpty5}
\begin{paste}{ugxIntegerBasicPageEmpty5}{ugxIntegerBasicPagePatch5}
\pastebutton{ugxIntegerBasicPageEmpty5}{\showpaste}
\tab{5}\spadcommand{x < 0\free{x }}
\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPagePatch6}
\begin{paste}{ugxIntegerBasicPageFull6}{ugxIntegerBasicPageEmpty6}
\pastebutton{ugxIntegerBasicPageFull6}{\hidepaste}
\tab{5}\spadcommand{x <= -1\free{x }}
\indentrel{3}\begin{verbatim}
   (6)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPageEmpty6}
\begin{paste}{ugxIntegerBasicPageEmpty6}{ugxIntegerBasicPagePatch6}
\pastebutton{ugxIntegerBasicPageEmpty6}{\showpaste}
\tab{5}\spadcommand{x <= -1\free{x }}
\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPagePatch7}
\begin{paste}{ugxIntegerBasicPageFull7}{ugxIntegerBasicPageEmpty7}
\pastebutton{ugxIntegerBasicPageFull7}{\hidepaste}
\tab{5}\spadcommand{negative?(x)\free{x }}
\indentrel{3}\begin{verbatim}
   (7)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPageEmpty7}
\begin{paste}{ugxIntegerBasicPageEmpty7}{ugxIntegerBasicPagePatch7}
\pastebutton{ugxIntegerBasicPageEmpty7}{\showpaste}
\tab{5}\spadcommand{negative?(x)\free{x }}
\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPagePatch8}
\begin{paste}{ugxIntegerBasicPageFull8}{ugxIntegerBasicPageEmpty8}
\pastebutton{ugxIntegerBasicPageFull8}{\hidepaste}
\tab{5}\spadcommand{x > 0\free{x }}
\indentrel{3}\begin{verbatim}
   (8)  false
                                          Type: Boolean
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPageEmpty8}
\begin{paste}{ugxIntegerBasicPageEmpty8}{ugxIntegerBasicPagePatch8}
\pastebutton{ugxIntegerBasicPageEmpty8}{\showpaste}
\tab{5}\spadcommand{x > 0\free{x }}
\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPagePatch9}
\begin{paste}{ugxIntegerBasicPageFull9}{ugxIntegerBasicPageEmpty9}
\pastebutton{ugxIntegerBasicPageFull9}{\hidepaste}
\tab{5}\spadcommand{x >= 1\free{x }}
\indentrel{3}\begin{verbatim}
   (9)  false
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPageEmpty9}
\begin{paste}{ugxIntegerBasicPageEmpty9}{ugxIntegerBasicPagePatch9}
\pastebutton{ugxIntegerBasicPageEmpty9}{\showpaste}
\tab{5}\spadcommand{x >= 1\free{x }}
\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPagePatch10}
\begin{paste}{ugxIntegerBasicPageFull10}{ugxIntegerBasicPageEmpty10}
\pastebutton{ugxIntegerBasicPageFull10}{\hidepaste}
\tab{5}\spadcommand{positive?(x)\free{x }}
\indentrel{3}\begin{verbatim}
   (10)  false
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPageEmpty10}
\begin{paste}{ugxIntegerBasicPageEmpty10}{ugxIntegerBasicPagePatch10}
\pastebutton{ugxIntegerBasicPageEmpty10}{\showpaste}
\tab{5}\spadcommand{positive?(x)\free{x }}
\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPagePatch11}
\begin{paste}{ugxIntegerBasicPageFull11}{ugxIntegerBasicPageEmpty11}
\pastebutton{ugxIntegerBasicPageFull11}{\hidepaste}
\tab{5}\spadcommand{zero?(x)\free{x }}
\indentrel{3}\begin{verbatim}
   (11)  false
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugxIntegerBasicPageEmpty11}
\begin{paste}{ugxIntegerBasicPageEmpty11}{ugxIntegerBasicPagePatch11}
\pastebutton{ugxIntegerBasicPageEmpty11}{\showpaste}
\tab{5}\spadcommand{zero?(x)\free{x }}
\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPagePatch12}
\begin{paste}{ugxIntegerBasicPageFull12}{ugxIntegerBasicPageEmpty12}
\pastebutton{ugxIntegerBasicPageFull12}{\hidepaste}
\tab{5}\spadcommand{one?(x)\free{x }}
\indentrel{3}\begin{verbatim}
   (12)  false
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPageEmpty12}
\begin{paste}{ugxIntegerBasicPageEmpty12}{ugxIntegerBasicPagePatch12}
\pastebutton{ugxIntegerBasicPageEmpty12}{\showpaste}
\tab{5}\spadcommand{one?(x)\free{x }}
\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPagePatch13}
\begin{paste}{ugxIntegerBasicPageFull13}{ugxIntegerBasicPageEmpty13}
\pastebutton{ugxIntegerBasicPageFull13}{\hidepaste}
\tab{5}\spadcommand{(x = -101)@Boolean\free{x }}
\indentrel{3}\begin{verbatim}
   (13)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPageEmpty13}
\begin{paste}{ugxIntegerBasicPageEmpty13}{ugxIntegerBasicPagePatch13}
\pastebutton{ugxIntegerBasicPageEmpty13}{\showpaste}
\tab{5}\spadcommand{(x = -101)@Boolean\free{x }}
\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPagePatch14}
\begin{paste}{ugxIntegerBasicPageFull14}{ugxIntegerBasicPageEmpty14}
\pastebutton{ugxIntegerBasicPageFull14}{\hidepaste}
\tab{5}\spadcommand{odd?(x)\free{x }}
\indentrel{3}\begin{verbatim}
   (14)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPageEmpty14}
\begin{paste}{ugxIntegerBasicPageEmpty14}{ugxIntegerBasicPagePatch14}
```

```
\pastebutton{ugxIntegerBasicPageEmpty14}{\showpaste}
\tab{5}\spadcommand{odd?(x)\free{x }}
\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPagePatch15}
\begin{paste}{ugxIntegerBasicPageFull15}{ugxIntegerBasicPageEmpty15}
\pastebutton{ugxIntegerBasicPageFull15}{\hidepaste}
\tab{5}\spadcommand{even?(x)\free{x }}
\indentrel{3}\begin{verbatim}
   (15)  false
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPageEmpty15}
\begin{paste}{ugxIntegerBasicPageEmpty15}{ugxIntegerBasicPagePatch15}
\pastebutton{ugxIntegerBasicPageEmpty15}{\showpaste}
\tab{5}\spadcommand{even?(x)\free{x }}
\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPagePatch16}
\begin{paste}{ugxIntegerBasicPageFull16}{ugxIntegerBasicPageEmpty16}
\pastebutton{ugxIntegerBasicPageFull16}{\hidepaste}
\tab{5}\spadcommand{gcd(56788,43688)}
\indentrel{3}\begin{verbatim}
   (16)  4
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPageEmpty16}
\begin{paste}{ugxIntegerBasicPageEmpty16}{ugxIntegerBasicPagePatch16}
\pastebutton{ugxIntegerBasicPageEmpty16}{\showpaste}
\tab{5}\spadcommand{gcd(56788,43688)}
\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPagePatch17}
\begin{paste}{ugxIntegerBasicPageFull17}{ugxIntegerBasicPageEmpty17}
\pastebutton{ugxIntegerBasicPageFull17}{\hidepaste}
\tab{5}\spadcommand{lcm(56788,43688)}
\indentrel{3}\begin{verbatim}
   (17)  620238536
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPageEmpty17}
\begin{paste}{ugxIntegerBasicPageEmpty17}{ugxIntegerBasicPagePatch17}
\pastebutton{ugxIntegerBasicPageEmpty17}{\showpaste}
\tab{5}\spadcommand{lcm(56788,43688)}
```

```
\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPagePatch18}
\begin{paste}{ugxIntegerBasicPageFull18}{ugxIntegerBasicPageEmpty18}
\pastebutton{ugxIntegerBasicPageFull18}{\hidepaste}
\tab{5}\spadcommand{max(678,567)}
\indentrel{3}\begin{verbatim}
   (18)  678
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPageEmpty18}
\begin{paste}{ugxIntegerBasicPageEmpty18}{ugxIntegerBasicPagePatch18}
\pastebutton{ugxIntegerBasicPageEmpty18}{\showpaste}
\tab{5}\spadcommand{max(678,567)}
\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPagePatch19}
\begin{paste}{ugxIntegerBasicPageFull19}{ugxIntegerBasicPageEmpty19}
\pastebutton{ugxIntegerBasicPageFull19}{\hidepaste}
\tab{5}\spadcommand{min(678,567)}
\indentrel{3}\begin{verbatim}
   (19)  567
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPageEmpty19}
\begin{paste}{ugxIntegerBasicPageEmpty19}{ugxIntegerBasicPagePatch19}
\pastebutton{ugxIntegerBasicPageEmpty19}{\showpaste}
\tab{5}\spadcommand{min(678,567)}
\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPagePatch20}
\begin{paste}{ugxIntegerBasicPageFull20}{ugxIntegerBasicPageEmpty20}
\pastebutton{ugxIntegerBasicPageFull20}{\hidepaste}
\tab{5}\spadcommand{reduce(max,[2,45,-89,78,100,-45])}
\indentrel{3}\begin{verbatim}
   (20)  100
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPageEmpty20}
\begin{paste}{ugxIntegerBasicPageEmpty20}{ugxIntegerBasicPagePatch20}
\pastebutton{ugxIntegerBasicPageEmpty20}{\showpaste}
\tab{5}\spadcommand{reduce(max,[2,45,-89,78,100,-45])}
\end{paste}\end{patch}
```

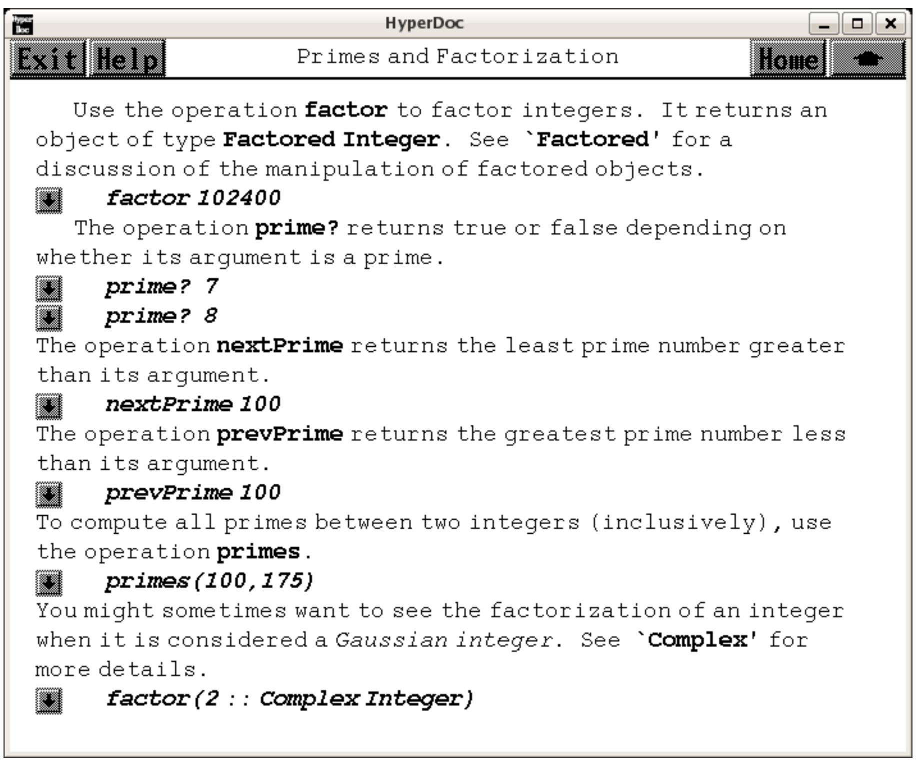
```
\begin{patch}{ugxIntegerBasicPagePatch21}
\begin{paste}{ugxIntegerBasicPageFull21}{ugxIntegerBasicPageEmpty21}
\pastebutton{ugxIntegerBasicPageFull21}{\hidepaste}
\tab{5}\spadcommand{reduce(min,[2,45,-89,78,100,-45])}
\indentrel{3}\begin{verbatim}
   (21)  - 89
                                            Type: Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPageEmpty21}
\begin{paste}{ugxIntegerBasicPageEmpty21}{ugxIntegerBasicPagePatch21}
\pastebutton{ugxIntegerBasicPageEmpty21}{\showpaste}
\tab{5}\spadcommand{reduce(min,[2,45,-89,78,100,-45])}
\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPagePatch22}
\begin{paste}{ugxIntegerBasicPageFull22}{ugxIntegerBasicPageEmpty22}
\pastebutton{ugxIntegerBasicPageFull22}{\hidepaste}
\tab{5}\spadcommand{reduce(gcd,[2,45,-89,78,100,-45])}
\indentrel{3}\begin{verbatim}
   (22)  1
                                    Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPageEmpty22}
\begin{paste}{ugxIntegerBasicPageEmpty22}{ugxIntegerBasicPagePatch22}
\pastebutton{ugxIntegerBasicPageEmpty22}{\showpaste}
\tab{5}\spadcommand{reduce(gcd,[2,45,-89,78,100,-45])}
\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPagePatch23}
\begin{paste}{ugxIntegerBasicPageFull23}{ugxIntegerBasicPageEmpty23}
\pastebutton{ugxIntegerBasicPageFull23}{\hidepaste}
\tab{5}\spadcommand{reduce(lcm,[2,45,-89,78,100,-45])}
\indentrel{3}\begin{verbatim}
   (23)  1041300
                                    Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPageEmpty23}
\begin{paste}{ugxIntegerBasicPageEmpty23}{ugxIntegerBasicPagePatch23}
\pastebutton{ugxIntegerBasicPageEmpty23}{\showpaste}
\tab{5}\spadcommand{reduce(lcm,[2,45,-89,78,100,-45])}
\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPagePatch24}
\begin{paste}{ugxIntegerBasicPageFull24}{ugxIntegerBasicPageEmpty24}
```

```
\pastebutton{ugxIntegerBasicPageFull24}{\hidepaste}
\tab{5}\spadcommand{13 / 4}
\indentrel{3}\begin{verbatim}
         13
   (24)
          4
                                 Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPageEmpty24}
\begin{paste}{ugxIntegerBasicPageEmpty24}{ugxIntegerBasicPagePatch24}
\pastebutton{ugxIntegerBasicPageEmpty24}{\showpaste}
\tab{5}\spadcommand{13 / 4}
\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPagePatch25}
\begin{paste}{ugxIntegerBasicPageFull25}{ugxIntegerBasicPageEmpty25}
\pastebutton{ugxIntegerBasicPageFull25}{\hidepaste}
\tab{5}\spadcommand{13 quo 4}
\indentrel{3}\begin{verbatim}
   (25)  3
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPageEmpty25}
\begin{paste}{ugxIntegerBasicPageEmpty25}{ugxIntegerBasicPagePatch25}
\pastebutton{ugxIntegerBasicPageEmpty25}{\showpaste}
\tab{5}\spadcommand{13 quo 4}
\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPagePatch26}
\begin{paste}{ugxIntegerBasicPageFull26}{ugxIntegerBasicPageEmpty26}
\pastebutton{ugxIntegerBasicPageFull26}{\hidepaste}
\tab{5}\spadcommand{13 rem 4}
\indentrel{3}\begin{verbatim}
   (26)  1
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPageEmpty26}
\begin{paste}{ugxIntegerBasicPageEmpty26}{ugxIntegerBasicPagePatch26}
\pastebutton{ugxIntegerBasicPageEmpty26}{\showpaste}
\tab{5}\spadcommand{13 rem 4}
\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPagePatch27}
\begin{paste}{ugxIntegerBasicPageFull27}{ugxIntegerBasicPageEmpty27}
```

```
\pastebutton{ugxIntegerBasicPageFull27}{\hidepaste}
\tab{5}\spadcommand{zero?(167604736446952 rem 2003644)}
\indentrel{3}\begin{verbatim}
   (27)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPageEmpty27}
\begin{paste}{ugxIntegerBasicPageEmpty27}{ugxIntegerBasicPagePatch27}
\pastebutton{ugxIntegerBasicPageEmpty27}{\showpaste}
\tab{5}\spadcommand{zero?(167604736446952 rem 2003644)}
\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPagePatch28}
\begin{paste}{ugxIntegerBasicPageFull28}{ugxIntegerBasicPageEmpty28}
\pastebutton{ugxIntegerBasicPageFull28}{\hidepaste}
\tab{5}\spadcommand{d := divide(13,4)\bound{d }}
\indentrel{3}\begin{verbatim}
   (28)  [quotient= 3,remainder= 1]
     Type: Record(quotient: Integer,remainder: Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPageEmpty28}
\begin{paste}{ugxIntegerBasicPageEmpty28}{ugxIntegerBasicPagePatch28}
\pastebutton{ugxIntegerBasicPageEmpty28}{\showpaste}
\tab{5}\spadcommand{d := divide(13,4)\bound{d }}
\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPagePatch29}
\begin{paste}{ugxIntegerBasicPageFull29}{ugxIntegerBasicPageEmpty29}
\pastebutton{ugxIntegerBasicPageFull29}{\hidepaste}
\tab{5}\spadcommand{d.quotient\free{d }}
\indentrel{3}\begin{verbatim}
   (29)  3
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPageEmpty29}
\begin{paste}{ugxIntegerBasicPageEmpty29}{ugxIntegerBasicPagePatch29}
\pastebutton{ugxIntegerBasicPageEmpty29}{\showpaste}
\tab{5}\spadcommand{d.quotient\free{d }}
\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPagePatch30}
\begin{paste}{ugxIntegerBasicPageFull30}{ugxIntegerBasicPageEmpty30}
\pastebutton{ugxIntegerBasicPageFull30}{\hidepaste}
\tab{5}\spadcommand{d.remainder\free{d }}
```

```
\indentrel{3}\begin{verbatim}
   (30)  1
                                    Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerBasicPageEmpty30}
\begin{paste}{ugxIntegerBasicPageEmpty30}{ugxIntegerBasicPagePatch30}
\pastebutton{ugxIntegerBasicPageEmpty30}{\showpaste}
\tab{5}\spadcommand{d.remainder\free{d }}
\end{paste}\end{patch}
```

Primes and Factorization



⇐ "Integers" (IntegerPage) 3.80 on page 1029
⇐ "Integer" (IntegerXmpPage) 3.55 on page 696
⇒ "Factored" (FactoredXmpPage) 3.43 on page 516
⇒ "Complex" (ComplexXmpPage) 3.16 on page 254

— **int.ht** —

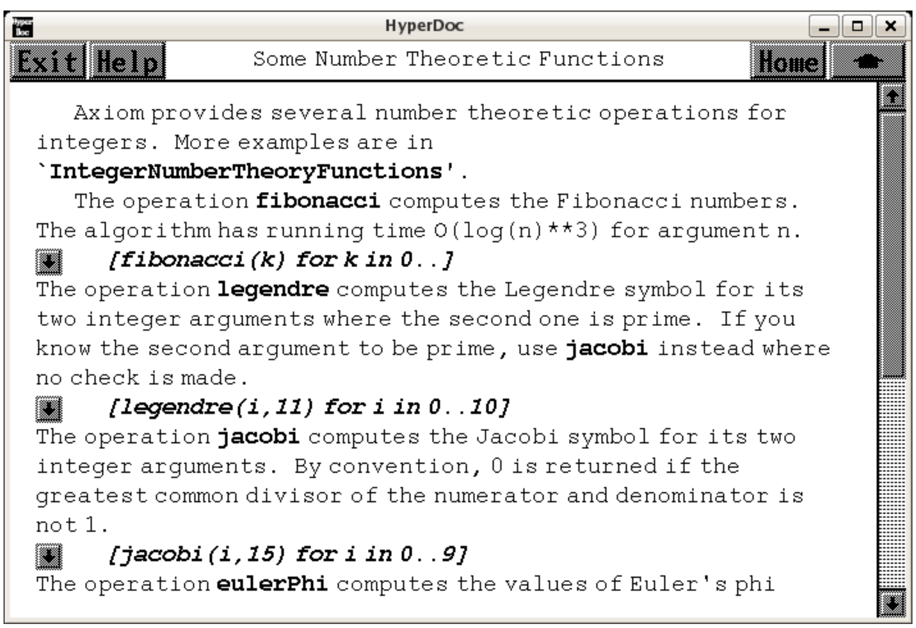
```
\begin{page}{ugxIntegerPrimesPage}{Primes and Factorization}
\beginscroll

\labelSpace{3pc}
\xtc{
Use the operation \spadfunFrom{factor}{Integer} to factor integers.
It returns an object of type \spadtype{Factored Integer}.
See \downlink{`Factored'}{FactoredXmpPage}\ignore{Factored}
for a discussion of the
manipulation of factored objects.
}{
\spadpaste{factor 102400}
}

\xtc{
The operation \spadfunFrom{prime?}{Integer} returns \spad{true} or
\spad{false} depending on whether its argument is a prime.
}{
\spadpaste{prime? 7}
}
\xtc{
}{
\spadpaste{prime? 8}
}
\xtc{
The operation \spadfunFrom{nextPrime}{IntegerPrimesPackage} returns the
least prime number greater than its argument.
}{
\spadpaste{nextPrime 100}
}
\xtc{
The operation
\spadfunFrom{prevPrime}{IntegerPrimesPackage} returns the greatest prime
number less than its argument.
}{
\spadpaste{prevPrime 100}
}
\xtc{
To compute all primes between two integers (inclusively), use the
operation \spadfunFrom{primes}{IntegerPrimesPackage}.
}{
\spadpaste{primes(100,175)}
}
\xtc{
You might sometimes want to see the factorization of an integer
when it is considered a {\it Gaussian integer}.
See \downlink{`Complex'}{ComplexXmpPage}\ignore{Complex} for more details.
}{
\spadpaste{factor(2 :: Complex Integer)}
}
```

```
\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxIntegerPrimesPagePatch1}
\begin{paste}{ugxIntegerPrimesPageFull1}{ugxIntegerPrimesPageEmpty1}
\pastebutton{ugxIntegerPrimesPageFull1}{\hidepaste}
\tab{5}\spadcommand{factor 102400}
\indentrel{3}\begin{verbatim}
         12 2
   (1)  2  5
                                 Type: Factored Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerPrimesPageEmpty1}
\begin{paste}{ugxIntegerPrimesPageEmpty1}{ugxIntegerPrimesPagePatch1}
\pastebutton{ugxIntegerPrimesPageEmpty1}{\showpaste}
\tab{5}\spadcommand{factor 102400}
\end{paste}\end{patch}

\begin{patch}{ugxIntegerPrimesPagePatch2}
\begin{paste}{ugxIntegerPrimesPageFull2}{ugxIntegerPrimesPageEmpty2}
\pastebutton{ugxIntegerPrimesPageFull2}{\hidepaste}
\tab{5}\spadcommand{prime? 7}
\indentrel{3}\begin{verbatim}
   (2)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerPrimesPageEmpty2}
\begin{paste}{ugxIntegerPrimesPageEmpty2}{ugxIntegerPrimesPagePatch2}
\pastebutton{ugxIntegerPrimesPageEmpty2}{\showpaste}
\tab{5}\spadcommand{prime? 7}
\end{paste}\end{patch}

\begin{patch}{ugxIntegerPrimesPagePatch3}
\begin{paste}{ugxIntegerPrimesPageFull3}{ugxIntegerPrimesPageEmpty3}
\pastebutton{ugxIntegerPrimesPageFull3}{\hidepaste}
\tab{5}\spadcommand{prime? 8}
\indentrel{3}\begin{verbatim}
   (3)  false
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerPrimesPageEmpty3}
\begin{paste}{ugxIntegerPrimesPageEmpty3}{ugxIntegerPrimesPagePatch3}
```

```
\pastebutton{ugxIntegerPrimesPageEmpty3}{\showpaste}
\tab{5}\spadcommand{prime? 8}
\end{paste}\end{patch}

\begin{patch}{ugxIntegerPrimesPagePatch4}
\begin{paste}{ugxIntegerPrimesPageFull4}{ugxIntegerPrimesPageEmpty4}
\pastebutton{ugxIntegerPrimesPageFull4}{\hidepaste}
\tab{5}\spadcommand{nextPrime 100}
\indentrel{3}\begin{verbatim}
   (4)  101
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerPrimesPageEmpty4}
\begin{paste}{ugxIntegerPrimesPageEmpty4}{ugxIntegerPrimesPagePatch4}
\pastebutton{ugxIntegerPrimesPageEmpty4}{\showpaste}
\tab{5}\spadcommand{nextPrime 100}
\end{paste}\end{patch}

\begin{patch}{ugxIntegerPrimesPagePatch5}
\begin{paste}{ugxIntegerPrimesPageFull5}{ugxIntegerPrimesPageEmpty5}
\pastebutton{ugxIntegerPrimesPageFull5}{\hidepaste}
\tab{5}\spadcommand{prevPrime 100}
\indentrel{3}\begin{verbatim}
   (5)  97
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerPrimesPageEmpty5}
\begin{paste}{ugxIntegerPrimesPageEmpty5}{ugxIntegerPrimesPagePatch5}
\pastebutton{ugxIntegerPrimesPageEmpty5}{\showpaste}
\tab{5}\spadcommand{prevPrime 100}
\end{paste}\end{patch}

\begin{patch}{ugxIntegerPrimesPagePatch6}
\begin{paste}{ugxIntegerPrimesPageFull6}{ugxIntegerPrimesPageEmpty6}
\pastebutton{ugxIntegerPrimesPageFull6}{\hidepaste}
\tab{5}\spadcommand{primes(100,175)}
\indentrel{3}\begin{verbatim}
   (6)
   [173, 167, 163, 157, 151, 149, 139, 137, 131, 127,
    113, 109, 107, 103, 101]
                                     Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerPrimesPageEmpty6}
\begin{paste}{ugxIntegerPrimesPageEmpty6}{ugxIntegerPrimesPagePatch6}
```

```
\pastebutton{ugxIntegerPrimesPageEmpty6}{\showpaste}
\tab{5}\spadcommand{primes(100,175)}
\end{paste}\end{patch}

\begin{patch}{ugxIntegerPrimesPagePatch7}
\begin{paste}{ugxIntegerPrimesPageFull7}{ugxIntegerPrimesPageEmpty7}
\pastebutton{ugxIntegerPrimesPageFull7}{\hidepaste}
\tab{5}\spadcommand{factor(2 :: Complex Integer)}
\indentrel{3}\begin{verbatim}
                       2
   (7)  - %i (1 + %i)
                           Type: Factored Complex Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerPrimesPageEmpty7}
\begin{paste}{ugxIntegerPrimesPageEmpty7}{ugxIntegerPrimesPagePatch7}
\pastebutton{ugxIntegerPrimesPageEmpty7}{\showpaste}
\tab{5}\spadcommand{factor(2 :: Complex Integer)}
\end{paste}\end{patch}
```

Some Number Theoretic Functions



⇐ "Integer" (IntegerXmpPage) 3.55 on page 696

⇒ "IntegerNumberTheoryFunctions" (IntNumberTheoryFnsXmpPage) 3.56 on page 722

— **int.ht** —

```
\begin{page}{ugxIntegerNTPage}{Some Number Theoretic Functions}
\beginscroll

Axiom provides several number theoretic operations for integers.
More examples are in \downlink{`IntegerNumberTheoryFunctions'}
{IntNumberTheoryFnsXmpPage}\ignore{IntegerNumberTheoryFunctions}.

\labelSpace{1pc}
\xtc{
The operation \spadfunFrom{fibonacci}{IntegerNumberTheoryFunctions}
computes the Fibonacci numbers.
The algorithm has running time
\texht{$O\,(\log^3(n))$}{O(\spad{log(n)**3})} for argument \spad{n}.
}{
\spadpaste{[fibonacci(k) for k in 0..]}
}
\xtc{
The operation \spadfunFrom{legendre}{IntegerNumberTheoryFunctions}
computes the Legendre symbol for its two integer arguments where the
second one is prime.
If you know the second argument to be prime, use
\spadfunFrom{jacobi}{IntegerNumberTheoryFunctions} instead where no check
is made.
}{
\spadpaste{[legendre(i,11) for i in 0..10]}
}
\xtc{
The operation \spadfunFrom{jacobi}{IntegerNumberTheoryFunctions} computes
the Jacobi symbol for its two integer arguments.
By convention, \spad{0} is returned if the greatest common divisor of the
numerator and denominator is not \spad{1}.
}{
\spadpaste{[jacobi(i,15) for i in 0..9]}
}
\xtc{
The operation \spadfunFrom{eulerPhi}{IntegerNumberTheoryFunctions} computes
the values of Euler's \texht{$\phi$}{phi}-function where
\texht{$\phi(n)$}{\spad{phi(n)}}
equals the number of positive integers
less than or equal to \spad{n} that are relatively prime to
the positive integer \spad{n}.
}{
\spadpaste{[eulerPhi i for i in 1..]}
}
\xtc{
The operation \spadfunFrom{moebiusMu}{IntegerNumberTheoryFunctions}
computes the \texht{M\"{o}bius $\mu$}{Moebius mu} function.
```

```
}{
\spadpaste{[moebiusMu i for i in 1..]}
}

\xtc{
Although they have somewhat limited utility, Axiom provides
Roman numerals.
}{
\spadpaste{a := roman(78) \bound{a}}
}
\xtc{
}{
\spadpaste{b := roman(87) \bound{b}}
}
\xtc{
}{
\spadpaste{a + b \free{a}\free{b}}
}
\xtc{
}{
\spadpaste{a * b \free{a}\free{b}}
}
\xtc{
}{
\spadpaste{b rem a \free{a}\free{b}}
}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxIntegerNTPagePatch1}
\begin{paste}{ugxIntegerNTPageFull1}{ugxIntegerNTPageEmpty1}
\pastebutton{ugxIntegerNTPageFull1}{\hidepaste}
\tab{5}\spadcommand{[fibonacci(k) for k in 0..]}
\indentrel{3}\begin{verbatim}
   (1)  [0,1,1,2,3,5,8,13,21,34,...]
                                   Type: Stream Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerNTPageEmpty1}
\begin{paste}{ugxIntegerNTPageEmpty1}{ugxIntegerNTPagePatch1}
\pastebutton{ugxIntegerNTPageEmpty1}{\showpaste}
\tab{5}\spadcommand{[fibonacci(k) for k in 0..]}
\end{paste}\end{patch}

\begin{patch}{ugxIntegerNTPagePatch2}
\begin{paste}{ugxIntegerNTPageFull2}{ugxIntegerNTPageEmpty2}
\pastebutton{ugxIntegerNTPageFull2}{\hidepaste}
\tab{5}\spadcommand{[legendre(i,11) for i in 0..10]}
```

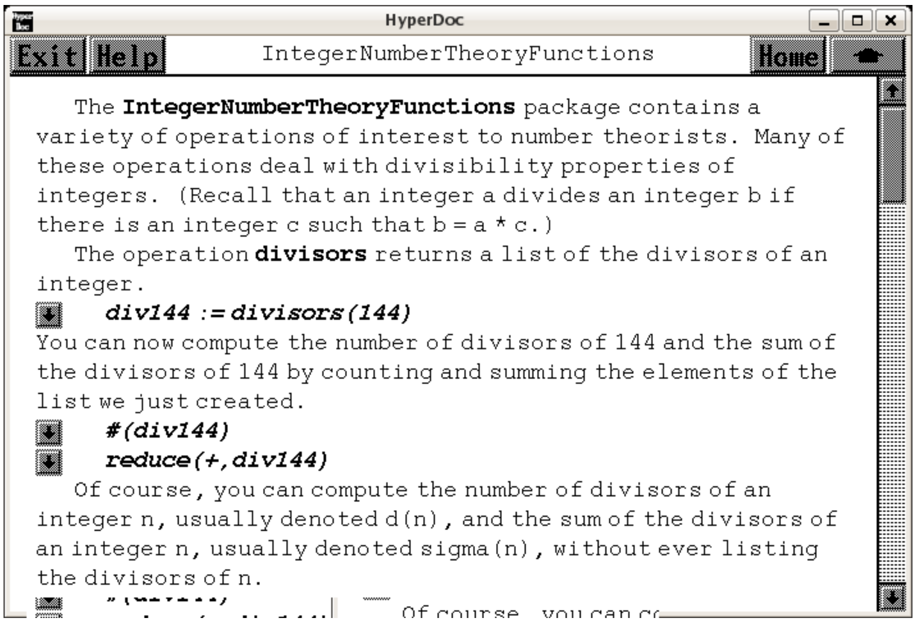
```
\indentrel{3}\begin{verbatim}
   (2)  [0,1,- 1,1,1,1,- 1,- 1,- 1,1,- 1]
                                     Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerNTPageEmpty2}
\begin{paste}{ugxIntegerNTPageEmpty2}{ugxIntegerNTPagePatch2}
\pastebutton{ugxIntegerNTPageEmpty2}{\showpaste}
\tab{5}\spadcommand{[legendre(i,11) for i in 0..10]}
\end{paste}\end{patch}

\begin{patch}{ugxIntegerNTPagePatch3}
\begin{paste}{ugxIntegerNTPageFull3}{ugxIntegerNTPageEmpty3}
\pastebutton{ugxIntegerNTPageFull3}{\hidepaste}
\tab{5}\spadcommand{[jacobi(i,15) for i in 0..9]}
\indentrel{3}\begin{verbatim}
   (3)  [0,1,1,0,1,0,0,- 1,1,0]
                                     Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerNTPageEmpty3}
\begin{paste}{ugxIntegerNTPageEmpty3}{ugxIntegerNTPagePatch3}
\pastebutton{ugxIntegerNTPageEmpty3}{\showpaste}
\tab{5}\spadcommand{[jacobi(i,15) for i in 0..9]}
\end{paste}\end{patch}

\begin{patch}{ugxIntegerNTPagePatch4}
\begin{paste}{ugxIntegerNTPageFull4}{ugxIntegerNTPageEmpty4}
\pastebutton{ugxIntegerNTPageFull4}{\hidepaste}
\tab{5}\spadcommand{[eulerPhi i for i in 1..]}
\indentrel{3}\begin{verbatim}
   (4)  [1,1,2,2,4,2,6,4,6,4,...]
                                   Type: Stream Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerNTPageEmpty4}
\begin{paste}{ugxIntegerNTPageEmpty4}{ugxIntegerNTPagePatch4}
\pastebutton{ugxIntegerNTPageEmpty4}{\showpaste}
\tab{5}\spadcommand{[eulerPhi i for i in 1..]}
\end{paste}\end{patch}

\begin{patch}{ugxIntegerNTPagePatch5}
\begin{paste}{ugxIntegerNTPageFull5}{ugxIntegerNTPageEmpty5}
\pastebutton{ugxIntegerNTPageFull5}{\hidepaste}
\tab{5}\spadcommand{[moebiusMu i for i in 1..]}
\indentrel{3}\begin{verbatim}
   (5)  [1,- 1,- 1,0,- 1,1,- 1,0,0,1,...]
```

```
                                    Type: Stream Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerNTPageEmpty5}
\begin{paste}{ugxIntegerNTPageEmpty5}{ugxIntegerNTPagePatch5}
\pastebutton{ugxIntegerNTPageEmpty5}{\showpaste}
\tab{5}\spadcommand{[moebiusMu i for i in 1..]}
\end{paste}\end{patch}

\begin{patch}{ugxIntegerNTPagePatch6}
\begin{paste}{ugxIntegerNTPageFull6}{ugxIntegerNTPageEmpty6}
\pastebutton{ugxIntegerNTPageFull6}{\hidepaste}
\tab{5}\spadcommand{a := roman(78)\bound{a }}
\indentrel{3}\begin{verbatim}
   (6)  LXXVIII
                                     Type: RomanNumeral
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerNTPageEmpty6}
\begin{paste}{ugxIntegerNTPageEmpty6}{ugxIntegerNTPagePatch6}
\pastebutton{ugxIntegerNTPageEmpty6}{\showpaste}
\tab{5}\spadcommand{a := roman(78)\bound{a }}
\end{paste}\end{patch}

\begin{patch}{ugxIntegerNTPagePatch7}
\begin{paste}{ugxIntegerNTPageFull7}{ugxIntegerNTPageEmpty7}
\pastebutton{ugxIntegerNTPageFull7}{\hidepaste}
\tab{5}\spadcommand{b := roman(87)\bound{b }}
\indentrel{3}\begin{verbatim}
   (7)  LXXXVII
                                     Type: RomanNumeral
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerNTPageEmpty7}
\begin{paste}{ugxIntegerNTPageEmpty7}{ugxIntegerNTPagePatch7}
\pastebutton{ugxIntegerNTPageEmpty7}{\showpaste}
\tab{5}\spadcommand{b := roman(87)\bound{b }}
\end{paste}\end{patch}

\begin{patch}{ugxIntegerNTPagePatch8}
\begin{paste}{ugxIntegerNTPageFull8}{ugxIntegerNTPageEmpty8}
\pastebutton{ugxIntegerNTPageFull8}{\hidepaste}
\tab{5}\spadcommand{a + b\free{a }\free{b }}
\indentrel{3}\begin{verbatim}
   (8)  CLXV
                                     Type: RomanNumeral
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerNTPageEmpty8}
\begin{paste}{ugxIntegerNTPageEmpty8}{ugxIntegerNTPagePatch8}
\pastebutton{ugxIntegerNTPageEmpty8}{\showpaste}
\tab{5}\spadcommand{a + b\free{a }\free{b }}
\end{paste}\end{patch}

\begin{patch}{ugxIntegerNTPagePatch9}
\begin{paste}{ugxIntegerNTPageFull9}{ugxIntegerNTPageEmpty9}
\pastebutton{ugxIntegerNTPageFull9}{\hidepaste}
\tab{5}\spadcommand{a * b\free{a }\free{b }}
\indentrel{3}\begin{verbatim}
   (9)  MMMMMMDCCLXXXVI
                                    Type: RomanNumeral
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerNTPageEmpty9}
\begin{paste}{ugxIntegerNTPageEmpty9}{ugxIntegerNTPagePatch9}
\pastebutton{ugxIntegerNTPageEmpty9}{\showpaste}
\tab{5}\spadcommand{a * b\free{a }\free{b }}
\end{paste}\end{patch}

\begin{patch}{ugxIntegerNTPagePatch10}
\begin{paste}{ugxIntegerNTPageFull10}{ugxIntegerNTPageEmpty10}
\pastebutton{ugxIntegerNTPageFull10}{\hidepaste}
\tab{5}\spadcommand{b rem a\free{a }\free{b }}
\indentrel{3}\begin{verbatim}
   (10)  IX
                                    Type: RomanNumeral
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxIntegerNTPageEmpty10}
\begin{paste}{ugxIntegerNTPageEmpty10}{ugxIntegerNTPagePatch10}
\pastebutton{ugxIntegerNTPageEmpty10}{\showpaste}
\tab{5}\spadcommand{b rem a\free{a }\free{b }}
\end{paste}\end{patch}
```

3.56 intheory.ht

IntegerNumberTheoryFunctions



⇐ "Integers" (IntegerPage) 3.80 on page 1029
⇐ "Integer" (IntegerXmpPage) 3.55 on page 696
⇐ "Some Number Theoretic Functions" (ugxIntegerNTPage) 3.55 on page 716

— **intheory.ht** —

```
\begin{page}{IntNumberTheoryFnsXmpPage}
{IntegerNumberTheoryFunctions}
\beginscroll

The \spadtype{IntegerNumberTheoryFunctions} package contains a variety
of operations of interest to number theorists.  Many of these
operations deal with divisibility properties of integers.  (Recall
that an integer \spad{a} divides an integer \spad{b} if there is an
integer \spad{c} such that \spad{b = a * c}.)

\xtc{
The operation \spadfunFrom{divisors}{IntegerNumberTheoryFunctions}
returns a list of the divisors of an integer.
}{
\spadpaste{div144 := divisors(144) \bound{div144}}
}
\xtc{
You can now compute the number of divisors of \spad{144} and the sum of
```

```
the divisors of \spad{144} by counting and summing the elements of the
list we just created.
}{
\spadpaste{\#(div144) \free{div144}}
}
\xtc{
}{
\spadpaste{reduce(+,div144) \free{div144}}
}

Of course, you can compute the number of divisors of an integer \spad{n},
usually denoted \spad{d(n)}, and the sum of the divisors of an integer
\spad{n}, usually denoted \spad{\texht{$\sigma$}{sigma}(n)},
without ever listing the divisors of \spad{n}.

\xtc{
In Axiom, you can simply call the operations
\spadfunFrom{numberOfDivisors}{IntegerNumberTheoryFunctions} and
\spadfunFrom{sumOfDivisors}{IntegerNumberTheoryFunctions}.
}{
\spadpaste{numberOfDivisors(144)}
}
\xtc{
}{
\spadpaste{sumOfDivisors(144)}
}

The key is that \spad{d(n)} and
\spad{\texht{$\sigma$}{sigma}(n)}
are ``multiplicative functions.''
This means that when \spad{n} and \spad{m} are relatively prime,
that is, when
\spad{n} and \spad{m} have no prime factor in common, then
\spad{d(nm) = d(n) d(m)} and
\spad{\texht{$\sigma$}{sigma}(nm) = \texht{$\sigma$}{sigma}(n)
\texht{$\sigma$}{sigma}(m)}.
Note that these functions are trivial to compute when \spad{n} is a prime
power and are computed for general \spad{n} from the prime factorization
of \spad{n}.
Other examples of multiplicative functions are
\spad{\texht{$\sigma_k$}{sigma_k}(n)}, the sum of the \eth{\spad{k}}
powers of the divisors of \spad{n} and
\texht{$\varphi(n)$}{\spad{phi(n)}}, the
number of integers between 1 and \spad{n} which are prime to \spad{n}.
The corresponding Axiom operations are called
\spadfunFrom{sumOfKthPowerDivisors}{IntegerNumberTheoryFunctions} and
\spadfunFrom{eulerPhi}{IntegerNumberTheoryFunctions}.

An interesting function is \spad{\texht{$\mu$}{mu}(n)},
the \texht{M\"{o}bius $\mu$}{Moebius mu} function, defined
```

```
as follows:
\spad{\texht{$\mu$}{mu}(1) = 1}, \spad{\texht{$\mu$}{mu}(n) = 0},
when \spad{n} is divisible by a
square, and
\spad{\texht{$\mu = {(-1)}^k$}{mu(n) = (-1) ** k}}, when \spad{n}
is the product of \spad{k} distinct primes.
The corresponding Axiom operation is
\spadfunFrom{moebiusMu}{IntegerNumberTheoryFunctions}.
This function occurs in the following theorem:

\noindent
{\bf Theorem} (\texht{M\"{o}bius}{Moebius} Inversion Formula): \newline
%\texht{\begin{quotation}\noindent}{\newline\indent{5}}
Let \spad{f(n)} be a function on the positive integers and let \spad{F(n)}
be defined by
\texht{\narrowDisplay{F(n) = \sum_{d \mid n} f(n)}}{\spad{F(n) =}
sum of \spad{f(n)} over \spad{d | n}}
where the sum is taken over the positive divisors of \spad{n}.
Then the values of \spad{f(n)} can be recovered from the values of
\spad{F(n)}:
\texht{\narrowDisplay{f(n) = \sum_{d \mid n} \mu(n) F({{n}\over {d}})}}
{\spad{f(n) =}
sum of \spad{mu(n) F(n/d)} over \spad{d | n},}
where again the sum is taken over the positive divisors of \spad{n}.

\xtc{
When \spad{f(n) = 1}, then \spad{F(n) = d(n)}.
Thus, if you sum \spad{\texht{$\mu$}{mu}(d) \texht{$\cdot$}{*} d(n/d)}
over the positive divisors
\spad{d} of \spad{n}, you should always get \spad{1}.
}{
\spadpaste{f1(n) == reduce(+,[moebiusMu(d) * numberOfDivisors(quo(n,d))
for d in divisors(n)]) \bound{f1}}
}
\xtc{
}{
\spadpaste{f1(200) \free{f1}}
}
\xtc{
}{
\spadpaste{f1(846) \free{f1}}
}
\xtc{
Similarly,
when \spad{f(n) = n}, then \spad{F(n) = \texht{$\sigma$}{sigma}(n)}.
Thus, if you sum \spad{\texht{$\mu$}{mu}(d) \texht{$\cdot$}{*}
\texht{$\sigma$}{sigma}(n/d)} over the positive divisors
\spad{d} of \spad{n}, you should always get \spad{n}.
}{
\spadpaste{f2(n) == reduce(+,[moebiusMu(d) * sumOfDivisors(quo(n,d))
```

```
for d in divisors(n)]) \bound{f2}}
}
\xtc{
}{
\spadpaste{f2(200) \free{f2}}
}
\xtc{
}{
\spadpaste{f2(846) \free{f2}}
}


The Fibonacci numbers are defined by \spad{F(1) = F(2) = 1} and
\spad{F(n) = F(n-1) + F(n-2)} for \spad{n = 3,4, ...}.
\xtc{
The operation \spadfunFrom{fibonacci}{IntegerNumberTheoryFunctions}
computes the \eth{\spad{n}} Fibonacci number.
}{
\spadpaste{fibonacci(25)}
}
\xtc{
}{
\spadpaste{[fibonacci(n) for n in 1..15]}
}
\xtc{
Fibonacci numbers can also be expressed as sums of binomial coefficients.
}{
\spadpaste{fib(n) == reduce(+,[binomial(n-1-k,k)
for k in 0..quo(n-1,2)]) \bound{fib}}
}
\xtc{
}{
\spadpaste{fib(25) \free{fib}}
}
\xtc{
}{
\spadpaste{[fib(n) for n in 1..15] \free{fib}}
}

Quadratic symbols can be computed with the operations
\spadfunFrom{legendre}{IntegerNumberTheoryFunctions} and
\spadfunFrom{jacobi}{IntegerNumberTheoryFunctions}.
The Legendre symbol
\texht{$\left({a \over p}\right)$}{\spad{(a/p)}}
is defined for integers \spad{a} and
\spad{p} with \spad{p} an odd prime number.
By definition, \texht{$\left({a \over p}\right)$}{\spad{(a/p) = +1}},
when \spad{a} is a square \spad{(mod p)},
\texht{$\left({a \over p}\right)$}{\spad{(a/p) = -1}},
when \spad{a} is not a square \spad{(mod p)}, and
```

```
\texht{$\left({a \over p}\right)$}{\spad{(a/p) = 0}},
when \spad{a} is divisible by \spad{p}.
\xtc{
You compute \texht{$\left({a \over p}\right)$}{\spad{(a/p)}}
via the command \spad{legendre(a,p)}.
}{
\spadpaste{legendre(3,5)}
}
\xtc{
}{
\spadpaste{legendre(23,691)}
}
The Jacobi symbol \texht{$\left({a \over n}\right)$}{\spad{(a/n)}}
is the usual extension of the Legendre
symbol, where \spad{n} is an arbitrary integer.
The most important property of the Jacobi symbol is the following:
if \spad{K} is a quadratic field with discriminant \spad{d} and quadratic
character \texht{$\chi$}{\spad{chi}},
then \texht{$\chi$}{\spad{chi}}\spad{(n) = (d/n)}.
Thus, you can use the Jacobi symbol
to compute, say, the class numbers of
imaginary quadratic fields from a standard class number formula.
\xtc{
This function computes the class number of the imaginary
quadratic field with discriminant \spad{d}.
}{
\spadpaste{h(d) == quo(reduce(+, [jacobi(d,k) for k in 1..quo(-d, 2)]),
2 - jacobi(d,2)) \bound{h}}
}
\xtc{
}{
\spadpaste{h(-163) \free{h}}
}
\xtc{
}{
\spadpaste{h(-499) \free{h}}
}
\xtc{
}{
\spadpaste{h(-1832) \free{h}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{IntNumberTheoryFnsXmpPagePatch1}
\begin{paste}{IntNumberTheoryFnsXmpPageFull1}{IntNumberTheoryFnsXmpPageEmpty1}
\pastebutton{IntNumberTheoryFnsXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{div144 := divisors(144)\bound{div144 }}
```

```
\indentrel{3}\begin{verbatim}
   (1)  [1,2,3,4,6,8,9,12,16,18,24,36,48,72,144]
                                    Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntNumberTheoryFnsXmpPageEmpty1}
\begin{paste}{IntNumberTheoryFnsXmpPageEmpty1}{IntNumberTheoryFnsXmpPagePatch1}
\pastebutton{IntNumberTheoryFnsXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{div144 := divisors(144)\bound{div144 }}
\end{paste}\end{patch}

\begin{patch}{IntNumberTheoryFnsXmpPagePatch2}
\begin{paste}{IntNumberTheoryFnsXmpPageFull2}{IntNumberTheoryFnsXmpPageEmpty2}
\pastebutton{IntNumberTheoryFnsXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{\#(div144)\free{div144 }}
\indentrel{3}\begin{verbatim}
   (2)  15
                                 Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntNumberTheoryFnsXmpPageEmpty2}
\begin{paste}{IntNumberTheoryFnsXmpPageEmpty2}{IntNumberTheoryFnsXmpPagePatch2}
\pastebutton{IntNumberTheoryFnsXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{\#(div144)\free{div144 }}
\end{paste}\end{patch}

\begin{patch}{IntNumberTheoryFnsXmpPagePatch3}
\begin{paste}{IntNumberTheoryFnsXmpPageFull3}{IntNumberTheoryFnsXmpPageEmpty3}
\pastebutton{IntNumberTheoryFnsXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{reduce(+,div144)\free{div144 }}
\indentrel{3}\begin{verbatim}
   (3)  403
                                 Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntNumberTheoryFnsXmpPageEmpty3}
\begin{paste}{IntNumberTheoryFnsXmpPageEmpty3}{IntNumberTheoryFnsXmpPagePatch3}
\pastebutton{IntNumberTheoryFnsXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{reduce(+,div144)\free{div144 }}
\end{paste}\end{patch}

\begin{patch}{IntNumberTheoryFnsXmpPagePatch4}
\begin{paste}{IntNumberTheoryFnsXmpPageFull4}{IntNumberTheoryFnsXmpPageEmpty4}
\pastebutton{IntNumberTheoryFnsXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{numberOfDivisors(144)}
\indentrel{3}\begin{verbatim}
   (4)  15
```

```
                                   Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntNumberTheoryFnsXmpPageEmpty4}
\begin{paste}{IntNumberTheoryFnsXmpPageEmpty4}{IntNumberTheoryFnsXmpPagePatch4}
\pastebutton{IntNumberTheoryFnsXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{numberOfDivisors(144)}
\end{paste}\end{patch}

\begin{patch}{IntNumberTheoryFnsXmpPagePatch5}
\begin{paste}{IntNumberTheoryFnsXmpPageFull5}{IntNumberTheoryFnsXmpPageEmpty5}
\pastebutton{IntNumberTheoryFnsXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{sumOfDivisors(144)}
\indentrel{3}\begin{verbatim}
   (5)  403
                                   Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntNumberTheoryFnsXmpPageEmpty5}
\begin{paste}{IntNumberTheoryFnsXmpPageEmpty5}{IntNumberTheoryFnsXmpPagePatch5}
\pastebutton{IntNumberTheoryFnsXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{sumOfDivisors(144)}
\end{paste}\end{patch}

\begin{patch}{IntNumberTheoryFnsXmpPagePatch6}
\begin{paste}{IntNumberTheoryFnsXmpPageFull6}{IntNumberTheoryFnsXmpPageEmpty6}
\pastebutton{IntNumberTheoryFnsXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{f1(n) == reduce(+,[moebiusMu(d) * numberOfDivisors(quo(n,d)) for d in diviso
\indentrel{3}\begin{verbatim}
                                              Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntNumberTheoryFnsXmpPageEmpty6}
\begin{paste}{IntNumberTheoryFnsXmpPageEmpty6}{IntNumberTheoryFnsXmpPagePatch6}
\pastebutton{IntNumberTheoryFnsXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{f1(n) == reduce(+,[moebiusMu(d) * numberOfDivisors(quo(n,d)) for d in diviso
\end{paste}\end{patch}

\begin{patch}{IntNumberTheoryFnsXmpPagePatch7}
\begin{paste}{IntNumberTheoryFnsXmpPageFull7}{IntNumberTheoryFnsXmpPageEmpty7}
\pastebutton{IntNumberTheoryFnsXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{f1(200)\free{f1 }}
\indentrel{3}\begin{verbatim}
   (7)  1
                                   Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{IntNumberTheoryFnsXmpPageEmpty7}
\begin{paste}{IntNumberTheoryFnsXmpPageEmpty7}{IntNumberTheoryFnsXmpPagePatch7}
\pastebutton{IntNumberTheoryFnsXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{f1(200)\free{f1 }}
\end{paste}\end{patch}

\begin{patch}{IntNumberTheoryFnsXmpPagePatch8}
\begin{paste}{IntNumberTheoryFnsXmpPageFull8}{IntNumberTheoryFnsXmpPageEmpty8}
\pastebutton{IntNumberTheoryFnsXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{f1(846)\free{f1 }}
\indentrel{3}\begin{verbatim}
   (8)  1
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntNumberTheoryFnsXmpPageEmpty8}
\begin{paste}{IntNumberTheoryFnsXmpPageEmpty8}{IntNumberTheoryFnsXmpPagePatch8}
\pastebutton{IntNumberTheoryFnsXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{f1(846)\free{f1 }}
\end{paste}\end{patch}

\begin{patch}{IntNumberTheoryFnsXmpPagePatch9}
\begin{paste}{IntNumberTheoryFnsXmpPageFull9}{IntNumberTheoryFnsXmpPageEmpty9}
\pastebutton{IntNumberTheoryFnsXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{f2(n) == reduce(+,[moebiusMu(d) * sumOfDivisors(quo(n,d)) for d in divisors(n)])\bound{
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntNumberTheoryFnsXmpPageEmpty9}
\begin{paste}{IntNumberTheoryFnsXmpPageEmpty9}{IntNumberTheoryFnsXmpPagePatch9}
\pastebutton{IntNumberTheoryFnsXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{f2(n) == reduce(+,[moebiusMu(d) * sumOfDivisors(quo(n,d)) for d in divisors(n)])\bound{
\end{paste}\end{patch}

\begin{patch}{IntNumberTheoryFnsXmpPagePatch10}
\begin{paste}{IntNumberTheoryFnsXmpPageFull10}{IntNumberTheoryFnsXmpPageEmpty10}
\pastebutton{IntNumberTheoryFnsXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{f2(200)\free{f2 }}
\indentrel{3}\begin{verbatim}
   (10)  200
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntNumberTheoryFnsXmpPageEmpty10}
\begin{paste}{IntNumberTheoryFnsXmpPageEmpty10}{IntNumberTheoryFnsXmpPagePatch10}
```

```
\pastebutton{IntNumberTheoryFnsXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{f2(200)\free{f2 }}
\end{paste}\end{patch}

\begin{patch}{IntNumberTheoryFnsXmpPagePatch11}
\begin{paste}{IntNumberTheoryFnsXmpPageFull11}{IntNumberTheoryFnsXmpPageEmpty11}
\pastebutton{IntNumberTheoryFnsXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{f2(846)\free{f2 }}
\indentrel{3}\begin{verbatim}
   (11)  846
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntNumberTheoryFnsXmpPageEmpty11}
\begin{paste}{IntNumberTheoryFnsXmpPageEmpty11}{IntNumberTheoryFnsXmpPagePatch11}
\pastebutton{IntNumberTheoryFnsXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{f2(846)\free{f2 }}
\end{paste}\end{patch}

\begin{patch}{IntNumberTheoryFnsXmpPagePatch12}
\begin{paste}{IntNumberTheoryFnsXmpPageFull12}{IntNumberTheoryFnsXmpPageEmpty12}
\pastebutton{IntNumberTheoryFnsXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{fibonacci(25)}
\indentrel{3}\begin{verbatim}
   (12)  75025
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntNumberTheoryFnsXmpPageEmpty12}
\begin{paste}{IntNumberTheoryFnsXmpPageEmpty12}{IntNumberTheoryFnsXmpPagePatch12}
\pastebutton{IntNumberTheoryFnsXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{fibonacci(25)}
\end{paste}\end{patch}

\begin{patch}{IntNumberTheoryFnsXmpPagePatch13}
\begin{paste}{IntNumberTheoryFnsXmpPageFull13}{IntNumberTheoryFnsXmpPageEmpty13}
\pastebutton{IntNumberTheoryFnsXmpPageFull13}{\hidepaste}
\tab{5}\spadcommand{[fibonacci(n) for n in 1..15]}
\indentrel{3}\begin{verbatim}
   (13)  [1,1,2,3,5,8,13,21,34,55,89,144,233,377,610]
                                     Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntNumberTheoryFnsXmpPageEmpty13}
\begin{paste}{IntNumberTheoryFnsXmpPageEmpty13}{IntNumberTheoryFnsXmpPagePatch13}
\pastebutton{IntNumberTheoryFnsXmpPageEmpty13}{\showpaste}
\tab{5}\spadcommand{[fibonacci(n) for n in 1..15]}
```

```
\end{paste}\end{patch}

\begin{patch}{IntNumberTheoryFnsXmpPagePatch14}
\begin{paste}{IntNumberTheoryFnsXmpPageFull14}{IntNumberTheoryFnsXmpPageEmpty14}
\pastebutton{IntNumberTheoryFnsXmpPageFull14}{\hidepaste}
\tab{5}\spadcommand{fib(n) == reduce(+,[binomial(n-1-k,k) for k in 0..quo(n-1,2)])\bound{fib }}
\indentrel{3}\begin{verbatim}
                                              Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntNumberTheoryFnsXmpPageEmpty14}
\begin{paste}{IntNumberTheoryFnsXmpPageEmpty14}{IntNumberTheoryFnsXmpPagePatch14}
\pastebutton{IntNumberTheoryFnsXmpPageEmpty14}{\showpaste}
\tab{5}\spadcommand{fib(n) == reduce(+,[binomial(n-1-k,k) for k in 0..quo(n-1,2)])\bound{fib }}
\end{paste}\end{patch}

\begin{patch}{IntNumberTheoryFnsXmpPagePatch15}
\begin{paste}{IntNumberTheoryFnsXmpPageFull15}{IntNumberTheoryFnsXmpPageEmpty15}
\pastebutton{IntNumberTheoryFnsXmpPageFull15}{\hidepaste}
\tab{5}\spadcommand{fib(25)\free{fib }}
\indentrel{3}\begin{verbatim}
   (15)  75025
                                   Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntNumberTheoryFnsXmpPageEmpty15}
\begin{paste}{IntNumberTheoryFnsXmpPageEmpty15}{IntNumberTheoryFnsXmpPagePatch15}
\pastebutton{IntNumberTheoryFnsXmpPageEmpty15}{\showpaste}
\tab{5}\spadcommand{fib(25)\free{fib }}
\end{paste}\end{patch}

\begin{patch}{IntNumberTheoryFnsXmpPagePatch16}
\begin{paste}{IntNumberTheoryFnsXmpPageFull16}{IntNumberTheoryFnsXmpPageEmpty16}
\pastebutton{IntNumberTheoryFnsXmpPageFull16}{\hidepaste}
\tab{5}\spadcommand{[fib(n) for n in 1..15]\free{fib }}
\indentrel{3}\begin{verbatim}
   (16)  [1,1,2,3,5,8,13,21,34,55,89,144,233,377,610]
                                      Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntNumberTheoryFnsXmpPageEmpty16}
\begin{paste}{IntNumberTheoryFnsXmpPageEmpty16}{IntNumberTheoryFnsXmpPagePatch16}
\pastebutton{IntNumberTheoryFnsXmpPageEmpty16}{\showpaste}
\tab{5}\spadcommand{[fib(n) for n in 1..15]\free{fib }}
\end{paste}\end{patch}

\begin{patch}{IntNumberTheoryFnsXmpPagePatch17}
```

```
\begin{paste}{IntNumberTheoryFnsXmpPageFull17}{IntNumberTheoryFnsXmpPageEmpty17}
\pastebutton{IntNumberTheoryFnsXmpPageFull17}{\hidepaste}
\tab{5}\spadcommand{legendre(3,5)}
\indentrel{3}\begin{verbatim}
   (17)  - 1
                                          Type: Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntNumberTheoryFnsXmpPageEmpty17}
\begin{paste}{IntNumberTheoryFnsXmpPageEmpty17}{IntNumberTheoryFnsXmpPagePatch17}
\pastebutton{IntNumberTheoryFnsXmpPageEmpty17}{\showpaste}
\tab{5}\spadcommand{legendre(3,5)}
\end{paste}\end{patch}

\begin{patch}{IntNumberTheoryFnsXmpPagePatch18}
\begin{paste}{IntNumberTheoryFnsXmpPageFull18}{IntNumberTheoryFnsXmpPageEmpty18}
\pastebutton{IntNumberTheoryFnsXmpPageFull18}{\hidepaste}
\tab{5}\spadcommand{legendre(23,691)}
\indentrel{3}\begin{verbatim}
   (18)  - 1
                                          Type: Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntNumberTheoryFnsXmpPageEmpty18}
\begin{paste}{IntNumberTheoryFnsXmpPageEmpty18}{IntNumberTheoryFnsXmpPagePatch18}
\pastebutton{IntNumberTheoryFnsXmpPageEmpty18}{\showpaste}
\tab{5}\spadcommand{legendre(23,691)}
\end{paste}\end{patch}

\begin{patch}{IntNumberTheoryFnsXmpPagePatch19}
\begin{paste}{IntNumberTheoryFnsXmpPageFull19}{IntNumberTheoryFnsXmpPageEmpty19}
\pastebutton{IntNumberTheoryFnsXmpPageFull19}{\hidepaste}
\tab{5}\spadcommand{h(d) == quo(reduce(+, [jacobi(d,k) for k in 1..quo(-d, 2)]), 2 - jacobi(d,2)
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntNumberTheoryFnsXmpPageEmpty19}
\begin{paste}{IntNumberTheoryFnsXmpPageEmpty19}{IntNumberTheoryFnsXmpPagePatch19}
\pastebutton{IntNumberTheoryFnsXmpPageEmpty19}{\showpaste}
\tab{5}\spadcommand{h(d) == quo(reduce(+, [jacobi(d,k) for k in 1..quo(-d, 2)]), 2 - jacobi(d,2)
\end{paste}\end{patch}

\begin{patch}{IntNumberTheoryFnsXmpPagePatch20}
\begin{paste}{IntNumberTheoryFnsXmpPageFull20}{IntNumberTheoryFnsXmpPageEmpty20}
\pastebutton{IntNumberTheoryFnsXmpPageFull20}{\hidepaste}
\tab{5}\spadcommand{h(-163)\free{h }}
```

```
\indentrel{3}\begin{verbatim}
   (20)  1
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntNumberTheoryFnsXmpPageEmpty20}
\begin{paste}{IntNumberTheoryFnsXmpPageEmpty20}{IntNumberTheoryFnsXmpPagePatch20}
\pastebutton{IntNumberTheoryFnsXmpPageEmpty20}{\showpaste}
\tab{5}\spadcommand{h(-163)\free{h }}
\end{paste}\end{patch}

\begin{patch}{IntNumberTheoryFnsXmpPagePatch21}
\begin{paste}{IntNumberTheoryFnsXmpPageFull21}{IntNumberTheoryFnsXmpPageEmpty21}
\pastebutton{IntNumberTheoryFnsXmpPageFull21}{\hidepaste}
\tab{5}\spadcommand{h(-499)\free{h }}
\indentrel{3}\begin{verbatim}
   (21)  3
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntNumberTheoryFnsXmpPageEmpty21}
\begin{paste}{IntNumberTheoryFnsXmpPageEmpty21}{IntNumberTheoryFnsXmpPagePatch21}
\pastebutton{IntNumberTheoryFnsXmpPageEmpty21}{\showpaste}
\tab{5}\spadcommand{h(-499)\free{h }}
\end{paste}\end{patch}

\begin{patch}{IntNumberTheoryFnsXmpPagePatch22}
\begin{paste}{IntNumberTheoryFnsXmpPageFull22}{IntNumberTheoryFnsXmpPageEmpty22}
\pastebutton{IntNumberTheoryFnsXmpPageFull22}{\hidepaste}
\tab{5}\spadcommand{h(-1832)\free{h }}
\indentrel{3}\begin{verbatim}
   (22)  26
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntNumberTheoryFnsXmpPageEmpty22}
\begin{paste}{IntNumberTheoryFnsXmpPageEmpty22}{IntNumberTheoryFnsXmpPagePatch22}
\pastebutton{IntNumberTheoryFnsXmpPageEmpty22}{\showpaste}
\tab{5}\spadcommand{h(-1832)\free{h }}
\end{paste}\end{patch}
```

3.57 kafile.ht

KeyedAccessFile

⇒ “notitle” (FileXmpPage) 3.40 on page 480
⇒ “notitle” (TextFileXmpPage) 3.107 on page 1307
⇒ “notitle” (LibraryXmpPage) 3.62 on page 840

— **kafile.ht** —

```
\begin{page}{KeyedAccessFileXmpPage}{KeyedAccessFile}
\beginscroll

The domain \spadtype{KeyedAccessFile(S)} provides files which can be
used as associative tables.  Data values are stored in these files and
can be retrieved according to their keys.  The keys must be strings so
this type behaves very much like the \spadtype{StringTable(S)} domain.
The difference is that keyed access files reside in secondary storage
while string tables are kept in memory.  For more information on
table-oriented operations, see the description of \spadtype{Table}.

\xtc{
Before a keyed access file can be used, it must first be opened.
A new file can be created by opening it for output.
}{
\spadpaste{ey: KeyedAccessFile(Integer) :=
open("/tmp/editor.year", "output")  \bound{ey}}
}
\xtc{
Just as for vectors, tables or lists, values are saved in a keyed
access file by setting elements.
}{
\spadpaste{ey."Char"     := 1986 \free{ey}\bound{eya}}
}
\xtc{
}{
\spadpaste{ey."Caviness" := 1985 \free{ey}\bound{eyb}}
}
\xtc{
}{
\spadpaste{ey."Fitch"    := 1984 \free{ey}\bound{eyc}}
}
\xtc{
Values are retrieved using application, in any of its syntactic forms.
}{
\spadpaste{ey."Char"     \free{eya}}
}
\xtc{
}{
```

```
\spadpaste{ey("Char")    \free{eya}}
}
\xtc{
}{
\spadpaste{ey "Char"     \free{eya}}
}
\xtc{
Attempting to retrieve a non-existent element in this way causes an
error.  If it is not known whether a key exists, you should use the
\spadfunFrom{search}{KeyedAccessFile} operation.
}{
\spadpaste{search("Char", ey)   \free{eya,eyb,eyc}\bound{eyaa}}
}
\xtc{
}{
\spadpaste{search("Smith", ey)  \free{eyaa}}
}
\xtc{
When an entry is no longer needed, it can be removed from the file.
}{
\spadpaste{remove!("Char", ey)  \free{eyaa}\bound{eybb}}
}
\xtc{
The \spadfunFrom{keys}{KeyedAccessFile} operation returns a list of
all the keys for a given file.
}{
\spadpaste{keys ey  \free{eybb}}
}
\xtc{
The \spadfunFrom{\#}{KeyedAccessFile} operation gives the
number of entries.
}{
\spadpaste{\#ey     \free{eybb}}
}

The table view of keyed access files provides safe operations.
That is, if the Axiom program is terminated between file operations,
the file is left in a consistent, current state.
This means, however, that the operations are somewhat costly.
For example, after each update the file is closed.
\xtc{
Here we add several more items to the file, then check its contents.
}{
\spadpaste{KE := Record(key: String, entry: Integer)  \bound{KE}}
}
\xtc{
}{
\spadpaste{reopen!(ey, "output")  \free{eybb,KE}\bound{eycc}}
}
\xtc{
```

```
If many items are to be added to a file at the same time, then
it is more efficient to use the
\spadfunFromX{write}{KeyedAccessFile} operation.
}{
\spadpaste{write!(ey, ["van Hulzen", 1983]\$KE) \bound{eyccA}\free{eycc}}
}
\xtc{
}{
\spadpaste{write!(ey, ["Calmet", 1982]\$KE) \bound{eyccB}\free{eycc}}
}
\xtc{
}{
\spadpaste{write!(ey, ["Wang", 1981]\$KE) \bound{eyccC}\free{eycc}}
}
\xtc{
}{
\spadpaste{close! ey   \free{eyccA,eyccB,eyccC}\bound{eydd}}
}
\xtc{
The \spadfunFromX{read}{KeyedAccessFile} operation is also available
from the file view, but it returns elements in a random order.  It is
generally clearer and more efficient to use the
\spadfunFrom{keys}{KeyedAccessFile} operation and to extract elements
by key.
}{
\spadpaste{keys ey     \free{eydd}}
}
\xtc{
}{
\spadpaste{members ey  \free{eydd}}
}
\noOutputXtc{
}{
\spadpaste{)system rm -r /tmp/editor.year  \free{ey}}
}

For more information on related topics,  see
\downlink{`File'}{FileXmpPage}\ignore{File},
\downlink{`TextFile'}{TextFileXmpPage}\ignore{TextFile}, and
\downlink{`Library'}{LibraryXmpPage}\ignore{Library}.
%
\showBlurb{KeyedAccessFile}
\endscroll
\autobuttons
\end{page}

\begin{patch}{KeyedAccessFileXmpPagePatch1}
\begin{paste}{KeyedAccessFileXmpPageFull1}{KeyedAccessFileXmpPageEmpty1}
\pastebutton{KeyedAccessFileXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{ey: KeyedAccessFile(Integer) := open("/tmp/editor.year", "output")\bound{ey
```

```
\indentrel{3}\begin{verbatim}
   (1)  "/tmp/editor.year"
                        Type: KeyedAccessFile Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{KeyedAccessFileXmpPageEmpty1}
\begin{paste}{KeyedAccessFileXmpPageEmpty1}{KeyedAccessFileXmpPagePatch1}
\pastebutton{KeyedAccessFileXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{ey: KeyedAccessFile(Integer) := open("/tmp/editor.year", "output")\bound{ey }}
\end{paste}\end{patch}

\begin{patch}{KeyedAccessFileXmpPagePatch2}
\begin{paste}{KeyedAccessFileXmpPageFull2}{KeyedAccessFileXmpPageEmpty2}
\pastebutton{KeyedAccessFileXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{ey."Char" := 1986\free{ey }\bound{eya }}
\indentrel{3}\begin{verbatim}
   (2)  1986
                                Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{KeyedAccessFileXmpPageEmpty2}
\begin{paste}{KeyedAccessFileXmpPageEmpty2}{KeyedAccessFileXmpPagePatch2}
\pastebutton{KeyedAccessFileXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{ey."Char" := 1986\free{ey }\bound{eya }}
\end{paste}\end{patch}

\begin{patch}{KeyedAccessFileXmpPagePatch3}
\begin{paste}{KeyedAccessFileXmpPageFull3}{KeyedAccessFileXmpPageEmpty3}
\pastebutton{KeyedAccessFileXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{ey."Caviness" := 1985\free{ey }\bound{eyb }}
\indentrel{3}\begin{verbatim}
   (3)  1985
                                Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{KeyedAccessFileXmpPageEmpty3}
\begin{paste}{KeyedAccessFileXmpPageEmpty3}{KeyedAccessFileXmpPagePatch3}
\pastebutton{KeyedAccessFileXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{ey."Caviness" := 1985\free{ey }\bound{eyb }}
\end{paste}\end{patch}

\begin{patch}{KeyedAccessFileXmpPagePatch4}
\begin{paste}{KeyedAccessFileXmpPageFull4}{KeyedAccessFileXmpPageEmpty4}
\pastebutton{KeyedAccessFileXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{ey."Fitch" := 1984\free{ey }\bound{eyc }}
\indentrel{3}\begin{verbatim}
   (4)  1984
```

```
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{KeyedAccessFileXmpPageEmpty4}
\begin{paste}{KeyedAccessFileXmpPageEmpty4}{KeyedAccessFileXmpPagePatch4}
\pastebutton{KeyedAccessFileXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{ey."Fitch" := 1984\free{ey }\bound{eyc }}
\end{paste}\end{patch}

\begin{patch}{KeyedAccessFileXmpPagePatch5}
\begin{paste}{KeyedAccessFileXmpPageFull5}{KeyedAccessFileXmpPageEmpty5}
\pastebutton{KeyedAccessFileXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{ey."Char"\free{eya }}
\indentrel{3}\begin{verbatim}
   (5)  1986
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{KeyedAccessFileXmpPageEmpty5}
\begin{paste}{KeyedAccessFileXmpPageEmpty5}{KeyedAccessFileXmpPagePatch5}
\pastebutton{KeyedAccessFileXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{ey."Char"\free{eya }}
\end{paste}\end{patch}

\begin{patch}{KeyedAccessFileXmpPagePatch6}
\begin{paste}{KeyedAccessFileXmpPageFull6}{KeyedAccessFileXmpPageEmpty6}
\pastebutton{KeyedAccessFileXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{ey("Char")\free{eya }}
\indentrel{3}\begin{verbatim}
   (6)  1986
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{KeyedAccessFileXmpPageEmpty6}
\begin{paste}{KeyedAccessFileXmpPageEmpty6}{KeyedAccessFileXmpPagePatch6}
\pastebutton{KeyedAccessFileXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{ey("Char")\free{eya }}
\end{paste}\end{patch}

\begin{patch}{KeyedAccessFileXmpPagePatch7}
\begin{paste}{KeyedAccessFileXmpPageFull7}{KeyedAccessFileXmpPageEmpty7}
\pastebutton{KeyedAccessFileXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{ey "Char"\free{eya }}
\indentrel{3}\begin{verbatim}
   (7)  1986
                                  Type: PositiveInteger
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{KeyedAccessFileXmpPageEmpty7}
\begin{paste}{KeyedAccessFileXmpPageEmpty7}{KeyedAccessFileXmpPagePatch7}
\pastebutton{KeyedAccessFileXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{ey "Char"\free{eya }}
\end{paste}\end{patch}

\begin{patch}{KeyedAccessFileXmpPagePatch8}
\begin{paste}{KeyedAccessFileXmpPageFull8}{KeyedAccessFileXmpPageEmpty8}
\pastebutton{KeyedAccessFileXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{search("Char", ey)\free{eya eyb eyc }\bound{eyaa }}
\indentrel{3}\begin{verbatim}
   (8)  1986
                               Type: Union(Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{KeyedAccessFileXmpPageEmpty8}
\begin{paste}{KeyedAccessFileXmpPageEmpty8}{KeyedAccessFileXmpPagePatch8}
\pastebutton{KeyedAccessFileXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{search("Char", ey)\free{eya eyb eyc }\bound{eyaa }}
\end{paste}\end{patch}

\begin{patch}{KeyedAccessFileXmpPagePatch9}
\begin{paste}{KeyedAccessFileXmpPageFull9}{KeyedAccessFileXmpPageEmpty9}
\pastebutton{KeyedAccessFileXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{search("Smith", ey)\free{eyaa }}
\indentrel{3}\begin{verbatim}
   (9)  "failed"
                              Type: Union("failed",...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{KeyedAccessFileXmpPageEmpty9}
\begin{paste}{KeyedAccessFileXmpPageEmpty9}{KeyedAccessFileXmpPagePatch9}
\pastebutton{KeyedAccessFileXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{search("Smith", ey)\free{eyaa }}
\end{paste}\end{patch}

\begin{patch}{KeyedAccessFileXmpPagePatch10}
\begin{paste}{KeyedAccessFileXmpPageFull10}{KeyedAccessFileXmpPageEmpty10}
\pastebutton{KeyedAccessFileXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{remove!("Char", ey)\free{eyaa }\bound{eybb }}
\indentrel{3}\begin{verbatim}
   (10)  1986
                               Type: Union(Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{KeyedAccessFileXmpPageEmpty10}
\begin{paste}{KeyedAccessFileXmpPageEmpty10}{KeyedAccessFileXmpPagePatch10}
\pastebutton{KeyedAccessFileXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{remove!("Char", ey)\free{eyaa }\bound{eybb }}
\end{paste}\end{patch}

\begin{patch}{KeyedAccessFileXmpPagePatch11}
\begin{paste}{KeyedAccessFileXmpPageFull11}{KeyedAccessFileXmpPageEmpty11}
\pastebutton{KeyedAccessFileXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{keys ey\free{eybb }}
\indentrel{3}\begin{verbatim}
   (11)  ["Fitch","Caviness"]
                                      Type: List String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{KeyedAccessFileXmpPageEmpty11}
\begin{paste}{KeyedAccessFileXmpPageEmpty11}{KeyedAccessFileXmpPagePatch11}
\pastebutton{KeyedAccessFileXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{keys ey\free{eybb }}
\end{paste}\end{patch}

\begin{patch}{KeyedAccessFileXmpPagePatch12}
\begin{paste}{KeyedAccessFileXmpPageFull12}{KeyedAccessFileXmpPageEmpty12}
\pastebutton{KeyedAccessFileXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{\#ey\free{eybb }}
\indentrel{3}\begin{verbatim}
   (12)  2
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{KeyedAccessFileXmpPageEmpty12}
\begin{paste}{KeyedAccessFileXmpPageEmpty12}{KeyedAccessFileXmpPagePatch12}
\pastebutton{KeyedAccessFileXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{\#ey\free{eybb }}
\end{paste}\end{patch}

\begin{patch}{KeyedAccessFileXmpPagePatch13}
\begin{paste}{KeyedAccessFileXmpPageFull13}{KeyedAccessFileXmpPageEmpty13}
\pastebutton{KeyedAccessFileXmpPageFull13}{\hidepaste}
\tab{5}\spadcommand{KE := Record(key: String, entry: Integer)\bound{KE }}
\indentrel{3}\begin{verbatim}
   (13)  Record(key: String,entry: Integer)
                                           Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{KeyedAccessFileXmpPageEmpty13}
\begin{paste}{KeyedAccessFileXmpPageEmpty13}{KeyedAccessFileXmpPagePatch13}
```

```
\pastebutton{KeyedAccessFileXmpPageEmpty13}{\showpaste}
\tab{5}\spadcommand{KE := Record(key: String, entry: Integer)\bound{KE }}
\end{paste}\end{patch}

\begin{patch}{KeyedAccessFileXmpPagePatch14}
\begin{paste}{KeyedAccessFileXmpPageFull14}{KeyedAccessFileXmpPageEmpty14}
\pastebutton{KeyedAccessFileXmpPageFull14}{\hidepaste}
\tab{5}\spadcommand{reopen!(ey, "output")\free{eybb KE }\bound{eycc }}
\indentrel{3}\begin{verbatim}
   (14)  "/tmp/editor.year"
                            Type: KeyedAccessFile Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{KeyedAccessFileXmpPageEmpty14}
\begin{paste}{KeyedAccessFileXmpPageEmpty14}{KeyedAccessFileXmpPagePatch14}
\pastebutton{KeyedAccessFileXmpPageEmpty14}{\showpaste}
\tab{5}\spadcommand{reopen!(ey, "output")\free{eybb KE }\bound{eycc }}
\end{paste}\end{patch}

\begin{patch}{KeyedAccessFileXmpPagePatch15}
\begin{paste}{KeyedAccessFileXmpPageFull15}{KeyedAccessFileXmpPageEmpty15}
\pastebutton{KeyedAccessFileXmpPageFull15}{\hidepaste}
\tab{5}\spadcommand{write!(ey, ["van Hulzen", 1983]$KE)\bound{eyccA }\free{eycc }}
\indentrel{3}\begin{verbatim}
   (15)  [key= "van Hulzen",entry= 1983]
               Type: Record(key: String,entry: Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{KeyedAccessFileXmpPageEmpty15}
\begin{paste}{KeyedAccessFileXmpPageEmpty15}{KeyedAccessFileXmpPagePatch15}
\pastebutton{KeyedAccessFileXmpPageEmpty15}{\showpaste}
\tab{5}\spadcommand{write!(ey, ["van Hulzen", 1983]$KE)\bound{eyccA }\free{eycc }}
\end{paste}\end{patch}

\begin{patch}{KeyedAccessFileXmpPagePatch16}
\begin{paste}{KeyedAccessFileXmpPageFull16}{KeyedAccessFileXmpPageEmpty16}
\pastebutton{KeyedAccessFileXmpPageFull16}{\hidepaste}
\tab{5}\spadcommand{write!(ey, ["Calmet", 1982]$KE)\bound{eyccB }\free{eycc }}
\indentrel{3}\begin{verbatim}
   (16)  [key= "Calmet",entry= 1982]
               Type: Record(key: String,entry: Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{KeyedAccessFileXmpPageEmpty16}
\begin{paste}{KeyedAccessFileXmpPageEmpty16}{KeyedAccessFileXmpPagePatch16}
\pastebutton{KeyedAccessFileXmpPageEmpty16}{\showpaste}
\tab{5}\spadcommand{write!(ey, ["Calmet", 1982]$KE)\bound{eyccB }\free{eycc }}
```

```
\end{paste}\end{patch}

\begin{patch}{KeyedAccessFileXmpPagePatch17}
\begin{paste}{KeyedAccessFileXmpPageFull17}{KeyedAccessFileXmpPageEmpty17}
\pastebutton{KeyedAccessFileXmpPageFull17}{\hidepaste}
\tab{5}\spadcommand{write!(ey, ["Wang", 1981]$KE)\bound{eyccC }\free{eycc }}
\indentrel{3}\begin{verbatim}
   (17)  [key= "Wang",entry= 1981]
              Type: Record(key: String,entry: Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{KeyedAccessFileXmpPageEmpty17}
\begin{paste}{KeyedAccessFileXmpPageEmpty17}{KeyedAccessFileXmpPagePatch17}
\pastebutton{KeyedAccessFileXmpPageEmpty17}{\showpaste}
\tab{5}\spadcommand{write!(ey, ["Wang", 1981]$KE)\bound{eyccC }\free{eycc }}
\end{paste}\end{patch}

\begin{patch}{KeyedAccessFileXmpPagePatch18}
\begin{paste}{KeyedAccessFileXmpPageFull18}{KeyedAccessFileXmpPageEmpty18}
\pastebutton{KeyedAccessFileXmpPageFull18}{\hidepaste}
\tab{5}\spadcommand{close! ey\free{eyccA eyccB eyccC }\bound{eydd }}
\indentrel{3}\begin{verbatim}
   (18)  "/tmp/editor.year"
                         Type: KeyedAccessFile Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{KeyedAccessFileXmpPageEmpty18}
\begin{paste}{KeyedAccessFileXmpPageEmpty18}{KeyedAccessFileXmpPagePatch18}
\pastebutton{KeyedAccessFileXmpPageEmpty18}{\showpaste}
\tab{5}\spadcommand{close! ey\free{eyccA eyccB eyccC }\bound{eydd }}
\end{paste}\end{patch}

\begin{patch}{KeyedAccessFileXmpPagePatch19}
\begin{paste}{KeyedAccessFileXmpPageFull19}{KeyedAccessFileXmpPageEmpty19}
\pastebutton{KeyedAccessFileXmpPageFull19}{\hidepaste}
\tab{5}\spadcommand{keys ey\free{eydd }}
\indentrel{3}\begin{verbatim}
   (19)
   ["Wang","Calmet","van Hulzen","Fitch","Caviness"]
                                    Type: List String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{KeyedAccessFileXmpPageEmpty19}
\begin{paste}{KeyedAccessFileXmpPageEmpty19}{KeyedAccessFileXmpPagePatch19}
\pastebutton{KeyedAccessFileXmpPageEmpty19}{\showpaste}
\tab{5}\spadcommand{keys ey\free{eydd }}
\end{paste}\end{patch}
```

```
\begin{patch}{KeyedAccessFileXmpPagePatch20}
\begin{paste}{KeyedAccessFileXmpPageFull20}{KeyedAccessFileXmpPageEmpty20}
\pastebutton{KeyedAccessFileXmpPageFull20}{\hidepaste}
\tab{5}\spadcommand{members ey\free{eydd }}
\indentrel{3}\begin{verbatim}
   (20)  [1981,1982,1983,1984,1985]
                                    Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{KeyedAccessFileXmpPageEmpty20}
\begin{paste}{KeyedAccessFileXmpPageEmpty20}{KeyedAccessFileXmpPagePatch20}
\pastebutton{KeyedAccessFileXmpPageEmpty20}{\showpaste}
\tab{5}\spadcommand{members ey\free{eydd }}
\end{paste}\end{patch}

\begin{patch}{KeyedAccessFileXmpPagePatch21}
\begin{paste}{KeyedAccessFileXmpPageFull21}{KeyedAccessFileXmpPageEmpty21}
\pastebutton{KeyedAccessFileXmpPageFull21}{\hidepaste}
\tab{5}\spadcommand{)system rm -r /tmp/editor.year\free{ey }}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{KeyedAccessFileXmpPageEmpty21}
\begin{paste}{KeyedAccessFileXmpPageEmpty21}{KeyedAccessFileXmpPagePatch21}
\pastebutton{KeyedAccessFileXmpPageEmpty21}{\showpaste}
\tab{5}\spadcommand{)system rm -r /tmp/editor.year\free{ey }}
\end{paste}\end{patch}
```

3.58 kernel.ht

Kernel

⇒ "notitle" (BasicOperatorXmpPage) 3.10 on page 169
⇒ "notitle" (ExpressionXmpPage) 3.33 on page 436

— **kernel.ht** —

```
\begin{page}{KernelXmpPage}{Kernel}
\beginscroll

A {\it kernel} is a symbolic function application (such as
```

```
\spad{sin(x + y)}) or a symbol (such as \spad{x}).
More precisely, a non-symbol
kernel over a set {\it S} is an operator applied to a given list of
arguments from {\it S}.  The operator has type
\axiomType{BasicOperator} (see
\downlink{`BasicOperator'}{BasicOperatorXmpPage}\ignore{BasicOperator})
and the kernel object is usually part of an expression object (see
\downlink{`Expression'}{ExpressionXmpPage}\ignore{Expression}).

Kernels are created implicitly for you when you
create expressions.
\xtc{
}{
\spadpaste{x :: Expression Integer}
}
\xtc{
You can directly create a ``symbol'' kernel by using the
\axiomFunFrom{kernel}{Kernel} operation.
}{
\spadpaste{kernel x}
}
\xtc{
This expression has two different kernels.
}{
\spadpaste{sin(x) + cos(x) \bound{sincos}}
}
\xtc{
The operator \axiomFunFrom{kernels}{Expression} returns a list
of the kernels in an object of type \axiomType{Expression}.
}{
\spadpaste{kernels \% \free{sincos}}
}
\xtc{
This expression also has two different kernels.
}{
\spadpaste{sin(x)**2 + sin(x) + cos(x) \bound{sincos2}}
}
\xtc{
The \spad{sin(x)} kernel is used twice.
}{
\spadpaste{kernels \% \free{sincos2}}
}
\xtc{
An expression need not contain any kernels.
}{
\spadpaste{kernels(1 :: Expression Integer)}
}
\xtc{
If one or more kernels are present, one of them is
designated the {\it main} kernel.
```

```
}{
\spadpaste{mainKernel(cos(x) + tan(x))}
}
\xtc{
Kernels can be nested. Use \axiomFunFrom{height}{Kernel} to determine
the nesting depth.
}{
\spadpaste{height kernel x}
}
\xtc{
This has height 2 because the \spad{x} has height 1 and then we apply
an operator to that.
}{
\spadpaste{height mainKernel(sin x)}
}
\xtc{
}{
\spadpaste{height mainKernel(sin cos x)}
}
\xtc{
}{
\spadpaste{height mainKernel(sin cos (tan x + sin x))}
}
\xtc{
Use the \axiomFunFrom{operator}{Kernel} operation to extract the
operator component of the kernel.
The operator has type \axiomType{BasicOperator}.
}{
\spadpaste{operator mainKernel(sin cos (tan x + sin x))}
}
\xtc{
Use the \axiomFunFrom{name}{Kernel} operation to extract the name of
the operator component of the kernel.  The name has type
\axiomType{Symbol}.  This is really just a shortcut for a two-step
process of extracting the operator and then calling
\axiomFunFrom{name}{BasicOperator} on the operator.
}{
\spadpaste{name mainKernel(sin cos (tan x + sin x))}
}
Axiom knows about functions such as \axiomFun{sin}, \axiomFun{cos}
and so on and can make kernels and then expressions using them.
To create a kernel and expression using an arbitrary operator, use
\axiomFunFrom{operator}{BasicOperator}.
\xtc{
Now \spad{f} can be used to create symbolic function applications.
}{
\spadpaste{f := operator 'f \bound{f}}
}
\xtc{
}{
```

```
\spadpaste{e := f(x, y, 10) \free{f}\bound{e}}
}
\xtc{
Use the \axiomFunFrom{is?}{Kernel} operation to learn if the
operator component of a kernel is equal to a given operator.
}{
\spadpaste{is?(e, f) \free{f e}}
}
\xtc{
You can also use a symbol or a string as the second
argument to \axiomFunFrom{is?}{Kernel}.
}{
\spadpaste{is?(e, 'f) \free{e}}
}
\xtc{
Use the \axiomFunFrom{argument}{Kernel} operation to get a list containing
the argument component of a kernel.
}{
\spadpaste{argument mainKernel e \free{f}\free{e}}
}

Conceptually, an object of type \axiomType{Expression} can be thought
of a quotient of multivariate polynomials, where the ``variables''
are kernels.
The arguments of the kernels are again expressions and so the
structure recurses.
See
\downlink{`Expression'}{ExpressionXmpPage}\ignore{Expression}
for examples of using kernels to
take apart expression objects.
\endscroll
\autobuttons
\end{page}

\begin{patch}{KernelXmpPagePatch1}
\begin{paste}{KernelXmpPageFull1}{KernelXmpPageEmpty1}
\pastebutton{KernelXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{x :: Expression Integer}
\indentrel{3}\begin{verbatim}
   (1)  x
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{KernelXmpPageEmpty1}
\begin{paste}{KernelXmpPageEmpty1}{KernelXmpPagePatch1}
\pastebutton{KernelXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{x :: Expression Integer}
\end{paste}\end{patch}
```

```
\begin{patch}{KernelXmpPagePatch2}
\begin{paste}{KernelXmpPageFull2}{KernelXmpPageEmpty2}
\pastebutton{KernelXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{kernel x}
\indentrel{3}\begin{verbatim}
   (2)  x
                        Type: Kernel Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{KernelXmpPageEmpty2}
\begin{paste}{KernelXmpPageEmpty2}{KernelXmpPagePatch2}
\pastebutton{KernelXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{kernel x}
\end{paste}\end{patch}

\begin{patch}{KernelXmpPagePatch3}
\begin{paste}{KernelXmpPageFull3}{KernelXmpPageEmpty3}
\pastebutton{KernelXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{sin(x) + cos(x)\bound{sincos }}
\indentrel{3}\begin{verbatim}
   (3)  sin(x) + cos(x)
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{KernelXmpPageEmpty3}
\begin{paste}{KernelXmpPageEmpty3}{KernelXmpPagePatch3}
\pastebutton{KernelXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{sin(x) + cos(x)\bound{sincos }}
\end{paste}\end{patch}

\begin{patch}{KernelXmpPagePatch4}
\begin{paste}{KernelXmpPageFull4}{KernelXmpPageEmpty4}
\pastebutton{KernelXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{kernels \%\free{sincos }}
\indentrel{3}\begin{verbatim}
   (4)  [sin(x),cos(x)]
                   Type: List Kernel Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{KernelXmpPageEmpty4}
\begin{paste}{KernelXmpPageEmpty4}{KernelXmpPagePatch4}
\pastebutton{KernelXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{kernels \%\free{sincos }}
\end{paste}\end{patch}

\begin{patch}{KernelXmpPagePatch5}
\begin{paste}{KernelXmpPageFull5}{KernelXmpPageEmpty5}
```

```
\pastebutton{KernelXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{sin(x)**2 + sin(x) + cos(x)\bound{sincos2 }}
\indentrel{3}\begin{verbatim}
              2
   (5)  sin(x)  + sin(x) + cos(x)
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{KernelXmpPageEmpty5}
\begin{paste}{KernelXmpPageEmpty5}{KernelXmpPagePatch5}
\pastebutton{KernelXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{sin(x)**2 + sin(x) + cos(x)\bound{sincos2 }}
\end{paste}\end{patch}

\begin{patch}{KernelXmpPagePatch6}
\begin{paste}{KernelXmpPageFull6}{KernelXmpPageEmpty6}
\pastebutton{KernelXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{kernels \%\free{sincos2 }}
\indentrel{3}\begin{verbatim}
   (6)  [sin(x),cos(x)]
                   Type: List Kernel Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{KernelXmpPageEmpty6}
\begin{paste}{KernelXmpPageEmpty6}{KernelXmpPagePatch6}
\pastebutton{KernelXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{kernels \%\free{sincos2 }}
\end{paste}\end{patch}

\begin{patch}{KernelXmpPagePatch7}
\begin{paste}{KernelXmpPageFull7}{KernelXmpPageEmpty7}
\pastebutton{KernelXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{kernels(1 :: Expression Integer)}
\indentrel{3}\begin{verbatim}
   (7)  []
                   Type: List Kernel Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{KernelXmpPageEmpty7}
\begin{paste}{KernelXmpPageEmpty7}{KernelXmpPagePatch7}
\pastebutton{KernelXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{kernels(1 :: Expression Integer)}
\end{paste}\end{patch}

\begin{patch}{KernelXmpPagePatch8}
\begin{paste}{KernelXmpPageFull8}{KernelXmpPageEmpty8}
\pastebutton{KernelXmpPageFull8}{\hidepaste}
```

```
\tab{5}\spadcommand{mainKernel(cos(x) + tan(x))}
\indentrel{3}\begin{verbatim}
   (8)  tan(x)
             Type: Union(Kernel Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{KernelXmpPageEmpty8}
\begin{paste}{KernelXmpPageEmpty8}{KernelXmpPagePatch8}
\pastebutton{KernelXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{mainKernel(cos(x) + tan(x))}
\end{paste}\end{patch}

\begin{patch}{KernelXmpPagePatch9}
\begin{paste}{KernelXmpPageFull9}{KernelXmpPageEmpty9}
\pastebutton{KernelXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{height kernel x}
\indentrel{3}\begin{verbatim}
   (9)  1
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{KernelXmpPageEmpty9}
\begin{paste}{KernelXmpPageEmpty9}{KernelXmpPagePatch9}
\pastebutton{KernelXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{height kernel x}
\end{paste}\end{patch}

\begin{patch}{KernelXmpPagePatch10}
\begin{paste}{KernelXmpPageFull10}{KernelXmpPageEmpty10}
\pastebutton{KernelXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{height mainKernel(sin x)}
\indentrel{3}\begin{verbatim}
   (10)  2
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{KernelXmpPageEmpty10}
\begin{paste}{KernelXmpPageEmpty10}{KernelXmpPagePatch10}
\pastebutton{KernelXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{height mainKernel(sin x)}
\end{paste}\end{patch}

\begin{patch}{KernelXmpPagePatch11}
\begin{paste}{KernelXmpPageFull11}{KernelXmpPageEmpty11}
\pastebutton{KernelXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{height mainKernel(sin cos x)}
\indentrel{3}\begin{verbatim}
```

```
   (11)  3
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{KernelXmpPageEmpty11}
\begin{paste}{KernelXmpPageEmpty11}{KernelXmpPagePatch11}
\pastebutton{KernelXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{height mainKernel(sin cos x)}
\end{paste}\end{patch}

\begin{patch}{KernelXmpPagePatch12}
\begin{paste}{KernelXmpPageFull12}{KernelXmpPageEmpty12}
\pastebutton{KernelXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{height mainKernel(sin cos (tan x + sin x))}
\indentrel{3}\begin{verbatim}
   (12)  4
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{KernelXmpPageEmpty12}
\begin{paste}{KernelXmpPageEmpty12}{KernelXmpPagePatch12}
\pastebutton{KernelXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{height mainKernel(sin cos (tan x + sin x))}
\end{paste}\end{patch}

\begin{patch}{KernelXmpPagePatch13}
\begin{paste}{KernelXmpPageFull13}{KernelXmpPageEmpty13}
\pastebutton{KernelXmpPageFull13}{\hidepaste}
\tab{5}\spadcommand{operator mainKernel(sin cos (tan x + sin x))}
\indentrel{3}\begin{verbatim}
   (13)  sin
                                   Type: BasicOperator
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{KernelXmpPageEmpty13}
\begin{paste}{KernelXmpPageEmpty13}{KernelXmpPagePatch13}
\pastebutton{KernelXmpPageEmpty13}{\showpaste}
\tab{5}\spadcommand{operator mainKernel(sin cos (tan x + sin x))}
\end{paste}\end{patch}

\begin{patch}{KernelXmpPagePatch14}
\begin{paste}{KernelXmpPageFull14}{KernelXmpPageEmpty14}
\pastebutton{KernelXmpPageFull14}{\hidepaste}
\tab{5}\spadcommand{name mainKernel(sin cos (tan x + sin x))}
\indentrel{3}\begin{verbatim}
   (14)  sin
                                          Type: Symbol
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{KernelXmpPageEmpty14}
\begin{paste}{KernelXmpPageEmpty14}{KernelXmpPagePatch14}
\pastebutton{KernelXmpPageEmpty14}{\showpaste}
\tab{5}\spadcommand{name mainKernel(sin cos (tan x + sin x))}
\end{paste}\end{patch}

\begin{patch}{KernelXmpPagePatch15}
\begin{paste}{KernelXmpPageFull15}{KernelXmpPageEmpty15}
\pastebutton{KernelXmpPageFull15}{\hidepaste}
\tab{5}\spadcommand{f := operator 'f\bound{f }}
\indentrel{3}\begin{verbatim}
   (15)  f
                                   Type: BasicOperator
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{KernelXmpPageEmpty15}
\begin{paste}{KernelXmpPageEmpty15}{KernelXmpPagePatch15}
\pastebutton{KernelXmpPageEmpty15}{\showpaste}
\tab{5}\spadcommand{f := operator 'f\bound{f }}
\end{paste}\end{patch}

\begin{patch}{KernelXmpPagePatch16}
\begin{paste}{KernelXmpPageFull16}{KernelXmpPageEmpty16}
\pastebutton{KernelXmpPageFull16}{\hidepaste}
\tab{5}\spadcommand{e := f(x, y, 10)\free{f }\bound{e }}
\indentrel{3}\begin{verbatim}
   (16)  f(x,y,10)
                              Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{KernelXmpPageEmpty16}
\begin{paste}{KernelXmpPageEmpty16}{KernelXmpPagePatch16}
\pastebutton{KernelXmpPageEmpty16}{\showpaste}
\tab{5}\spadcommand{e := f(x, y, 10)\free{f }\bound{e }}
\end{paste}\end{patch}

\begin{patch}{KernelXmpPagePatch17}
\begin{paste}{KernelXmpPageFull17}{KernelXmpPageEmpty17}
\pastebutton{KernelXmpPageFull17}{\hidepaste}
\tab{5}\spadcommand{is?(e, f)\free{f e }}
\indentrel{3}\begin{verbatim}
   (17)  true
                                         Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{KernelXmpPageEmpty17}
\begin{paste}{KernelXmpPageEmpty17}{KernelXmpPagePatch17}
\pastebutton{KernelXmpPageEmpty17}{\showpaste}
\tab{5}\spadcommand{is?(e, f)\free{f e }}
\end{paste}\end{patch}

\begin{patch}{KernelXmpPagePatch18}
\begin{paste}{KernelXmpPageFull18}{KernelXmpPageEmpty18}
\pastebutton{KernelXmpPageFull18}{\hidepaste}
\tab{5}\spadcommand{is?(e, 'f)\free{e }}
\indentrel{3}\begin{verbatim}
   (18)  true
                                               Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{KernelXmpPageEmpty18}
\begin{paste}{KernelXmpPageEmpty18}{KernelXmpPagePatch18}
\pastebutton{KernelXmpPageEmpty18}{\showpaste}
\tab{5}\spadcommand{is?(e, 'f)\free{e }}
\end{paste}\end{patch}

\begin{patch}{KernelXmpPagePatch19}
\begin{paste}{KernelXmpPageFull19}{KernelXmpPageEmpty19}
\pastebutton{KernelXmpPageFull19}{\hidepaste}
\tab{5}\spadcommand{argument mainKernel e\free{f }\free{e }}
\indentrel{3}\begin{verbatim}
   (19)  [x,y,10]
                            Type: List Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{KernelXmpPageEmpty19}
\begin{paste}{KernelXmpPageEmpty19}{KernelXmpPagePatch19}
\pastebutton{KernelXmpPageEmpty19}{\showpaste}
\tab{5}\spadcommand{argument mainKernel e\free{f }\free{e }}
\end{paste}\end{patch}
```

———-

3.59 lazm3pk.ht

LazardSetSolvingPackage

— lazm3pk.ht —

```
\begin{page}{LazardSetSolvingPackageXmpPage}{LazardSetSolvingPackage}
\beginscroll
The \spadtype{LazardSetSolvingPackage} package constructor solves
polynomial systems by means of Lazard triangular sets.  However one
condition is relaxed: Regular triangular sets whose saturated ideals
have positive dimension are not necessarily normalized.

The decompositions are computed in two steps.  First the algorithm of
Moreno Maza (implemented in the \spadtype{RegularTriangularSet} domain
constructor) is called.  Then the resulting decompositions are
converted into lists of square-free regular triangular sets and the
redundant components are removed.  Moreover, zero-dimensional regular
triangular sets are normalized.

Note that the way of understanding triangular decompositions
is detailed in the example of the \spadtype{RegularTriangularSet}
constructor.

The \spadtype{LazardSetSolvingPackage} constructor takes six arguments.
The first one, {\bf R}, is the coefficient ring of the polynomials;
it must belong to the category \spadtype{GcdDomain}.
The second one, {\bf E}, is the exponent monoid of the polynomials;
it must belong to the category \spadtype{OrderedAbelianMonoidSup}.
the third one, {\bf V}, is the ordered set of variables;
it must belong to the category \spadtype{OrderedSet}.
The fourth one is the polynomial ring;
it must belong to the category
\spadtype{RecursivePolynomialCategory(R,E,V)}.
The fifth one is a domain of the category
\spadtype{RegularTriangularSetCategory(R,E,V,P)}
and the last one is a domain of the category
\spadtype{SquareFreeRegularTriangularSetCategory(R,E,V,P)}.
The abbreviation for \spadtype{LazardSetSolvingPackage} is
\spad{LAZM3PK}.

{\bf N.B.} For the purpose of solving zero-dimensional algebraic
systems, see also \spadtype{LexTriangularPackage} and
\spadtype{ZeroDimensionalSolvePackage}.
These packages are easier to call than \spad{LAZM3PK}.
Moreover, the \spadtype{ZeroDimensionalSolvePackage}
package  provides operations
to compute either the complex roots or the real roots.

We illustrate now the use of the \spadtype{LazardSetSolvingPackage}
package constructor with two examples (Butcher and Vermeer).


\xtc{
Define the coefficient ring.
}{
```

```
\spadpaste{R := Integer \bound{R}}
}


\xtc{
Define the list of variables,
}{
\spadpaste{ls : List Symbol := [b1,x,y,z,t,v,u,w] \bound{ls}}
}

\xtc{
and make it an ordered set;
}{
\spadpaste{V := OVAR(ls) \free{ls} \bound{V}}
}

\xtc{
then define the exponent monoid.
}{
\spadpaste{E := IndexedExponents V \free{V} \bound{E}}
}

\xtc{
Define the polynomial ring.
}{
\spadpaste{P := NSMP(R, V) \free{R} \free{V} \bound{P}}
}

\xtc{
Let the variables be polynomial.
}{
\spadpaste{b1: P := 'b1  \free{P} \bound{b1}}
}
\xtc{
}{
\spadpaste{x: P := 'x \free{P} \bound{x}}
}
\xtc{
}{
\spadpaste{y: P := 'y \free{P} \bound{y}}
}
\xtc{
}{
\spadpaste{z: P := 'z \free{P} \bound{z}}
}
\xtc{
}{
\spadpaste{t: P := 't \free{P} \bound{t}}
}
\xtc{
```

```
}{
\spadpaste{u: P := 'u  \free{P} \bound{u}}
}
\xtc{
}{
\spadpaste{v: P := 'v  \free{P} \bound{v}}
}
\xtc{
}{
\spadpaste{w: P := 'w  \free{P} \bound{w}}
}

\xtc{
Now call the \spadtype{RegularTriangularSet} domain constructor.
}{
\spadpaste{T := REGSET(R,E,V,P) \free{R} \free{E} \free{V}
\free{P} \bound{T} }
}

\xtc{
Define a polynomial system (the Butcher example).
}{
\spadpaste{p0 := b1 + y + z - t - w \free{b1} \free{x} \free{y}
\free{z} \free{t} \free{u} \free{v} \free{w} \bound{p0}}
}
\xtc{
}{
\spadpaste{p1 := 2*z*u + 2*y*v + 2*t*w - 2*w**2 - w - 1 \free{b1}
\free{x} \free{y} \free{z} \free{t} \free{u} \free{v} \free{w} \bound{p1}}
}
\xtc{
}{
\spadpaste{p2 := 3*z*u**2 + 3*y*v**2 - 3*t*w**2 + 3*w**3 +
3*w**2 - t + 4*w \free{b1} \free{x} \free{y} \free{z} \free{t}
\free{u} \free{v} \free{w} \bound{p2}}
}
\xtc{
}{
\spadpaste{p3 := 6*x*z*v - 6*t*w**2 + 6*w**3 - 3*t*w +
6*w**2 - t + 4*w \free{b1} \free{x} \free{y} \free{z}
\free{t} \free{u} \free{v} \free{w} \bound{p3}}
}
\xtc{
}{
\spadpaste{p4 := 4*z*u**3+ 4*y*v**3+ 4*t*w**3- 4*w**4 -
6*w**3+ 4*t*w- 10*w**2- w- 1 \free{b1} \free{x} \free{y}
\free{z} \free{t} \free{u} \free{v} \free{w} \bound{p4}}
}
\xtc{
}{
```

```
\spadpaste{p5 := 8*x*z*u*v +8*t*w**3 -8*w**4 +4*t*w**2 -
12*w**3 +4*t*w -14*w**2 -3*w -1  \free{b1} \free{x} \free{y}
\free{z} \free{t} \free{u} \free{v} \free{w} \bound{p5}}
}
\xtc{
}{
\spadpaste{p6 := 12*x*z*v**2+12*t*w**3 -12*w**4 +12*t*w**2 -
18*w**3 +8*t*w -14*w**2 -w -1 \free{b1} \free{x} \free{y}
\free{z} \free{t} \free{u} \free{v} \free{w} \bound{p6}}
}
\xtc{
}{
\spadpaste{p7 := -24*t*w**3 + 24*w**4 - 24*t*w**2 + 36*w**3 -
8*t*w + 26*w**2 + 7*w + 1 \free{b1} \free{x} \free{y} \free{z}
\free{t} \free{u} \free{v} \free{w} \bound{p7}}
}
\xtc{
}{
\spadpaste{lp := [p0, p1, p2, p3, p4, p5, p6, p7] \free{p0}
\free{p1} \free{p2} \free{p3} \free{p4} \free{p5} \free{p6}
\free{p7} \bound{lp}}
}

\xtc{
First of all, let us solve this system in the sense of Lazard
by means of the \spadtype{REGSET} constructor:
}{
\spadpaste{lts := zeroSetSplit(lp,false)$T\free{lp}\free{T}\bound{lts}}
}

\xtc{
We can get the dimensions of each component
of a decomposition as follows.
}{
\spadpaste{[coHeight(ts) for ts in lts] \free{lts}}
}

The first five sets have a simple shape.
However, the last one, which has dimension zero, can be simplified
by using Lazard triangular sets.

\xtc{
Thus we call the \spadtype{SquareFreeRegularTriangularSet} domain
constructor,
}{
\spadpaste{ST := SREGSET(R,E,V,P) \free{R} \free{E} \free{V}
\free{P} \bound{ST} }
}

\xtc{
```

```
and set the \spadtype{LAZM3PK} package constructor to our situation.
}{
\spadpaste{pack := LAZM3PK(R,E,V,P,T,ST) \free{R} \free{E}
\free{V} \free{P} \free{T} \free{ST} \bound{pack} }
}

\xtc{
We are ready to solve the system by means of Lazard triangular sets:
}{
\spadpaste{zeroSetSplit(lp,false)$pack \free{lp} \free{pack}}
}

We see the sixth triangular set is {\em nicer} now:
each one of its polynomials has a constant initial.

We follow with the Vermeer example. The ordering is the usual one
for this system.

\xtc{
Define the polynomial system.
}{
\spadpaste{f0 := (w - v) ** 2 + (u - t) ** 2 - 1 \free{b1} \free{x}
\free{y} \free{z} \free{t} \free{u} \free{v} \free{w} \bound{f0}}
}
\xtc{
}{
\spadpaste{f1 := t ** 2 - v ** 3 \free{b1} \free{x} \free{y} \free{z}
\free{t} \free{u} \free{v} \free{w} \bound{f1}}
}
\xtc{
}{
\spadpaste{f2 := 2 * t * (w - v) + 3 * v ** 2 * (u - t) \free{b1}
\free{x} \free{y} \free{z} \free{t} \free{u} \free{v} \free{w}
\bound{f2}}
}
\xtc{
}{
\spadpaste{f3 := (3 * z * v ** 2 - 1) * (2 * z * t - 1) \free{b1}
\free{x} \free{y} \free{z} \free{t} \free{u} \free{v} \free{w}
\bound{f3}}
}
\xtc{
}{
\spadpaste{lf := [f0, f1, f2, f3] \free{f0} \free{f1} \free{f2}
\free{f3} \bound{lf}}
}


\xtc{
First of all, let us solve this system in the sense of Kalkbrener by
```

```
means of the \spadtype{REGSET}
constructor:
}{
\spadpaste{zeroSetSplit(lf,true)$T \free{lf} \free{T}}
}

We have obtained one regular chain (i.e. regular triangular set) with
dimension 1.  This set is in fact a characterist set of the (radical
of) of the ideal generated by the input system {\bf lf}.  Thus we have
only the {\em generic points} of the variety associated with {\bf lf}
(for the elimination ordering given by {\bf ls}).

So let us get now a full description of this variety.
\xtc{
Hence, we solve this system in the sense of Lazard by means of the
\spadtype{REGSET}
constructor:
}{
\spadpaste{zeroSetSplit(lf,false)$T \free{lf} \free{T}}
}

We retrieve our regular chain of dimension 1 and we get three regular
chains of dimension 0 corresponding to the {\em degenerated cases}.
We want now to simplify these zero-dimensional regular chains by using
Lazard triangular sets.  Moreover, this will allow us to prove that
the above decomposition has no redundant component.  {\bf N.B.}
Generally, decompositions computed by the \spadtype{REGSET}
constructor do not have redundant components.  However, to be sure
that no redundant component occurs one needs to use the
\spadtype{SREGSET} or \spadtype{LAZM3PK} constructors.

\xtc{
So let us solve the input system  in the sense of Lazard by means of
the \spadtype{LAZM3PK} constructor:
}{
\spadpaste{zeroSetSplit(lf,false)$pack \free{lf} \free{pack}}
}

Due to square-free factorization, we obtained now four
zero-dimensional regular chains.  Moreover, each of them is normalized
(the initials are constant).  Note that these zero-dimensional
components may be investigated further with the
\spadtype{ZeroDimensionalSolvePackage} package constructor.

\endscroll
\autobuttons
\end{page}

\begin{patch}{LazardSetSolvingPackageXmpPagePatch1}
\begin{paste}{LazardSetSolvingPackageXmpPageFull1}{LazardSetSolvingPackageXmpPageEmpty1}
```

```
\pastebutton{LazardSetSolvingPackageXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{R := Integer\bound{R }}
\indentrel{3}\begin{verbatim}
   (1)  Integer
                                         Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPageEmpty1}
\begin{paste}{LazardSetSolvingPackageXmpPageEmpty1}{LazardSetSolvingPackageXmpPagePatch1}
\pastebutton{LazardSetSolvingPackageXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{R := Integer\bound{R }}
\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPagePatch2}
\begin{paste}{LazardSetSolvingPackageXmpPageFull2}{LazardSetSolvingPackageXmpPageEmpty2}
\pastebutton{LazardSetSolvingPackageXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{ls : List Symbol := [b1,x,y,z,t,v,u,w]\bound{ls }}
\indentrel{3}\begin{verbatim}
   (2)  [b1,x,y,z,t,v,u,w]
                                    Type: List Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPageEmpty2}
\begin{paste}{LazardSetSolvingPackageXmpPageEmpty2}{LazardSetSolvingPackageXmpPagePatch2}
\pastebutton{LazardSetSolvingPackageXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{ls : List Symbol := [b1,x,y,z,t,v,u,w]\bound{ls }}
\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPagePatch3}
\begin{paste}{LazardSetSolvingPackageXmpPageFull3}{LazardSetSolvingPackageXmpPageEmpty3}
\pastebutton{LazardSetSolvingPackageXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{V := OVAR(ls)\free{ls }\bound{V }}
\indentrel{3}\begin{verbatim}
   (3)  OrderedVariableList [b1,x,y,z,t,v,u,w]
                                         Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPageEmpty3}
\begin{paste}{LazardSetSolvingPackageXmpPageEmpty3}{LazardSetSolvingPackageXmpPagePatch3}
\pastebutton{LazardSetSolvingPackageXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{V := OVAR(ls)\free{ls }\bound{V }}
\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPagePatch4}
\begin{paste}{LazardSetSolvingPackageXmpPageFull4}{LazardSetSolvingPackageXmpPageEmpty4}
\pastebutton{LazardSetSolvingPackageXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{E := IndexedExponents V\free{V }\bound{E }}
```

```
\indentrel{3}\begin{verbatim}
   (4)
   IndexedExponents OrderedVariableList [b1,x,y,z,t,v,u,w]
                                         Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPageEmpty4}
\begin{paste}{LazardSetSolvingPackageXmpPageEmpty4}{LazardSetSolvingPackageXmpPagePatch4}
\pastebutton{LazardSetSolvingPackageXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{E := IndexedExponents V\free{V }\bound{E }}
\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPagePatch5}
\begin{paste}{LazardSetSolvingPackageXmpPageFull5}{LazardSetSolvingPackageXmpPageEmpty5}
\pastebutton{LazardSetSolvingPackageXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{P := NSMP(R, V)\free{R }\free{V }\bound{P }}
\indentrel{3}\begin{verbatim}
   (5)
  NewSparseMultivariatePolynomial(Integer,OrderedVariable
  List [b1,x,y,z,t,v,u,w])
                                         Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPageEmpty5}
\begin{paste}{LazardSetSolvingPackageXmpPageEmpty5}{LazardSetSolvingPackageXmpPagePatch5}
\pastebutton{LazardSetSolvingPackageXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{P := NSMP(R, V)\free{R }\free{V }\bound{P }}
\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPagePatch6}
\begin{paste}{LazardSetSolvingPackageXmpPageFull6}{LazardSetSolvingPackageXmpPageEmpty6}
\pastebutton{LazardSetSolvingPackageXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{b1: P := 'b1\free{P }\bound{b1 }}
\indentrel{3}\begin{verbatim}
   (6)  b1
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [b1,x,y,z,t,v,u,w])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPageEmpty6}
\begin{paste}{LazardSetSolvingPackageXmpPageEmpty6}{LazardSetSolvingPackageXmpPagePatch6}
\pastebutton{LazardSetSolvingPackageXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{b1: P := 'b1\free{P }\bound{b1 }}
\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPagePatch7}
\begin{paste}{LazardSetSolvingPackageXmpPageFull7}{LazardSetSolvingPackageXmpPageEmpty7}
\pastebutton{LazardSetSolvingPackageXmpPageFull7}{\hidepaste}
```

```
\tab{5}\spadcommand{x: P := 'x\free{P }\bound{x }}
\indentrel{3}\begin{verbatim}
   (7)  x
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [b1,x,y,z,t,v,u,w])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPageEmpty7}
\begin{paste}{LazardSetSolvingPackageXmpPageEmpty7}{LazardSetSolvingPackageXmpPagePatch7}
\pastebutton{LazardSetSolvingPackageXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{x: P := 'x\free{P }\bound{x }}
\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPagePatch8}
\begin{paste}{LazardSetSolvingPackageXmpPageFull8}{LazardSetSolvingPackageXmpPageEmpty8}
\pastebutton{LazardSetSolvingPackageXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{y: P := 'y\free{P }\bound{y }}
\indentrel{3}\begin{verbatim}
   (8)  y
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [b1,x,y,z,t,v,u,w])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPageEmpty8}
\begin{paste}{LazardSetSolvingPackageXmpPageEmpty8}{LazardSetSolvingPackageXmpPagePatch8}
\pastebutton{LazardSetSolvingPackageXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{y: P := 'y\free{P }\bound{y }}
\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPagePatch9}
\begin{paste}{LazardSetSolvingPackageXmpPageFull9}{LazardSetSolvingPackageXmpPageEmpty9}
\pastebutton{LazardSetSolvingPackageXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{z: P := 'z\free{P }\bound{z }}
\indentrel{3}\begin{verbatim}
   (9)  z
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [b1,x,y,z,t,v,u,w])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPageEmpty9}
\begin{paste}{LazardSetSolvingPackageXmpPageEmpty9}{LazardSetSolvingPackageXmpPagePatch9}
\pastebutton{LazardSetSolvingPackageXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{z: P := 'z\free{P }\bound{z }}
\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPagePatch10}
\begin{paste}{LazardSetSolvingPackageXmpPageFull10}{LazardSetSolvingPackageXmpPageEmpty10}
\pastebutton{LazardSetSolvingPackageXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{t: P := 't\free{P }\bound{t }}
\indentrel{3}\begin{verbatim}
```

```
   (10)  t
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [b1,x,y,z,t,v,u,w])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPageEmpty10}
\begin{paste}{LazardSetSolvingPackageXmpPageEmpty10}{LazardSetSolvingPackageXmpPagePatch10}
\pastebutton{LazardSetSolvingPackageXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{t: P := 't\free{P }\bound{t }}
\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPagePatch11}
\begin{paste}{LazardSetSolvingPackageXmpPageFull11}{LazardSetSolvingPackageXmpPageEmpty11}
\pastebutton{LazardSetSolvingPackageXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{u: P := 'u\free{P }\bound{u }}
\indentrel{3}\begin{verbatim}
   (11)  u
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [b1,x,y,z,t,v,u,w])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPageEmpty11}
\begin{paste}{LazardSetSolvingPackageXmpPageEmpty11}{LazardSetSolvingPackageXmpPagePatch11}
\pastebutton{LazardSetSolvingPackageXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{u: P := 'u\free{P }\bound{u }}
\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPagePatch12}
\begin{paste}{LazardSetSolvingPackageXmpPageFull12}{LazardSetSolvingPackageXmpPageEmpty12}
\pastebutton{LazardSetSolvingPackageXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{v: P := 'v\free{P }\bound{v }}
\indentrel{3}\begin{verbatim}
   (12)  v
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [b1,x,y,z,t,v,u,w])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPageEmpty12}
\begin{paste}{LazardSetSolvingPackageXmpPageEmpty12}{LazardSetSolvingPackageXmpPagePatch12}
\pastebutton{LazardSetSolvingPackageXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{v: P := 'v\free{P }\bound{v }}
\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPagePatch13}
\begin{paste}{LazardSetSolvingPackageXmpPageFull13}{LazardSetSolvingPackageXmpPageEmpty13}
\pastebutton{LazardSetSolvingPackageXmpPageFull13}{\hidepaste}
\tab{5}\spadcommand{w: P := 'w\free{P }\bound{w }}
\indentrel{3}\begin{verbatim}
   (13)  w
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [b1,x,y,z,t,v,u,w])
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPageEmpty13}
\begin{paste}{LazardSetSolvingPackageXmpPageEmpty13}{LazardSetSolvingPackageXmpPagePatch13}
\pastebutton{LazardSetSolvingPackageXmpPageEmpty13}{\showpaste}
\tab{5}\spadcommand{w: P := 'w\free{P }\bound{w }}
\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPagePatch14}
\begin{paste}{LazardSetSolvingPackageXmpPageFull14}{LazardSetSolvingPackageXmpPageEmpty14}
\pastebutton{LazardSetSolvingPackageXmpPageFull14}{\hidepaste}
\tab{5}\spadcommand{T := REGSET(R,E,V,P)\free{R }\free{E }\free{V }\free{P }\bound{T }}
\indentrel{3}\begin{verbatim}
   (14)
  RegularTriangularSet(Integer,IndexedExponents OrderedVa
  riableList [b1,x,y,z,t,v,u,w],OrderedVariableList [b1,x
  ,y,z,t,v,u,w],NewSparseMultivariatePolynomial(Integer,O
  rderedVariableList [b1,x,y,z,t,v,u,w]))
                                            Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPageEmpty14}
\begin{paste}{LazardSetSolvingPackageXmpPageEmpty14}{LazardSetSolvingPackageXmpPagePatch14}
\pastebutton{LazardSetSolvingPackageXmpPageEmpty14}{\showpaste}
\tab{5}\spadcommand{T := REGSET(R,E,V,P)\free{R }\free{E }\free{V }\free{P }\bound{T }}
\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPagePatch15}
\begin{paste}{LazardSetSolvingPackageXmpPageFull15}{LazardSetSolvingPackageXmpPageEmpty15}
\pastebutton{LazardSetSolvingPackageXmpPageFull15}{\hidepaste}
\tab{5}\spadcommand{p0 := b1 + y + z - t - w\free{b1 }\free{x }\free{y }\free{z }\free{t }\free{u }\free{v }
\indentrel{3}\begin{verbatim}
   (15)  b1 + y + z - t - w
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [b1,x,y,z,t,v,u,w])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPageEmpty15}
\begin{paste}{LazardSetSolvingPackageXmpPageEmpty15}{LazardSetSolvingPackageXmpPagePatch15}
\pastebutton{LazardSetSolvingPackageXmpPageEmpty15}{\showpaste}
\tab{5}\spadcommand{p0 := b1 + y + z - t - w\free{b1 }\free{x }\free{y }\free{z }\free{t }\free{u }\free{v }
\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPagePatch16}
\begin{paste}{LazardSetSolvingPackageXmpPageFull16}{LazardSetSolvingPackageXmpPageEmpty16}
\pastebutton{LazardSetSolvingPackageXmpPageFull16}{\hidepaste}
\tab{5}\spadcommand{p1 := 2*z*u + 2*y*v + 2*t*w - 2*w**2 - w - 1\free{b1 }\free{x }\free{y }\free{z }\free{
\indentrel{3}\begin{verbatim}
```

```
                              2
   (16)  2v y + 2u z + 2w t - 2w  - w - 1
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [b1,x,y,z,t,v,u,w])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPageEmpty16}
\begin{paste}{LazardSetSolvingPackageXmpPageEmpty16}{LazardSetSolvingPackageXmpPagePatch16}
\pastebutton{LazardSetSolvingPackageXmpPageEmpty16}{\showpaste}
\tab{5}\spadcommand{p1 := 2*z*u + 2*y*v + 2*t*w - 2*w**2 - w - 1\free{b1 }\free{x }\free{y }\fre
\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPagePatch17}
\begin{paste}{LazardSetSolvingPackageXmpPageFull17}{LazardSetSolvingPackageXmpPageEmpty17}
\pastebutton{LazardSetSolvingPackageXmpPageFull17}{\hidepaste}
\tab{5}\spadcommand{p2 := 3*z*u**2 + 3*y*v**2 - 3*t*w**2 + 3*w**3 + 3*w**2 - t + 4*w\free{b1 }\f
\indentrel{3}\begin{verbatim}
           2      2         2           3     2
   (17)  3v y + 3u z + (- 3w  - 1)t + 3w  + 3w  + 4w
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [b1,x,y,z,t,v,u,w])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPageEmpty17}
\begin{paste}{LazardSetSolvingPackageXmpPageEmpty17}{LazardSetSolvingPackageXmpPagePatch17}
\pastebutton{LazardSetSolvingPackageXmpPageEmpty17}{\showpaste}
\tab{5}\spadcommand{p2 := 3*z*u**2 + 3*y*v**2 - 3*t*w**2 + 3*w**3 + 3*w**2 - t + 4*w\free{b1 }\f
\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPagePatch18}
\begin{paste}{LazardSetSolvingPackageXmpPageFull18}{LazardSetSolvingPackageXmpPageEmpty18}
\pastebutton{LazardSetSolvingPackageXmpPageFull18}{\hidepaste}
\tab{5}\spadcommand{p3 := 6*x*z*v - 6*t*w**2 + 6*w**3 - 3*t*w + 6*w**2 - t + 4*w\free{b1 }\free{
\indentrel{3}\begin{verbatim}
                       2                3     2
   (18)  6v z x + (- 6w  - 3w - 1)t + 6w  + 6w  + 4w
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [b1,x,y,z,t,v,u,w])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPageEmpty18}
\begin{paste}{LazardSetSolvingPackageXmpPageEmpty18}{LazardSetSolvingPackageXmpPagePatch18}
\pastebutton{LazardSetSolvingPackageXmpPageEmpty18}{\showpaste}
\tab{5}\spadcommand{p3 := 6*x*z*v - 6*t*w**2 + 6*w**3 - 3*t*w + 6*w**2 - t + 4*w\free{b1 }\free{
\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPagePatch19}
\begin{paste}{LazardSetSolvingPackageXmpPageFull19}{LazardSetSolvingPackageXmpPageEmpty19}
\pastebutton{LazardSetSolvingPackageXmpPageFull19}{\hidepaste}
\tab{5}\spadcommand{p4 := 4*z*u**3+ 4*y*v**3+ 4*t*w**3- 4*w**4 - 6*w**3+ 4*t*w- 10*w**2- w- 1\fr
```

```
\indentrel{3}\begin{verbatim}
   (19)
     3      3       3            4     3      2
   4v y + 4u z + (4w  + 4w)t - 4w  - 6w  - 10w  - w - 1
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [b1,x,y,z,t,v,u,w])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPageEmpty19}
\begin{paste}{LazardSetSolvingPackageXmpPageEmpty19}{LazardSetSolvingPackageXmpPagePatch19}
\pastebutton{LazardSetSolvingPackageXmpPageEmpty19}{\showpaste}
\tab{5}\spadcommand{p4 := 4*z*u**3+ 4*y*v**3+ 4*t*w**3- 4*w**4 - 6*w**3+ 4*t*w- 10*w**2- w- 1\free{b1 }\free
\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPagePatch20}
\begin{paste}{LazardSetSolvingPackageXmpPageFull20}{LazardSetSolvingPackageXmpPageEmpty20}
\pastebutton{LazardSetSolvingPackageXmpPageFull20}{\hidepaste}
\tab{5}\spadcommand{p5 := 8*x*z*u*v +8*t*w**3 -8*w**4 +4*t*w**2 -12*w**3 +4*t*w -14*w**2 -3*w -1\free{b1 }\
\indentrel{3}\begin{verbatim}
   (20)
                  3     2            4      3      2
     8u v z x + (8w  + 4w  + 4w)t - 8w  - 12w  - 14w  - 3w
   +
     - 1
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [b1,x,y,z,t,v,u,w])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPageEmpty20}
\begin{paste}{LazardSetSolvingPackageXmpPageEmpty20}{LazardSetSolvingPackageXmpPagePatch20}
\pastebutton{LazardSetSolvingPackageXmpPageEmpty20}{\showpaste}
\tab{5}\spadcommand{p5 := 8*x*z*u*v +8*t*w**3 -8*w**4 +4*t*w**2 -12*w**3 +4*t*w -14*w**2 -3*w -1\free{b1 }\
\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPagePatch21}
\begin{paste}{LazardSetSolvingPackageXmpPageFull21}{LazardSetSolvingPackageXmpPageEmpty21}
\pastebutton{LazardSetSolvingPackageXmpPageFull21}{\hidepaste}
\tab{5}\spadcommand{p6 := 12*x*z*v**2+12*t*w**3 -12*w**4 +12*t*w**2 -18*w**3 +8*t*w -14*w**2 -w -1\free{b1 
\indentrel{3}\begin{verbatim}
   (21)
       2          3      2             4      3      2
     12v z x + (12w  + 12w  + 8w)t - 12w  - 18w  - 14w
   +
     - w - 1
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [b1,x,y,z,t,v,u,w])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPageEmpty21}
\begin{paste}{LazardSetSolvingPackageXmpPageEmpty21}{LazardSetSolvingPackageXmpPagePatch21}
```

```
\pastebutton{LazardSetSolvingPackageXmpPageEmpty21}{\showpaste}
\tab{5}\spadcommand{p6 := 12*x*z*v**2+12*t*w**3 -12*w**4 +12*t*w**2 -18*w**3 +8*t*w -14*w**2 -w
\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPagePatch22}
\begin{paste}{LazardSetSolvingPackageXmpPageFull22}{LazardSetSolvingPackageXmpPageEmpty22}
\pastebutton{LazardSetSolvingPackageXmpPageFull22}{\hidepaste}
\tab{5}\spadcommand{p7 := -24*t*w**3 + 24*w**4 - 24*t*w**2 + 36*w**3 - 8*t*w + 26*w**2 + 7*w + 1
\indentrel{3}\begin{verbatim}
   (22)
         3      2             4      3      2
   (- 24w  - 24w  - 8w)t + 24w  + 36w  + 26w  + 7w + 1
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [b1,x,y,z,t,v,u,w])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPageEmpty22}
\begin{paste}{LazardSetSolvingPackageXmpPageEmpty22}{LazardSetSolvingPackageXmpPagePatch22}
\pastebutton{LazardSetSolvingPackageXmpPageEmpty22}{\showpaste}
\tab{5}\spadcommand{p7 := -24*t*w**3 + 24*w**4 - 24*t*w**2 + 36*w**3 - 8*t*w + 26*w**2 + 7*w + 1
\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPagePatch23}
\begin{paste}{LazardSetSolvingPackageXmpPageFull23}{LazardSetSolvingPackageXmpPageEmpty23}
\pastebutton{LazardSetSolvingPackageXmpPageFull23}{\hidepaste}
\tab{5}\spadcommand{lp := [p0, p1, p2, p3, p4, p5, p6, p7]\free{p0 }\free{p1 }\free{p2 }\free{p3
\indentrel{3}\begin{verbatim}
   (23)
                                                  2
   [b1 + y + z - t - w, 2v y + 2u z + 2w t - 2w  - w - 1,
      2      2          2           3     2
    3v y + 3u z + (- 3w  - 1)t + 3w  + 3w  + 4w,
                2                 3     2
    6v z x + (- 6w  - 3w - 1)t + 6w  + 6w  + 4w,
      3      3       3            4     3      2
    4v y + 4u z + (4w  + 4w)t - 4w  - 6w  - 10w  - w - 1,

                    3     2            4      3      2
       8u v z x + (8w  + 4w  + 4w)t - 8w  - 12w  - 14w
     +
       - 3w - 1
     ,

          2          3      2             4      3      2
       12v z x + (12w  + 12w  + 8w)t - 12w  - 18w  - 14w
     +
       - w - 1
     ,
         3      2             4      3      2
    (- 24w  - 24w  - 8w)t + 24w  + 36w  + 26w  + 7w + 1]
```

```
Type: List NewSparseMultivariatePolynomial(Integer,OrderedVariableList [b1,x,y,z,t,v,u,w])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPageEmpty23}
\begin{paste}{LazardSetSolvingPackageXmpPageEmpty23}{LazardSetSolvingPackageXmpPagePatch23}
\pastebutton{LazardSetSolvingPackageXmpPageEmpty23}{\showpaste}
\tab{5}\spadcommand{lp := [p0, p1, p2, p3, p4, p5, p6, p7]\free{p0 }\free{p1 }\free{p2 }\free{p3 }\free{p4 
\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPagePatch24}
\begin{paste}{LazardSetSolvingPackageXmpPageFull24}{LazardSetSolvingPackageXmpPageEmpty24}
\pastebutton{LazardSetSolvingPackageXmpPageFull24}{\hidepaste}
\tab{5}\spadcommand{lts := zeroSetSplit(lp,false)$T\free{lp }\free{T }\bound{lts }}
\indentrel{3}\begin{verbatim}
   (24)
   [{w + 1,u,v,t + 1,b1 + y + z + 2},
    {w + 1,v,t + 1,z,b1 + y + 2},
    {w + 1,t + 1,z,y,b1 + 2},
    {w + 1,v - u,t + 1,y + z,x,b1 + 2},
    {w + 1,u,t + 1,y,x,b1 + z + 2},

          5       4      3     2
     {144w  + 216w  + 96w  + 6w  - 11w - 1,
          2                 5       4      3     2
      (12w  + 9w + 1)u - 72w  - 108w  - 42w  - 9w  - 3w,
          2                 4      3      2
      (12w  + 9w + 1)v + 36w  + 54w  + 18w ,
          3      2             4      3      2
      (24w  + 24w  + 8w)t - 24w  - 36w  - 26w  - 7w - 1,

                      2                 2
         (12u v - 12u )z + (12w v + 12w  + 4)t + (3w - 5)v
       +
            4      3     2
         36w  + 42w  + 6w  - 16w
       ,
                                 2
      2v y + 2u z + 2w t - 2w  - w - 1,
                    2                3     2
      6v z x + (- 6w  - 3w - 1)t + 6w  + 6w  + 4w,
      b1 + y + z - t - w}
     ]
Type: List RegularTriangularSet(Integer,IndexedExponents OrderedVariableList [b1,x,y,z,t,v,u,w],OrderedVaria
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPageEmpty24}
\begin{paste}{LazardSetSolvingPackageXmpPageEmpty24}{LazardSetSolvingPackageXmpPagePatch24}
\pastebutton{LazardSetSolvingPackageXmpPageEmpty24}{\showpaste}
```

```
\tab{5}\spadcommand{lts := zeroSetSplit(lp,false)$T\free{lp }\free{T }\bound{lts }}
\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPagePatch25}
\begin{paste}{LazardSetSolvingPackageXmpPageFull25}{LazardSetSolvingPackageXmpPageEmpty25}
\pastebutton{LazardSetSolvingPackageXmpPageFull25}{\hidepaste}
\tab{5}\spadcommand{[coHeight(ts) for ts in lts]\free{lts }}
\indentrel{3}\begin{verbatim}
   (25)  [3,3,3,2,2,0]
                           Type: List NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPageEmpty25}
\begin{paste}{LazardSetSolvingPackageXmpPageEmpty25}{LazardSetSolvingPackageXmpPagePatch25}
\pastebutton{LazardSetSolvingPackageXmpPageEmpty25}{\showpaste}
\tab{5}\spadcommand{[coHeight(ts) for ts in lts]\free{lts }}
\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPagePatch26}
\begin{paste}{LazardSetSolvingPackageXmpPageFull26}{LazardSetSolvingPackageXmpPageEmpty26}
\pastebutton{LazardSetSolvingPackageXmpPageFull26}{\hidepaste}
\tab{5}\spadcommand{ST := SREGSET(R,E,V,P)\free{R }\free{E }\free{V }\free{P }\bound{ST }}
\indentrel{3}\begin{verbatim}
   (26)
  SquareFreeRegularTriangularSet(Integer,IndexedExponents
   OrderedVariableList [b1,x,y,z,t,v,u,w],OrderedVariable
  List [b1,x,y,z,t,v,u,w],NewSparseMultivariatePolynomial
  (Integer,OrderedVariableList [b1,x,y,z,t,v,u,w]))
                                            Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPageEmpty26}
\begin{paste}{LazardSetSolvingPackageXmpPageEmpty26}{LazardSetSolvingPackageXmpPagePatch26}
\pastebutton{LazardSetSolvingPackageXmpPageEmpty26}{\showpaste}
\tab{5}\spadcommand{ST := SREGSET(R,E,V,P)\free{R }\free{E }\free{V }\free{P }\bound{ST }}
\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPagePatch27}
\begin{paste}{LazardSetSolvingPackageXmpPageFull27}{LazardSetSolvingPackageXmpPageEmpty27}
\pastebutton{LazardSetSolvingPackageXmpPageFull27}{\hidepaste}
\tab{5}\spadcommand{pack := LAZM3PK(R,E,V,P,T,ST)\free{R }\free{E }\free{V }\free{P }\free{T }\f
\indentrel{3}\begin{verbatim}
   (27)
  LazardSetSolvingPackage(Integer,IndexedExponents Ordere
  dVariableList [b1,x,y,z,t,v,u,w],OrderedVariableList [b
  1,x,y,z,t,v,u,w],NewSparseMultivariatePolynomial(Intege
  r,OrderedVariableList [b1,x,y,z,t,v,u,w]),RegularTriang
  ularSet(Integer,IndexedExponents OrderedVariableList [b
```

```
  1,x,y,z,t,v,u,w],OrderedVariableList [b1,x,y,z,t,v,u,w]
  ,NewSparseMultivariatePolynomial(Integer,OrderedVariabl
  eList [b1,x,y,z,t,v,u,w])),SquareFreeRegularTriangularS
  et(Integer,IndexedExponents OrderedVariableList [b1,x,y
  ,z,t,v,u,w],OrderedVariableList [b1,x,y,z,t,v,u,w],NewS
  parseMultivariatePolynomial(Integer,OrderedVariableList
   [b1,x,y,z,t,v,u,w])))
                                            Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPageEmpty27}
\begin{paste}{LazardSetSolvingPackageXmpPageEmpty27}{LazardSetSolvingPackageXmpPagePatch27}
\pastebutton{LazardSetSolvingPackageXmpPageEmpty27}{\showpaste}
\tab{5}\spadcommand{pack := LAZM3PK(R,E,V,P,T,ST)\free{R }\free{E }\free{V }\free{P }\free{T }\free{ST }\bou
\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPagePatch28}
\begin{paste}{LazardSetSolvingPackageXmpPageFull28}{LazardSetSolvingPackageXmpPageEmpty28}
\pastebutton{LazardSetSolvingPackageXmpPageFull28}{\hidepaste}
\tab{5}\spadcommand{zeroSetSplit(lp,false)$pack\free{lp }\free{pack }}
\indentrel{3}\begin{verbatim}
   (28)
   [{w + 1,t + 1,z,y,b1 + 2},
    {w + 1,v,t + 1,z,b1 + y + 2},
    {w + 1,u,v,t + 1,b1 + y + z + 2},
    {w + 1,v - u,t + 1,y + z,x,b1 + 2},
    {w + 1,u,t + 1,y,x,b1 + z + 2},

          5       4      3     2
     {144w  + 216w  + 96w  + 6w  - 11w - 1,
             4      3      2
      u - 24w  - 36w  - 14w  + w + 1,
              4      3      2
      3v - 48w  - 60w  - 10w  + 8w + 2,
             4      3      2
      t - 24w  - 36w  - 14w  - w + 1,
                 4       3       2
      486z - 2772w  - 4662w  - 2055w  + 30w + 127,
                   4        3       2
      2916y - 22752w  - 30312w  - 8220w  + 2064w + 1561,
                 4        3       2
      356x - 3696w  - 4536w  - 968w  + 822w + 371,
                    4        3        2
      2916b1 - 30600w  - 46692w  - 20274w  - 8076w + 593}
     ]
Type: List SquareFreeRegularTriangularSet(Integer,IndexedExponents OrderedVariableList [b1,x,y,z,t,v,u,w],O
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{LazardSetSolvingPackageXmpPageEmpty28}
\begin{paste}{LazardSetSolvingPackageXmpPageEmpty28}{LazardSetSolvingPackageXmpPagePatch28}
\pastebutton{LazardSetSolvingPackageXmpPageEmpty28}{\showpaste}
\tab{5}\spadcommand{zeroSetSplit(lp,false)$pack\free{lp }\free{pack }}
\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPagePatch29}
\begin{paste}{LazardSetSolvingPackageXmpPageFull29}{LazardSetSolvingPackageXmpPageEmpty29}
\pastebutton{LazardSetSolvingPackageXmpPageFull29}{\hidepaste}
\tab{5}\spadcommand{f0 := (w - v) ** 2 + (u - t) ** 2 - 1\free{b1 }\free{x }\free{y }\free{z }\f
\indentrel{3}\begin{verbatim}
          2           2           2    2
   (29)  t  - 2u t + v  - 2w v + u  + w  - 1
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [b1,x,y,z,t,v,u,w])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPageEmpty29}
\begin{paste}{LazardSetSolvingPackageXmpPageEmpty29}{LazardSetSolvingPackageXmpPagePatch29}
\pastebutton{LazardSetSolvingPackageXmpPageEmpty29}{\showpaste}
\tab{5}\spadcommand{f0 := (w - v) ** 2 + (u - t) ** 2 - 1\free{b1 }\free{x }\free{y }\free{z }\f
\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPagePatch30}
\begin{paste}{LazardSetSolvingPackageXmpPageFull30}{LazardSetSolvingPackageXmpPageEmpty30}
\pastebutton{LazardSetSolvingPackageXmpPageFull30}{\hidepaste}
\tab{5}\spadcommand{f1 := t ** 2 - v ** 3\free{b1 }\free{x }\free{y }\free{z }\free{t }\free{u }
\indentrel{3}\begin{verbatim}
          2    3
   (30)  t  - v
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [b1,x,y,z,t,v,u,w])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPageEmpty30}
\begin{paste}{LazardSetSolvingPackageXmpPageEmpty30}{LazardSetSolvingPackageXmpPagePatch30}
\pastebutton{LazardSetSolvingPackageXmpPageEmpty30}{\showpaste}
\tab{5}\spadcommand{f1 := t ** 2 - v ** 3\free{b1 }\free{x }\free{y }\free{z }\free{t }\free{u }
\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPagePatch31}
\begin{paste}{LazardSetSolvingPackageXmpPageFull31}{LazardSetSolvingPackageXmpPageEmpty31}
\pastebutton{LazardSetSolvingPackageXmpPageFull31}{\hidepaste}
\tab{5}\spadcommand{f2 := 2 * t * (w - v) + 3 * v ** 2 * (u - t)\free{b1 }\free{x }\free{y }\fre
\indentrel{3}\begin{verbatim}
              2                    2
   (31)  (- 3v  - 2v + 2w)t + 3u v
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [b1,x,y,z,t,v,u,w])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{LazardSetSolvingPackageXmpPageEmpty31}
\begin{paste}{LazardSetSolvingPackageXmpPageEmpty31}{LazardSetSolvingPackageXmpPagePatch31}
\pastebutton{LazardSetSolvingPackageXmpPageEmpty31}{\showpaste}
\tab{5}\spadcommand{f2 := 2 * t * (w - v) + 3 * v ** 2 * (u - t)\free{b1 }\free{x }\free{y }\free{z }\free{t
\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPagePatch32}
\begin{paste}{LazardSetSolvingPackageXmpPageFull32}{LazardSetSolvingPackageXmpPageEmpty32}
\pastebutton{LazardSetSolvingPackageXmpPageFull32}{\hidepaste}
\tab{5}\spadcommand{f3 := (3 * z * v ** 2 - 1) * (2 * z * t - 1)\free{b1 }\free{x }\free{y }\free{z }\free{t
\indentrel{3}\begin{verbatim}
           2   2             2
   (32)  6v t z  + (- 2t - 3v )z + 1
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [b1,x,y,z,t,v,u,w])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPageEmpty32}
\begin{paste}{LazardSetSolvingPackageXmpPageEmpty32}{LazardSetSolvingPackageXmpPagePatch32}
\pastebutton{LazardSetSolvingPackageXmpPageEmpty32}{\showpaste}
\tab{5}\spadcommand{f3 := (3 * z * v ** 2 - 1) * (2 * z * t - 1)\free{b1 }\free{x }\free{y }\free{z }\free{t
\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPagePatch33}
\begin{paste}{LazardSetSolvingPackageXmpPageFull33}{LazardSetSolvingPackageXmpPageEmpty33}
\pastebutton{LazardSetSolvingPackageXmpPageFull33}{\hidepaste}
\tab{5}\spadcommand{lf := [f0, f1, f2, f3]\free{f0 }\free{f1 }\free{f2 }\free{f3 }\bound{lf }}
\indentrel{3}\begin{verbatim}
   (33)
     2           2           2    2       2    3
   [t  - 2u t + v  - 2w v + u  + w  - 1, t  - v ,
        2                  2
    (- 3v  - 2v + 2w)t + 3u v ,
      2   2            2
    6v t z  + (- 2t - 3v )z + 1]
Type: List NewSparseMultivariatePolynomial(Integer,OrderedVariableList [b1,x,y,z,t,v,u,w])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPageEmpty33}
\begin{paste}{LazardSetSolvingPackageXmpPageEmpty33}{LazardSetSolvingPackageXmpPagePatch33}
\pastebutton{LazardSetSolvingPackageXmpPageEmpty33}{\showpaste}
\tab{5}\spadcommand{lf := [f0, f1, f2, f3]\free{f0 }\free{f1 }\free{f2 }\free{f3 }\bound{lf }}
\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPagePatch34}
\begin{paste}{LazardSetSolvingPackageXmpPageFull34}{LazardSetSolvingPackageXmpPageEmpty34}
\pastebutton{LazardSetSolvingPackageXmpPageFull34}{\hidepaste}
\tab{5}\spadcommand{zeroSetSplit(lf,true)$T\free{lf }\free{T }}
```

```
\indentrel{3}\begin{verbatim}
   (34)
   [
     {
           6           3       2                 4
         729u  + (- 1458w  + 729w  - 4158w - 1685)u
       +
               6        5        4        3       2
             729w  - 1458w  - 2619w  - 4892w  - 297w
           +
             5814w + 427
        *
            2
           u
       +
           8       7        6        5        4        3
         729w  + 216w  - 2900w  - 2376w  + 3870w  + 4072w
       +
                2
         - 1188w  - 1656w + 529
       ,

                4           3       2                 2
             2187u  + (- 4374w  - 972w  - 12474w - 2868)u
           +
                6        5         4        3        2
             2187w  - 1944w  - 10125w  - 4800w  + 2501w
           +
             4968w - 1587
        *
           v
       +
              3       2  2       6        5        4
         (1944w  - 108w )u  + 972w  + 3024w  - 1080w
       +
             3        2
         496w  + 1116w
       ,
         2                 2
      (3v  + 2v - 2w)t - 3u v ,
                        2  2           2
       ((4v - 4w)t - 6u v )z  + (2t + 3v )z - 1}
     ]
Type: List RegularTriangularSet(Integer,IndexedExponents OrderedVariableList [b1,x,y,z,t,v,u,w],
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPageEmpty34}
\begin{paste}{LazardSetSolvingPackageXmpPageEmpty34}{LazardSetSolvingPackageXmpPagePatch34}
\pastebutton{LazardSetSolvingPackageXmpPageEmpty34}{\showpaste}
```

```
\tab{5}\spadcommand{zeroSetSplit(lf,true)$T\free{lf }\free{T }}
\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPagePatch35}
\begin{paste}{LazardSetSolvingPackageXmpPageFull35}{LazardSetSolvingPackageXmpPageEmpty35}
\pastebutton{LazardSetSolvingPackageXmpPageFull35}{\hidepaste}
\tab{5}\spadcommand{zeroSetSplit(lf,false)$T\free{lf }\free{T }}
\indentrel{3}\begin{verbatim}
   (35)
   [
     {
            6           3       2                 4
         729u  + (- 1458w  + 729w  - 4158w - 1685)u
       +
                 6        5        4        3       2
             729w  - 1458w  - 2619w  - 4892w  - 297w
           +
             5814w + 427
        *
            2
           u
       +
            8       7        6        5        4        3
         729w  + 216w  - 2900w  - 2376w  + 3870w  + 4072w
       +
                2
         - 1188w  - 1656w + 529
       ,

                 4           3       2                  2
             2187u  + (- 4374w  - 972w  - 12474w - 2868)u
           +
                  6        5         4        3        2
             2187w  - 1944w  - 10125w  - 4800w  + 2501w
           +
             4968w - 1587
        *
           v
       +
              3       2  2       6        5        4
         (1944w  - 108w )u  + 972w  + 3024w  - 1080w
       +
            3        2
         496w  + 1116w
       ,
          2                   2
      (3v  + 2v - 2w)t - 3u v ,
                         2  2           2
      ((4v - 4w)t - 6u v )z  + (2t + 3v )z - 1}
     ,
```

```
    4     3      2
{27w  + 4w  - 54w  - 36w + 23, u,
                2             2          2
 (12w + 2)v - 9w  - 2w + 9, 6t  - 2v - 3w  + 2w + 3,
 2t z - 1}
,

{
          6          5         4          3         2
    59049w  + 91854w  - 45198w  + 145152w  + 63549w
  +
    60922w + 21420
  ,

                      5                   4
        31484448266904w  - 18316865522574w
      +
                      3                  2
        23676995746098w  + 6657857188965w
      +
        8904703998546w + 3890631403260
   *
      2
     u
  +
                    5                   4
    94262810316408w  - 82887296576616w
  +
                    3                   2
    89801831438784w  + 28141734167208w
  +
    38070359425432w + 16003865949120
  ,

        2          2
    (243w  + 36w + 85)v
  +
       2       3      2                     3     2
  (- 81u  - 162w  + 36w  + 154w + 72)v - 72w  + 4w
  ,
    2                  2
 (3v  + 2v - 2w)t - 3u v ,
                  2  2          2
 ((4v - 4w)t - 6u v )z  + (2t + 3v )z - 1}
,

    4     3      2
{27w  + 4w  - 54w  - 36w + 23, u,
                2             2          2
```

```
      (12w + 2)v - 9w  - 2w + 9, 6t  - 2v - 3w  + 2w + 3,
        2
      3v z - 1}
     ]
Type: List RegularTriangularSet(Integer,IndexedExponents OrderedVariableList [b1,x,y,z,t,v,u,w],OrderedVari
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPageEmpty35}
\begin{paste}{LazardSetSolvingPackageXmpPageEmpty35}{LazardSetSolvingPackageXmpPagePatch35}
\pastebutton{LazardSetSolvingPackageXmpPageEmpty35}{\showpaste}
\tab{5}\spadcommand{zeroSetSplit(lf,false)$T\free{lf }\free{T }}
\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPagePatch36}
\begin{paste}{LazardSetSolvingPackageXmpPageFull36}{LazardSetSolvingPackageXmpPageEmpty36}
\pastebutton{LazardSetSolvingPackageXmpPageFull36}{\hidepaste}
\tab{5}\spadcommand{zeroSetSplit(lf,false)$pack\free{lf }\free{pack }}
\indentrel{3}\begin{verbatim}
   (36)
   [
     {
             6           3       2                 4
         729u  + (- 1458w  + 729w  - 4158w - 1685)u
       +
                 6        5        4        3       2
             729w  - 1458w  - 2619w  - 4892w  - 297w
           +
             5814w + 427
        *
            2
           u
       +
             8       7        6        5        4        3
         729w  + 216w  - 2900w  - 2376w  + 3870w  + 4072w
       +
               2
         - 1188w  - 1656w + 529
       ,

                4           3      2                  2
             2187u  + (- 4374w  - 972w  - 12474w - 2868)u
           +
                  6        5         4        3        2
             2187w  - 1944w  - 10125w  - 4800w  + 2501w
           +
             4968w - 1587
        *
           v
       +
```

```
          3        2  2        6         5         4
    (1944w  - 108w )u  + 972w  + 3024w  - 1080w
  +
         3        2
    496w  + 1116w
  ,
    2                    2
 (3v  + 2v - 2w)t - 3u v ,
                    2  2            2
 ((4v - 4w)t - 6u v )z  + (2t + 3v )z - 1}
,

    2                  2
{81w  + 18w + 28, 729u  - 1890w - 533,
    2
 81v  + (- 162w + 27)v - 72w - 112,
 11881t + (972w + 2997)u v + (- 11448w - 11536)u,

                     2
    641237934604288z
  +
            (78614584763904w + 26785578742272)u
          +
            236143618655616w + 70221988585728
       *
          v
      +
        (358520253138432w + 101922133759488)u
      +
        142598803536000w + 54166419595008
   *
      z
  +
    (32655103844499w - 44224572465882)u v
  +
    (43213900115457w - 32432039102070)u
  }
,

    4     3      2
{27w  + 4w  - 54w  - 36w + 23, u,
            3     2
 218v - 162w  + 3w  + 160w + 153,
     2      3      2
 109t  - 27w  - 54w  + 63w + 80,
                 3      2
 1744z + (- 1458w  + 27w  + 1440w + 505)t}
,

    4     3      2
```

```
     {27w  + 4w  - 54w  - 36w + 23, u,
                3     2
      218v - 162w  + 3w  + 160w + 153,
          2      3      2
      109t  - 27w  - 54w  + 63w + 80,
                 3     2
      1308z + 162w  - 3w  - 814w - 153}
     ,

         4       3        2
     {729w  + 972w  - 1026w  + 1684w + 765,
        2      2
      81u  + 72w  + 16w - 72,
                3       2
      702v - 162w  - 225w  + 40w - 99,
                    3       2
      11336t + (324w  - 603w  - 1718w - 1557)u,

                     2
         595003968z
       +
                             3             2
                - 963325386w  - 898607682w  + 1516286466w
              +
                - 3239166186
            *
              u
          +
                        3             2
            - 1579048992w  - 1796454288w  + 2428328160w
          +
            - 4368495024
        *
          z
       +
                      3             2
            9713133306w  + 9678670317w  - 16726834476w
          +
            28144233593
        *
          u
       }
     ]
Type: List SquareFreeRegularTriangularSet(Integer,IndexedExponents OrderedVariableList [b1,x,y,z,t,v,u,w],Or
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LazardSetSolvingPackageXmpPageEmpty36}
\begin{paste}{LazardSetSolvingPackageXmpPageEmpty36}{LazardSetSolvingPackageXmpPagePatch36}
\pastebutton{LazardSetSolvingPackageXmpPageEmpty36}{\showpaste}
```

```
\tab{5}\spadcommand{zeroSetSplit(lf,false)$pack\free{lf }\free{pack }}
\end{paste}\end{patch}
```

———-

3.60 lexp.ht

LieExponentials

— lexp.ht —

```
\begin{page}{LieExponentialsXmpPage}{LieExponentials}
\beginscroll
\xtc{
}{
\spadpaste{ a: Symbol := 'a \bound{a}}
}
\xtc{
}{
\spadpaste{ b: Symbol := 'b \bound{b}}
}

Declarations of domains
\xtc{
}{
\spadpaste{ coef     := Fraction(Integer) \bound{coef}}
}
\xtc{
}{
\spadpaste{ group    := LieExponentials(Symbol, coef, 3)
\free{coef} \bound{group}}
}
\xtc{
}{
\spadpaste{ lpoly    := LiePolynomial(Symbol, coef)
\free{coef} \bound{lpoly}}
}
\xtc{
}{
\spadpaste{ poly     := XPBWPolynomial(Symbol, coef)
\free{coef} \bound{poly}}
}

Calculations
\xtc{
}{
```

```
\spadpaste{ ea := exp(a::lpoly)$group \free{a} \free{lpoly}
\free{group} \bound{ea}}
}
\xtc{
}{
\spadpaste{ eb := exp(b::lpoly)$group \free{b} \free{lpoly}
\free{group}  \bound{eb}}
}
\xtc{
}{
\spadpaste{ g: group := ea*eb \free{ea} \free{eb} \bound{g}}
}
\xtc{
}{
\spadpaste{ g :: poly \free{g} \free{poly}}
}
\xtc{
}{
\spadpaste{ log(g)$group \free{g} \free{group}}
}
\xtc{
}{
\spadpaste{ g1: group := inv(g) \free{g} \free{group} \bound{g1}}
}
\xtc{
}{
\spadpaste{ g*g1 \free{g} \free{g1}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{LieExponentialsXmpPagePatch1}
\begin{paste}{LieExponentialsXmpPageFull1}{LieExponentialsXmpPageEmpty1}
\pastebutton{LieExponentialsXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{ a: Symbol := 'a\bound{a }}
\indentrel{3}\begin{verbatim}
   (1)  a
                                            Type: Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LieExponentialsXmpPageEmpty1}
\begin{paste}{LieExponentialsXmpPageEmpty1}{LieExponentialsXmpPagePatch1}
\pastebutton{LieExponentialsXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{ a: Symbol := 'a\bound{a }}
\end{paste}\end{patch}

\begin{patch}{LieExponentialsXmpPagePatch2}
```

```
\begin{paste}{LieExponentialsXmpPageFull2}{LieExponentialsXmpPageEmpty2}
\pastebutton{LieExponentialsXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{ b: Symbol := 'b\bound{b }}
\indentrel{3}\begin{verbatim}
   (2)  b
                                          Type: Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LieExponentialsXmpPageEmpty2}
\begin{paste}{LieExponentialsXmpPageEmpty2}{LieExponentialsXmpPagePatch2}
\pastebutton{LieExponentialsXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{ b: Symbol := 'b\bound{b }}
\end{paste}\end{patch}

\begin{patch}{LieExponentialsXmpPagePatch3}
\begin{paste}{LieExponentialsXmpPageFull3}{LieExponentialsXmpPageEmpty3}
\pastebutton{LieExponentialsXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{ coef := Fraction(Integer)\bound{coef }}
\indentrel{3}\begin{verbatim}
   (3)  Fraction Integer
                                          Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LieExponentialsXmpPageEmpty3}
\begin{paste}{LieExponentialsXmpPageEmpty3}{LieExponentialsXmpPagePatch3}
\pastebutton{LieExponentialsXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{ coef := Fraction(Integer)\bound{coef }}
\end{paste}\end{patch}

\begin{patch}{LieExponentialsXmpPagePatch4}
\begin{paste}{LieExponentialsXmpPageFull4}{LieExponentialsXmpPageEmpty4}
\pastebutton{LieExponentialsXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{ group := LieExponentials(Symbol, coef, 3)\free{coef }\bound{group }}
\indentrel{3}\begin{verbatim}
   (4)  LieExponentials(Symbol,Fraction Integer,3)
                                          Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LieExponentialsXmpPageEmpty4}
\begin{paste}{LieExponentialsXmpPageEmpty4}{LieExponentialsXmpPagePatch4}
\pastebutton{LieExponentialsXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{ group := LieExponentials(Symbol, coef, 3)\free{coef }\bound{group }}
\end{paste}\end{patch}

\begin{patch}{LieExponentialsXmpPagePatch5}
\begin{paste}{LieExponentialsXmpPageFull5}{LieExponentialsXmpPageEmpty5}
\pastebutton{LieExponentialsXmpPageFull5}{\hidepaste}
```

```
\tab{5}\spadcommand{ lpoly := LiePolynomial(Symbol, coef)\free{coef }\bound{lpoly }}
\indentrel{3}\begin{verbatim}
   (5)  LiePolynomial(Symbol,Fraction Integer)
                                           Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LieExponentialsXmpPageEmpty5}
\begin{paste}{LieExponentialsXmpPageEmpty5}{LieExponentialsXmpPagePatch5}
\pastebutton{LieExponentialsXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{ lpoly := LiePolynomial(Symbol, coef)\free{coef }\bound{lpoly }}
\end{paste}\end{patch}

\begin{patch}{LieExponentialsXmpPagePatch6}
\begin{paste}{LieExponentialsXmpPageFull6}{LieExponentialsXmpPageEmpty6}
\pastebutton{LieExponentialsXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{ poly := XPBWPolynomial(Symbol, coef)\free{coef }\bound{poly }}
\indentrel{3}\begin{verbatim}
   (6)  XPBWPolynomial(Symbol,Fraction Integer)
                                           Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LieExponentialsXmpPageEmpty6}
\begin{paste}{LieExponentialsXmpPageEmpty6}{LieExponentialsXmpPagePatch6}
\pastebutton{LieExponentialsXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{ poly := XPBWPolynomial(Symbol, coef)\free{coef }\bound{poly }}
\end{paste}\end{patch}

\begin{patch}{LieExponentialsXmpPagePatch7}
\begin{paste}{LieExponentialsXmpPageFull7}{LieExponentialsXmpPageEmpty7}
\pastebutton{LieExponentialsXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{ ea := exp(a::lpoly)$group\free{a }\free{lpoly }\free{group }\bound{ea }}
\indentrel{3}\begin{verbatim}
         [a]
   (7)  e
       Type: LieExponentials(Symbol,Fraction Integer,3)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LieExponentialsXmpPageEmpty7}
\begin{paste}{LieExponentialsXmpPageEmpty7}{LieExponentialsXmpPagePatch7}
\pastebutton{LieExponentialsXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{ ea := exp(a::lpoly)$group\free{a }\free{lpoly }\free{group }\bound{ea }}
\end{paste}\end{patch}

\begin{patch}{LieExponentialsXmpPagePatch8}
\begin{paste}{LieExponentialsXmpPageFull8}{LieExponentialsXmpPageEmpty8}
\pastebutton{LieExponentialsXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{ eb := exp(b::lpoly)$group\free{b }\free{lpoly }\free{group }\bound{eb }}
```

```
\indentrel{3}\begin{verbatim}
         [b]
   (8)  e
       Type: LieExponentials(Symbol,Fraction Integer,3)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LieExponentialsXmpPageEmpty8}
\begin{paste}{LieExponentialsXmpPageEmpty8}{LieExponentialsXmpPagePatch8}
\pastebutton{LieExponentialsXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{ eb := exp(b::lpoly)$group\free{b }\free{lpoly }\free{group }\bound{eb }}
\end{paste}\end{patch}

\begin{patch}{LieExponentialsXmpPagePatch9}
\begin{paste}{LieExponentialsXmpPageFull9}{LieExponentialsXmpPageEmpty9}
\pastebutton{LieExponentialsXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{ g: group := ea*eb\free{ea }\free{eb }\bound{g }}
\indentrel{3}\begin{verbatim}
             1     2        1   2
              [a b ]        [a b]
         [b] 2        [a b] 2       [a]
   (9)  e   e        e     e       e
       Type: LieExponentials(Symbol,Fraction Integer,3)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LieExponentialsXmpPageEmpty9}
\begin{paste}{LieExponentialsXmpPageEmpty9}{LieExponentialsXmpPagePatch9}
\pastebutton{LieExponentialsXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{ g: group := ea*eb\free{ea }\free{eb }\bound{g }}
\end{paste}\end{patch}

\begin{patch}{LieExponentialsXmpPagePatch10}
\begin{paste}{LieExponentialsXmpPageFull10}{LieExponentialsXmpPageEmpty10}
\pastebutton{LieExponentialsXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{ g :: poly\free{g }\free{poly }}
\indentrel{3}\begin{verbatim}
   (10)
                     1                            1
     1 + [a] + [b] +  [a][a] + [a b] + [b][a] +  [b][b]
                     2                            2
   +
     1             1   2                1     2
      [a][a][a] +  [a b] + [a b][a] +  [a b ]
     6             2                    2
   +
     1                        1             1
      [b][a][a] + [b][a b] +  [b][b][a] +  [b][b][b]
     2                        2             6
          Type: XPBWPolynomial(Symbol,Fraction Integer)
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LieExponentialsXmpPageEmpty10}
\begin{paste}{LieExponentialsXmpPageEmpty10}{LieExponentialsXmpPagePatch10}
\pastebutton{LieExponentialsXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{ g :: poly\free{g }\free{poly }}
\end{paste}\end{patch}

\begin{patch}{LieExponentialsXmpPagePatch11}
\begin{paste}{LieExponentialsXmpPageFull11}{LieExponentialsXmpPageEmpty11}
\pastebutton{LieExponentialsXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{ log(g)$group\free{g }\free{group }}
\indentrel{3}\begin{verbatim}
                     1          1   2      1     2
   (11)  [a] + [b] +  [a b] +  [a b] +  [a b ]
                     2         12         12
           Type: LiePolynomial(Symbol,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LieExponentialsXmpPageEmpty11}
\begin{paste}{LieExponentialsXmpPageEmpty11}{LieExponentialsXmpPagePatch11}
\pastebutton{LieExponentialsXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{ log(g)$group\free{g }\free{group }}
\end{paste}\end{patch}

\begin{patch}{LieExponentialsXmpPagePatch12}
\begin{paste}{LieExponentialsXmpPageFull12}{LieExponentialsXmpPageEmpty12}
\pastebutton{LieExponentialsXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{ g1: group := inv(g)\free{g }\free{group }\bound{g1 }}
\indentrel{3}\begin{verbatim}
          - [b] - [a]
   (12)  e     e
       Type: LieExponentials(Symbol,Fraction Integer,3)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LieExponentialsXmpPageEmpty12}
\begin{paste}{LieExponentialsXmpPageEmpty12}{LieExponentialsXmpPagePatch12}
\pastebutton{LieExponentialsXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{ g1: group := inv(g)\free{g }\free{group }\bound{g1 }}
\end{paste}\end{patch}

\begin{patch}{LieExponentialsXmpPagePatch13}
\begin{paste}{LieExponentialsXmpPageFull13}{LieExponentialsXmpPageEmpty13}
\pastebutton{LieExponentialsXmpPageFull13}{\hidepaste}
\tab{5}\spadcommand{ g*g1\free{g }\free{g1 }}
\indentrel{3}\begin{verbatim}
   (13)  1
```

```
      Type: LieExponentials(Symbol,Fraction Integer,3)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LieExponentialsXmpPageEmpty13}
\begin{paste}{LieExponentialsXmpPageEmpty13}{LieExponentialsXmpPagePatch13}
\pastebutton{LieExponentialsXmpPageEmpty13}{\showpaste}
\tab{5}\spadcommand{ g*g1\free{g }\free{g1 }}
\end{paste}\end{patch}
```

———

3.61 lextripk.ht

LexTriangularPackage

— lextripk.ht —

```
\begin{page}{LexTriangularPkgXmpPage}{LexTriangularPackage}
\beginscroll
The \spadtype{LexTriangularPackage} package constructor provides
an implementation of the {\em lexTriangular} algorithm
(D. Lazard "Solving Zero-dimensional Algebraic Systems",
J. of Symbol. Comput., 1992).
This algorithm decomposes a zero-dimensional variety into zero-sets of
regular triangular sets.  Thus the input system must have a finite
number of complex solutions.  Moreover, this system needs to be a
lexicographical Groebner basis.

This package takes two arguments: the coefficient-ring {\bf R} of the
polynomials, which must be a \spadtype{GcdDomain} and their set of
variables given by {\bf ls} a \spadtype{List Symbol}.  The type of the
input polynomials must be
\spadtype{NewSparseMultivariatePolynomial(R,V)} where {\bf V} is
\spadtype{OrderedVariableList(ls)}.  The abbreviation for
\spadtype{LexTriangularPackage} is \spadtype{LEXTRIPK}.  The main
operations are \axiomOpFrom{lexTriangular}{LexTriangularPackage} and
\axiomOpFrom{squareFreeLexTriangular}{LexTriangularPackage}.  The
later provide decompositions by means of square-free regular
triangular sets, built with the \spadtype{SREGSET} constructor,
whereas the former uses the \spadtype{REGSET} constructor.  Note that
these constructors also implement another algorithm for solving
algebraic systems by means of regular triangular sets; in that case no
computations of Groebner bases are needed and the input system may
have any dimension (i.e. it may have an infinite number of solutions).
```

```
The implementation of the {\em lexTriangular} algorithm provided in
the \spadtype{LexTriangularPackage} constructor differs from that
reported in "Computations of gcd over algebraic towers of simple
extensions" by M. Moreno Maza and R. Rioboo (in proceedings of
AAECC11, Paris, 1995).  Indeed, the
\axiomOpFrom{squareFreeLexTriangular}{LexTriangularPackage} operation
removes all multiplicities of the solutions (i.e. the computed
solutions are pairwise different) and the
\axiomOpFrom{lexTriangular}{LexTriangularPackage} operation may keep
some multiplicities; this latter operation runs generally faster than
the former.

The interest of the {\em lexTriangular} algorithm is due to the
following experimental remark.  For some examples, a triangular
decomposition of a zero-dimensional variety can be computed faster via
a lexicographical Groebner basis computation than by using a direct
method (like that of \spadtype{SREGSET} and \spadtype{REGSET}).  This
happens typically when the total degree of the system relies
essentially on its smallest variable (like in the {\em Katsura}
systems).  When this is not the case, the direct method may give
better timings (like in the {\em Rose} system).

Of course, the direct method can also be applied to a lexicographical
Groebner basis.  However, the {\em lexTriangular} algorithm takes
advantage of the structure of this basis and avoids many unnecessary
computations which are performed by the direct method.

For this purpose of solving algebraic systems with a finite number of
solutions, see also the \spadtype{ZeroDimensionalSolvePackage}.  It
allows to use both strategies (the lexTriangular algorithm and the
direct method) for computing either the complex or real roots of a
system.

Note that the way of understanding triangular decompositions
is detailed in the example of the \spadtype{RegularTriangularSet}
constructor.

Since the \spadtype{LEXTRIPK} package constructor is limited to
zero-dimensional systems, it provides a
\axiomOpFrom{zeroDimensional?}{LexTriangularPackage} operation to
check whether this requirement holds.  There is also a
\axiomOpFrom{groebner}{LexTriangularPackage} operation to compute the
lexicographical Groebner basis of a set of polynomials with type
\spadtype{NewSparseMultivariatePolynomial(R,V)}.  The elimination
ordering is that given by {\bf ls} (the greatest variable being the
first element of {\bf ls}).  This basis is computed by the {\em FLGM}
algorithm (Faugere et al. "Efficient Computation of Zero-Dimensional
Groebner Bases by Change of Ordering" , J. of Symbol. Comput., 1993)
implemented in the \spadtype{LinGroebnerPackage} package constructor.
Once a lexicographical Groebner basis is computed,
```

```
then one can call the operations
\axiomOpFrom{lexTriangular}{LexTriangularPackage}
and \axiomOpFrom{squareFreeLexTriangular}{LexTriangularPackage}.
Note that these operations admit an optional argument
to produce normalized triangular sets.
There is also a \axiomOpFrom{zeroSetSplit}{LexTriangularPackage}
operation which does all the job from the input system;
an error is produced if this system is not zero-dimensional.

Let us illustrate the facilities of the \spadtype{LEXTRIPK} constructor
by a famous example, the {\em cyclic-6 root} system.
\xtc{
Define the coefficient ring.
}{
\spadpaste{R := Integer \bound{R}}
}
\xtc{
Define the list of variables,
}{
\spadpaste{ls : List Symbol := [a,b,c,d,e,f] \bound{ls}}
}
\xtc{
and make it an ordered set.
}{
\spadpaste{V := OVAR(ls) \free{ls} \bound{V}}
}
\xtc{
Define the polynomial ring.
}{
\spadpaste{P := NSMP(R, V) \free{R} \free{V} \bound{P}}
}
\xtc{
Define the polynomials.
}{
\spadpaste{p1: P :=  a*b*c*d*e*f - 1 \free{P} \bound{p1}}
}
\xtc{
}{
\spadpaste{p2: P := a*b*c*d*e +a*b*c*d*f +a*b*c*e*f +a*b*d*e*f +
a*c*d*e*f +b*c*d*e*f  \free{P} \bound{p2}}
}
\xtc{
}{
\spadpaste{p3: P :=  a*b*c*d + a*b*c*f + a*b*e*f + a*d*e*f +
b*c*d*e + c*d*e*f \free{P} \bound{p3}}
}
\xtc{
}{
\spadpaste{p4: P := a*b*c + a*b*f + a*e*f + b*c*d + c*d*e + d*e*f
\free{P} \bound{p4}}
```

```
}
\xtc{
}{
\spadpaste{p5: P := a*b + a*f + b*c + c*d + d*e + e*f \free{P}
\bound{p5}}
}
\xtc{
}{
\spadpaste{p6: P := a + b + c + d + e + f  \free{P} \bound{p6}}
}
\xtc{
}{
\spadpaste{lp := [p1, p2, p3, p4, p5, p6] \free{p1} \free{p2}
\free{p3} \free{p4} \free{p5} \free{p6} \bound{lp}}
}
\xtc{
Now call \spadtype{LEXTRIPK} .
}{
\spadpaste{lextripack :=  LEXTRIPK(R,ls) \free{R} \free{ls}
\bound{lextripack}}
}
\xtc{
Compute the lexicographical Groebner basis of the system.
This may take between 5 minutes and one hour, depending on your machine.
}{
\spadpaste{lg := groebner(lp)$lextripack \free{lp}
\free{lextripack} \bound{lg}}
}
\xtc{
Apply lexTriangular to compute a decomposition into regular triangular sets.
This should not take more than 5 seconds.
}{
\spadpaste{lexTriangular(lg,false)$lextripack \free{lg}
\free{lextripack}}
}
Note that the first set of the decomposition is normalized
(all initials are integer numbers) but not the second one
(normalized triangular sets are defined in the
description of the \spadtype{NormalizedTriangularSetCategory}
constructor).
\xtc{
So apply now lexTriangular to produce normalized triangular sets.
}{
\spadpaste{lts := lexTriangular(lg,true)$lextripack \free{lg}
\free{lextripack} \bound{lts}}
}
\xtc{
We check that all initials are constant.
}{
\spadpaste{[[init(p) for p in (ts :: List(P))] for ts in lts]
```

```
\free{lts}}
}
Note that each triangular set in {\bf lts} is a lexicographical
Groebner basis.
Recall that a point belongs to the variety
associated with {\bf lp} if and only if
it belongs to that associated with one triangular set {\bf ts}
in {\bf lts}.

By running the
\axiomOpFrom{squareFreeLexTriangular}{LexTriangularPackage} operation,
we retrieve the above decomposition.
\xtc{
}{
\spadpaste{squareFreeLexTriangular(lg,true)$lextripack \free{lg}
\free{lextripack}}
}
Thus the solutions given by {\bf lts} are pairwise different.
\xtc{
We count them as follows.
}{
\spadpaste{reduce(+,[degree(ts) for ts in lts]) \free{lts}}
}

We can investigate the triangular decomposition {\bf lts} by
using the \spadtype{ZeroDimensionalSolvePackage}.
\xtc{
This requires to add an extra variable (smaller than the others) as follows.
}{
\spadpaste{ls2 : List Symbol := concat(ls,new()$Symbol) \free{ls}
\bound{ls2}}
}

\xtc{
Then we call the package.
}{
\spadpaste{zdpack := ZDSOLVE(R,ls,ls2) \free{R} \free{ls}
\free{ls2} \bound{zdpack}}
}


\xtc{
We compute a univariate representation of the variety associated with
the input system as follows.
}{
\spadpaste{concat [univariateSolve(ts)$zdpack for ts in lts]
\free{lts} \free{zdpack}}
}
Since the
\axiomOpFrom{univariateSolve}{ZeroDimensionalSolvePackage} operation may
```

```
split a regular set, it returns a list. This explains the use
of \axiomOpFrom{concat}{List}.

Look at the last item of the result. It consists of two parts.  For
any complex root {\bf ?} of the univariate polynomial in the first
part, we get a tuple of univariate polynomials (in {\bf a}, ...,
{\bf f} respectively) by replacing {\bf \%A} by {\bf ?} in the second
part. Each of these tuples {\bf t} describes a point of the variety
associated with {\bf lp} by equaling to zero the polynomials in
{\bf t}.

Note that the way of reading these univariate representations is
explained also in the example illustrating the
\spadtype{ZeroDimensionalSolvePackage} constructor.

\xtc{
Now, we compute the points of the variety with real coordinates.
}{
\spadpaste{concat [realSolve(ts)$zdpack for ts in lts] \free{lts}
\free{zdpack}}
}
We obtain 24 points given by lists of elements in the
\spadtype{RealClosure} of \spadtype{Fraction} of {\bf R}.
In each list, the first value corresponds to the indeterminate {\bf f},
the second to {\bf e} and so on.
See \spadtype{ZeroDimensionalSolvePackage} to learn more about
the \axiomOpFrom{realSolve}{ZeroDimensionalSolvePackage} operation.

\endscroll
\autobuttons
\end{page}

\begin{patch}{LexTriangularPackageXmpPagePatch1}
\begin{paste}{LexTriangularPackageXmpPageFull1}{LexTriangularPackageXmpPageEmpty1}
\pastebutton{LexTriangularPackageXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{R := Integer\bound{R }}
\indentrel{3}\begin{verbatim}
   (1)  Integer
                                           Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LexTriangularPackageXmpPageEmpty1}
\begin{paste}{LexTriangularPackageXmpPageEmpty1}{LexTriangularPackageXmpPagePatch1}
\pastebutton{LexTriangularPackageXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{R := Integer\bound{R }}
\end{paste}\end{patch}

\begin{patch}{LexTriangularPackageXmpPagePatch2}
\begin{paste}{LexTriangularPackageXmpPageFull2}{LexTriangularPackageXmpPageEmpty2}
```

```
\pastebutton{LexTriangularPackageXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{ls : List Symbol := [a,b,c,d,e,f]\bound{ls }}
\indentrel{3}\begin{verbatim}
   (2)  [a,b,c,d,e,f]
                                       Type: List Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LexTriangularPackageXmpPageEmpty2}
\begin{paste}{LexTriangularPackageXmpPageEmpty2}{LexTriangularPackageXmpPagePatch2}
\pastebutton{LexTriangularPackageXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{ls : List Symbol := [a,b,c,d,e,f]\bound{ls }}
\end{paste}\end{patch}

\begin{patch}{LexTriangularPackageXmpPagePatch3}
\begin{paste}{LexTriangularPackageXmpPageFull3}{LexTriangularPackageXmpPageEmpty3}
\pastebutton{LexTriangularPackageXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{V := OVAR(ls)\free{ls }\bound{V }}
\indentrel{3}\begin{verbatim}
   (3)  OrderedVariableList [a,b,c,d,e,f]
                                            Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LexTriangularPackageXmpPageEmpty3}
\begin{paste}{LexTriangularPackageXmpPageEmpty3}{LexTriangularPackageXmpPagePatch3}
\pastebutton{LexTriangularPackageXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{V := OVAR(ls)\free{ls }\bound{V }}
\end{paste}\end{patch}

\begin{patch}{LexTriangularPackageXmpPagePatch4}
\begin{paste}{LexTriangularPackageXmpPageFull4}{LexTriangularPackageXmpPageEmpty4}
\pastebutton{LexTriangularPackageXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{P := NSMP(R, V)\free{R }\free{V }\bound{P }}
\indentrel{3}\begin{verbatim}
   (4)
  NewSparseMultivariatePolynomial(Integer,OrderedVariable
  List [a,b,c,d,e,f])
                                            Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LexTriangularPackageXmpPageEmpty4}
\begin{paste}{LexTriangularPackageXmpPageEmpty4}{LexTriangularPackageXmpPagePatch4}
\pastebutton{LexTriangularPackageXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{P := NSMP(R, V)\free{R }\free{V }\bound{P }}
\end{paste}\end{patch}

\begin{patch}{LexTriangularPackageXmpPagePatch5}
\begin{paste}{LexTriangularPackageXmpPageFull5}{LexTriangularPackageXmpPageEmpty5}
```

```
\pastebutton{LexTriangularPackageXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{p1: P := a*b*c*d*e*f - 1\free{P }\bound{p1 }}
\indentrel{3}\begin{verbatim}
   (5)  f e d c b a - 1
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [a,b,c,d,e,f])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LexTriangularPackageXmpPageEmpty5}
\begin{paste}{LexTriangularPackageXmpPageEmpty5}{LexTriangularPackageXmpPagePatch5}
\pastebutton{LexTriangularPackageXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{p1: P := a*b*c*d*e*f - 1\free{P }\bound{p1 }}
\end{paste}\end{patch}

\begin{patch}{LexTriangularPackageXmpPagePatch6}
\begin{paste}{LexTriangularPackageXmpPageFull6}{LexTriangularPackageXmpPageEmpty6}
\pastebutton{LexTriangularPackageXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{p2: P := a*b*c*d*e +a*b*c*d*f +a*b*c*e*f +a*b*d*e*f +a*c*d*e*f +b*c*d*e*f\free{P }\bound
\indentrel{3}\begin{verbatim}
   (6)
   ((((e + f)d + f e)c + f e d)b + f e d c)a + f e d c b
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [a,b,c,d,e,f])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LexTriangularPackageXmpPageEmpty6}
\begin{paste}{LexTriangularPackageXmpPageEmpty6}{LexTriangularPackageXmpPagePatch6}
\pastebutton{LexTriangularPackageXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{p2: P := a*b*c*d*e +a*b*c*d*f +a*b*c*e*f +a*b*d*e*f +a*c*d*e*f +b*c*d*e*f\free{P }\bound
\end{paste}\end{patch}

\begin{patch}{LexTriangularPackageXmpPagePatch7}
\begin{paste}{LexTriangularPackageXmpPageFull7}{LexTriangularPackageXmpPageEmpty7}
\pastebutton{LexTriangularPackageXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{p3: P := a*b*c*d + a*b*c*f + a*b*e*f + a*d*e*f + b*c*d*e + c*d*e*f\free{P }\bound{p3 }}
\indentrel{3}\begin{verbatim}
   (7)  (((d + f)c + f e)b + f e d)a + e d c b + f e d c
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [a,b,c,d,e,f])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LexTriangularPackageXmpPageEmpty7}
\begin{paste}{LexTriangularPackageXmpPageEmpty7}{LexTriangularPackageXmpPagePatch7}
\pastebutton{LexTriangularPackageXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{p3: P := a*b*c*d + a*b*c*f + a*b*e*f + a*d*e*f + b*c*d*e + c*d*e*f\free{P }\bound{p3 }}
\end{paste}\end{patch}

\begin{patch}{LexTriangularPackageXmpPagePatch8}
\begin{paste}{LexTriangularPackageXmpPageFull8}{LexTriangularPackageXmpPageEmpty8}
\pastebutton{LexTriangularPackageXmpPageFull8}{\hidepaste}
```

```
\tab{5}\spadcommand{p4: P := a*b*c + a*b*f + a*e*f + b*c*d + c*d*e + d*e*f\free{P }\bound{p4 }}
\indentrel{3}\begin{verbatim}
   (8)  ((c + f)b + f e)a + d c b + e d c + f e d
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [a,b,c,d,e,f])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LexTriangularPackageXmpPageEmpty8}
\begin{paste}{LexTriangularPackageXmpPageEmpty8}{LexTriangularPackageXmpPagePatch8}
\pastebutton{LexTriangularPackageXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{p4: P := a*b*c + a*b*f + a*e*f + b*c*d + c*d*e + d*e*f\free{P }\bound{p4 }}
\end{paste}\end{patch}

\begin{patch}{LexTriangularPackageXmpPagePatch9}
\begin{paste}{LexTriangularPackageXmpPageFull9}{LexTriangularPackageXmpPageEmpty9}
\pastebutton{LexTriangularPackageXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{p5: P := a*b + a*f + b*c + c*d + d*e + e*f\free{P }\bound{p5 }}
\indentrel{3}\begin{verbatim}
   (9)  (b + f)a + c b + d c + e d + f e
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [a,b,c,d,e,f])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LexTriangularPackageXmpPageEmpty9}
\begin{paste}{LexTriangularPackageXmpPageEmpty9}{LexTriangularPackageXmpPagePatch9}
\pastebutton{LexTriangularPackageXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{p5: P := a*b + a*f + b*c + c*d + d*e + e*f\free{P }\bound{p5 }}
\end{paste}\end{patch}

\begin{patch}{LexTriangularPackageXmpPagePatch10}
\begin{paste}{LexTriangularPackageXmpPageFull10}{LexTriangularPackageXmpPageEmpty10}
\pastebutton{LexTriangularPackageXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{p6: P := a + b + c + d + e + f\free{P }\bound{p6 }}
\indentrel{3}\begin{verbatim}
   (10)  a + b + c + d + e + f
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [a,b,c,d,e,f])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LexTriangularPackageXmpPageEmpty10}
\begin{paste}{LexTriangularPackageXmpPageEmpty10}{LexTriangularPackageXmpPagePatch10}
\pastebutton{LexTriangularPackageXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{p6: P := a + b + c + d + e + f\free{P }\bound{p6 }}
\end{paste}\end{patch}

\begin{patch}{LexTriangularPackageXmpPagePatch11}
\begin{paste}{LexTriangularPackageXmpPageFull11}{LexTriangularPackageXmpPageEmpty11}
\pastebutton{LexTriangularPackageXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{lp := [p1, p2, p3, p4, p5, p6]\free{p1 }\free{p2 }\free{p3 }\free{p4 }\free{
\indentrel{3}\begin{verbatim}
```

```
   (11)
   [f e d c b a - 1,
    ((((e + f)d + f e)c + f e d)b + f e d c)a + f e d c b,
    (((d + f)c + f e)b + f e d)a + e d c b + f e d c,
    ((c + f)b + f e)a + d c b + e d c + f e d,
    (b + f)a + c b + d c + e d + f e,
    a + b + c + d + e + f]
Type: List NewSparseMultivariatePolynomial(Integer,OrderedVariableList [a,b,c,d,e,f])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LexTriangularPackageXmpPageEmpty11}
\begin{paste}{LexTriangularPackageXmpPageEmpty11}{LexTriangularPackageXmpPagePatch11}
\pastebutton{LexTriangularPackageXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{lp := [p1, p2, p3, p4, p5, p6]\free{p1 }\free{p2 }\free{p3 }\free{p4 }\free{p5 }\free{p6
\end{paste}\end{patch}

\begin{patch}{LexTriangularPackageXmpPagePatch12}
\begin{paste}{LexTriangularPackageXmpPageFull12}{LexTriangularPackageXmpPageEmpty12}
\pastebutton{LexTriangularPackageXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{lextripack := LEXTRIPK(R,ls)\free{R }\free{ls }\bound{lextripack }}
\indentrel{3}\begin{verbatim}
   (12)  LexTriangularPackage(Integer,[a,b,c,d,e,f])
                                           Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LexTriangularPackageXmpPageEmpty12}
\begin{paste}{LexTriangularPackageXmpPageEmpty12}{LexTriangularPackageXmpPagePatch12}
\pastebutton{LexTriangularPackageXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{lextripack := LEXTRIPK(R,ls)\free{R }\free{ls }\bound{lextripack }}
\end{paste}\end{patch}

\begin{patch}{LexTriangularPackageXmpPagePatch13}
\begin{paste}{LexTriangularPackageXmpPageFull13}{LexTriangularPackageXmpPageEmpty13}
\pastebutton{LexTriangularPackageXmpPageFull13}{\hidepaste}
\tab{5}\spadcommand{lg := groebner(lp)$lextripack\free{lp }\free{lextripack }\bound{lg }}
\indentrel{3}\begin{verbatim}
   (13)
   [a + b + c + d + e + f,

                        2
      3968379498283200b  + 15873517993132800f b
    +
                        2
      3968379498283200d  + 15873517993132800f d
    +
                        3 5                     4 4
      3968379498283200f e  - 15873517993132800f e
    +
```

```
                    5 3
  23810276989699200f e
+
                      6                        2
  (206355733910726400f  + 230166010900425600)e
+
                          43                         37
      - 729705987316687f   + 1863667496867205421f
    +
                              31
      2916748537717311044461f
    +
                              25
      3652859946911069217745f
    +
                          19
      549961185828911895f
    +
                                13
      - 3650484040387684392 69f
    +
                              7
    - 2923828204315040276 69f  - 2271898467631865497f
   *
    e
+
                     44                           38
  - 3988812642545399f   + 10187423878429609997f
+
                          32
  15943775234243140536 37f
+
                          26                         20
  19947393084399162380 65f   + 1596840088052642815f
+
                            14
  - 19934941183011621454 13f
+
                          8                          2
- 15960497422896898150 53f  - 1148817133015966 7449f
,

  (23810276989699200c - 23810276989699200f)b
+
                    2
  23810276989699200c  + 71430830969097600f c
+
                      2
  - 23810276989699200d  - 95241107958796800f d
+
```

```
                       3 5                        4 4
  - 55557312975964800f e  + 174608697924460800f e
+
                        5 3
  - 174608697924460800f e
+
                          6                         2
  (- 2428648252949318400f  - 2611193709870345600)e
+
                        43                        37
      8305444561289527f   - 21212087151945459641f
    +
                                31
      - 331981588309345138538lf
    +
                                25
      - 415769164626165713644 5f
    +
                             19
      - 6072721607510764095f
    +
                              13
      41549867090364602216 49f
    +
                            7
    3327761311138587096749f  + 25885340608290841637f
 *
    e
+
                    44                         38
  45815897629010329f   - 117013765582151891207f
+
                             32
  - 18313166848970865074187f
+
                             26
  - 22909971239649297438915f
+
                          20
  - 1613325076130515726 5f
+
                           14
  2289730585763617825 6623f
+
                         8                         2
18329944781867242497923f  + 130258531002020420699f
,

  (7936758996566400d - 7936758996566400f)b
+
```

```
                                                 3 5
   - 7936758996566400f d - 7936758996566400f e
 +
                     4 4                       5 3
   23810276989699200f e  - 23810276989699200f e
 +
                         6                       2
   (- 337312257354072000f  - 369059293340337600)e
 +
                         43                       37
       1176345388640471f   - 3004383582891473073f
     +
                                31
       - 470203502707246105653f
     +
                                25
       - 588858183402644348085f
     +
                             19
       - 856939308623513535f
     +
                              13
       588472674242340526377f
     +
                              7
       471313241958371103517f  + 3659742549078552381f
  *
     e
 +
                    44                       38
   6423170513956901f   - 16404772137036480803f
 +
                             32
   - 2567419165227528774463f
 +
                             26
   - 3211938090825682172335f
 +
                          20
   - 2330490332697587485f
 +
                           14
   3210100109444754864587f
 +
                           8                         2
   2569858315395162617847f  + 1832608948742773575lf
 ,

   (11905138494849600e - 11905138494849600f)b
 +
```

```
                         3 5                     4 4
   - 3968379498283200f e  + 15873517993132800f e
  +
                          5 3
   - 27778656487982400f e
  +
                            6                        2
   (- 208339923659868000f  - 240086959646133600)e
  +
                          43                        37
       786029984751110f   - 2007519008182245250f
     +
                                  31
       - 3141880629080738070 90f
     +
                                  25
       - 3934236675379295752 50f
     +
                               19
       - 550329120654394950f
     +
                                13
       393196408728889612770f
     +
                                7
       314892372799176495730f  + 240938651514666853 0f
   *
     e
  +
                       44                         38
   4177638546747827f   - 1066968529460257638 1f
  +
                               32
   - 1669852980419949524601f
  +
                               26
   - 2089077057287904170745f
  +
                            20
   - 1569899763580278795f
  +
                             14
   2087864026859015573349f
  +
                             8                         2
   1671496085945199577969f  + 1194025722621628017 7f
  ,

                         6
   (11905138494849600f  - 11905138494849600)b
```

```
+
                      2 5                     3 4
  - 15873517993132800f e  + 39683794982832000f e
+
                      4 3
  - 39683794982832000f e
+
                        11                      5  2
  (- 686529653202993600f   - 607162063237329600f )e
+
                     42                        36
      65144531306704f   - 166381280901088652f
    +
                              30
      - 26033434502470283472f
    +
                              24
      - 31696259583860650140f
    +
                          18                        12
      971492093167581360f   + 32220085033691389548f
    +
                            6
      25526177666070529808f  + 138603268355749244
   *
    e
+
                  43                        37
  167620036074811f   - 428102417974791473f
+
                          31                        25
  - 66997243801231679313f   - 83426716722148750485f
+
                      19                        13
  203673895369980765f   + 83523056326010432457f
+
                        7
  66995789640238066937f  + 478592855549587901f
,

              3                    2
  801692827936c  + 2405078483808f c
+
                 2                 45
  - 2405078483808f c - 13752945467f
+
                 39                  33
  35125117815561f   + 5496946957826433f
+
                  27                  21
```

```
  6834659447749117f   - 44484880462461f
+
                    15                     9
  - 6873406230093057f   - 5450844938762633f
+
              3
  1216586044571f
,

  (23810276989699200d - 23810276989699200f)c
+
                    2
  23810276989699200d  + 71430830969097600f d
+
                  3 5                     4 4
  7936758996566400f e  - 31747035986265600f e
+
                    5 3
  31747035986265600f e
+
                      6                       2
  (404774708824886400f  + 396837949828320000)e
+
                          43                        37
      - 1247372229446701f   + 3185785654596621203f
    +
                             31
      4985948668499747751463f
    +
                             25
      6245425458457910479 35f
    +
                          19
      931085755769682885f
    +
                               13
      - 624150663582417063387f
    +
                             7
    - 4998818593883604756 47f  - 3926885313819527351f
 *
    e
+
                    44                        38
  - 7026011547118141f   + 1794442705195069 1243f
+
                        32
  28083835225939866 03543f
+
                        26                        20
```

```
  3513624142354807530135f   + 2860757006705537685f
+
                           14
  - 3511356735642190737267f
+
                         8                        2
- 2811332494697103819887f  - 2031501163152284731 1f
,

  (7936758996566400e - 7936758996566400f)c
+
                      43                     37
      - 4418748183673f   + 11285568707456559f
    +
                          31                       25
      1765998617294451019f   + 2173749283622606155f
    +
                          19                       13
      - 55788292195402895f   - 2215291421788292951f
    +
                            7
      - 1718142665347430851f  + 30256569458230237f
 *
    e
+
                44                     38
  4418748183673f   - 11285568707456559f
+
                        32                       26
  - 1765998617294451019f   - 2173749283622606155f
+
                    20                       14
  55788292195402895f   + 2215291421788292951f
+
                      8                     2
  1718142665347430851f  - 30256569458230237f
,

                  6
  (72152354514240f  - 72152354514240)c
+
              43                    37
  40950859449f   - 104588980990367f
+
                      31                     25
  - 16367227395575307f   - 20268523416527355f
+
                  19                     13
  442205002259535f   + 20576059935789063f
+
```

```
                     7
  15997133796970563f  - 275099892785581f
,

                    3                       2
  1984189749141600d  + 5952569247424800f d
+
                      2                      4 5
  - 5952569247424800f d - 3968379498283200f e
+
                     5 4                     3
  15873517993132800f e  + 17857707742274400e
+
                         7                          2
  (- 148814231185620000f  - 162703559429611200f)e
+
                         44                        38
      - 390000914678878f    + 996062704593756434f
    +
                             32
      155886323972034823914f
    +
                             26                     20
      194745956143985421330f    + 6205077595574430f
    +
                               14
      - 194596512653299068786f
    +
                             8                        2
    - 155796897940756922666f  - 1036375759077320978f
 *
    e
+
                     45                       39
  - 374998630035991f    + 957747106595453993f
+
                         33                          27
  149889155566764891693f    + 187154171443494641685f
+
                         21                          15
  - 127129015426348065f    - 187241533243115040417f
+
                           9                       3
  - 149719983567976534037f  - 836654081239648061f
,

  (5952569247424800e - 5952569247424800f)d
+
                      3 5                     4 4
  - 3968379498283200f e  + 9920948745708000f e
```

```
+
                    5 3
  - 3968379498283200f e
+
                       6                        2
  (- 148814231185620000f  - 150798420934761600)e
+
                     43                        37
      492558110242553f   - 1257992359608074599f
    +
                             31
      - 196883094539368513959f
    +
                             25
      - 246562115745735428055f
    +
                          19
      - 325698701993885505f
    +
                           13
      246417769883651808111f
    +
                          7
      197327352068200652911f  + 1523373796389332143f
 *
    e
+
                  44                        38
  2679481081803026f   - 6843392695421906608f
+
                          32
  - 1071020459642646913578f
+
                          26                        20
  - 1339789169692041240060f   - 852746750910750210f
+
                        14
  1339105101971878401312f
+
                         8                        2
  1071900289758712984762f  + 7555239072072727756f
,

                    6
  (11905138494849600f  - 11905138494849600)d
+
                    2 5                       3 4
  - 7936758996566400f e  + 31747035986265600f e
+
                     4 3
```

```
  - 31747035986265600f e
 +
                        11                        5  2
  (- 420648226818019200f   - 404774708824886400f )e
 +
                      42                       36
      15336187600889f   - 39169739565161107f
    +
                             30
      - 6127176127489690827f
    +
                             24
      - 7217708742310509615f
    +
                          18                         12
      538628483890722735f   + 7506804353843507643f
    +
                           6
      5886160769782607203f  + 63576108396535879
   *
    e
 +
                 43                       37
  71737781777066f   - 183218856207557938f
 +
                          31                         25
  - 28672874271132276078f   - 35625223686939812010f
 +
                      19                         13
  164831339634084390f   + 35724160423073052642f
 +
                        7
  28627022578664910622f  + 187459987029680506f
 ,

                   6                       5
  1322793166094400e  - 3968379498283200f e
 +
                   2 4                     3 3
  3968379498283200f e  - 5291172664377600f e
 +
                        10                        4  2
  (- 230166010900425600f   - 226197631402142400f )e
 +
                            47
      - 152375364610443885f
    +
                             41
      3891666260648548904 15f
    +
```

```
                              35
      60906097841360558987335f
    +
                              29
      76167367934608798697275f
    +
                           23
      27855066785995181125f
    +
                                17
      - 76144952817052723145495f
    +
                                11
      - 60933629892463517546975f
    +
                              5
      - 411415071682002547795f
 *
    e
+
                    42                      36
  - 209493533143822f   + 535045979490560586f
+
                       30                         24
  83737947964973553146f   + 104889507084213371570f
+
                     18                         12
  167117997269207870f   - 104793725781390615514f
+
                         6
  - 83842685189903180394f  - 569978796672974242
,

                  6                   3
  (25438330117200f   + 25438330117200)e
+
                  7                    2
  (76314990351600f   + 76314990351600f)e
+
                      44                    38
      - 1594966552735f   + 4073543370415745f
    +
                         32                      26
      637527159231148925f   + 797521176113606525f
    +
                      20                      14
      530440941097175f   - 797160527306433145f
    +
                           8                    2
      - 638132320196044965f  - 4510507167940725f
```

```
  *
     e
+
                 45                      39
 - 6036376800443f   + 15416903421476909f
+
                     33                       27
 2412807646192304449f   + 3017679923028013705f
+
                  21                       15
 1422320037411955f   - 3016560402417843941f
+
                        9                     3
 - 2414249368183033161f  - 16561862361763873f
,

                12                  2
 (1387545279120f   - 1387545279120)e
+
                43                  37
     4321823003f   - 11037922310209f
   +
                        31                    25
     - 1727510711947989f   - 2165150991154425f
   +
                     19                    13
     - 5114342560755f   + 2162682824948601f
   +
                     7
     1732620732685741f  + 13506088516033f
  *
     e
+
             44                   38
 24177661775f   - 61749727185325f
+
                     32                      26
 - 9664106795754225f   - 12090487758628245f
+
                 20                      14
 - 8787672733575f   + 12083693383005045f
+
                  8                  2
 9672870290826025f  + 68544102808525f
,

  48       42         36          30          18
 f   - 2554f   - 399710f   - 499722f   + 499722f
+
        12        6
```

```
      399710f   + 2554f  - 1
     ]
Type: List NewSparseMultivariatePolynomial(Integer,OrderedVariableList [a,b,c,d,e,f])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LexTriangularPackageXmpPageEmpty13}
\begin{paste}{LexTriangularPackageXmpPageEmpty13}{LexTriangularPackageXmpPagePatch13}
\pastebutton{LexTriangularPackageXmpPageEmpty13}{\showpaste}
\tab{5}\spadcommand{lg := groebner(lp)$lextripack\free{lp }\free{lextripack }\bound{lg }}
\end{paste}\end{patch}

\begin{patch}{LexTriangularPackageXmpPagePatch14}
\begin{paste}{LexTriangularPackageXmpPageFull14}{LexTriangularPackageXmpPageEmpty14}
\pastebutton{LexTriangularPackageXmpPageFull14}{\hidepaste}
\tab{5}\spadcommand{lexTriangular(lg,false)$lextripack\free{lg }\free{lextripack }}
\indentrel{3}\begin{verbatim}
   (14)
   [
      6
     {f  + 1,
       6      5     2 4     3 3     4 2     5
      e  - 3f e  + 3f e  - 4f e  + 3f e  - 3f e - 1,
            2 5     3 4     4 3     5 2
      3d + f e  - 4f e  + 4f e  - 2f e  - 2e + 2f, c + f,
             2 5     3 4     4 3      5 2
      3b + 2f e  - 5f e  + 5f e  - 10f e  - 4e + 7f,
           2 5     3 4     4 3     5 2
      a - f e  + 3f e  - 3f e  + 4f e  + 3e - 3f}
     ,

      6                     2           2
     {f  - 1, e - f, d - f, c  + 4f c + f ,
                         2
      (c - f)b - f c - 5f , a + b + c + 3f}
     ,
     6                        2           2
    {f  - 1,e - f,d - f,c - f,b  + 4f b + f ,a + b + 4f},

      6             2           2
     {f  - 1, e - f, d  + 4f d + f ,
                         2
      (d - f)c - f d - 5f , b - f, a + c + d + 3f}
     ,

     {
         36        30          24          18          12
        f   - 2554f   - 399709f   - 502276f   - 399709f
       +
               6
```

```
  - 2554f   + 1
,

             12              2
  (161718564f   - 161718564)e
+
               31              25                 19
      - 504205f   + 1287737951f   + 201539391380f
    +
                 13                 7
    253982817368f   + 201940704665f  + 1574134601f
 *
    e
+
            32              26                  20
  - 2818405f   + 7198203911f   + 1126548149060f
+
              14                  8              2
1416530563364f   + 1127377589345f  + 7988820725f
,

               6
  (693772639560f  - 693772639560)d
+
                 2 5                 3 4
  - 462515093040f e  + 1850060372160f e
+
                  4 3
  - 1850060372160f e
+
                    11                   5  2
  (- 24513299931120f   - 23588269745040f )e
+
                  30                  24
      - 890810428f   + 2275181044754f
    +
                      18                    12
      355937263869776f   + 413736880104344f
    +
                      6
      342849304487996f  + 3704966481878
 *
    e
+
               31                  25
  - 4163798003f   + 10634395752169f
+
                   19                    13
  1664161760192806f   + 2079424391370694f
+
```

```
                    7
    1668153650635921f  + 10924274392693f
  ,

                 6                             31
    (12614047992f  - 12614047992)c - 7246825f
  +
                25                   19
    18508536599f   + 2896249516034f
  +
                13                   7
  3581539649666f   + 2796477571739f  - 48094301893f
  ,

                  6
    (693772639560f  - 693772639560)b
  +
                     2 5                 3 4
    - 925030186080f e  + 2312575465200f e
  +
                      4 3
    - 2312575465200f e
  +
                        11                   5  2
    (- 40007555547960f   - 35382404617560f )e
  +
                    30                  24
        - 3781280823f   + 9657492291789f
      +
                          18                      12
        1511158913397906f   + 1837290892286154f
      +
                          6
        1487216006594361f  + 8077238712093
   *
      e
  +
                 31                   25
    - 9736390478f   + 24866827916734f
  +
                        19                    13
    3891495681905296f   + 4872556418871424f
  +
                        7
    3904047887269606f  + 27890075838538f
  ,
 a + b + c + d + e + f}
,

  6       2           2                      2
```

```
     {f  - 1, e  + 4f e + f , (e - f)d - f e - 5f ,
      c - f, b - f, a + d + e + 3f}
     ]
         Type: List RegularChain(Integer,[a,b,c,d,e,f])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LexTriangularPackageXmpPageEmpty14}
\begin{paste}{LexTriangularPackageXmpPageEmpty14}{LexTriangularPackageXmpPagePatch14}
\pastebutton{LexTriangularPackageXmpPageEmpty14}{\showpaste}
\tab{5}\spadcommand{lexTriangular(lg,false)$lextripack\free{lg }\free{lextripack }}
\end{paste}\end{patch}

\begin{patch}{LexTriangularPackageXmpPagePatch15}
\begin{paste}{LexTriangularPackageXmpPageFull15}{LexTriangularPackageXmpPageEmpty15}
\pastebutton{LexTriangularPackageXmpPageFull15}{\hidepaste}
\tab{5}\spadcommand{lts := lexTriangular(lg,true)$lextripack\free{lg }\free{lextripack }\bound{lts }}
\indentrel{3}\begin{verbatim}
   (15)
   [
      6
     {f  + 1,
       6      5     2 4     3 3     4 2     5
      e  - 3f e  + 3f e  - 4f e  + 3f e  - 3f e - 1,
            2 5     3 4     4 3     5 2
      3d + f e  - 4f e  + 4f e  - 2f e  - 2e + 2f, c + f,
             2 5     3 4     4 3      5 2
      3b + 2f e  - 5f e  + 5f e  - 10f e  - 4e + 7f,
           2 5     3 4     4 3     5 2
      a - f e  + 3f e  - 3f e  + 4f e  + 3e - 3f}
     ,
      6                 2          2
    {f  - 1,e - f,d - f,c  + 4f c + f ,b + c + 4f,a - f},
      6                       2          2
    {f  - 1,e - f,d - f,c - f,b  + 4f b + f ,a + b + 4f},
      6           2          2
    {f  - 1,e - f,d  + 4f d + f ,c + d + 4f,b - f,a - f},

     {
         36        30          24          18          12
        f   - 2554f   - 399709f   - 502276f   - 399709f
      +
                6
        - 2554f  + 1
      ,

                      2
        1387545279120e
      +
                        31                   25
```

```
        4321823003f    - 11037922310209f
      +
                           19                      13
        - 1727506390124986f    - 2176188913464634f
      +
                           7
        - 1732620732685741f  - 13506088516033f
   *
      e
  +
                 32                   26
    24177661775f    - 61749727185325f
  +
                       20                      14
    - 9664082618092450f    - 12152237485813570f
  +
                       8                   2
    - 9672870290826025f  - 68544102808525f
  ,

    1387545279120d
  +
                     30                   24
        - 1128983050f    + 2883434331830f
      +
                         18                     12
        451234998755840f    + 562426491685760f
      +
                         6
        447129055314890f  - 165557857270
   *
      e
  +
                  31                   25
    - 1816935351f    + 4640452214013f
  +
                    19                     13
    726247129626942f    + 912871801716798f
  +
                    7
    726583262666877f  + 4909358645961f
  ,

                                  31                   25
    1387545279120c + 778171189f    - 1987468196267f
  +
                      19                     13
    - 310993556954378f    - 383262822316802f
  +
                      7
```

```
        - 300335488637543f  + 5289595037041f
       ,

         1387545279120b
       +
                       30                  24
           1128983050f   - 2883434331830f
         +
                              18                    12
           - 451234998755840f   - 562426491685760f
         +
                              6
           - 447129055314890f  + 165557857270
        *
           e
       +
                     31                  25
         - 3283058841f   + 8384938292463f
       +
                          19                     13
         1312252817452422f   + 1646579934064638f
       +
                          7
         1306372958656407f  + 4694680112151f
       ,

                                                            31
         1387545279120a + 1387545279120e + 4321823003f
       +
                          25                     19
         - 11037922310209f   - 1727506390124986f
       +
                            13                    7
         - 2176188913464634f   - 1732620732685741f
       +
         - 13506088516033f
       }
     ,
      6      2           2
    {f  - 1,e  + 4f e + f ,d + e + 4f,c - f,b - f,a - f}]
         Type: List RegularChain(Integer,[a,b,c,d,e,f])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LexTriangularPackageXmpPageEmpty15}
\begin{paste}{LexTriangularPackageXmpPageEmpty15}{LexTriangularPackageXmpPagePatch15}
\pastebutton{LexTriangularPackageXmpPageEmpty15}{\showpaste}
\tab{5}\spadcommand{lts := lexTriangular(lg,true)$lextripack\free{lg }\free{lextripack }\bound{lts }}
\end{paste}\end{patch}
```

```
\begin{patch}{LexTriangularPackageXmpPagePatch16}
\begin{paste}{LexTriangularPackageXmpPageFull16}{LexTriangularPackageXmpPageEmpty16}
\pastebutton{LexTriangularPackageXmpPageFull16}{\hidepaste}
\tab{5}\spadcommand{[[init(p) for p in (ts :: List(P))] for ts in lts]\free{lts }}
\indentrel{3}\begin{verbatim}
   (16)
   [[1,3,1,3,1,1], [1,1,1,1,1,1], [1,1,1,1,1,1],
    [1,1,1,1,1,1],

     [1387545279120, 1387545279120, 1387545279120,
      1387545279120, 1387545279120, 1]
     ,
    [1,1,1,1,1,1]]
Type: List List NewSparseMultivariatePolynomial(Integer,OrderedVariableList [a,b,c,d,e,f])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LexTriangularPackageXmpPageEmpty16}
\begin{paste}{LexTriangularPackageXmpPageEmpty16}{LexTriangularPackageXmpPagePatch16}
\pastebutton{LexTriangularPackageXmpPageEmpty16}{\showpaste}
\tab{5}\spadcommand{[[init(p) for p in (ts :: List(P))] for ts in lts]\free{lts }}
\end{paste}\end{patch}

\begin{patch}{LexTriangularPackageXmpPagePatch17}
\begin{paste}{LexTriangularPackageXmpPageFull17}{LexTriangularPackageXmpPageEmpty17}
\pastebutton{LexTriangularPackageXmpPageFull17}{\hidepaste}
\tab{5}\spadcommand{squareFreeLexTriangular(lg,true)$lextripack\free{lg }\free{lextripack }}
\indentrel{3}\begin{verbatim}
   (17)
   [
      6
     {f  + 1,
       6      5     2 4     3 3     4 2     5
      e  - 3f e  + 3f e  - 4f e  + 3f e  - 3f e - 1,
            2 5     3 4     4 3     5 2
      3d + f e  - 4f e  + 4f e  - 2f e  - 2e + 2f, c + f,
             2 5     3 4     4 3      5 2
      3b + 2f e  - 5f e  + 5f e  - 10f e  - 4e + 7f,
           2 5     3 4     4 3     5 2
      a - f e  + 3f e  - 3f e  + 4f e  + 3e - 3f}
     ,
     6                 2          2
    {f  - 1,e - f,d - f,c  + 4f c + f ,b + c + 4f,a - f},
     6                       2          2
    {f  - 1,e - f,d - f,c - f,b  + 4f b + f ,a + b + 4f},
     6           2          2
    {f  - 1,e - f,d  + 4f d + f ,c + d + 4f,b - f,a - f},

     {
         36        30          24          18          12
```

```
  f   - 2554f    - 399709f    - 502276f    - 399709f
+
         6
  - 2554f  + 1
,

                 2
  1387545279120e
+
                  31                   25
      4321823003f   - 11037922310209f
    +
                          19                     13
      - 1727506390124986f   - 2176188913464634f
    +
                          7
      - 1732620732685741f  - 13506088516033f
 *
    e
+
               32                   26
  24177661775f   - 61749727185325f
+
                      20                     14
  - 9664082618092450f   - 12152237485813570f
+
                      8                  2
  - 9672870290826025f  - 68544102808525f
,

  1387545279120d
+
                    30                  24
      - 1128983050f   + 2883434331830f
    +
                       18                   12
      451234998755840f   + 562426491685760f
    +
                       6
      447129055314890f  - 165557857270
 *
    e
+
                 31                  25
  - 1816935351f   + 4640452214013f
+
                   19                   13
  726247129626942f   + 912871801716798f
+
                 7
```

```
      726583262666877f  + 4909358645961f
     ,

                                  31                  25
      1387545279120c + 778171189f   - 1987468196267f
     +
                        19                     13
      - 310993556954378f   - 383262822316802f
     +
                        7
      - 300335488637543f  + 5289595037041f
     ,

      1387545279120b
     +
                     30                  24
          1128983050f   - 2883434331830f
        +
                            18                     12
          - 451234998755840f   - 562426491685760f
        +
                            6
          - 447129055314890f  + 165557857270
      *
         e
     +
                   31                  25
      - 3283058841f   + 8384938292463f
     +
                       19                     13
      1312252817452422f   + 1646579934064638f
     +
                       7
      1306372958656407f  + 4694680112151f
     ,

                                                        31
      1387545279120a + 1387545279120e + 4321823003f
     +
                       25                     19
      - 11037922310209f   - 1727506390124986f
     +
                         13                     7
      - 2176188913464634f   - 1732620732685741f
     +
      - 13506088516033f
     }
   ,
    6      2           2
  {f  - 1,e  + 4f e + f ,d + e + 4f,c - f,b - f,a - f}]
```

```
Type: List SquareFreeRegularTriangularSet(Integer,IndexedExponents OrderedVariableList [a,b,c,d,e,f],Ordered
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LexTriangularPackageXmpPageEmpty17}
\begin{paste}{LexTriangularPackageXmpPageEmpty17}{LexTriangularPackageXmpPagePatch17}
\pastebutton{LexTriangularPackageXmpPageEmpty17}{\showpaste}
\tab{5}\spadcommand{squareFreeLexTriangular(lg,true)$lextripack\free{lg }\free{lextripack }}
\end{paste}\end{patch}

\begin{patch}{LexTriangularPackageXmpPagePatch18}
\begin{paste}{LexTriangularPackageXmpPageFull18}{LexTriangularPackageXmpPageEmpty18}
\pastebutton{LexTriangularPackageXmpPageFull18}{\hidepaste}
\tab{5}\spadcommand{reduce(+,[degree(ts) for ts in lts])\free{lts }}
\indentrel{3}\begin{verbatim}
   (18)  156
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LexTriangularPackageXmpPageEmpty18}
\begin{paste}{LexTriangularPackageXmpPageEmpty18}{LexTriangularPackageXmpPagePatch18}
\pastebutton{LexTriangularPackageXmpPageEmpty18}{\showpaste}
\tab{5}\spadcommand{reduce(+,[degree(ts) for ts in lts])\free{lts }}
\end{paste}\end{patch}

\begin{patch}{LexTriangularPackageXmpPagePatch19}
\begin{paste}{LexTriangularPackageXmpPageFull19}{LexTriangularPackageXmpPageEmpty19}
\pastebutton{LexTriangularPackageXmpPageFull19}{\hidepaste}
\tab{5}\spadcommand{ls2 : List Symbol := concat(ls,new()$Symbol)\free{ls }\bound{ls2 }}
\indentrel{3}\begin{verbatim}
   (19)  [a,b,c,d,e,f,%A]
                                      Type: List Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LexTriangularPackageXmpPageEmpty19}
\begin{paste}{LexTriangularPackageXmpPageEmpty19}{LexTriangularPackageXmpPagePatch19}
\pastebutton{LexTriangularPackageXmpPageEmpty19}{\showpaste}
\tab{5}\spadcommand{ls2 : List Symbol := concat(ls,new()$Symbol)\free{ls }\bound{ls2 }}
\end{paste}\end{patch}

\begin{patch}{LexTriangularPackageXmpPagePatch20}
\begin{paste}{LexTriangularPackageXmpPageFull20}{LexTriangularPackageXmpPageEmpty20}
\pastebutton{LexTriangularPackageXmpPageFull20}{\hidepaste}
\tab{5}\spadcommand{zdpack := ZDSOLVE(R,ls,ls2)\free{R }\free{ls }\free{ls2 }\bound{zdpack }}
\indentrel{3}\begin{verbatim}
   (20)
  ZeroDimensionalSolvePackage(Integer,[a,b,c,d,e,f],[a,b,
  c,d,e,f,%A])
```

```
                                              Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LexTriangularPackageXmpPageEmpty20}
\begin{paste}{LexTriangularPackageXmpPageEmpty20}{LexTriangularPackageXmpPagePatch20}
\pastebutton{LexTriangularPackageXmpPageEmpty20}{\showpaste}
\tab{5}\spadcommand{zdpack := ZDSOLVE(R,ls,ls2)\free{R }\free{ls }\free{ls2 }\bound{zdpack }}
\end{paste}\end{patch}

\begin{patch}{LexTriangularPackageXmpPagePatch21}
\begin{paste}{LexTriangularPackageXmpPageFull21}{LexTriangularPackageXmpPageEmpty21}
\pastebutton{LexTriangularPackageXmpPageFull21}{\hidepaste}
\tab{5}\spadcommand{concat [univariateSolve(ts)$zdpack for ts in lts]\free{lts }\free{zdpack }}
\indentrel{3}\begin{verbatim}
   (21)
   [
                    4      2
     [complexRoots= ?  - 13?  + 49,

       coordinates =
                 3                 3
         [7a + %A  - 6%A, 21b + %A  + %A,
                   3                 3
          21c - 2%A  + 19%A, 7d - %A  + 6%A,
                 3                 3
          21e - %A  - %A, 21f + 2%A  - 19%A]
       ]
     ,

                    4      2
     [complexRoots= ?  + 11?  + 49,

       coordinates =
                   3                  3
         [35a + 3%A  + 19%A, 35b + %A  + 18%A,
                   3                 3
          35c - 2%A  - %A, 35d - 3%A  - 19%A,
                 3                  3
          35e - %A  - 18%A, 35f + 2%A  + %A]
       ]
     ,

     [
       complexRoots =
             8      7      6       5       4       3
           ?  - 12?  + 58?  - 120?  + 207?  - 360?
         +
                2
           802?  - 1332? + 1369
```

```
     ,

     coordinates =
       [
                               7            6             5
          43054532a + 33782%A  - 546673%A  + 3127348%A
        +
                    4             3              2
          - 6927123%A  + 4365212%A  - 25086957%A
        +
          39582814%A - 107313172
        ,

                               7            6             5
          43054532b - 33782%A  + 546673%A  - 3127348%A
        +
                  4             3              2
          6927123%A  - 4365212%A  + 25086957%A
        +
          - 39582814%A + 107313172
        ,

                               7            6             5
          21527266c - 22306%A  + 263139%A  - 1166076%A
        +
                  4             3              2
          1821805%A  - 2892788%A  + 10322663%A
        +
          - 9026596%A + 12950740
        ,

                               7            6             5
          43054532d + 22306%A  - 263139%A  + 1166076%A
        +
                    4             3              2
          - 1821805%A  + 2892788%A  - 10322663%A
        +
          30553862%A - 12950740
        ,

                               7            6             5
          43054532e - 22306%A  + 263139%A  - 1166076%A
        +
                  4             3              2
          1821805%A  - 2892788%A  + 10322663%A
        +
          - 30553862%A + 12950740
        ,

                               7            6             5
```

```
          21527266f + 22306%A  - 263139%A  + 1166076%A
         +
                      4             3              2
          - 1821805%A  + 2892788%A  - 10322663%A
         +
          9026596%A - 12950740
         ]
     ]
    ,

    [
     complexRoots =
         8      7      6       5       4       3
        ?  + 12?  + 58?  + 120?  + 207?  + 360?
       +
            2
        802?  + 1332? + 1369
     ,

     coordinates =
       [
                            7            6             5
          43054532a + 33782%A  + 546673%A  + 3127348%A
         +
                    4             3              2
          6927123%A  + 4365212%A  + 25086957%A
         +
          39582814%A + 107313172
         ,

                            7            6             5
          43054532b - 33782%A  - 546673%A  - 3127348%A
         +
                      4             3              2
          - 6927123%A  - 4365212%A  - 25086957%A
         +
          - 39582814%A - 107313172
         ,

                            7            6             5
          21527266c - 22306%A  - 263139%A  - 1166076%A
         +
                      4             3              2
          - 1821805%A  - 2892788%A  - 10322663%A
         +
          - 9026596%A - 12950740
         ,

                            7            6             5
          43054532d + 22306%A  + 263139%A  + 1166076%A
```

```
     +
               4             3              2
       1821805%A  + 2892788%A  + 10322663%A
     +
       30553862%A + 12950740
     ,

                             7            6             5
       43054532e - 22306%A  - 263139%A  - 1166076%A
     +
                 4             3              2
       - 1821805%A  - 2892788%A  - 10322663%A
     +
       - 30553862%A - 12950740
     ,

                             7            6             5
       21527266f + 22306%A  + 263139%A  + 1166076%A
     +
               4             3              2
       1821805%A  + 2892788%A  + 10322663%A
     +
       9026596%A + 12950740
     ]
 ]
,

               4    2
[complexRoots= ?  - ?  + 1,

  coordinates =
                   3              3
    [a - %A, b + %A  - %A, c + %A , d + %A,
           3             3
     e - %A  + %A, f - %A ]
  ]
,

               8     6      4      2
[complexRoots= ?  + 4?  + 12?  + 16?  + 4,

  coordinates =
            7      5       3
    [4a - 2%A  - 7%A  - 20%A  - 22%A,
            7      5       3
     4b + 2%A  + 7%A  + 20%A  + 22%A,
           7      5       3
     4c + %A  + 3%A  + 10%A  + 10%A,
           7      5       3
     4d + %A  + 3%A  + 10%A  + 6%A,
```

```
           7      5       3
     4e - %A  - 3%A  - 10%A  - 6%A,
           7      5       3
     4f - %A  - 3%A  - 10%A  - 10%A]
  ]
,

               4     3      2
[complexRoots= ?  + 6?  + 30?  + 36? + 36,

  coordinates =
            3      2
    [30a - %A  - 5%A  - 30%A - 6,
           3      2
     6b + %A  + 5%A  + 24%A + 6,
            3      2
     30c - %A  - 5%A  - 6,
            3      2
     30d - %A  - 5%A  - 30%A - 6,
            3      2
     30e - %A  - 5%A  - 30%A - 6,
            3      2
     30f - %A  - 5%A  - 30%A - 6]
  ]
,

               4     3      2
[complexRoots= ?  - 6?  + 30?  - 36? + 36,

  coordinates =
            3      2
    [30a - %A  + 5%A  - 30%A + 6,
           3      2
     6b + %A  - 5%A  + 24%A - 6,
            3      2
     30c - %A  + 5%A  + 6,
            3      2
     30d - %A  + 5%A  - 30%A + 6,
            3      2
     30e - %A  + 5%A  - 30%A + 6,
            3      2
     30f - %A  + 5%A  - 30%A + 6]
  ]
,

               2
[complexRoots= ?  + 6? + 6,

  coordinates =
    [a + 1,b - %A - 5,c + %A + 1,d + 1,e + 1,f + 1]
```

```
  ]
,

                2
[complexRoots= ?  - 6? + 6,

  coordinates =
    [a - 1,b - %A + 5,c + %A - 1,d - 1,e - 1,f - 1]
  ]
,

                4     3      2
[complexRoots= ?  + 6?  + 30?  + 36? + 36,

  coordinates =
            3      2
    [6a + %A  + 5%A  + 24%A + 6,
             3      2
     30b - %A  - 5%A  - 6,
             3      2
     30c - %A  - 5%A  - 30%A - 6,
             3      2
     30d - %A  - 5%A  - 30%A - 6,
             3      2
     30e - %A  - 5%A  - 30%A - 6,
             3      2
     30f - %A  - 5%A  - 30%A - 6]
  ]
,

                4     3      2
[complexRoots= ?  - 6?  + 30?  - 36? + 36,

  coordinates =
            3      2
    [6a + %A  - 5%A  + 24%A - 6,
             3      2
     30b - %A  + 5%A  + 6,
             3      2
     30c - %A  + 5%A  - 30%A + 6,
             3      2
     30d - %A  + 5%A  - 30%A + 6,
             3      2
     30e - %A  + 5%A  - 30%A + 6,
             3      2
     30f - %A  + 5%A  - 30%A + 6]
  ]
,

                2
```

```
[complexRoots= ?  + 6? + 6,

  coordinates =
    [a - %A - 5,b + %A + 1,c + 1,d + 1,e + 1,f + 1]
  ]
,

                2
[complexRoots= ?  - 6? + 6,

  coordinates =
    [a - %A + 5,b + %A - 1,c - 1,d - 1,e - 1,f - 1]
  ]
,

                4     3      2
[complexRoots= ?  + 6?  + 30?  + 36? + 36,

  coordinates =
             3      2
    [30a - %A  - 5%A  - 30%A - 6,
             3      2
     30b - %A  - 5%A  - 30%A - 6,
            3      2
     6c + %A  + 5%A  + 24%A + 6,
             3      2
     30d - %A  - 5%A  - 6,
             3      2
     30e - %A  - 5%A  - 30%A - 6,
             3      2
     30f - %A  - 5%A  - 30%A - 6]
  ]
,

                4     3      2
[complexRoots= ?  - 6?  + 30?  - 36? + 36,

  coordinates =
             3      2
    [30a - %A  + 5%A  - 30%A + 6,
             3      2
     30b - %A  + 5%A  - 30%A + 6,
            3      2
     6c + %A  - 5%A  + 24%A - 6,
             3      2
     30d - %A  + 5%A  + 6,
             3      2
     30e - %A  + 5%A  - 30%A + 6,
             3      2
     30f - %A  + 5%A  - 30%A + 6]
```

```
  ]
,

                 2
[complexRoots= ?  + 6? + 6,

  coordinates =
    [a + 1,b + 1,c - %A - 5,d + %A + 1,e + 1,f + 1]
  ]
,

                 2
[complexRoots= ?  - 6? + 6,

  coordinates =
    [a - 1,b - 1,c - %A + 5,d + %A - 1,e - 1,f - 1]
  ]
,

[
  complexRoots =
     8     7      6      5      4     2
    ?  + 6?  + 16?  + 24?  + 18?  - 8?  + 4
  ,

  coordinates =
    [
                 7      6       5       4      3
        2a + 2%A  + 9%A  + 18%A  + 19%A  + 4%A
      +
              2
        - 10%A  - 2%A + 4
      ,

                 7      6       5       4      3
        2b + 2%A  + 9%A  + 18%A  + 19%A  + 4%A
      +
              2
        - 10%A  - 4%A + 4
      ,
            7      6      5      4      3
       2c - %A  - 4%A  - 8%A  - 9%A  - 4%A  - 2%A - 4,
            7      6      5      4      3
       2d + %A  + 4%A  + 8%A  + 9%A  + 4%A  + 2%A + 4,

                 7      6       5       4      3
        2e - 2%A  - 9%A  - 18%A  - 19%A  - 4%A
      +
            2
        10%A  + 4%A - 4
```

```
       ,

                 7       6        5        4       3
         2f - 2%A  - 9%A  - 18%A  - 19%A  - 4%A
       +
             2
         10%A  + 2%A - 4
       ]
   ]
  ,

  [
   complexRoots =
        8       7       6        5        4        3
       ?  + 12?  + 64?  + 192?  + 432?  + 768?
     +
            2
       1024?  + 768? + 256
   ,

   coordinates =
     [
                     7         6         5          4
         1408a - 19%A  - 200%A  - 912%A  - 2216%A
       +
                 3          2
         - 4544%A  - 6784%A  - 6976%A - 1792
       ,

                     7         6          5          4
         1408b - 37%A  - 408%A  - 1952%A  - 5024%A
       +
                  3           2
         - 10368%A  - 16768%A  - 17920%A - 5120
       ,

                     7         6          5          4
         1408c + 37%A  + 408%A  + 1952%A  + 5024%A
       +
                3           2
         10368%A  + 16768%A  + 17920%A + 5120
       ,

                     7         6         5          4
         1408d + 19%A  + 200%A  + 912%A  + 2216%A
       +
               3          2
         4544%A  + 6784%A  + 6976%A + 1792
       ,
      2e + %A, 2f - %A]
```

```
  ]
,

                 8     6      4      2
[complexRoots= ?  + 4?  + 12?  + 16?  + 4,

  coordinates =
           7      5       3
    [4a - %A  - 3%A  - 10%A  - 6%A,
           7      5       3
     4b - %A  - 3%A  - 10%A  - 10%A,
            7      5       3
     4c - 2%A  - 7%A  - 20%A  - 22%A,
            7      5       3
     4d + 2%A  + 7%A  + 20%A  + 22%A,
           7      5       3
     4e + %A  + 3%A  + 10%A  + 10%A,
           7      5       3
     4f + %A  + 3%A  + 10%A  + 6%A]
  ]
,

                 8      6      4       2
[complexRoots= ?  + 16?  - 96?  + 256?  + 256,

  coordinates =
             7       5        3
    [512a - %A  - 12%A  + 176%A  - 448%A,
             7       5       3
     128b - %A  - 16%A  + 96%A  - 256%A,
             7       5       3
     128c + %A  + 16%A  - 96%A  + 256%A,
             7       5        3
     512d + %A  + 12%A  - 176%A  + 448%A, 2e + %A,
     2f - %A]
  ]
,

[
  complexRoots =
      8      7      6       5       4       3
     ?  - 12?  + 64?  - 192?  + 432?  - 768?
    +
          2
     1024?  - 768? + 256
  ,

  coordinates =
    [
                 7        6        5         4
```

```
        1408a - 19%A  + 200%A  - 912%A  + 2216%A
      +
                3         2
        - 4544%A  + 6784%A  - 6976%A + 1792
      ,

                    7        6         5         4
        1408b - 37%A  + 408%A  - 1952%A  + 5024%A
      +
                 3          2
        - 10368%A  + 16768%A  - 17920%A + 5120
      ,

                    7        6         5         4
        1408c + 37%A  - 408%A  + 1952%A  - 5024%A
      +
               3          2
        10368%A  - 16768%A  + 17920%A - 5120
      ,

                    7        6        5         4
        1408d + 19%A  - 200%A  + 912%A  - 2216%A
      +
              3         2
        4544%A  - 6784%A  + 6976%A - 1792
      ,
     2e + %A, 2f - %A]
  ]
,

[
  complexRoots =
     8     7      6      5      4     2
    ?  - 6?  + 16?  - 24?  + 18?  - 8?  + 4
  ,

  coordinates =
    [
                7      6       5       4      3
        2a + 2%A  - 9%A  + 18%A  - 19%A  + 4%A
      +
            2
        10%A  - 2%A - 4
      ,

                7      6       5       4      3
        2b + 2%A  - 9%A  + 18%A  - 19%A  + 4%A
      +
            2
        10%A  - 4%A - 4
```

```
      ,
            7       6       5       4       3
     2c - %A  + 4%A  - 8%A  + 9%A  - 4%A  - 2%A + 4,
            7       6       5       4       3
     2d + %A  - 4%A  + 8%A  - 9%A  + 4%A  + 2%A - 4,

                7       6        5        4       3
        2e - 2%A  + 9%A  - 18%A  + 19%A  - 4%A
      +
              2
        - 10%A  + 4%A + 4
      ,

                7       6        5        4       3
        2f - 2%A  + 9%A  - 18%A  + 19%A  - 4%A
      +
              2
        - 10%A  + 2%A + 4
      ]
   ]
  ,

                 4      2
  [complexRoots= ?  + 12?  + 144,

    coordinates =
              2                2                2
      [12a - %A  - 12, 12b - %A  - 12, 12c - %A  - 12,
              2               2
       12d - %A  - 12, 6e + %A  + 3%A + 12,
             2
       6f + %A  - 3%A + 12]
    ]
  ,

                 4     3      2
  [complexRoots= ?  + 6?  + 30?  + 36? + 36,

    coordinates =
             3      2
      [6a - %A  - 5%A  - 24%A - 6,
              3      2
       30b + %A  + 5%A  + 30%A + 6,
              3      2
       30c + %A  + 5%A  + 30%A + 6,
              3      2
       30d + %A  + 5%A  + 30%A + 6,
              3      2
       30e + %A  + 5%A  + 30%A + 6,
              3      2
```

```
    30f + %A  + 5%A  + 6]
  ]
,

                4     3      2
[complexRoots= ?  - 6?  + 30?  - 36? + 36,

  coordinates =
            3      2
    [6a - %A  + 5%A  - 24%A + 6,
             3      2
     30b + %A  - 5%A  + 30%A - 6,
             3      2
     30c + %A  - 5%A  + 30%A - 6,
             3      2
     30d + %A  - 5%A  + 30%A - 6,
             3      2
     30e + %A  - 5%A  + 30%A - 6,
             3      2
     30f + %A  - 5%A  - 6]
  ]
,

                4      2
[complexRoots= ?  + 12?  + 144,

  coordinates =
             2               2               2
    [12a + %A  + 12, 12b + %A  + 12, 12c + %A  + 12,
             2              2
     12d + %A  + 12, 6e - %A  + 3%A - 12,
            2
     6f - %A  - 3%A - 12]
  ]
,

                2
[complexRoots= ?  - 12,

  coordinates =
    [a - 1,b - 1,c - 1,d - 1,2e + %A + 4,2f - %A + 4]
  ]
,

                2
[complexRoots= ?  + 6? + 6,

  coordinates =
    [a + %A + 5,b - 1,c - 1,d - 1,e - 1,f - %A - 1]
  ]
```

```
,

                 2
[complexRoots= ?  - 6? + 6,

  coordinates =
    [a + %A - 5,b + 1,c + 1,d + 1,e + 1,f - %A + 1]
  ]
,

                 2
[complexRoots= ?  - 12,

  coordinates =
    [a + 1,b + 1,c + 1,d + 1,2e + %A - 4,2f - %A - 4]
  ]
,

                 4     3      2
[complexRoots= ?  + 6?  + 30?  + 36? + 36,

  coordinates =
             3      2
    [30a - %A  - 5%A  - 30%A - 6,
             3      2
     30b - %A  - 5%A  - 30%A - 6,
             3      2
     30c - %A  - 5%A  - 30%A - 6,
            3      2
     6d + %A  + 5%A  + 24%A + 6,
             3      2
     30e - %A  - 5%A  - 6,
             3      2
     30f - %A  - 5%A  - 30%A - 6]
  ]
,

                 4     3      2
[complexRoots= ?  - 6?  + 30?  - 36? + 36,

  coordinates =
             3      2
    [30a - %A  + 5%A  - 30%A + 6,
             3      2
     30b - %A  + 5%A  - 30%A + 6,
             3      2
     30c - %A  + 5%A  - 30%A + 6,
            3      2
     6d + %A  - 5%A  + 24%A - 6,
             3      2
```

```
          30e - %A  + 5%A  + 6,
                 3      2
          30f - %A  + 5%A  - 30%A + 6]
        ]
      ,

                         2
      [complexRoots= ?  + 6? + 6,

        coordinates =
          [a + 1,b + 1,c + 1,d - %A - 5,e + %A + 1,f + 1]
        ]
      ,

                         2
      [complexRoots= ?  - 6? + 6,

        coordinates =
          [a - 1,b - 1,c - 1,d - %A + 5,e + %A - 1,f - 1]
        ]
      ]
Type: List Record(complexRoots: SparseUnivariatePolynomial Integer,coordinates: List Polynomial
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LexTriangularPackageXmpPageEmpty21}
\begin{paste}{LexTriangularPackageXmpPageEmpty21}{LexTriangularPackageXmpPagePatch21}
\pastebutton{LexTriangularPackageXmpPageEmpty21}{\showpaste}
\tab{5}\spadcommand{concat [univariateSolve(ts)$zdpack for ts in lts]\free{lts }\free{zdpack }}
\end{paste}\end{patch}

\begin{patch}{LexTriangularPackageXmpPagePatch22}
\begin{paste}{LexTriangularPackageXmpPageFull22}{LexTriangularPackageXmpPageEmpty22}
\pastebutton{LexTriangularPackageXmpPageFull22}{\hidepaste}
\tab{5}\spadcommand{concat [realSolve(ts)$zdpack for ts in lts]\free{lts }\free{zdpack }}
\indentrel{3}\begin{verbatim}
   (22)
   [[%R1,%R1,%R1,%R5,- %R5 - 4%R1,%R1],
    [%R1,%R1,%R1,%R6,- %R6 - 4%R1,%R1],
    [%R2,%R2,%R2,%R3,- %R3 - 4%R2,%R2],
    [%R2,%R2,%R2,%R4,- %R4 - 4%R2,%R2],
    [%R7,%R7,%R7,%R7,%R11,- %R11 - 4%R7],
    [%R7,%R7,%R7,%R7,%R12,- %R12 - 4%R7],
    [%R8,%R8,%R8,%R8,%R9,- %R9 - 4%R8],
    [%R8,%R8,%R8,%R8,%R10,- %R10 - 4%R8],
    [%R13,%R13,%R17,- %R17 - 4%R13,%R13,%R13],
    [%R13,%R13,%R18,- %R18 - 4%R13,%R13,%R13],
    [%R14,%R14,%R15,- %R15 - 4%R14,%R14,%R14],
    [%R14,%R14,%R16,- %R16 - 4%R14,%R14,%R14],
```

```
[%R19, %R29,

     7865521      31   6696179241     25
    %R19   -  %R19
   6006689520          2002229840
  +
     25769893181     19   1975912990729     13
   -  %R19   -  %R19
       49235160             3003344760
  +
     1048460696489     7   21252634831
   -  %R19  -  %R19
       2002229840           6006689520
  ,

       778171189       31   1987468196267     25
   -  %R19   +  %R19
     1387545279120         1387545279120
  +
   155496778477189     19   191631411158401     13
    %R19   +  %R19
     693772639560            693772639560
  +
   300335488637543     7   755656433863
    %R19  -  %R19
    1387545279120         198220754160
  ,

    1094352947      31   2794979430821     25
    %R19   -  %R19
   462515093040          462515093040
  +
     218708802908737     19   91476663003591     13
   -  %R19   -  %R19
       231257546520            77085848840
  +
     145152550961823     7   1564893370717
   -  %R19  -  %R19
       154171697680          462515093040
  ,

              4321823003      31
   - %R29 -  %R19
            1387545279120
  +
   180949546069      25   863753195062493     19
    %R19   +  %R19
    22746643920             693772639560
  +
   1088094456732317     13   1732620732685741     7
```

```
      %R19    +  %R19
       693772639560               1387545279120
   +
     13506088516033
      %R19
      1387545279120
   ]
 ,

 [%R19, %R30,

       7865521       31    6696179241      25
      %R19    -  %R19
     6006689520           2002229840
   +
       25769893181      19    1975912990729      13
     -  %R19    -  %R19
         49235160                3003344760
   +
       1048460696489      7    21252634831
     -  %R19   -  %R19
         2002229840             6006689520
   ,

         778171189        31    1987468196267      25
     -  %R19    +  %R19
       1387545279120           1387545279120
   +
     155496778477189      19    191631411158401      13
      %R19    +  %R19
       693772639560              693772639560
   +
     300335488637543      7    755656433863
      %R19   -  %R19
      1387545279120           198220754160
   ,

      1094352947        31    2794979430821      25
      %R19    -  %R19
     462515093040             462515093040
   +
       218708802908737      19    91476663003591      13
     -  %R19    -  %R19
         231257546520               77085848840
   +
       145152550961823      7    1564893370717
     -  %R19   -  %R19
         154171697680             462515093040
   ,
```

```
            4321823003      31
    - %R30 -  %R19
          1387545279120
  +
    180949546069     25   863753195062493     19
     %R19    +  %R19
     22746643920             693772639560
  +
    1088094456732317     13   1732620732685741     7
     %R19    +  %R19
      693772639560              1387545279120
  +
    13506088516033
     %R19
     1387545279120
  ]
 ,

 [%R20, %R27,

      7865521       31   6696179241      25
     %R20    -  %R20
    6006689520           2002229840
  +
      25769893181      19   1975912990729      13
    -  %R20    -  %R20
        49235160             3003344760
  +
      1048460696489      7   21252634831
    -  %R20   -  %R20
        2002229840           6006689520
  ,

        778171189       31   1987468196267      25
    -  %R20    +  %R20
      1387545279120          1387545279120
  +
    155496778477189      19   191631411158401      13
     %R20    +  %R20
      693772639560             693772639560
  +
    300335488637543      7   755656433863
     %R20   -  %R20
     1387545279120          198220754160
  ,

     1094352947       31   2794979430821      25
     %R20    -  %R20
    462515093040           462515093040
  +
```

```
      218708802908737      19   91476663003591      13
    - %R20   - %R20
        231257546520             77085848840
  +
      145152550961823      7   1564893370717
    - %R20  - %R20
        154171697680           462515093040
  ,

               4321823003       31
    - %R27 -  %R20
             1387545279120
  +
    180949546069      25   863753195062493      19
     %R20   + %R20
     22746643920             693772639560
  +
    1088094456732317      13   1732620732685741      7
     %R20   + %R20
      693772639560               1387545279120
  +
    13506088516033
     %R20
     1387545279120
  ]
,

[%R20, %R28,

      7865521       31   6696179241      25
     %R20   -  %R20
    6006689520           2002229840
  +
      25769893181      19   1975912990729      13
    - %R20   -  %R20
        49235160               3003344760
  +
      1048460696489      7   21252634831
    - %R20  - %R20
        2002229840            6006689520
  ,

        778171189        31   1987468196267      25
    - %R20   +  %R20
      1387545279120          1387545279120
  +
    155496778477189      19   191631411158401      13
     %R20   + %R20
      693772639560              693772639560
  +
```

```
        300335488637543      7   755656433863
         %R20  -  %R20
         1387545279120           198220754160
      ,

         1094352947       31   2794979430821      25
         %R20   -  %R20
        462515093040            462515093040
      +
          218708802908737      19    91476663003591      13
        -  %R20   -  %R20
            231257546520              77085848840
      +
          145152550961823      7   1564893370717
        -  %R20  -  %R20
            154171697680           462515093040
      ,

                   4321823003       31
        - %R28 -  %R20
                 1387545279120
      +
        180949546069      25   863753195062493      19
         %R20   +  %R20
         22746643920             693772639560
      +
        1088094456732317      13   1732620732685741      7
         %R20   +  %R20
          693772639560               1387545279120
      +
        13506088516033
         %R20
         1387545279120
      ]
    ,

    [%R21, %R25,

          7865521       31   6696179241      25
         %R21   -  %R21
        6006689520           2002229840
      +
          25769893181      19   1975912990729      13
        -  %R21   -  %R21
            49235160              3003344760
      +
          1048460696489      7   21252634831
        -  %R21  -  %R21
            2002229840           6006689520
      ,
```

```
       778171189        31   1987468196267      25
    - %R21    + %R21
      1387545279120          1387545279120
  +
    155496778477189      19   191631411158401      13
     %R21    + %R21
      693772639560              693772639560
  +
    300335488637543      7   755656433863
     %R21  -  %R21
     1387545279120          198220754160
  ,

     1094352947       31   2794979430821      25
     %R21   -  %R21
    462515093040            462515093040
  +
      218708802908737      19   91476663003591      13
    -  %R21   -  %R21
        231257546520              77085848840
  +
      145152550961823      7   1564893370717
    -  %R21  -  %R21
        154171697680            462515093040
  ,

              4321823003       31
    - %R25 -  %R21
            1387545279120
  +
    180949546069      25   863753195062493      19
     %R21    + %R21
     22746643920              693772639560
  +
    1088094456732317      13   1732620732685741      7
     %R21    + %R21
      693772639560               1387545279120
  +
    13506088516033
     %R21
     1387545279120
  ]
,

[%R21, %R26,

      7865521       31   6696179241      25
     %R21   -  %R21
    6006689520          2002229840
```

```
+
    25769893181       19   1975912990729      13
  -  %R21    -  %R21
      49235160               3003344760
+
    1048460696489      7   21252634831
  -  %R21   -  %R21
      2002229840             6006689520
,

      778171189        31   1987468196267      25
  -  %R21    +  %R21
    1387545279120          1387545279120
+
  155496778477189      19   191631411158401      13
   %R21    +  %R21
    693772639560              693772639560
+
  300335488637543      7   755656433863
   %R21   -  %R21
   1387545279120          198220754160
,

   1094352947        31   2794979430821      25
   %R21    -  %R21
  462515093040            462515093040
+
    218708802908737      19   91476663003591      13
  -  %R21    -  %R21
      231257546520              77085848840
+
    145152550961823      7   1564893370717
  -  %R21   -  %R21
      154171697680            462515093040
,

             4321823003        31
  - %R26 -  %R21
           1387545279120
+
  180949546069       25   863753195062493      19
   %R21    +  %R21
   22746643920              693772639560
+
  1088094456732317      13   1732620732685741      7
   %R21    +  %R21
    693772639560              1387545279120
+
  13506088516033
   %R21
```

```
      1387545279120
   ]
  ,

  [%R22, %R23,

       7865521       31   6696179241      25
      %R22    -  %R22
     6006689520           2002229840
   +
       25769893181      19   1975912990729      13
     -  %R22    -  %R22
         49235160              3003344760
   +
       1048460696489      7   21252634831
     -  %R22   -  %R22
         2002229840            6006689520
   ,

         778171189        31   1987468196267      25
     -  %R22    +  %R22
       1387545279120           1387545279120
   +
     155496778477189      19   191631411158401      13
      %R22    +  %R22
       693772639560              693772639560
   +
     300335488637543      7   755656433863
      %R22   -  %R22
      1387545279120           198220754160
   ,

      1094352947       31   2794979430821      25
      %R22    -  %R22
     462515093040            462515093040
   +
       218708802908737      19   91476663003591      13
     -  %R22    -  %R22
         231257546520              77085848840
   +
       145152550961823      7   1564893370717
     -  %R22   -  %R22
         154171697680            462515093040
   ,

               4321823003       31
     - %R23 -  %R22
             1387545279120
   +
     180949546069      25   863753195062493      19
```

```
     %R22   +  %R22
     22746643920            693772639560
   +
    1088094456732317     13   1732620732685741     7
     %R22   +  %R22
      693772639560              1387545279120
   +
    13506088516033
     %R22
     1387545279120
   ]
  ,

  [%R22, %R24,

      7865521       31   6696179241     25
     %R22   -  %R22
    6006689520           2002229840
   +
      25769893181     19   1975912990729     13
    -  %R22   -  %R22
        49235160             3003344760
   +
      1048460696489     7   21252634831
    -  %R22  -  %R22
        2002229840           6006689520
   ,

        778171189       31   1987468196267     25
    -  %R22   +  %R22
      1387545279120          1387545279120
   +
    155496778477189     19   191631411158401     13
     %R22   +  %R22
      693772639560             693772639560
   +
    300335488637543     7   755656433863
     %R22  -  %R22
     1387545279120          198220754160
   ,

     1094352947       31   2794979430821     25
     %R22   -  %R22
    462515093040           462515093040
   +
      218708802908737     19   91476663003591     13
    -  %R22   -  %R22
        231257546520             77085848840
   +
      145152550961823     7   1564893370717
```

```
         -  %R22  -  %R22
             154171697680           462515093040
       ,

                    4321823003      31
         - %R24 -  %R22
                  1387545279120
       +
         180949546069     25   863753195062493     19
          %R22   +  %R22
          22746643920            693772639560
       +
         1088094456732317     13   1732620732685741     7
          %R22   +  %R22
           693772639560              1387545279120
       +
         13506088516033
          %R22
          1387545279120
       ]
     ,
    [%R31,%R35,- %R35 - 4%R31,%R31,%R31,%R31],
    [%R31,%R36,- %R36 - 4%R31,%R31,%R31,%R31],
    [%R32,%R33,- %R33 - 4%R32,%R32,%R32,%R32],
    [%R32,%R34,- %R34 - 4%R32,%R32,%R32,%R32]]
           Type: List List RealClosure Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LexTriangularPackageXmpPageEmpty22}
\begin{paste}{LexTriangularPackageXmpPageEmpty22}{LexTriangularPackageXmpPagePatch22}
\pastebutton{LexTriangularPackageXmpPageEmpty22}{\showpaste}
\tab{5}\spadcommand{concat [realSolve(ts)$zdpack for ts in lts]\free{lts }\free{zdpack }}
\end{paste}\end{patch}
```

———-

3.62 lib.ht

Library

⇒ “notitle” (FileXmpPage) 3.40 on page 480
⇒ “notitle” (TextFileXmpPage) 3.107 on page 1307
⇒ “notitle” (KeyedAccessFileXmpPage) 3.57 on page 734

— **lib.ht** —

```
\begin{page}{LibraryXmpPage}{Library}
\beginscroll

The \spadtype{Library} domain provides a simple way to store Axiom
values in a file.  This domain is similar to
\spadtype{KeyedAccessFile} but fewer declarations are needed and items
of different types can be saved together in the same file.

\xtc{
To create a library, you supply a file name.
}{
\spadpaste{stuff := library "/tmp/Neat.stuff" \bound{stuff}}
}
\xtc{
Now values can be saved by key in the file.
The keys should be mnemonic, just as the field names are for records.
They can be given either as strings or symbols.
}{
\spadpaste{stuff.int    := 32**2     \free{stuff}\bound{stuffa}}
}
\xtc{
}{
\spadpaste{stuff."poly" := x**2 + 1  \free{stuffa}\bound{stuffb}}
}
\xtc{
}{
\spadpaste{stuff.str    := "Hello"   \free{stuffb}\bound{stuffc}}
}
\xtc{
You obtain
the set of available keys using the \spadfunFrom{keys}{Library} operation.
}{
\spadpaste{keys stuff \free{stuffa,stuffb,stuffc}\bound{stuffabc}}
}
\xtc{
You extract values  by giving the desired key in this way.
}{
\spadpaste{stuff.poly     \free{stuffb}}
}
\xtc{
}{
\spadpaste{stuff("poly")  \free{stuffb}}
}
\noOutputXtc{
When the file is no longer needed, you should remove it from the
file system.
}{
\spadpaste{)system rm -rf /tmp/Neat.stuff  \free{stuff}\bound{rmstuff}}
}
```

```
For more information on related topics, see
\downlink{`File'}{FileXmpPage}\ignore{File},
\downlink{`TextFile'}{TextFileXmpPage}\ignore{TextFile}, and
\downlink{`KeyedAccessFile'}{KeyedAccessFileXmpPage}
\ignore{KeyedAccessFile}.
\showBlurb{Library}
\endscroll
\autobuttons
\end{page}

\begin{patch}{LibraryXmpPagePatch1}
\begin{paste}{LibraryXmpPageFull1}{LibraryXmpPageEmpty1}
\pastebutton{LibraryXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{stuff := library "/tmp/Neat.stuff"\bound{stuff }}
\indentrel{3}\begin{verbatim}
   (1)  "/tmp/Neat.stuff"
                                          Type: Library
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LibraryXmpPageEmpty1}
\begin{paste}{LibraryXmpPageEmpty1}{LibraryXmpPagePatch1}
\pastebutton{LibraryXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{stuff := library "/tmp/Neat.stuff"\bound{stuff }}
\end{paste}\end{patch}

\begin{patch}{LibraryXmpPagePatch2}
\begin{paste}{LibraryXmpPageFull2}{LibraryXmpPageEmpty2}
\pastebutton{LibraryXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{stuff.int := 32**2\free{stuff }\bound{stuffa }}
\indentrel{3}\begin{verbatim}
   (2)  1024
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LibraryXmpPageEmpty2}
\begin{paste}{LibraryXmpPageEmpty2}{LibraryXmpPagePatch2}
\pastebutton{LibraryXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{stuff.int := 32**2\free{stuff }\bound{stuffa }}
\end{paste}\end{patch}

\begin{patch}{LibraryXmpPagePatch3}
\begin{paste}{LibraryXmpPageFull3}{LibraryXmpPageEmpty3}
\pastebutton{LibraryXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{stuff."poly" := x**2 + 1\free{stuffa }\bound{stuffb }}
\indentrel{3}\begin{verbatim}
         2
   (3)  x  + 1
                               Type: Polynomial Integer
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LibraryXmpPageEmpty3}
\begin{paste}{LibraryXmpPageEmpty3}{LibraryXmpPagePatch3}
\pastebutton{LibraryXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{stuff."poly" := x**2 + 1\free{stuffa }\bound{stuffb }}
\end{paste}\end{patch}

\begin{patch}{LibraryXmpPagePatch4}
\begin{paste}{LibraryXmpPageFull4}{LibraryXmpPageEmpty4}
\pastebutton{LibraryXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{stuff.str := "Hello"\free{stuffb }\bound{stuffc }}
\indentrel{3}\begin{verbatim}
   (4)  "Hello"
                                          Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LibraryXmpPageEmpty4}
\begin{paste}{LibraryXmpPageEmpty4}{LibraryXmpPagePatch4}
\pastebutton{LibraryXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{stuff.str := "Hello"\free{stuffb }\bound{stuffc }}
\end{paste}\end{patch}

\begin{patch}{LibraryXmpPagePatch5}
\begin{paste}{LibraryXmpPageFull5}{LibraryXmpPageEmpty5}
\pastebutton{LibraryXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{keys stuff\free{stuffa stuffb stuffc }\bound{stuffabc }}
\indentrel{3}\begin{verbatim}
   (5)  ["str","poly","int"]
                                     Type: List String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LibraryXmpPageEmpty5}
\begin{paste}{LibraryXmpPageEmpty5}{LibraryXmpPagePatch5}
\pastebutton{LibraryXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{keys stuff\free{stuffa stuffb stuffc }\bound{stuffabc }}
\end{paste}\end{patch}

\begin{patch}{LibraryXmpPagePatch6}
\begin{paste}{LibraryXmpPageFull6}{LibraryXmpPageEmpty6}
\pastebutton{LibraryXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{stuff.poly\free{stuffb }}
\indentrel{3}\begin{verbatim}
         2
   (6)  x  + 1
                              Type: Polynomial Integer
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LibraryXmpPageEmpty6}
\begin{paste}{LibraryXmpPageEmpty6}{LibraryXmpPagePatch6}
\pastebutton{LibraryXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{stuff.poly\free{stuffb }}
\end{paste}\end{patch}

\begin{patch}{LibraryXmpPagePatch7}
\begin{paste}{LibraryXmpPageFull7}{LibraryXmpPageEmpty7}
\pastebutton{LibraryXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{stuff("poly")\free{stuffb }}
\indentrel{3}\begin{verbatim}
        2
   (7)  x  + 1
                                 Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LibraryXmpPageEmpty7}
\begin{paste}{LibraryXmpPageEmpty7}{LibraryXmpPagePatch7}
\pastebutton{LibraryXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{stuff("poly")\free{stuffb }}
\end{paste}\end{patch}

\begin{patch}{LibraryXmpPagePatch8}
\begin{paste}{LibraryXmpPageFull8}{LibraryXmpPageEmpty8}
\pastebutton{LibraryXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{)system rm -rf /tmp/Neat.stuff\free{stuff }\bound{rmstuff }}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LibraryXmpPageEmpty8}
\begin{paste}{LibraryXmpPageEmpty8}{LibraryXmpPagePatch8}
\pastebutton{LibraryXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{)system rm -rf /tmp/Neat.stuff\free{stuff }\bound{rmstuff }}
\end{paste}\end{patch}
```

———

3.63 link.ht

The Axiom Link to NAG Software

⇒ "Introduction to the NAG Library Link" (nagLinkIntroPage) 18 on page 2494

⇒ "Access the Link from HyperDoc" (htxl1) 3.63 on page 845
⇒ "Browser pages for individual routines" (LispFunctions) 3.71 on page 952
⇒ "NAG Library Documentation" (FoundationLibraryDocPage) 3.1 on page 125

— **link.ht** —

```
\begin{page}{htxl}{The Axiom Link to NAG Software}
\beginscroll
\beginmenu
\item \menumemolink{Introduction to the NAG Library Link}
{nagLinkIntroPage}
\menumemolink{Access the Link from HyperDoc}{htxl1}
\menulispmemolink{Browser pages for individual routines}
{(|kSearch| "Nag*")}
\menumemolink{NAG Library Documentation}{FoundationLibraryDocPage}
\endmenu
\endscroll
\end{page}
```

———-

Use of the Link from HyperDoc

⇒ "notitle" (c02) 3.63 on page 846
⇒ "notitle" (c05) 3.63 on page 847
⇒ "notitle" (c06) 3.63 on page 847
⇒ "notitle" (d01) 3.63 on page 849
⇒ "notitle" (d02) 3.63 on page 851
⇒ "notitle" (d03) 3.63 on page 852
⇒ "notitle" (e01) 3.63 on page 853
⇒ "notitle" (e02) 3.63 on page 854
⇒ "notitle" (e04) 3.63 on page 856
⇒ "notitle" (f01) 3.63 on page 857
⇒ "notitle" (f02) 3.63 on page 858
⇒ "notitle" (f04) 3.63 on page 860
⇒ "notitle" (f07) 3.63 on page 862
⇒ "notitle" (s) 3.63 on page 863

— **link.ht** —

```
\begin{page}{htxl1}{Use of the Link from HyperDoc}
Click on the chapter of routines that you would like to use.
\beginscroll
\beginmenu
\menumemolink{C02}{c02}\tab{8} Zeros of Polynomials
\menumemolink{C05}{c05}\tab{8}
Roots of One or More Transcendental Equations
```

```
\menumemolink{C06}{c06}\tab{8} Summation of Series
\menumemolink{D01}{d01}\tab{8} Quadrature
\menumemolink{D02}{d02}\tab{8} Ordinary Differential Equations
\menumemolink{D03}{d03}\tab{8} Partial Differential Equations
\menumemolink{E01}{e01}\tab{8} Interpolation
\menumemolink{E02}{e02}\tab{8} Curve and Surface Fitting
\menumemolink{E04}{e04}\tab{8} Minimizing or Maximizing a Function
\menumemolink{F01}{f01}\tab{8} Matrix Operations, Including Inversion
\menumemolink{F02}{f02}\tab{8} Eigenvalues and Eigenvectors
\menumemolink{F04}{f04}\tab{8} Simultaneous Linear Equations
\menumemolink{F07}{f07}\tab{8} Linear Equations (LAPACK)
\menumemolink{S}{s}\tab{8} Approximations of Special Functions
\endmenu
\endscroll
\end{page}
```

———-

C02 Zeros of Polynomials

⇒ “Foundation Library Chapter c02 Manual Page” (manpageXXc02) 22.2 on page 2845
⇒ “Browse” (LispFunctions) 3.71 on page 952
⇒ “C02AFF” (LispFunctions) 3.71 on page 952
⇒ “C02AGF” (LispFunctions) 3.71 on page 952

— **link.ht** —

```
\begin{page}{c02}{C02 Zeros of Polynomials}
\beginscroll
\centerline{What would you like to do?}
\newline
\beginmenu
\item Read
\menuwindowlink{Foundation Library Chapter c02 Manual Page}
{manpageXXc02}
\item or
\menulispwindowlink{Browse}{(|kSearch| "NagPolynomialRootsPackage")}
\tab{10} through this chapter
\item or use the routines:
\menulispdownlink{C02AFF}{(|c02aff|)}\space{}
\tab{10}  All zeros of a complex polynomial
\menulispdownlink{C02AGF}{(|c02agf|)}\space{}
\tab{10}  All zeros of a real polynomial
\endmenu
\endscroll
\autobuttons
\end{page}
```

———-

C05 Roots of One or More Transcendental Equations

⇒ "Foundation Library Chapter c05 Manual Page" (manpageXXc05) 22.2 on page 2860
⇒ "Browse" (LispFunctions) 3.71 on page 952
⇒ "C05ADF" (LispFunctions) 3.71 on page 952
⇒ "C05NBF" (LispFunctions) 3.71 on page 952
⇒ "C05PBF" (LispFunctions) 3.71 on page 952

— **link.ht** —

```
\begin{page}{c05}{C05 Roots of One or More Transcendental Equations}
\beginscroll
\centerline{What would you like to do?}
\newline
\beginmenu
\item Read
\menuwindowlink{Foundation Library Chapter c05 Manual Page}
{manpageXXc05}
\item or
\menulispwindowlink{Browse}{(|kSearch| "NagRootFindingPackage")}
\tab{10} through this chapter
\item or use the routines:
\menulispdownlink{C05ADF}{(|c05adf|)}\space{}
\tab{10}  Zero of continuous function in given interval,
Bus and Dekker algorithm
\menulispdownlink{C05NBF}{(|c05nbf|)}\space{}
\tab{10}  Solution of system of nonlinear equations using
function values only
\menulispdownlink{C05PBF}{(|c05pbf|)}\space{}
\tab{10}  Solution of system of nonlinear equations using
1st derivatives
\endmenu
\endscroll
\autobuttons
\end{page}
```

———-

C06 Summation of Series

⇒ "Foundation Library Chapter c06 Manual Page" (manpageXXc06) 22.2 on page 2881

⇒ “Browse” (LispFunctions) 3.71 on page 952
⇒ “C06EAF” (LispFunctions) 3.71 on page 952
⇒ “C06EBF” (LispFunctions) 3.71 on page 952
⇒ “C06ECF” (LispFunctions) 3.71 on page 952
⇒ “C06EKF” (LispFunctions) 3.71 on page 952
⇒ “C06FPF” (LispFunctions) 3.71 on page 952
⇒ “C06FQF” (LispFunctions) 3.71 on page 952
⇒ “C06FRF” (LispFunctions) 3.71 on page 952
⇒ “C06FUF” (LispFunctions) 3.71 on page 952
⇒ “C06GBF” (LispFunctions) 3.71 on page 952
⇒ “C06GCF” (LispFunctions) 3.71 on page 952
⇒ “C06GQF” (LispFunctions) 3.71 on page 952
⇒ “C06GSF” (LispFunctions) 3.71 on page 952

— **link.ht** —

```
\begin{page}{c06}{C06 Summation of Series}
\beginscroll
\centerline{What would you like to do?}
\newline
\beginmenu
\item Read
\menuwindowlink{Foundation Library Chapter c06 Manual Page}
{manpageXXc06}
\item or
\menulispwindowlink{Browse}{(|kSearch| "NagSeriesSummationPackage")}
\tab{10} through this chapter
\item or use the routines:
\menulispdownlink{C06EAF}{(|c06eaf|)}\space{}
\tab{10}  Single 1-D real discrete Fourier transform, no extra
workspace
\menulispdownlink{C06EBF}{(|c06ebf|)}\space{}
\tab{10}  Single 1-D Hermitian discrete Fourier transform, no
extra workspace
\menulispdownlink{C06ECF}{(|c06ecf|)}\space{}
\tab{10}  Single 1-D complex discrete Fourier transform, no
extra workspace
\menulispdownlink{C06EKF}{(|c06ekf|)}\space{}
\tab{10}  Circular convolution or correlation of two real
vectors, no extra
workspace
\menulispdownlink{C06FPF}{(|c06fpf|)}\space{}
\tab{10}  Multiple 1-D real discrete Fourier transforms
\menulispdownlink{C06FQF}{(|c06fqf|)}\space{}
\tab{10}  Multiple 1-D Hermitian discrete Fourier transforms
\menulispdownlink{C06FRF}{(|c06frf|)}\space{}
\tab{10}  Multiple 1-D complex discrete Fourier transforms
\menulispdownlink{C06FUF}{(|c06fuf|)}\space{}
\tab{10}  2-D complex discrete Fourier transforms
\menulispdownlink{C06GBF}{(|c06gbf|)}\space{}
```

```
\tab{10}  Complex conjugate of Hermitian sequence
\menulispdownlink{C06GCF}{(|c06gcf|)}\space{}
\tab{10}  Complex conjugate of complex sequence
\menulispdownlink{C06GQF}{(|c06gqf|)}\space{}
\tab{10}  Complex conjugate of multiple Hermitian sequences
\menulispdownlink{C06GSF}{(|c06gsf|)}\space{}
\tab{10}  Convert Hermitian sequences to general complex sequences
\endmenu
\endscroll
\autobuttons
\end{page}
```

———-

D01 Quadrature

⇒ "Foundation Library Chapter d01 Manual Page" (manpageXXd01) 22.3 on page 2930
⇒ "Browse" (LispFunctions) 3.71 on page 952
⇒ "D01AJF" (LispFunctions) 3.71 on page 952
⇒ "D01AKF" (LispFunctions) 3.71 on page 952
⇒ "D01ALF" (LispFunctions) 3.71 on page 952
⇒ "D01AMF" (LispFunctions) 3.71 on page 952
⇒ "D01ANF" (LispFunctions) 3.71 on page 952
⇒ "D01APF" (LispFunctions) 3.71 on page 952
⇒ "D01AQF" (LispFunctions) 3.71 on page 952
⇒ "D01ASF" (LispFunctions) 3.71 on page 952
⇒ "D01BBF" (LispFunctions) 3.71 on page 952
⇒ "D01FCF" (LispFunctions) 3.71 on page 952
⇒ "D01GAF" (LispFunctions) 3.71 on page 952
⇒ "D01GBF" (LispFunctions) 3.71 on page 952

— **link.ht** —

```
\begin{page}{d01}{D01 Quadrature}
\beginscroll
\centerline{What would you like to do?}
\newline
\beginmenu
\item Read
\menuwindowlink{Foundation Library Chapter d01 Manual Page}
{manpageXXd01}
\item or
\menulispwindowlink{Browse}{(|kSearch| "NagIntegrationPackage")}
\tab{10} through this chapter
\item or use the routines:
\menulispdownlink{D01AJF}{(|d01ajf|)}\space{}
```

```
\tab{10}  1-D quadrature, adaptive, finite interval, strategy
due to Plessens
and de Doncker, allowing for badly-behaved integrands
\menulispdownlink{D01AKF}{(|d01akf|)}\space{}
\tab{10}  1-D quadrature, adaptive, finite interval, method
suitable for
oscillating functions
\menulispdownlink{D01ALF}{(|d01alf|)}\space{}
\tab{10}  1-D quadrature, adaptive, finite interval, allowing for
singularities at user specified points
\menulispdownlink{D01AMF}{(|d01amf|)}\space{}
\tab{10}  1-D quadrature, adaptive, infinite or semi-finite interval
\menulispdownlink{D01ANF}{(|d01anf|)}\space{}
\tab{10}  1-D quadrature, adaptive, finite interval, weight function
 cos(\omega x) or sin(\omega x)
\menulispdownlink{D01APF}{(|d01apf|)}\space{}
\tab{10}  1-D quadrature, adaptive, finite interval, weight function
with end point singularities of algebraico-logarithmic type
\menulispdownlink{D01AQF}{(|d01aqf|)}\space{}
\tab{10}  1-D quadrature, adaptive, finite interval, weight function
1/(x-c), Cauchy principle value (Hilbert transform)
\menulispdownlink{D01ASF}{(|d01asf|)}\space{}
\tab{10}  1-D quadrature, adaptive, semi-infinite interval,
weight function
cos(\omega x) or sin(\omega x)
\menulispdownlink{D01BBF}{(|d01bbf|)}\space{}
\tab{10} Pre-computed weights and abscissae for Gaussian
quadrature rules,
restricted choice of rule
\menulispdownlink{D01FCF}{(|d01fcf|)}\space{}
\tab{10} Multi-dimensional adaptive quadrature over
hyper-rectangle
\menulispdownlink{D01GAF}{(|d01gaf|)}\space{}
\tab{10} 1-D quadrature, integration of function defined
by data values,
Gill-Miller method
\menulispdownlink{D01GBF}{(|d01gbf|)}\space{}
\tab{10} Multi-dimensional quadrature over hyper-rectangle,
Monte Carlo method
\endmenu
\endscroll
\autobuttons
\end{page}
```

———

D02 Ordinary Differential Equations

⇒ "notitle" (manpageXXd02) 22.3 on page 3011

— **link.ht** —

```
\begin{page}{d02}{D02 Ordinary Differential Equations}
\pageto{Foundation Library Chapter d02 Manual Page}{manpageXXd02}
\pageto{Browse}{LispFunctions}
\pageto{D02BBF}{LispFunctions}
\pageto{D02BHF}{LispFunctions}
\pageto{D02CJF}{LispFunctions}
\pageto{D02EJF}{LispFunctions}
\pageto{D02GAF}{LispFunctions}
\pageto{D02GBF}{LispFunctions}
\pageto{D02KEF}{LispFunctions}
\pageto{D02RAF}{LispFunctions}
\beginscroll
\centerline{What would you like to do?}
\newline
\beginmenu
\item Read
\menuwindowlink{Foundation Library Chapter d02 Manual Page}
{manpageXXd02}
\item or
\menulispwindowlink{Browse}
{(|kSearch| "NagOrdinaryDifferentialEquationsPackage")}
\tab{10} through this chapter
\item or use the routines:
\menulispdownlink{D02BBF}{(|d02bbf|)}\space{}
\tab{10}  ODEs, IVP, Runge-Kutta-Merson method, over a range,
intermediate output
\menulispdownlink{D02BHF}{(|d02bhf|)}\space{}
\tab{10}  ODEs, IVP, Runge-Kutta-Merson method, until function of
solution is zero
\menulispdownlink{D02CJF}{(|d02cjf|)}\space{}
\tab{10}  ODEs, IVP, Adams method, until function of solution
is zero,
intermediate output
\menulispdownlink{D02EJF}{(|d02ejf|)}\space{}
\tab{10}  ODEs, stiff IVP, BDF method, until function of solution
is zero,
intermediate output
\menulispdownlink{D02GAF}{(|d02gaf|)}\space{}
\tab{10}  ODEs, boundary value problem, finite difference technique
with deferred correction, simple nonlinear problem
\menulispdownlink{D02GBF}{(|d02gbf|)}\space{}
\tab{10}  ODEs, boundary value problem, finite difference technique
with deferred correction, general nonlinear problem
\menulispdownlink{D02KEF}{(|d02kef|)}\space{}
```

```
\tab{10}  2nd order Sturm-Liouville problem, regular/singular system,
finite/infinite range, eigenvalue and eigenfunction, user-specified
break-points
\menulispdownlink{D02RAF}{(|d02raf|)}\space{}
\tab{10}  ODEs, general nonlinear boundary value problem, finite
difference technique with deferred correction, continuation facility
\endmenu
\endscroll
\autobuttons
\end{page}
```

———-

D03 Partial Differential Equations

⇒ "Foundation Library Chapter d03 Manual Page" (manpageXXd03) 22.3 on page 3104
⇒ "Browse" (LispFunctions) 3.71 on page 952
⇒ "D03EDF" (LispFunctions) 3.71 on page 952
⇒ "D03EEF" (LispFunctions) 3.71 on page 952
⇒ "D03FAF" (LispFunctions) 3.71 on page 952

— link.ht —

```
\begin{page}{d03}{D03 Partial Differential Equations}
\beginscroll
\centerline{What would you like to do?}
\newline
\beginmenu
\item Read
\menuwindowlink{Foundation Library Chapter d03 Manual Page}
{manpageXXd03}
\item or
\menulispwindowlink{Browse}
{(|kSearch| "NagPartialDifferentialEquationsPackage")}
\tab{10} through this chapter
\item or use the routines:
\menulispdownlink{D03EDF}{(|d03edf|)}\space{}
\tab{10}  Elliptic PDE, solution of finite difference equations
by a multigrid technique
\menulispdownlink{D03EEF}{(|d03eef|)}\space{}
\tab{10}  Discretize a 2nd order elliptic PDE on a rectangle
\menulispdownlink{D03FAF}{(|d03faf|)}\space{}
\tab{10}  Elliptic PDE, Helmholtz equation, 3-D Cartesian co-ordinates
\endmenu
\endscroll
\autobuttons
\end{page}
```

———-

E01 Interpolation

⇒ "Foundation Library Chapter e01 Manual Page" (manpageXXe01) 22.4 on page 3142
⇒ "Browse" (LispFunctions) 3.71 on page 952
⇒ "E01BAF" (LispFunctions) 3.71 on page 952
⇒ "E01BEF" (LispFunctions) 3.71 on page 952
⇒ "E01BFF" (LispFunctions) 3.71 on page 952
⇒ "E01BGF" (LispFunctions) 3.71 on page 952
⇒ "E01BHF" (LispFunctions) 3.71 on page 952
⇒ "E01DAF" (LispFunctions) 3.71 on page 952
⇒ "E01SAF" (LispFunctions) 3.71 on page 952
⇒ "E01SEF" (LispFunctions) 3.71 on page 952

— **link.ht** —

```
\begin{page}{e01}{E01 Interpolation}
\beginscroll
\centerline{What would you like to do?}
\beginmenu
\item Read
\menuwindowlink{Foundation Library Chapter e01 Manual Page}
{manpageXXe01}
\item or
\menulispwindowlink{Browse}{(|kSearch| "NagInterpolationPackage")}
\tab{10} through this chapter
\item or use the routines:
\menulispdownlink{E01BAF}{(|e01baf|)}\space{}
\tab{10}  Interpolating functions, cubic spline interpolant,
one variable
\menulispdownlink{E01BEF}{(|e01bef|)}\space{}
\tab{10}  Interpolating functions, monotonicity-preserving, piecewise
cubic Hermite, one variable
\menulispdownlink{E01BFF}{(|e01bff|)}\space{}
\tab{10}  Interpolated values, interpolant computed by E01BEF, function
only, one variable
\menulispdownlink{E01BGF}{(|e01bgf|)}\space{}
\tab{10}  Interpolated values, interpolant computed by E01BEF, function
and 1st derivative, one variable
\menulispdownlink{E01BHF}{(|e01bhf|)}\space{}
\tab{10}  Interpolated values, interpolant computed by E01BEF, definite
integral, one variable
\menulispdownlink{E01DAF}{(|e01daf|)}\space{}
\tab{10}  Interpolating functions, fitting bicubic spline, data on a
rectangular grid
```

```
\menulispdownlink{E01SAF}{(|e01saf|)}\space{}
\tab{10}  Interpolating functions, method of Renka and Cline,
two variables
\menulispdownlink{E01SEF}{(|e01sef|)}\space{}
\tab{10}  Interpolating functions, modified Shepherd's method,
two variables
\endmenu
\endscroll
\autobuttons
\end{page}
```

————-

E02 Curve and Surface Fitting

⇒ "Foundation Library Chapter e02 Manual Page" (manpageXXe02) 22.4 on page 3185
⇒ "Browse" (LispFunctions) 3.71 on page 952
⇒ "E02ADF" (LispFunctions) 3.71 on page 952
⇒ "E02AEF" (LispFunctions) 3.71 on page 952
⇒ "E02AGF" (LispFunctions) 3.71 on page 952
⇒ "E02AHF" (LispFunctions) 3.71 on page 952
⇒ "E02AJF" (LispFunctions) 3.71 on page 952
⇒ "E02AKF" (LispFunctions) 3.71 on page 952
⇒ "E02BAF" (LispFunctions) 3.71 on page 952
⇒ "E02BBF" (LispFunctions) 3.71 on page 952
⇒ "E02BCF" (LispFunctions) 3.71 on page 952
⇒ "E02BDF" (LispFunctions) 3.71 on page 952
⇒ "E02BEF" (LispFunctions) 3.71 on page 952
⇒ "E02DAF" (LispFunctions) 3.71 on page 952
⇒ "E02DCF" (LispFunctions) 3.71 on page 952
⇒ "E02DDF" (LispFunctions) 3.71 on page 952
⇒ "E02DEF" (LispFunctions) 3.71 on page 952
⇒ "E02DFF" (LispFunctions) 3.71 on page 952
⇒ "E02GAF" (LispFunctions) 3.71 on page 952
⇒ "E02ZAF" (LispFunctions) 3.71 on page 952

— **link.ht** —

```
\begin{page}{e02}{E02 Curve and Surface Fitting}
\beginscroll
\centerline{What would you like to do?}
\beginmenu
\item Read
\menuwindowlink{Foundation Library Chapter e02 Manual Page}
{manpageXXe02}
\item or
```

```
\menulispwindowlink{Browse}{(|kSearch| "NagFittingPackage")}
\tab{10} through this chapter
\item or use the routines:
\menulispdownlink{E02ADF}{(|e02adf|)}\space{}
\tab{10}  Least-squares curve fit, by polynomials, arbitrary
data points
\menulispdownlink{E02AEF}{(|e02aef|)}\space{}
\tab{10}  Evaluation of fitted polynomial in one variable from
Chebyshev series
form (simplified parameter list)
\menulispdownlink{E02AGF}{(|e02agf|)}\space{}
\tab{10}  Least-squares polynomial fit, values and derivatives
may be
constrained, arbitrary data points
\menulispdownlink{E02AHF}{(|e02ahf|)}\space{}
\tab{10}  Derivative of fitted polynomial in Chebyshev series form
\menulispdownlink{E02AJF}{(|e02ajf|)}\space{}
\tab{10}  Integral of fitted polynomial in Chebyshev series form
\menulispdownlink{E02AKF}{(|e02akf|)}\space{}
\tab{10}  Evaluation of fitted polynomial in one variable, from
Chebyshev
series form
\menulispdownlink{E02BAF}{(|e02baf|)}\space{}
\tab{10}  Least-squares curve cubic spline fit (including
interpolation)
\menulispdownlink{E02BBF}{(|e02bbf|)}\space{}
\tab{10}  Evaluation of fitted cubic spline, function only
\menulispdownlink{E02BCF}{(|e02bcf|)}\space{}
\tab{10}  Evaluation of fitted cubic spline, function and derivatives
\menulispdownlink{E02BDF}{(|e02bdf|)}\space{}
\tab{10}  Evaluation of fitted cubic spline, definite integral
\menulispdownlink{E02BEF}{(|e02bef|)}\space{}
\tab{10}  Least-squares curve cubic spline fit, automatic knot placement
\menulispdownlink{E02DAF}{(|e02daf|)}\space{}
\tab{10}  Least-squares surface fit, bicubic splines
\menulispdownlink{E02DCF}{(|e02dcf|)}\space{}
\tab{10}  Least-squares surface fit by bicubic splines with automatic
knot placement, data on a rectangular grid
\menulispdownlink{E02DDF}{(|e02ddf|)}\space{}
\tab{10}  Least-squares surface fit by bicubic splines with automatic
knot placement, scattered data
\menulispdownlink{E02DEF}{(|e02def|)}\space{}
\tab{10}  Evaluation of a fitted bicubic spline at a vector of points
\menulispdownlink{E02DFF}{(|e02dff|)}\space{}
\tab{10}  Evaluation of a fitted bicubic spline at a mesh of points
\menulispdownlink{E02GAF}{(|e02gaf|)}\space{}
\tab{10}  \htbitmap{l1}-approximation by general linear function
\menulispdownlink{E02ZAF}{(|e02zaf|)}\space{}
\tab{10}  Sort 2-D sata into panels for fitting bicubic splines
\endmenu
```

```
\endscroll
\autobuttons
\end{page}
```

———-

E04 Minimizing or Maximizing a Function

⇒ "Foundation Library Chapter e04 Manual Page" (manpageXXe04) 22.4 on page 3322
⇒ "Browse" (LispFunctions) 3.71 on page 952
⇒ "E04DGF" (LispFunctions) 3.71 on page 952
⇒ "E04FDF" (LispFunctions) 3.71 on page 952
⇒ "E04GCF" (LispFunctions) 3.71 on page 952
⇒ "E04JAF" (LispFunctions) 3.71 on page 952
⇒ "E04MBF" (LispFunctions) 3.71 on page 952
⇒ "E04NAF" (LispFunctions) 3.71 on page 952
⇒ "E04UCF" (LispFunctions) 3.71 on page 952
⇒ "E04YCF" (LispFunctions) 3.71 on page 952

— **link.ht** —

```
\begin{page}{e04}{E04 Minimizing or Maximizing a Function}
\beginscroll
\centerline{What would you like to do?}
\newline
\beginmenu
\item Read
\menuwindowlink{Foundation Library Chapter e04 Manual Page}
{manpageXXe04}
\item or
\menulispwindowlink{Browse}{(|kSearch| "NagOptimisationPackage")}
\tab{10} through this chapter
\item or use the routines:
\menulispdownlink{E04DGF}{(|e04dgf|)}\space{}
\tab{10}  Unconstrained minimum, pre-conditioned conjugate gradient
algorithm, function of several variables using 1st derivatives
\menulispdownlink{E04FDF}{(|e04fdf|)}\space{}
\tab{10}  Unconstrained minimum of a sum of squares, combined
Gauss-Newton and modified Newton algorithm using function values only
\menulispdownlink{E04GCF}{(|e04gcf|)}\space{}
\tab{10}  Unconstrained minimum, of a sum of squares, combined
Gauss-Newton and modified Newton algorithm using 1st derivatives
\menulispdownlink{E04JAF}{(|e04jaf|)}\space{}
\tab{10}  Minimum, function of several variables, quasi-Newton
algorithm, simple bounds, using function values only
\menulispdownlink{E04MBF}{(|e04mbf|)}\space{}
\tab{10}  Linear programming problem
```

```
\menulispdownlink{E04NAF}{(|e04naf|)}\space{}
\tab{10}  Quadratic programming problem
\menulispdownlink{E04UCF}{(|e04ucf|)}\space{}
\tab{10}  Minimum, function of several variables, sequential QP method,
nonlinear constraints, using function values and optionally 1st
derivatives
\menulispdownlink{E04YCF}{(|e04ycf|)}\space{}
\tab{10}  Covariance matrix for non-linear least-squares problem
\endmenu
\endscroll
\autobuttons
\end{page}
```

———-

F01 Matrix Operations - Including Inversion

⇒ "Foundation Library Chapter f Manual Page" (manpageXXf) 22.5 on page 3478
⇒ "Foundation Library Chapter f01 Manual Page" (manpageXXf01) 22.5 on page 3482
⇒ "Browse" (LispFunctions) 3.71 on page 952
⇒ "F01BRF" (LispFunctions) 3.71 on page 952
⇒ "F01BSF" (LispFunctions) 3.71 on page 952
⇒ "F01MAF" (LispFunctions) 3.71 on page 952
⇒ "F01MCF" (LispFunctions) 3.71 on page 952
⇒ "F01QCF" (LispFunctions) 3.71 on page 952
⇒ "F01QDF" (LispFunctions) 3.71 on page 952
⇒ "F01QEF" (LispFunctions) 3.71 on page 952
⇒ "F01RCF" (LispFunctions) 3.71 on page 952
⇒ "F01RDF" (LispFunctions) 3.71 on page 952
⇒ "F01REF" (LispFunctions) 3.71 on page 952

— **link.ht** —

```
\begin{page}{f01}{F01 Matrix Operations - Including Inversion}
\beginscroll
\centerline{What would you like to do?}
\beginmenu
\item Read
\menuwindowlink{Foundation Library Chapter f Manual Page}
{manpageXXf}
\menuwindowlink{Foundation Library Chapter f01 Manual Page}
{manpageXXf01}
\item or
\menulispwindowlink{Browse}{(|kSearch| "NagMatrixOperationsPackage")}
\tab{10} through this chapter
\item or use the routines:
```

```
\menulispdownlink{F01BRF}{(|f01brf|)}\space{}
\tab{10}  {\it LU} factorization of real sparse matrix
\menulispdownlink{F01BSF}{(|f01bsf|)}\space{}
\tab{10}  {\it LU} factorization of real sparse matrix with known
sparsity pattern
\menulispdownlink{F01MAF}{(|f01maf|)}\space{}
\tab{10}  \htbitmap{llt} factorization of real sparse
symmetric positive-definite matrix
\menulispdownlink{F01MCF}{(|f01mcf|)}\space{}
\tab{10}  \htbitmap{ldlt} factorization of real
symmetric positive-definite variable-bandwith matrix
\menulispdownlink{F01QCF}{(|f01qcf|)}\space{}
\tab{10}  {\it QR} factorization of real {\it m} by {\it n} matrix
(m \htbitmap{great=} n)
\menulispdownlink{F01QDF}{(|f01qdf|)}\space{}
\tab{10} Operations with orthogonal matrices, compute {\it QB} or
\htbitmap{f01qdf} after factorization by F01QCF or F01QFF
\menulispdownlink{F01QEF}{(|f01qef|)}\space{}
\tab{10} Operations with orthogonal matrices, form columns of {\it Q}
after factorization by F01QCF or F01QFF
\menulispdownlink{F01RCF}{(|f01rcf|)}\space{}
\tab{10} {\it QR} factorization of complex {\it m} by {\it n} matrix
(m \htbitmap{great=} n)
\menulispdownlink{F01RDF}{(|f01rdf|)}\space{}
\tab{10} Operations with unitary matrices, compute {\it QB} or
\htbitmap{f01rdf} after factorization by F01RCF
\menulispdownlink{F01REF}{(|f01ref|)}\space{}
\tab{10} Operations with unitary matrices, form columns of {\it Q}
after factorization by F01RCF
\endmenu
\endscroll
\autobuttons
\end{page}
```

F02 Eigenvalues and Eigenvectors

⇒ "Foundation Library Chapter f Manual Page" (manpageXXf) 22.5 on page 3478
⇒ "Foundation Library Chapter f02 Manual Page" (manpageXXf02) 22.5 on page 3543
⇒ "Browse" (LispFunctions) 3.71 on page 952
⇒ "F02AAF" (LispFunctions) 3.71 on page 952
⇒ "F02ABF" (LispFunctions) 3.71 on page 952
⇒ "F02ADF" (LispFunctions) 3.71 on page 952
⇒ "F02AEF" (LispFunctions) 3.71 on page 952
⇒ "F02AFF" (LispFunctions) 3.71 on page 952

⇒ "F02AGF" (LispFunctions) 3.71 on page 952
⇒ "F02AJF" (LispFunctions) 3.71 on page 952
⇒ "F02AKF" (LispFunctions) 3.71 on page 952
⇒ "F02AWF" (LispFunctions) 3.71 on page 952
⇒ "F02AXF" (LispFunctions) 3.71 on page 952
⇒ "F02BBF" (LispFunctions) 3.71 on page 952
⇒ "F02BJF" (LispFunctions) 3.71 on page 952
⇒ "F02FJF" (LispFunctions) 3.71 on page 952
⇒ "F02WEF" (LispFunctions) 3.71 on page 952
⇒ "F02XEF" (LispFunctions) 3.71 on page 952

— **link.ht** —

```
\begin{page}{f02}{F02 Eigenvalues and Eigenvectors}
\beginscroll
\centerline{What would you like to do?}
\beginmenu
\item Read
\menuwindowlink{Foundation Library Chapter f Manual Page}{manpageXXf}
\menuwindowlink{Foundation Library Chapter f02 Manual Page}
{manpageXXf02}
\item or
\menulispwindowlink{Browse}{(|kSearch| "NagEigenPackage")}
\tab{10} through this chapter
\item or use the routines:
\menulispdownlink{F02AAF}{(|f02aaf|)}\space{}
\tab{10}  All eigenvalues of real symmetric matrix (Black box)
\menulispdownlink{F02ABF}{(|f02abf|)}\space{}
\tab{10}  All eigenvalues and eigenvectors of real symmetric
matrix (Black box)
\menulispdownlink{F02ADF}{(|f02adf|)}\space{}
\tab{10}  All eigenvalues of generalized real eigenproblem of the form
Ax = \lambda Bx where A and B are symmetric and B is positive definite
\menulispdownlink{F02AEF}{(|f02aef|)}\space{}
\tab{10}  All eigenvalues and eigenvectors of generalized real
eigenproblem of the form Ax = \lambda Bx where A and B are symmetric
and B is positive definite
\menulispdownlink{F02AFF}{(|f02aff|)}\space{}
\tab{10}  All eigenvalues of real matrix (Black box)
\menulispdownlink{F02AGF}{(|f02agf|)}\space{}
\tab{10}  All eigenvalues and eigenvectors of real matrix (Black box)
\menulispdownlink{F02AJF}{(|f02ajf|)}\space{}
\tab{10}  All eigenvalues of complex matrix (Black box)
\menulispdownlink{F02AKF}{(|f02akf|)}\space{}
\tab{10}  All eigenvalues and eigenvectors of complex matrix
(Black box)
\menulispdownlink{F02AWF}{(|f02awf|)}\space{}
\tab{10}  All eigenvalues of complex Hermitian matrix (Black box)
\menulispdownlink{F02AXF}{(|f02axf|)}\space{}
\tab{10}  All eigenvalues and eigenvectors of complex Hermitian
```

```
matrix (Black box)
\menulispdownlink{F02BBF}{(|f02bbf|)}\space{}
\tab{10}  Selected eigenvalues and eigenvectors of real symmetric
matrix (Black box)
\menulispdownlink{F02BJF}{(|f02bjf|)}\space{}
\tab{10}  All eigenvalues and optionally eigenvectors of generalized
eigenproblem by {\it QZ} algorithm, real matrices (Black box)
\menulispdownlink{F02FJF}{(|f02fjf|)}\space{}
\tab{10}  Selected eigenvalues and eigenvectors of sparse symmetric
eigenproblem
\menulispdownlink{F02WEF}{(|f02wef|)}\space{}
\tab{10}  SVD of real matrix
\menulispdownlink{F02XEF}{(|f02xef|)}\space{}
\tab{10}  SVD of complex matrix
\endmenu
\endscroll
\autobuttons
\end{page}
```

———-

F04 Simultaneous Linear Equations

⇒ "Foundation Library Chapter f Manual Page" (manpageXXf) 22.5 on page 3478
⇒ "Foundation Library Chapter f04 Manual Page" (manpageXXf04) 22.5 on page 3615
⇒ "Browse" (LispFunctions) 3.71 on page 952
⇒ "F04ADF" (LispFunctions) 3.71 on page 952
⇒ "F04ARF" (LispFunctions) 3.71 on page 952
⇒ "F04ASF" (LispFunctions) 3.71 on page 952
⇒ "F04ATF" (LispFunctions) 3.71 on page 952
⇒ "F04AXF" (LispFunctions) 3.71 on page 952
⇒ "F04FAF" (LispFunctions) 3.71 on page 952
⇒ "F04JGF" (LispFunctions) 3.71 on page 952
⇒ "F04MAF" (LispFunctions) 3.71 on page 952
⇒ "F04MBF" (LispFunctions) 3.71 on page 952
⇒ "F04MCF" (LispFunctions) 3.71 on page 952
⇒ "F04QAF" (LispFunctions) 3.71 on page 952

— **link.ht** —

```
\begin{page}{f04}{F04 Simultaneous Linear Equations}
\beginscroll
\centerline{What would you like to do?}
\beginmenu
\item Read
\menuwindowlink{Foundation Library Chapter f Manual Page}
```

```
{manpageXXf}
\menuwindowlink{Foundation Library Chapter f04 Manual Page}
{manpageXXf04}
\item or
\menulispwindowlink{Browse}
{(|kSearch| "NagLinearEquationSolvingPackage")}\tab{10} through
this chapter
\item or use the routines:
\menulispdownlink{F04ADF}{(|f04adf|)}\space{}
\tab{10}  Solution of complex simultaneous linear equations, with
multiple right-hand sides (Black box)
\menulispdownlink{F04ARF}{(|f04arf|)}\space{}
\tab{10}  Solution of real simultaneous linear equations, one
right-hand side
(Black box)
\menulispdownlink{F04ASF}{(|f04asf|)}\space{}
\tab{10}  Solution of real symmetric positive-definite simultaneous
linear equations, one right-hand side using iterative refinement
(Black box)
\menulispdownlink{F04ATF}{(|f04atf|)}\space{}
\tab{10}  Solution of real simultaneous linear equations, one
right-hand side
using iterative refinement (Black box)
\menulispdownlink{F04AXF}{(|f04axf|)}\space{}
\tab{10}  Approximate solution of real sparse simultaneous linear
equations (coefficient matrix already factorized by F01BRF or F01BSF)
\menulispdownlink{F04FAF}{(|f04faf|)}\space{}
\tab{10}  Solution of real symmetric positive-definite tridiagonal
simultaneous linear equations, one right-hand side (Black box)
\menulispdownlink{F04JGF}{(|f04jgf|)}\space{}
\tab{10}  Least-squares (if rank = n) or minimal least-squares
(if rank < n) solution of m real equations in n unknowns, rank
\htbitmap{less=} n, m \htbitmap{great=} n
\menulispdownlink{F04MAF}{(|f04maf|)}\space{}
\tab{10}  Real sparse symmetric positive-definite simultaneous linear
equations(coefficient matrix already factorized)
\menulispdownlink{F04MBF}{(|f04mbf|)}\space{}
\tab{10}  Real sparse symmetric simultaneous linear equations
\menulispdownlink{F04MCF}{(|f04mcf|)}\space{}
\tab{10}  Approximate solution of real symmetric positive-definite
variable-bandwidth simultaneous linear equations (coefficient matrix
already factorized)
\menulispdownlink{F04QAF}{(|f04qaf|)}\space{}
\tab{10}  Sparse linear least-squares problem, {\it m} real equations
in {\it n} unknowns
\endmenu
\endscroll
\autobuttons
\end{page}
```

F07 Linear Equations (LAPACK)

⇒ "Foundation Library Chapter f Manual Page" (manpageXXf) 22.5 on page 3478
⇒ "Foundation Library Chapter f07 Manual Page" (manpageXXf07) 22.5 on page 3718
⇒ "Browse" (LispFunctions) 3.71 on page 952
⇒ "F07ADF" (LispFunctions) 3.71 on page 952
⇒ "F07AEF" (LispFunctions) 3.71 on page 952
⇒ "F07FDF" (LispFunctions) 3.71 on page 952
⇒ "F07FEF" (LispFunctions) 3.71 on page 952

— **link.ht** —

```
\begin{page}{f07}{F07 Linear Equations (LAPACK)}
\beginscroll
\centerline{What would you like to do?}
\beginmenu
\item Read
\menuwindowlink{Foundation Library Chapter f Manual Page}{manpageXXf}
\menuwindowlink{Foundation Library Chapter f07 Manual Page}{manpageXXf07}
\item or
\menulispwindowlink{Browse}
{(|kSearch| "NagLapack")}\tab{10} through this chapter
\item or use the routines:
\menulispdownlink{F07ADF}{(|f07adf|)}\space{}
\tab{10}  (DGETRF) {\it LU} factorization of real
{\it m} by {\it n} matrix
\menulispdownlink{F07AEF}{(|f07aef|)}\space{}
\tab{10}  (DGETRS) Solution of real system of linear equations,
multiple right hand sides, matrix factorized by F07ADF
\menulispdownlink{F07FDF}{(|f07fdf|)}\space{}
\tab{10}  (DPOTRF) Cholesky factorization of real symmetric
positive-definite matrix
\menulispdownlink{F07FEF}{(|f07fef|)}\space{}
\tab{10}  (DPOTRS) Solution of real symmetric positive-definite
system of linear equations, multiple right-hand sides, matrix
already factorized by F07FDF
\endmenu
\endscroll
\autobuttons
\end{page}
```

S – Approximations of Special Functions

⇒ "Foundation Library Chapter s Manual Page" (manpageXXs) 22.6 on page 3754
⇒ "Browse" (LispFunctions) 3.71 on page 952
⇒ "S01EAF" (LispFunctions) 3.71 on page 952
⇒ "S13AAF" (LispFunctions) 3.71 on page 952
⇒ "S13ACF" (LispFunctions) 3.71 on page 952
⇒ "S13ADF" (LispFunctions) 3.71 on page 952
⇒ "S14AAF" (LispFunctions) 3.71 on page 952
⇒ "S14ABF" (LispFunctions) 3.71 on page 952
⇒ "S14BAF" (LispFunctions) 3.71 on page 952
⇒ "S15ADF" (LispFunctions) 3.71 on page 952
⇒ "S15AEF" (LispFunctions) 3.71 on page 952
⇒ "S17ACF" (LispFunctions) 3.71 on page 952
⇒ "S17ADF" (LispFunctions) 3.71 on page 952
⇒ "S17AEF" (LispFunctions) 3.71 on page 952
⇒ "S17AFF" (LispFunctions) 3.71 on page 952
⇒ "S17AGF" (LispFunctions) 3.71 on page 952
⇒ "S17AHF" (LispFunctions) 3.71 on page 952
⇒ "S17AJF" (LispFunctions) 3.71 on page 952
⇒ "S17AKF" (LispFunctions) 3.71 on page 952
⇒ "S17DCF" (LispFunctions) 3.71 on page 952
⇒ "S17DEF" (LispFunctions) 3.71 on page 952
⇒ "S17DGF" (LispFunctions) 3.71 on page 952
⇒ "S17DHF" (LispFunctions) 3.71 on page 952
⇒ "S17DLF" (LispFunctions) 3.71 on page 952
⇒ "S18ACF" (LispFunctions) 3.71 on page 952
⇒ "S18ADF" (LispFunctions) 3.71 on page 952
⇒ "S18AEF" (LispFunctions) 3.71 on page 952
⇒ "S18AFF" (LispFunctions) 3.71 on page 952
⇒ "S18DCF" (LispFunctions) 3.71 on page 952
⇒ "S18DEF" (LispFunctions) 3.71 on page 952
⇒ "S19AAF" (LispFunctions) 3.71 on page 952
⇒ "S19ABF" (LispFunctions) 3.71 on page 952
⇒ "S19ACF" (LispFunctions) 3.71 on page 952
⇒ "S19ADF" (LispFunctions) 3.71 on page 952
⇒ "S20ACF" (LispFunctions) 3.71 on page 952
⇒ "S20ADF" (LispFunctions) 3.71 on page 952
⇒ "S21BAF" (LispFunctions) 3.71 on page 952
⇒ "S21BBF" (LispFunctions) 3.71 on page 952
⇒ "S21BCF" (LispFunctions) 3.71 on page 952
⇒ "S21BDF" (LispFunctions) 3.71 on page 952

— **link.ht** —

```
\begin{page}{s}{S \space{2} Approximations of Special Functions}
\beginscroll
\centerline{What would you like to do?}
\beginmenu
\item Read
\menuwindowlink{Foundation Library Chapter s Manual Page}{manpageXXs}
\item or
\menulispwindowlink{Browse}{(|kSearch| "NagSpecialFunctionsPackage")}
\tab{10} through this chapter
\item or use the routines:
\menulispdownlink{S01EAF}{(|s01eaf|)}\space{}
\tab{10}  Complex exponential {\em exp(z)}
\menulispdownlink{S13AAF}{(|s13aaf|)}\space{}
\tab{10}  Exponential integral \htbitmap{s13aaf2}
\menulispdownlink{S13ACF}{(|s13acf|)}\space{}
\tab{10}  Cosine integral {\em Ci(x)}
\menulispdownlink{S13ADF}{(|s13adf|)}\space{}
\tab{10}  Sine integral {\em Si(x)}
\menulispdownlink{S14AAF}{(|s14aaf|)}\space{}
\tab{10}  Gamma function \Gamma
\menulispdownlink{S14ABF}{(|s14abf|)}\space{}
\tab{10}  Log Gamma function {\em ln \Gamma}
\menulispdownlink{S14BAF}{(|s14baf|)}\space{}
\tab{10}  Incomplete gamma functions P(a,x) and Q(a,x)
\menulispdownlink{S15ADF}{(|s15adf|)}\space{}
\tab{10}  Complement of error function {\em erfc x }
\menulispdownlink{S15AEF}{(|s15aef|)}\space{}
\tab{10}  Error function {\em erf x}
\menulispdownlink{S17ACF}{(|s17acf|)}\space{}
\tab{10}  Bessel function \space{1} \htbitmap{s17acf}
\menulispdownlink{S17ADF}{(|s17adf|)}\space{}
\tab{10}  Bessel function \space{1} \htbitmap{s17adf}
\menulispdownlink{S17AEF}{(|s17aef|)}\space{}
\tab{10}  Bessel function \space{1} \htbitmap{s17aef1}
\menulispdownlink{S17AFF}{(|s17aff|)}\space{}
\tab{10}  Bessel function \space{1} \htbitmap{s17aff1}
\menulispdownlink{S17AGF}{(|s17agf|)}
\tab{10}  Airy function {\em Ai(x)}
\menulispdownlink{S17AHF}{(|s17ahf|)}
\tab{10}  Airy function {\em Bi(x)}
\menulispdownlink{S17AJF}{(|s17ajf|)}
\tab{10}  Airy function {\em Ai'(x)}
\menulispdownlink{S17AKF}{(|s17akf|)}
\tab{10}  Airy function {\em Bi'(x)}
\menulispdownlink{S17DCF}{(|s17dcf|)}
\tab{10} Bessel function \htbitmap{s17dcf}, real a \space{1}
\htbitmap{great=} 0, complex z, v = 0,1,2,...
\menulispdownlink{S17DEF}{(|s17def|)}
\tab{10} Bessel function \htbitmap{s17def}, real a \space{1}
\htbitmap{great=} 0, complex z, v = 0,1,2,...
```

```
\menulispdownlink{S17DGF}{(|s17dgf|)}
\tab{10} Airy function {\em Ai(z)} and {\em Ai'(z)}, complex z
\menulispdownlink{S17DHF}{(|s17dhf|)}
\tab{10} Airy function {\em Bi(z)} and {\em Bi'(z)}, complex z
\menulispdownlink{S17DLF}{(|s17dlf|)}
\tab{10} Hankel function \vspace{-32} \htbitmap{s17dlf}
\vspace{-37}, j = 1,2, real a \space{1} \htbitmap{great=} 0,
complex z, v = 0,1,2,... \newline
\menulispdownlink{S18ACF}{(|s18acf|)}
\tab{10} Modified Bessel function \space{1} \htbitmap{s18acf1}
\menulispdownlink{S18ADF}{(|s18adf|)}
\tab{10} Modified Bessel function \space{1} \htbitmap{s18adf1}
\menulispdownlink{S18AEF}{(|s18aef|)}
\tab{10} Modified Bessel function \space{1} \htbitmap{s18aef1}
\menulispdownlink{S18AFF}{(|s18aff|)}
\tab{10} Modified Bessel function \space{1} \htbitmap{s18aff1}
\menulispdownlink{S18DCF}{(|s18dcf|)}
\tab{10} Modified bessel function \htbitmap{s18dcf}, real a \space{1}
\htbitmap{great=} 0, complex z, v = 0,1,2,...
\menulispdownlink{S18DEF}{(|s18def|)}
\tab{10} Modified bessel function \htbitmap{s18def}, real a \space{1}
\htbitmap{great=} 0, complex z, v = 0,1,2,...
\menulispdownlink{S19AAF}{(|s19aaf|)}
\tab{10} Kelvin function {\em ber x}
\menulispdownlink{S19ABF}{(|s19abf|)}
\tab{10} Kelvin function {\em bei x}
\menulispdownlink{S19ACF}{(|s19acf|)}
\tab{10} Kelvin function {\em ker x}
\menulispdownlink{S19ADF}{(|s19adf|)}
\tab{10} Kelvin function {\em kei x}
\menulispdownlink{S20ACF}{(|s20acf|)}
\tab{10} Fresnel integral {\em S(x)}
\menulispdownlink{S20ADF}{(|s20adf|)}
\tab{10} Fresnel integral {\em C(x)}
\menulispdownlink{S21BAF}{(|s21baf|)}
\tab{10} Degenerate symmetrised elliptic integral of 1st kind
\space{1} \htbitmap{s21baf1}
\menulispdownlink{S21BBF}{(|s21bbf|)}
\tab{10} Symmetrised elliptic integral of 1st kind \space{1}
\vspace{-28} \htbitmap{s21bbf1} \vspace{-40}
\menulispdownlink{S21BCF}{(|s21bcf|)}
\tab{10} Symmetrised elliptic integral of 2nd kind \space{1}
\vspace{-28} \htbitmap{s21bcf1} \vspace{-40}
\menulispdownlink{S21BDF}{(|s21bdf|)}
\tab{10} Symmetrised elliptic integral of 3rd kind \space{1}
\vspace{-26} \htbitmap{s21bdf1} \vspace{-40}
\endmenu
\endscroll
\autobuttons
\end{page}
```

3.64 list.ht

List

⇒ “notitle” (ugxListCreatePage) 3.64 on page 867
⇒ “notitle” (ugxListAccessPage) 3.64 on page 869
⇒ “notitle” (ugxListChangePage) 3.64 on page 875
⇒ “notitle” (ugxListOtherPage) 3.64 on page 879
⇒ “notitle” (ugxListDotPage) 3.64 on page 882

— **list.ht** —

```
\begin{page}{ListXmpPage}{List}
\beginscroll
A \spadgloss{list} is a finite collection of elements in a specified
order that can contain duplicates.
A list is a convenient structure to work with because it is easy
to add or remove elements and the length need not be constant.
There are many different kinds of lists in Axiom, but the
default types (and those used most often) are created by the
\spadtype{List} constructor.
For example, there are objects of type \spadtype{List Integer},
\spadtype{List Float} and \spadtype{List Polynomial Fraction Integer}.
Indeed, you can even have \spadtype{List List List Boolean}
(that is, lists of lists of lists of Boolean values).
You can have lists of any type of Axiom object.

\beginmenu
    \menudownlink{{9.47.1. Creating Lists}}{ugxListCreatePage}
    \menudownlink{{9.47.2. Accessing List Elements}}{ugxListAccessPage}
    \menudownlink{{9.47.3. Changing List Elements}}{ugxListChangePage}
    \menudownlink{{9.47.4. Other Functions}}{ugxListOtherPage}
    \menudownlink{{9.47.5. Dot, Dot}}{ugxListDotPage}
\endmenu
\endscroll
\autobuttons
\end{page}
```

Creating Lists

— list.ht —

```
\begin{page}{ugxListCreatePage}{Creating Lists}
\beginscroll

The easiest way to create a list with, for example, the elements
\spad{2, 4, 5, 6} is to enclose the elements with square
brackets and separate the elements with commas.
\xtc{
The spaces after the commas are optional, but they do improve the
readability.
}{
\spadpaste{[2, 4, 5, 6]}
}
\xtc{
To create a list with the single element \spad{1}, you can use
either \spad{[1]} or the operation \spadfunFrom{list}{List}.
}{
\spadpaste{[1]}
}
\xtc{
}{
\spadpaste{list(1)}
}
\xtc{
Once created, two lists \spad{k} and \spad{m} can be
concatenated by issuing \spad{append(k,m)}.
\spadfunFrom{append}{List} does {\it not} physically join the lists,
but rather produces a new list with the elements coming from the two
arguments.
}{
\spadpaste{append([1,2,3],[5,6,7])}
}
\xtc{
Use \spadfunFrom{cons}{List} to append an element onto the front of a
list.
}{
\spadpaste{cons(10,[9,8,7])}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxListCreatePagePatch1}
\begin{paste}{ugxListCreatePageFull1}{ugxListCreatePageEmpty1}
\pastebutton{ugxListCreatePageFull1}{\hidepaste}
```

```
\tab{5}\spadcommand{[2, 4, 5, 6]}
\indentrel{3}\begin{verbatim}
   (1)  [2,4,5,6]
                              Type: List PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxListCreatePageEmpty1}
\begin{paste}{ugxListCreatePageEmpty1}{ugxListCreatePagePatch1}
\pastebutton{ugxListCreatePageEmpty1}{\showpaste}
\tab{5}\spadcommand{[2, 4, 5, 6]}
\end{paste}\end{patch}

\begin{patch}{ugxListCreatePagePatch2}
\begin{paste}{ugxListCreatePageFull2}{ugxListCreatePageEmpty2}
\pastebutton{ugxListCreatePageFull2}{\hidepaste}
\tab{5}\spadcommand{[1]}
\indentrel{3}\begin{verbatim}
   (2)  [1]
                              Type: List PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxListCreatePageEmpty2}
\begin{paste}{ugxListCreatePageEmpty2}{ugxListCreatePagePatch2}
\pastebutton{ugxListCreatePageEmpty2}{\showpaste}
\tab{5}\spadcommand{[1]}
\end{paste}\end{patch}

\begin{patch}{ugxListCreatePagePatch3}
\begin{paste}{ugxListCreatePageFull3}{ugxListCreatePageEmpty3}
\pastebutton{ugxListCreatePageFull3}{\hidepaste}
\tab{5}\spadcommand{list(1)}
\indentrel{3}\begin{verbatim}
   (3)  [1]
                              Type: List PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxListCreatePageEmpty3}
\begin{paste}{ugxListCreatePageEmpty3}{ugxListCreatePagePatch3}
\pastebutton{ugxListCreatePageEmpty3}{\showpaste}
\tab{5}\spadcommand{list(1)}
\end{paste}\end{patch}

\begin{patch}{ugxListCreatePagePatch4}
\begin{paste}{ugxListCreatePageFull4}{ugxListCreatePageEmpty4}
\pastebutton{ugxListCreatePageFull4}{\hidepaste}
\tab{5}\spadcommand{append([1,2,3],[5,6,7])}
\indentrel{3}\begin{verbatim}
```

```
   (4)  [1,2,3,5,6,7]
                              Type: List PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxListCreatePageEmpty4}
\begin{paste}{ugxListCreatePageEmpty4}{ugxListCreatePagePatch4}
\pastebutton{ugxListCreatePageEmpty4}{\showpaste}
\tab{5}\spadcommand{append([1,2,3],[5,6,7])}
\end{paste}\end{patch}

\begin{patch}{ugxListCreatePagePatch5}
\begin{paste}{ugxListCreatePageFull5}{ugxListCreatePageEmpty5}
\pastebutton{ugxListCreatePageFull5}{\hidepaste}
\tab{5}\spadcommand{cons(10,[9,8,7])}
\indentrel{3}\begin{verbatim}
   (5)  [10,9,8,7]
                              Type: List PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxListCreatePageEmpty5}
\begin{paste}{ugxListCreatePageEmpty5}{ugxListCreatePagePatch5}
\pastebutton{ugxListCreatePageEmpty5}{\showpaste}
\tab{5}\spadcommand{cons(10,[9,8,7])}
\end{paste}\end{patch}
```

———-

Accessing List Elements

— list.ht —

```
\begin{page}{ugxListAccessPage}{Accessing List Elements}
\beginscroll

\labelSpace{4pc}
\xtc{
To determine whether a list has any elements, use the operation
\spadfunFrom{empty?}{List}.
}{
\spadpaste{empty? [x+1]}
}
\xtc{
Alternatively, equality with the list constant \spadfunFrom{nil}{List}
can be tested.
}{
```

```
\spadpaste{([] = nil)@Boolean}
}

\xtc{
We'll use this in some of the following examples.
}{
\spadpaste{k := [4,3,7,3,8,5,9,2] \bound{k}}
}
\xtc{
Each of the next four expressions extracts the \spadfunFrom{first}{List}
element of \spad{k}.
}{
\spadpaste{first k \free{k}}
}
\xtc{
}{
\spadpaste{k.first \free{k}}
}
\xtc{
}{
\spadpaste{k.1 \free{k}}
}
\xtc{
}{
\spadpaste{k(1) \free{k}}
}
The last two forms generalize to \spad{k.i} and \spad{k(i)},
respectively, where
\texht{$ 1 \leq i \leq n$}{\spad{1 <= i <= n}}
and
\spad{n} equals the length of \spad{k}.
\xtc{
This length is calculated by \spadopFrom{\#}{List}.
}{
\spadpaste{n := \#k \free{k}}
}
Performing an operation such as \spad{k.i} is sometimes
referred to as {\it indexing into k} or
{\it elting into k}.
The latter phrase comes about because the name of the operation
that extracts elements is called \spadfunFrom{elt}{List}.
That is, \spad{k.3} is just alternative syntax for \spad{elt(k,3)}.
It is important to remember that list indices
begin with 1.
If we issue \spad{k := [1,3,2,9,5]} then \spad{k.4}
returns \spad{9}.
It is an error to use an index that is not in the range from
\spad{1} to the length of the list.

\xtc{
```

```
The last element of a list is extracted by any of the
following three expressions.
}{
\spadpaste{last k \free{k}}
}
\xtc{
}{
\spadpaste{k.last \free{k}}
}
\xtc{
This form computes the index of the last element and then
extracts the element from the list.
}{
\spadpaste{k.(\#k) \free{k}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxListAccessPagePatch1}
\begin{paste}{ugxListAccessPageFull1}{ugxListAccessPageEmpty1}
\pastebutton{ugxListAccessPageFull1}{\hidepaste}
\tab{5}\spadcommand{empty? [x+1]}
\indentrel{3}\begin{verbatim}
   (1)  false
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxListAccessPageEmpty1}
\begin{paste}{ugxListAccessPageEmpty1}{ugxListAccessPagePatch1}
\pastebutton{ugxListAccessPageEmpty1}{\showpaste}
\tab{5}\spadcommand{empty? [x+1]}
\end{paste}\end{patch}

\begin{patch}{ugxListAccessPagePatch2}
\begin{paste}{ugxListAccessPageFull2}{ugxListAccessPageEmpty2}
\pastebutton{ugxListAccessPageFull2}{\hidepaste}
\tab{5}\spadcommand{([] = nil)@Boolean}
\indentrel{3}\begin{verbatim}
   (2)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxListAccessPageEmpty2}
\begin{paste}{ugxListAccessPageEmpty2}{ugxListAccessPagePatch2}
\pastebutton{ugxListAccessPageEmpty2}{\showpaste}
\tab{5}\spadcommand{([] = nil)@Boolean}
```

```
\end{paste}\end{patch}

\begin{patch}{ugxListAccessPagePatch3}
\begin{paste}{ugxListAccessPageFull3}{ugxListAccessPageEmpty3}
\pastebutton{ugxListAccessPageFull3}{\hidepaste}
\tab{5}\spadcommand{k := [4,3,7,3,8,5,9,2]\bound{k }}
\indentrel{3}\begin{verbatim}
   (3)  [4,3,7,3,8,5,9,2]
                            Type: List PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxListAccessPageEmpty3}
\begin{paste}{ugxListAccessPageEmpty3}{ugxListAccessPagePatch3}
\pastebutton{ugxListAccessPageEmpty3}{\showpaste}
\tab{5}\spadcommand{k := [4,3,7,3,8,5,9,2]\bound{k }}
\end{paste}\end{patch}

\begin{patch}{ugxListAccessPagePatch4}
\begin{paste}{ugxListAccessPageFull4}{ugxListAccessPageEmpty4}
\pastebutton{ugxListAccessPageFull4}{\hidepaste}
\tab{5}\spadcommand{first k\free{k }}
\indentrel{3}\begin{verbatim}
   (4)  4
                                 Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxListAccessPageEmpty4}
\begin{paste}{ugxListAccessPageEmpty4}{ugxListAccessPagePatch4}
\pastebutton{ugxListAccessPageEmpty4}{\showpaste}
\tab{5}\spadcommand{first k\free{k }}
\end{paste}\end{patch}

\begin{patch}{ugxListAccessPagePatch5}
\begin{paste}{ugxListAccessPageFull5}{ugxListAccessPageEmpty5}
\pastebutton{ugxListAccessPageFull5}{\hidepaste}
\tab{5}\spadcommand{k.first\free{k }}
\indentrel{3}\begin{verbatim}
   (5)  4
                                 Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxListAccessPageEmpty5}
\begin{paste}{ugxListAccessPageEmpty5}{ugxListAccessPagePatch5}
\pastebutton{ugxListAccessPageEmpty5}{\showpaste}
\tab{5}\spadcommand{k.first\free{k }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugxListAccessPagePatch6}
\begin{paste}{ugxListAccessPageFull6}{ugxListAccessPageEmpty6}
\pastebutton{ugxListAccessPageFull6}{\hidepaste}
\tab{5}\spadcommand{k.1\free{k }}
\indentrel{3}\begin{verbatim}
   (6)  4
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxListAccessPageEmpty6}
\begin{paste}{ugxListAccessPageEmpty6}{ugxListAccessPagePatch6}
\pastebutton{ugxListAccessPageEmpty6}{\showpaste}
\tab{5}\spadcommand{k.1\free{k }}
\end{paste}\end{patch}

\begin{patch}{ugxListAccessPagePatch7}
\begin{paste}{ugxListAccessPageFull7}{ugxListAccessPageEmpty7}
\pastebutton{ugxListAccessPageFull7}{\hidepaste}
\tab{5}\spadcommand{k(1)\free{k }}
\indentrel{3}\begin{verbatim}
   (7)  4
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxListAccessPageEmpty7}
\begin{paste}{ugxListAccessPageEmpty7}{ugxListAccessPagePatch7}
\pastebutton{ugxListAccessPageEmpty7}{\showpaste}
\tab{5}\spadcommand{k(1)\free{k }}
\end{paste}\end{patch}

\begin{patch}{ugxListAccessPagePatch8}
\begin{paste}{ugxListAccessPageFull8}{ugxListAccessPageEmpty8}
\pastebutton{ugxListAccessPageFull8}{\hidepaste}
\tab{5}\spadcommand{n := \#k\free{k }}
\indentrel{3}\begin{verbatim}
   (8)  8
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxListAccessPageEmpty8}
\begin{paste}{ugxListAccessPageEmpty8}{ugxListAccessPagePatch8}
\pastebutton{ugxListAccessPageEmpty8}{\showpaste}
\tab{5}\spadcommand{n := \#k\free{k }}
\end{paste}\end{patch}

\begin{patch}{ugxListAccessPagePatch9}
\begin{paste}{ugxListAccessPageFull9}{ugxListAccessPageEmpty9}
```

```
\pastebutton{ugxListAccessPageFull9}{\hidepaste}
\tab{5}\spadcommand{last k\free{k }}
\indentrel{3}\begin{verbatim}
   (9)  2
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxListAccessPageEmpty9}
\begin{paste}{ugxListAccessPageEmpty9}{ugxListAccessPagePatch9}
\pastebutton{ugxListAccessPageEmpty9}{\showpaste}
\tab{5}\spadcommand{last k\free{k }}
\end{paste}\end{patch}

\begin{patch}{ugxListAccessPagePatch10}
\begin{paste}{ugxListAccessPageFull10}{ugxListAccessPageEmpty10}
\pastebutton{ugxListAccessPageFull10}{\hidepaste}
\tab{5}\spadcommand{k.last\free{k }}
\indentrel{3}\begin{verbatim}
   (10)  2
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxListAccessPageEmpty10}
\begin{paste}{ugxListAccessPageEmpty10}{ugxListAccessPagePatch10}
\pastebutton{ugxListAccessPageEmpty10}{\showpaste}
\tab{5}\spadcommand{k.last\free{k }}
\end{paste}\end{patch}

\begin{patch}{ugxListAccessPagePatch11}
\begin{paste}{ugxListAccessPageFull11}{ugxListAccessPageEmpty11}
\pastebutton{ugxListAccessPageFull11}{\hidepaste}
\tab{5}\spadcommand{k.(\#k)\free{k }}
\indentrel{3}\begin{verbatim}
   (11)  2
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxListAccessPageEmpty11}
\begin{paste}{ugxListAccessPageEmpty11}{ugxListAccessPagePatch11}
\pastebutton{ugxListAccessPageEmpty11}{\showpaste}
\tab{5}\spadcommand{k.(\#k)\free{k }}
\end{paste}\end{patch}
```

Changing List Elements

— **list.ht** —

```
\begin{page}{ugxListChangePage}{Changing List Elements}
\beginscroll

\labelSpace{4pc}
\xtc{
We'll use this in some of the following examples.
}{
\spadpaste{k := [4,3,7,3,8,5,9,2] \bound{k}}
}
\xtc{
List elements are reset by using the \spad{k.i} form on
the left-hand side of an assignment.
This expression resets the first element of \spad{k} to \spad{999}.
}{
\spadpaste{k.1 := 999 \free{k}\bound{k1}}
}
\xtc{
As with indexing into a list, it is an error to use an index
that is not within the proper bounds.
Here you see that \spad{k} was modified.
}{
\spadpaste{k \free{k1}}
}

The operation that performs the assignment of an element to a particular
position in a list is called \spadfunFrom{setelt}{List}.
This operation is {\it destructive} in that it changes the list.
In the above example, the assignment returned the value \spad{999} and
\spad{k} was modified.
For this reason, lists are called \spadglos{mutable} objects: it is
possible to change part of a list (mutate it) rather than always returning
a new list reflecting the intended modifications.
\xtc{
Moreover, since lists can share structure, changes to one list can
sometimes affect others.
}{
\spadpaste{k := [1,2] \bound{k2}}
}
\xtc{
}{
\spadpaste{m := cons(0,k) \free{k2}\bound{m}}
}
\xtc{
Change the second element of \spad{m}.
}{
```

```
\spadpaste{m.2 := 99 \free{m}\bound{m2}}
}
\xtc{
See, \spad{m} was altered.
}{
\spadpaste{m \free{m2}}
}
\xtc{
But what about \spad{k}?
It changed too!
}{
\spadpaste{k  \free{m2 k2}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxListChangePagePatch1}
\begin{paste}{ugxListChangePageFull1}{ugxListChangePageEmpty1}
\pastebutton{ugxListChangePageFull1}{\hidepaste}
\tab{5}\spadcommand{k := [4,3,7,3,8,5,9,2]\bound{k }}
\indentrel{3}\begin{verbatim}
   (1)  [4,3,7,3,8,5,9,2]
                            Type: List PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxListChangePageEmpty1}
\begin{paste}{ugxListChangePageEmpty1}{ugxListChangePagePatch1}
\pastebutton{ugxListChangePageEmpty1}{\showpaste}
\tab{5}\spadcommand{k := [4,3,7,3,8,5,9,2]\bound{k }}
\end{paste}\end{patch}

\begin{patch}{ugxListChangePagePatch2}
\begin{paste}{ugxListChangePageFull2}{ugxListChangePageEmpty2}
\pastebutton{ugxListChangePageFull2}{\hidepaste}
\tab{5}\spadcommand{k.1 := 999\free{k }\bound{k1 }}
\indentrel{3}\begin{verbatim}
   (2)  999
                                 Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxListChangePageEmpty2}
\begin{paste}{ugxListChangePageEmpty2}{ugxListChangePagePatch2}
\pastebutton{ugxListChangePageEmpty2}{\showpaste}
\tab{5}\spadcommand{k.1 := 999\free{k }\bound{k1 }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugxListChangePagePatch3}
\begin{paste}{ugxListChangePageFull3}{ugxListChangePageEmpty3}
\pastebutton{ugxListChangePageFull3}{\hidepaste}
\tab{5}\spadcommand{k\free{k1 }}
\indentrel{3}\begin{verbatim}
   (3)  [999,3,7,3,8,5,9,2]
                             Type: List PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxListChangePageEmpty3}
\begin{paste}{ugxListChangePageEmpty3}{ugxListChangePagePatch3}
\pastebutton{ugxListChangePageEmpty3}{\showpaste}
\tab{5}\spadcommand{k\free{k1 }}
\end{paste}\end{patch}

\begin{patch}{ugxListChangePagePatch4}
\begin{paste}{ugxListChangePageFull4}{ugxListChangePageEmpty4}
\pastebutton{ugxListChangePageFull4}{\hidepaste}
\tab{5}\spadcommand{k := [1,2]\bound{k2 }}
\indentrel{3}\begin{verbatim}
   (4)  [1,2]
                             Type: List PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxListChangePageEmpty4}
\begin{paste}{ugxListChangePageEmpty4}{ugxListChangePagePatch4}
\pastebutton{ugxListChangePageEmpty4}{\showpaste}
\tab{5}\spadcommand{k := [1,2]\bound{k2 }}
\end{paste}\end{patch}

\begin{patch}{ugxListChangePagePatch5}
\begin{paste}{ugxListChangePageFull5}{ugxListChangePageEmpty5}
\pastebutton{ugxListChangePageFull5}{\hidepaste}
\tab{5}\spadcommand{m := cons(0,k)\free{k2 }\bound{m }}
\indentrel{3}\begin{verbatim}
   (5)  [0,1,2]
                                     Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxListChangePageEmpty5}
\begin{paste}{ugxListChangePageEmpty5}{ugxListChangePagePatch5}
\pastebutton{ugxListChangePageEmpty5}{\showpaste}
\tab{5}\spadcommand{m := cons(0,k)\free{k2 }\bound{m }}
\end{paste}\end{patch}

\begin{patch}{ugxListChangePagePatch6}
\begin{paste}{ugxListChangePageFull6}{ugxListChangePageEmpty6}
```

```
\pastebutton{ugxListChangePageFull6}{\hidepaste}
\tab{5}\spadcommand{m.2 := 99\free{m }\bound{m2 }}
\indentrel{3}\begin{verbatim}
   (6)  99
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxListChangePageEmpty6}
\begin{paste}{ugxListChangePageEmpty6}{ugxListChangePagePatch6}
\pastebutton{ugxListChangePageEmpty6}{\showpaste}
\tab{5}\spadcommand{m.2 := 99\free{m }\bound{m2 }}
\end{paste}\end{patch}

\begin{patch}{ugxListChangePagePatch7}
\begin{paste}{ugxListChangePageFull7}{ugxListChangePageEmpty7}
\pastebutton{ugxListChangePageFull7}{\hidepaste}
\tab{5}\spadcommand{m\free{m2 }}
\indentrel{3}\begin{verbatim}
   (7)  [0,99,2]
                                     Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxListChangePageEmpty7}
\begin{paste}{ugxListChangePageEmpty7}{ugxListChangePagePatch7}
\pastebutton{ugxListChangePageEmpty7}{\showpaste}
\tab{5}\spadcommand{m\free{m2 }}
\end{paste}\end{patch}

\begin{patch}{ugxListChangePagePatch8}
\begin{paste}{ugxListChangePageFull8}{ugxListChangePageEmpty8}
\pastebutton{ugxListChangePageFull8}{\hidepaste}
\tab{5}\spadcommand{k\free{m2 k2 }}
\indentrel{3}\begin{verbatim}
   (8)  [99,2]
                             Type: List PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxListChangePageEmpty8}
\begin{paste}{ugxListChangePageEmpty8}{ugxListChangePagePatch8}
\pastebutton{ugxListChangePageEmpty8}{\showpaste}
\tab{5}\spadcommand{k\free{m2 k2 }}
\end{paste}\end{patch}
```

———

Other Functions

— list.ht —

```
\begin{page}{ugxListOtherPage}{Other Functions}
\beginscroll

\labelSpace{3pc}
\xtc{
An operation that is used frequently in list processing is that
which returns all elements in a list after the first element.
}{
\spadpaste{k := [1,2,3] \bound{k}}
}
\xtc{
Use the \spadfunFrom{rest}{List} operation to do this.
}{
\spadpaste{rest k \free{k}}
}

\xtc{
To remove duplicate elements in a list \spad{k}, use
\spadfunFrom{removeDuplicates}{List}.
}{
\spadpaste{removeDuplicates [4,3,4,3,5,3,4]}
}
\xtc{
To get a list with elements in the order opposite to those in
a list \spad{k}, use \spadfunFrom{reverse}{List}.
}{
\spadpaste{reverse [1,2,3,4,5,6]}
}
\xtc{
To test whether an element is in a list, use \spadfunFrom{member?}{List}:
\spad{member?(a,k)} returns \spad{true} or \spad{false}
depending on whether \spad{a} is in \spad{k} or not.
}{
\spadpaste{member?(1/2,[3/4,5/6,1/2])}
}
\xtc{
}{
\spadpaste{member?(1/12,[3/4,5/6,1/2])}
}

As an exercise, the reader should determine how to get a
list containing all but the last of the elements in a given non-empty
list \spad{k}.\footnote{\spad{reverse(rest(reverse(k)))} works.}

\endscroll
```

```
\autobuttons
\end{page}

\begin{patch}{ugxListOtherPagePatch1}
\begin{paste}{ugxListOtherPageFull1}{ugxListOtherPageEmpty1}
\pastebutton{ugxListOtherPageFull1}{\hidepaste}
\tab{5}\spadcommand{k := [1,2,3]\bound{k }}
\indentrel{3}\begin{verbatim}
   (1)  [1,2,3]
                            Type: List PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxListOtherPageEmpty1}
\begin{paste}{ugxListOtherPageEmpty1}{ugxListOtherPagePatch1}
\pastebutton{ugxListOtherPageEmpty1}{\showpaste}
\tab{5}\spadcommand{k := [1,2,3]\bound{k }}
\end{paste}\end{patch}

\begin{patch}{ugxListOtherPagePatch2}
\begin{paste}{ugxListOtherPageFull2}{ugxListOtherPageEmpty2}
\pastebutton{ugxListOtherPageFull2}{\hidepaste}
\tab{5}\spadcommand{rest k\free{k }}
\indentrel{3}\begin{verbatim}
   (2)  [2,3]
                            Type: List PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxListOtherPageEmpty2}
\begin{paste}{ugxListOtherPageEmpty2}{ugxListOtherPagePatch2}
\pastebutton{ugxListOtherPageEmpty2}{\showpaste}
\tab{5}\spadcommand{rest k\free{k }}
\end{paste}\end{patch}

\begin{patch}{ugxListOtherPagePatch3}
\begin{paste}{ugxListOtherPageFull3}{ugxListOtherPageEmpty3}
\pastebutton{ugxListOtherPageFull3}{\hidepaste}
\tab{5}\spadcommand{removeDuplicates [4,3,4,3,5,3,4]}
\indentrel{3}\begin{verbatim}
   (3)  [4,3,5]
                            Type: List PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxListOtherPageEmpty3}
\begin{paste}{ugxListOtherPageEmpty3}{ugxListOtherPagePatch3}
\pastebutton{ugxListOtherPageEmpty3}{\showpaste}
\tab{5}\spadcommand{removeDuplicates [4,3,4,3,5,3,4]}
\end{paste}\end{patch}
```

```
\begin{patch}{ugxListOtherPagePatch4}
\begin{paste}{ugxListOtherPageFull4}{ugxListOtherPageEmpty4}
\pastebutton{ugxListOtherPageFull4}{\hidepaste}
\tab{5}\spadcommand{reverse [1,2,3,4,5,6]}
\indentrel{3}\begin{verbatim}
   (4)  [6,5,4,3,2,1]
                             Type: List PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxListOtherPageEmpty4}
\begin{paste}{ugxListOtherPageEmpty4}{ugxListOtherPagePatch4}
\pastebutton{ugxListOtherPageEmpty4}{\showpaste}
\tab{5}\spadcommand{reverse [1,2,3,4,5,6]}
\end{paste}\end{patch}

\begin{patch}{ugxListOtherPagePatch5}
\begin{paste}{ugxListOtherPageFull5}{ugxListOtherPageEmpty5}
\pastebutton{ugxListOtherPageFull5}{\hidepaste}
\tab{5}\spadcommand{member?(1/2,[3/4,5/6,1/2])}
\indentrel{3}\begin{verbatim}
   (5)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxListOtherPageEmpty5}
\begin{paste}{ugxListOtherPageEmpty5}{ugxListOtherPagePatch5}
\pastebutton{ugxListOtherPageEmpty5}{\showpaste}
\tab{5}\spadcommand{member?(1/2,[3/4,5/6,1/2])}
\end{paste}\end{patch}

\begin{patch}{ugxListOtherPagePatch6}
\begin{paste}{ugxListOtherPageFull6}{ugxListOtherPageEmpty6}
\pastebutton{ugxListOtherPageFull6}{\hidepaste}
\tab{5}\spadcommand{member?(1/12,[3/4,5/6,1/2])}
\indentrel{3}\begin{verbatim}
   (6)  false
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxListOtherPageEmpty6}
\begin{paste}{ugxListOtherPageEmpty6}{ugxListOtherPagePatch6}
\pastebutton{ugxListOtherPageEmpty6}{\showpaste}
\tab{5}\spadcommand{member?(1/12,[3/4,5/6,1/2])}
\end{paste}\end{patch}
```

Dot, Dot

— list.ht —

```
\begin{page}{ugxListDotPage}{Dot, Dot}
\beginscroll

Certain lists are used so often that Axiom provides an easy way
of constructing them.
If \spad{n} and \spad{m} are integers, then \spad{expand [n..m]}
creates a list containing \spad{n, n+1, ... m}.
If \spad{n > m} then the list is empty.
It is actually permissible to leave off the \spad{m} in the
dot-dot construction (see below).

\xtc{
The dot-dot notation can be used more than once in a list
construction and with specific elements being given.
Items separated by dots are called {\it segments.}
}{
\spadpaste{[1..3,10,20..23]}
}
\xtc{
Segments can be expanded into the range of items between the
endpoints by using \spadfunFrom{expand}{Segment}.
}{
\spadpaste{expand [1..3,10,20..23]}
}
\xtc{
What happens if we leave off a number on the right-hand side of
\spadopFrom{..}{UniversalSegment}?
}{
\spadpaste{expand [1..]}
}
What is created in this case is a \spadtype{Stream} which is a
generalization of a list.
See \downlink{`Stream'}{StreamXmpPage}\ignore{Stream} for more information.
\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxListDotPagePatch1}
\begin{paste}{ugxListDotPageFull1}{ugxListDotPageEmpty1}
\pastebutton{ugxListDotPageFull1}{\hidepaste}
\tab{5}\spadcommand{[1..3,10,20..23]}
\indentrel{3}\begin{verbatim}
```

```
   (1)  [1..3,10..10,20..23]
                     Type: List Segment PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxListDotPageEmpty1}
\begin{paste}{ugxListDotPageEmpty1}{ugxListDotPagePatch1}
\pastebutton{ugxListDotPageEmpty1}{\showpaste}
\tab{5}\spadcommand{[1..3,10,20..23]}
\end{paste}\end{patch}

\begin{patch}{ugxListDotPagePatch2}
\begin{paste}{ugxListDotPageFull2}{ugxListDotPageEmpty2}
\pastebutton{ugxListDotPageFull2}{\hidepaste}
\tab{5}\spadcommand{expand [1..3,10,20..23]}
\indentrel{3}\begin{verbatim}
   (2)  [1,2,3,10,20,21,22,23]
                                     Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxListDotPageEmpty2}
\begin{paste}{ugxListDotPageEmpty2}{ugxListDotPagePatch2}
\pastebutton{ugxListDotPageEmpty2}{\showpaste}
\tab{5}\spadcommand{expand [1..3,10,20..23]}
\end{paste}\end{patch}

\begin{patch}{ugxListDotPagePatch3}
\begin{paste}{ugxListDotPageFull3}{ugxListDotPageEmpty3}
\pastebutton{ugxListDotPageFull3}{\hidepaste}
\tab{5}\spadcommand{expand [1..]}
\indentrel{3}\begin{verbatim}
   (3)  [1,2,3,4,5,6,7,8,9,10,...]
                                   Type: Stream Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxListDotPageEmpty3}
\begin{paste}{ugxListDotPageEmpty3}{ugxListDotPagePatch3}
\pastebutton{ugxListDotPageEmpty3}{\showpaste}
\tab{5}\spadcommand{expand [1..]}
\end{paste}\end{patch}
```

3.65 lodo.ht

LinearOrdinaryDifferentialOperator

⇒ "notitle" (ugxLinearOrdinaryDifferentialOperatorSeriesPage) 3.65 on page 884

— lodo.ht —

```
\begin{page}{LinearOrdinaryDifferentialOperatorXmpPage}
{LinearOrdinaryDifferentialOperator}
\beginscroll

\spadtype{LinearOrdinaryDifferentialOperator(A, diff)} is the
domain of linear ordinary differential operators with coefficients
in a ring \spad{A} with a given derivation.
%This includes the cases of operators which are polynomials in \spad{D}
%acting upon scalar or vector expressions of a single variable.
%The coefficients of the operator polynomials can be integers, rational
%functions, matrices or elements of other domains.
\showBlurb{LinearOrdinaryDifferentialOperator}

\beginmenu
menudownlink{{9.44.1. Differential Operators with Series Coefficients}}
{ugxLinearOrdinaryDifferentialOperatorSeriesPage}
\endmenu
\endscroll
\autobuttons
\end{page}
```

———-

Differential Operators with Series Coefficients

— lodo.ht —

```
\begin{page}{ugxLinearOrdinaryDifferentialOperatorSeriesPage}
{Differential Operators with Series Coefficients}
\beginscroll

\noindent
{\bf Problem:}
Find the first few coefficients of \spad{exp(x)/x**i} of \spad{Dop phi}
where
\begin{verbatim}
Dop := D**3 + G/x**2 * D + H/x**3 - 1
phi := sum(s[i]*exp(x)/x**i, i = 0..)
```

```
\end{verbatim}

\noindent
{\bf Solution:}
\xtc{
Define the differential.
}{
\spadpaste{Dx: LODO(EXPR INT, f +-> D(f, x)) \bound{Dxd}}
}
\xtc{
}{
\spadpaste{Dx := D() \free{Dxd}\bound{Dx}}
}
\xtc{
Now define the differential operator \spad{Dop}.
}{
\spadpaste{Dop:= Dx**3 + G/x**2*Dx + H/x**3 - 1 \free{Dx}\bound{Dop}}
}
\xtc{
}{
\spadpaste{n == 3 \bound{n3}}
}
\xtc{
}{
\spadpaste{phi == reduce(+,[subscript(s,[i])*exp(x)/x**i for i in 0..n])
\bound{phi}}
}
\xtc{
}{
\spadpaste{phi1 ==  Dop(phi) / exp x \bound{phi1}\free{Dop phi}}
}
\xtc{
}{
\spadpaste{phi2 == phi1 *x**(n+3) \bound{phi2}\free{phi1}}
}
\xtc{
}{
\spadpaste{phi3 == retract(phi2)@(POLY INT) \bound{phi3}\free{phi2}}
}
\xtc{
}{
\spadpaste{pans == phi3 ::UP(x,POLY INT) \free{phi3}\bound{pans}}
}
\xtc{
}{
\spadpaste{pans1 == [coefficient(pans, (n+3-i) :: NNI) for i in 2..n+1]
\bound{pans1}\free{pans}}
}
\xtc{
}{
```

```
\spadpaste{leq == solve(pans1,[subscript(s,[i]) for i in 1..n])
\bound{leq}\free{pans1}}
}
\xtc{
Evaluate this for several values of \spad{n}.
}{
\spadpaste{leq \free{n3 leq}}
}
\xtc{
}{
\spadpaste{n==4 \bound{n4}}
}
\xtc{
}{
\spadpaste{leq \free{n4 leq}}
}
\xtc{
}{
\spadpaste{n==7 \bound{n7}}
}
\xtc{
}{
\spadpaste{leq \free{n7 leq}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorSeriesPagePatch1}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorSeriesPageFull1}{ugxLinearOrdinaryDifferentia
\pastebutton{ugxLinearOrdinaryDifferentialOperatorSeriesPageFull1}{\hidepaste}
\tab{5}\spadcommand{Dx: LODO(EXPR INT, f +-> D(f, x))\bound{Dxd }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty1}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty1}{ugxLinearOrdinaryDifferenti
\pastebutton{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty1}{\showpaste}
\tab{5}\spadcommand{Dx: LODO(EXPR INT, f +-> D(f, x))\bound{Dxd }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorSeriesPagePatch2}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorSeriesPageFull2}{ugxLinearOrdinaryDifferentia
\pastebutton{ugxLinearOrdinaryDifferentialOperatorSeriesPageFull2}{\hidepaste}
\tab{5}\spadcommand{Dx := D()\free{Dxd }\bound{Dx }}
\indentrel{3}\begin{verbatim}
   (2)  D
```

```
Type: LinearOrdinaryDifferentialOperator(Expression Integer,theMap NIL)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty2}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty2}{ugxLinearOrdinaryDifferentialOperatorSe
\pastebutton{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty2}{\showpaste}
\tab{5}\spadcommand{Dx := D()\free{Dxd }\bound{Dx }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorSeriesPagePatch3}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorSeriesPageFull3}{ugxLinearOrdinaryDifferentialOperatorSer
\pastebutton{ugxLinearOrdinaryDifferentialOperatorSeriesPageFull3}{\hidepaste}
\tab{5}\spadcommand{Dop:= Dx**3 + G/x**2*Dx + H/x**3 - 1\free{Dx }\bound{Dop }}
\indentrel{3}\begin{verbatim}
                      3
        3    G     - x  + H
   (3)  D  +  D +
             2        3
            x        x
Type: LinearOrdinaryDifferentialOperator(Expression Integer,theMap NIL)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty3}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty3}{ugxLinearOrdinaryDifferentialOperatorSe
\pastebutton{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty3}{\showpaste}
\tab{5}\spadcommand{Dop:= Dx**3 + G/x**2*Dx + H/x**3 - 1\free{Dx }\bound{Dop }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorSeriesPagePatch4}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorSeriesPageFull4}{ugxLinearOrdinaryDifferentialOperatorSer
\pastebutton{ugxLinearOrdinaryDifferentialOperatorSeriesPageFull4}{\hidepaste}
\tab{5}\spadcommand{n == 3\bound{n3 }}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty4}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty4}{ugxLinearOrdinaryDifferentialOperatorSe
\pastebutton{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty4}{\showpaste}
\tab{5}\spadcommand{n == 3\bound{n3 }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorSeriesPagePatch5}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorSeriesPageFull5}{ugxLinearOrdinaryDifferentialOperatorSer
\pastebutton{ugxLinearOrdinaryDifferentialOperatorSeriesPageFull5}{\hidepaste}
\tab{5}\spadcommand{phi == reduce(+,[subscript(s,[i])*exp(x)/x**i for i in 0..n])\bound{phi }}
\indentrel{3}\begin{verbatim}
```

```
                                        Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty5}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty5}{ugxLinearOrdinaryDifferenti
\pastebutton{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty5}{\showpaste}
\tab{5}\spadcommand{phi == reduce(+,[subscript(s,[i])*exp(x)/x**i for i in 0..n])\bound{phi }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorSeriesPagePatch6}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorSeriesPageFull6}{ugxLinearOrdinaryDifferentia
\pastebutton{ugxLinearOrdinaryDifferentialOperatorSeriesPageFull6}{\hidepaste}
\tab{5}\spadcommand{phi1 == Dop(phi) / exp x\bound{phi1 }\free{Dop phi }}
\indentrel{3}\begin{verbatim}
                                        Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty6}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty6}{ugxLinearOrdinaryDifferenti
\pastebutton{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty6}{\showpaste}
\tab{5}\spadcommand{phi1 == Dop(phi) / exp x\bound{phi1 }\free{Dop phi }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorSeriesPagePatch7}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorSeriesPageFull7}{ugxLinearOrdinaryDifferentia
\pastebutton{ugxLinearOrdinaryDifferentialOperatorSeriesPageFull7}{\hidepaste}
\tab{5}\spadcommand{phi2 == phi1 *x**(n+3)\bound{phi2 }\free{phi1 }}
\indentrel{3}\begin{verbatim}
                                        Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty7}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty7}{ugxLinearOrdinaryDifferenti
\pastebutton{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty7}{\showpaste}
\tab{5}\spadcommand{phi2 == phi1 *x**(n+3)\bound{phi2 }\free{phi1 }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorSeriesPagePatch8}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorSeriesPageFull8}{ugxLinearOrdinaryDifferentia
\pastebutton{ugxLinearOrdinaryDifferentialOperatorSeriesPageFull8}{\hidepaste}
\tab{5}\spadcommand{phi3 == retract(phi2)@(POLY INT)\bound{phi3 }\free{phi2 }}
\indentrel{3}\begin{verbatim}
                                        Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty8}
```

```
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty8}{ugxLinearOrdinaryDifferentialOperatorSe
\pastebutton{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty8}{\showpaste}
\tab{5}\spadcommand{phi3 == retract(phi2)@(POLY INT)\bound{phi3 }\free{phi2 }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorSeriesPagePatch9}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorSeriesPageFull9}{ugxLinearOrdinaryDifferentialOperatorSer
\pastebutton{ugxLinearOrdinaryDifferentialOperatorSeriesPageFull9}{\hidepaste}
\tab{5}\spadcommand{pans == phi3 ::UP(x,POLY INT)\free{phi3 }\bound{pans }}
\indentrel{3}\begin{verbatim}
                                              Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty9}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty9}{ugxLinearOrdinaryDifferentialOperatorSe
\pastebutton{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty9}{\showpaste}
\tab{5}\spadcommand{pans == phi3 ::UP(x,POLY INT)\free{phi3 }\bound{pans }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorSeriesPagePatch10}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorSeriesPageFull10}{ugxLinearOrdinaryDifferentialOperatorSe
\pastebutton{ugxLinearOrdinaryDifferentialOperatorSeriesPageFull10}{\hidepaste}
\tab{5}\spadcommand{pans1 == [coefficient(pans, (n+3-i) :: NNI) for i in 2..n+1]\bound{pans1 }\free{pans }}
\indentrel{3}\begin{verbatim}
                                              Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty10}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty10}{ugxLinearOrdinaryDifferentialOperatorS
\pastebutton{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty10}{\showpaste}
\tab{5}\spadcommand{pans1 == [coefficient(pans, (n+3-i) :: NNI) for i in 2..n+1]\bound{pans1 }\free{pans }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorSeriesPagePatch11}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorSeriesPageFull11}{ugxLinearOrdinaryDifferentialOperatorSe
\pastebutton{ugxLinearOrdinaryDifferentialOperatorSeriesPageFull11}{\hidepaste}
\tab{5}\spadcommand{leq == solve(pans1,[subscript(s,[i]) for i in 1..n])\bound{leq }\free{pans1 }}
\indentrel{3}\begin{verbatim}
                                              Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty11}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty11}{ugxLinearOrdinaryDifferentialOperatorS
\pastebutton{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty11}{\showpaste}
\tab{5}\spadcommand{leq == solve(pans1,[subscript(s,[i]) for i in 1..n])\bound{leq }\free{pans1 }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugxLinearOrdinaryDifferentialOperatorSeriesPagePatch12}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorSeriesPageFull12}{ugxLinearOrdinaryDifferenti
\pastebutton{ugxLinearOrdinaryDifferentialOperatorSeriesPageFull12}{\hidepaste}
\tab{5}\spadcommand{leq\free{n3 leq }}
\indentrel{3}\begin{verbatim}
   (12)
   [
                              2
          s G      3s H + s G  + 6s G
           0         0     0       0
     [s = , s = ,
       1   3    2          18
                               3        2
          (9s G + 54s )H + s G  + 18s G  + 72s G
             0       0      0        0        0
      s = ]
       3                    162
     ]
   Type: List List Equation Fraction Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty12}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty12}{ugxLinearOrdinaryDifferent
\pastebutton{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty12}{\showpaste}
\tab{5}\spadcommand{leq\free{n3 leq }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorSeriesPagePatch13}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorSeriesPageFull13}{ugxLinearOrdinaryDifferenti
\pastebutton{ugxLinearOrdinaryDifferentialOperatorSeriesPageFull13}{\hidepaste}
\tab{5}\spadcommand{n==4\bound{n4 }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty13}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty13}{ugxLinearOrdinaryDifferent
\pastebutton{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty13}{\showpaste}
\tab{5}\spadcommand{n==4\bound{n4 }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorSeriesPagePatch14}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorSeriesPageFull14}{ugxLinearOrdinaryDifferenti
\pastebutton{ugxLinearOrdinaryDifferentialOperatorSeriesPageFull14}{\hidepaste}
\tab{5}\spadcommand{leq\free{n4 leq }}
\indentrel{3}\begin{verbatim}
   (14)
   [
```

```
                          2
          s G      3s H + s G  + 6s G
           0         0     0       0
     [s = , s = ,
       1   3    2          18
                           3        2
          (9s G + 54s )H + s G  + 18s G  + 72s G
             0       0      0        0        0
      s = ,
       3                   162

       s  =
        4
                 2         2                          4
             27s H  + (18s G  + 378s G + 1296s )H + s G
                0         0         0         0      0
           +
                 3         2
             36s G  + 396s G  + 1296s G
                0         0          0
        /
           1944
       ]
     ]
   Type: List List Equation Fraction Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty14}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty14}{ugxLinearOrdinaryDifferentialOperatorS
\pastebutton{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty14}{\showpaste}
\tab{5}\spadcommand{leq\free{n4 leq }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorSeriesPagePatch15}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorSeriesPageFull15}{ugxLinearOrdinaryDifferentialOperatorSe
\pastebutton{ugxLinearOrdinaryDifferentialOperatorSeriesPageFull15}{\hidepaste}
\tab{5}\spadcommand{n==7\bound{n7 }}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty15}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty15}{ugxLinearOrdinaryDifferentialOperatorS
\pastebutton{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty15}{\showpaste}
\tab{5}\spadcommand{n==7\bound{n7 }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorSeriesPagePatch16}
```

```
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorSeriesPageFull16}{ugxLinearOrdinaryDifferenti
\pastebutton{ugxLinearOrdinaryDifferentialOperatorSeriesPageFull16}{\hidepaste}
\tab{5}\spadcommand{leq\free{n7 leq }}
\indentrel{3}\begin{verbatim}
   (16)
   [
                           2
          s G      3s H + s G  + 6s G
           0         0     0       0
     [s = , s = ,
       1   3    2           18
                            3        2
          (9s G + 54s )H + s G  + 18s G  + 72s G
             0       0      0        0        0
      s = ,
       3                    162

       s  =
        4
                 2         2                           4
             27s H  + (18s G  + 378s G + 1296s )H + s G
                0         0         0         0      0
           +
                 3         2
             36s G  + 396s G  + 1296s G
                0         0          0
        /
           1944
       ,

       s  =
        5
                                 2
             (135s G + 2268s )H
                  0         0
           +
                   3          2
             (30s G  + 1350s G  + 16416s G + 38880s )H
                 0          0           0          0
           +
              5        4          3          2
           s G  + 60s G  + 1188s G  + 9504s G  + 25920s G
            0        0          0          0           0
        /
           29160
       ,

       s  =
        6
                   3          2                        2
```

```
      405s H  + (405s G  + 18468s G + 174960s )H
          0          0           0           0
    +
               4          3           2
          45s G  + 3510s G  + 88776s G  + 777600s G
             0          0           0            0
        +
          1166400s
                  0
     *
        H
    +
         6        5          4           3
      s G  + 90s G  + 2628s G  + 27864s G
       0        0          0           0
    +
              2
      90720s G
            0
 /
    524880
,

s  =
 7
                          3
      (2835s G + 91854s )H
            0          0
    +
                3           2
          945s G  + 81648s G  + 2082996s G
              0           0             0
        +
          14171760s
                   0
     *
         2
        H
    +
               5          4            3
          63s G  + 7560s G  + 317520s G
             0          0            0
        +
                    2
          5554008s G  + 34058880s G
                  0              0
     *
        H
    +
         7         6          5           4
```

```
            s G  + 126s G  + 4788s G  + 25272s G
             0        0         0          0
          +
                        3              2
            - 1744416s G  - 26827200s G  - 97977600s G
                       0              0              0
       /
          11022480
      ]
    ]
   Type: List List Equation Fraction Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty16}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty16}{ugxLinearOrdinaryDifferent
\pastebutton{ugxLinearOrdinaryDifferentialOperatorSeriesPageEmpty16}{\showpaste}
\tab{5}\spadcommand{leq\free{n7 leq }}
\end{paste}\end{patch}
```

————-

3.66 lodo1.ht

LinearOrdinaryDifferentialOperator1

⇒ "notitle" (ugxLinearOrdinaryDifferentialOperatorOneRatPage) 3.66 on page 895

— **lodo1.ht** —

```
\begin{page}{LinearOrdinaryDifferentialOperatorOneXmpPage}
{LinearOrdinaryDifferentialOperator1}
\beginscroll

\spadtype{LinearOrdinaryDifferentialOperator1(A)} is the domain of
linear ordinary differential operators with coefficients in the
differential ring \spad{A}.
%This includes the cases of operators which are polynomials in \spad{D}
%acting upon scalar or vector expressions of a single variable.
%The coefficients of the operator polynomials can be integers, rational
%functions, matrices or elements of other domains.
\showBlurb{LinearOrdinaryDifferentialOperator1}

\beginmenu
    \menudownlink{
{9.45.1. Differential Operators with Rational Function Coefficients}}
{ugxLinearOrdinaryDifferentialOperatorOneRatPage}
```

```
\endmenu
\endscroll
\autobuttons
\end{page}
```

———-

Differential Operators with Rational Function Coefficients

— lodo1.ht —

```
\begin{page}{ugxLinearOrdinaryDifferentialOperatorOneRatPage}
{Differential Operators with Rational Function Coefficients}
\beginscroll

This example shows differential operators with rational function
coefficients.  In this case operator multiplication is non-commutative and,
since the coefficients form a field, an operator division algorithm exists.

\labelSpace{2pc}
\xtc{
We begin by defining \spad{RFZ} to be the rational functions in
\spad{x} with integer coefficients and \spad{Dx} to be the differential
operator for \spad{d/dx}.
}{
\spadpaste{RFZ := Fraction UnivariatePolynomial('x, Integer) \bound{RFZ0}}
}
\xtc{
}{
\spadpaste{x : RFZ := 'x \free{RFZ0}\bound{RFZ}}
}
\xtc{
}{
\spadpaste{Dx : LODO1 RFZ := D()\bound{Dx}\free{RFZ}}
}
\xtc{
Operators are created using the usual arithmetic operations.
}{
\spadpaste{b : LODO1 RFZ := 3*x**2*Dx**2 + 2*Dx + 1/x  \free{Dx}\bound{b}}
}
\xtc{
}{
\spadpaste{a : LODO1 RFZ := b*(5*x*Dx + 7)            \free{b Dx}\bound{a}}
}
\xtc{
Operator multiplication corresponds to functional composition.
}{
```

```
\spadpaste{p := x**2 + 1/x**2 \bound{p}\free{RFZ}}
}
\xtc{
Since operator coefficients depend on \spad{x}, the multiplication is
not commutative.
}{
\spadpaste{(a*b - b*a) p \free{a b p}}
}

When the coefficients of operator polynomials come from a field, as in
this case, it is possible to define operator division.  Division on
the left and division on the right yield different results when the
multiplication is non-commutative.

The results of
\spadfunFrom{leftDivide}{LinearOrdinaryDifferentialOperator1} and
\spadfunFrom{rightDivide}{LinearOrdinaryDifferentialOperator1} are
quotient-remainder pairs satisfying: \newline
%
\spad{leftDivide(a,b) = [q, r]} such that  \spad{a = b*q + r} \newline
\spad{rightDivide(a,b) = [q, r]} such that  \spad{a = q*b + r} \newline
%
\xtc{
In both cases, the
\spadfunFrom{degree}{LinearOrdinaryDifferentialOperator1}
of the remainder, \spad{r}, is less than
the degree of \spad{b}.
}{
\spadpaste{ld := leftDivide(a,b) \bound{ld}\free{a b}}
}
\xtc{
}{
\spadpaste{a = b * ld.quotient + ld.remainder \free{a b ld}}
}
\xtc{
The operations of left and right division
are so-called because the quotient is obtained by dividing
\spad{a} on that side by \spad{b}.
}{
\spadpaste{rd := rightDivide(a,b) \bound{rd}\free{a b}}
}
\xtc{
}{
\spadpaste{a = rd.quotient * b + rd.remainder \free{a b rd}}
}

\xtc{
Operations \spadfunFrom{rightQuotient}{LinearOrdinaryDifferentialOperator1}
and \spadfunFrom{rightRemainder}{LinearOrdinaryDifferentialOperator1}
are available if only one of the quotient or remainder are of
```

```
interest to you.
This is the quotient from right division.
}{
\spadpaste{rightQuotient(a,b) \free{a b}}
}
\xtc{
This is the remainder from right division.
The corresponding ``left'' functions
\spadfunFrom{leftQuotient}{LinearOrdinaryDifferentialOperator1} and
\spadfunFrom{leftRemainder}{LinearOrdinaryDifferentialOperator1}
are also available.
}{
\spadpaste{rightRemainder(a,b) \free{a b}}
}

\xtc{
For exact division, the operations
\spadfunFrom{leftExactQuotient}{LinearOrdinaryDifferentialOperator1} and
\spadfunFrom{rightExactQuotient}{LinearOrdinaryDifferentialOperator1}
are supplied.
These return the quotient but only if the remainder is zero.
The call \spad{rightExactQuotient(a,b)} would yield an error.
}{
\spadpaste{leftExactQuotient(a,b) \free{a b}}
}

\xtc{
The division operations allow the computation of left and right
greatest common divisors
(\spadfunFrom{leftGcd}{LinearOrdinaryDifferentialOperator1} and
\spadfunFrom{rightGcd}{LinearOrdinaryDifferentialOperator1}) via
remainder sequences, and consequently the computation of left and
right least common
multiples (\spadfunFrom{rightLcm}{LinearOrdinaryDifferentialOperator1}
and
\spadfunFrom{leftLcm}{LinearOrdinaryDifferentialOperator1}).
}{
\spadpaste{e := leftGcd(a,b) \bound{e}\free{a b}}
}
\xtc{
Note that a greatest common divisor doesn't necessarily divide \spad{a} and
{9.45.1.}{Differential Operators with Rational Function Coefficients}
\spad{b} on both sides.
Here the left greatest common divisor does not divide \spad{a} on the right.
}{
\spadpaste{leftRemainder(a, e) \free{a e}}
}
\xtc{
}{
\spadpaste{rightRemainder(a, e) \free{a e}}
```

```
}

\xtc{
Similarly, a least common multiple
is not necessarily divisible from both sides.
}{
\spadpaste{f := rightLcm(a,b) \bound{f}\free{a b}}
}
\xtc{
}{
\spadpaste{rightRemainder(f, b) \free{f b}}
}
\xtc{
}{
\spadpaste{leftRemainder(f, b) \free{f b}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorOneRatPagePatch1}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorOneRatPageFull1}{ugxLinearOrdinaryDifferentia
\pastebutton{ugxLinearOrdinaryDifferentialOperatorOneRatPageFull1}{\hidepaste}
\tab{5}\spadcommand{RFZ := Fraction UnivariatePolynomial('x, Integer)\bound{RFZ0 }}
\indentrel{3}\begin{verbatim}
   (1)  Fraction UnivariatePolynomial(x,Integer)
                                            Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty1}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty1}{ugxLinearOrdinaryDifferenti
\pastebutton{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty1}{\showpaste}
\tab{5}\spadcommand{RFZ := Fraction UnivariatePolynomial('x, Integer)\bound{RFZ0 }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorOneRatPagePatch2}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorOneRatPageFull2}{ugxLinearOrdinaryDifferentia
\pastebutton{ugxLinearOrdinaryDifferentialOperatorOneRatPageFull2}{\hidepaste}
\tab{5}\spadcommand{x : RFZ := 'x\free{RFZ0 }\bound{RFZ }}
\indentrel{3}\begin{verbatim}
   (2)  x
         Type: Fraction UnivariatePolynomial(x,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty2}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty2}{ugxLinearOrdinaryDifferenti
\pastebutton{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty2}{\showpaste}
```

```
\tab{5}\spadcommand{x : RFZ := 'x\free{RFZ0 }\bound{RFZ }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorOneRatPagePatch3}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorOneRatPageFull3}{ugxLinearOrdinaryDifferentialOperatorOne
\pastebutton{ugxLinearOrdinaryDifferentialOperatorOneRatPageFull3}{\hidepaste}
\tab{5}\spadcommand{Dx : LODO1 RFZ := D()\bound{Dx }\free{RFZ }}
\indentrel{3}\begin{verbatim}
   (3)  D
Type: LinearOrdinaryDifferentialOperator1 Fraction UnivariatePolynomial(x,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty3}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty3}{ugxLinearOrdinaryDifferentialOperatorOn
\pastebutton{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty3}{\showpaste}
\tab{5}\spadcommand{Dx : LODO1 RFZ := D()\bound{Dx }\free{RFZ }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorOneRatPagePatch4}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorOneRatPageFull4}{ugxLinearOrdinaryDifferentialOperatorOne
\pastebutton{ugxLinearOrdinaryDifferentialOperatorOneRatPageFull4}{\hidepaste}
\tab{5}\spadcommand{b : LODO1 RFZ := 3*x**2*Dx**2 + 2*Dx + 1/x\free{Dx }\bound{b }}
\indentrel{3}\begin{verbatim}
          2 2       1
   (4)  3x D  + 2D +
                    x
Type: LinearOrdinaryDifferentialOperator1 Fraction UnivariatePolynomial(x,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty4}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty4}{ugxLinearOrdinaryDifferentialOperatorOn
\pastebutton{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty4}{\showpaste}
\tab{5}\spadcommand{b : LODO1 RFZ := 3*x**2*Dx**2 + 2*Dx + 1/x\free{Dx }\bound{b }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorOneRatPagePatch5}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorOneRatPageFull5}{ugxLinearOrdinaryDifferentialOperatorOne
\pastebutton{ugxLinearOrdinaryDifferentialOperatorOneRatPageFull5}{\hidepaste}
\tab{5}\spadcommand{a : LODO1 RFZ := b*(5*x*Dx + 7)\free{b Dx }\bound{a }}
\indentrel{3}\begin{verbatim}
           3 3       2       2         7
   (5)  15x D  + (51x  + 10x)D  + 29D +
                                       x
Type: LinearOrdinaryDifferentialOperator1 Fraction UnivariatePolynomial(x,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty5}
```

```
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty5}{ugxLinearOrdinaryDifferenti
\pastebutton{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty5}{\showpaste}
\tab{5}\spadcommand{a : LODO1 RFZ := b*(5*x*Dx + 7)\free{b Dx }\bound{a }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorOneRatPagePatch6}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorOneRatPageFull6}{ugxLinearOrdinaryDifferentia
\pastebutton{ugxLinearOrdinaryDifferentialOperatorOneRatPageFull6}{\hidepaste}
\tab{5}\spadcommand{p := x**2 + 1/x**2\bound{p }\free{RFZ }}
\indentrel{3}\begin{verbatim}
         4
        x  + 1
   (6)
           2
          x
         Type: Fraction UnivariatePolynomial(x,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty6}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty6}{ugxLinearOrdinaryDifferenti
\pastebutton{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty6}{\showpaste}
\tab{5}\spadcommand{p := x**2 + 1/x**2\bound{p }\free{RFZ }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorOneRatPagePatch7}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorOneRatPageFull7}{ugxLinearOrdinaryDifferentia
\pastebutton{ugxLinearOrdinaryDifferentialOperatorOneRatPageFull7}{\hidepaste}
\tab{5}\spadcommand{(a*b - b*a) p\free{a b p }}
\indentrel{3}\begin{verbatim}
             4
        - 75x  + 540x - 75
   (7)
                 4
                x
         Type: Fraction UnivariatePolynomial(x,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty7}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty7}{ugxLinearOrdinaryDifferenti
\pastebutton{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty7}{\showpaste}
\tab{5}\spadcommand{(a*b - b*a) p\free{a b p }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorOneRatPagePatch8}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorOneRatPageFull8}{ugxLinearOrdinaryDifferentia
\pastebutton{ugxLinearOrdinaryDifferentialOperatorOneRatPageFull8}{\hidepaste}
\tab{5}\spadcommand{ld := leftDivide(a,b)\bound{ld }\free{a b }}
\indentrel{3}\begin{verbatim}
```

```
   (8)  [quotient= 5x D + 7,remainder= 0]
Type: Record(quotient: LinearOrdinaryDifferentialOperator1 Fraction UnivariatePolynomial(x,Integer),remainde
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty8}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty8}{ugxLinearOrdinaryDifferentialOperatorOn
\pastebutton{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty8}{\showpaste}
\tab{5}\spadcommand{ld := leftDivide(a,b)\bound{ld }\free{a b }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorOneRatPagePatch9}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorOneRatPageFull9}{ugxLinearOrdinaryDifferentialOperatorOne
\pastebutton{ugxLinearOrdinaryDifferentialOperatorOneRatPageFull9}{\hidepaste}
\tab{5}\spadcommand{a = b * ld.quotient + ld.remainder\free{a b ld }}
\indentrel{3}\begin{verbatim}
   (9)
      3 3       2        2         7
   15x D  + (51x  + 10x)D  + 29D +  =
                                   x
        3 3       2        2         7
     15x D  + (51x  + 10x)D  + 29D +
                                     x
Type: Equation LinearOrdinaryDifferentialOperator1 Fraction UnivariatePolynomial(x,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty9}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty9}{ugxLinearOrdinaryDifferentialOperatorOn
\pastebutton{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty9}{\showpaste}
\tab{5}\spadcommand{a = b * ld.quotient + ld.remainder\free{a b ld }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorOneRatPagePatch10}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorOneRatPageFull10}{ugxLinearOrdinaryDifferentialOperatorOn
\pastebutton{ugxLinearOrdinaryDifferentialOperatorOneRatPageFull10}{\hidepaste}
\tab{5}\spadcommand{rd := rightDivide(a,b)\bound{rd }\free{a b }}
\indentrel{3}\begin{verbatim}
                                              5
   (10)  [quotient= 5x D + 7,remainder= 10D + ]
                                              x
Type: Record(quotient: LinearOrdinaryDifferentialOperator1 Fraction UnivariatePolynomial(x,Integer),remainde
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty10}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty10}{ugxLinearOrdinaryDifferentialOperator(
\pastebutton{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty10}{\showpaste}
\tab{5}\spadcommand{rd := rightDivide(a,b)\bound{rd }\free{a b }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugxLinearOrdinaryDifferentialOperatorOneRatPagePatch11}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorOneRatPageFull11}{ugxLinearOrdinaryDifferenti
\pastebutton{ugxLinearOrdinaryDifferentialOperatorOneRatPageFull11}{\hidepaste}
\tab{5}\spadcommand{a = rd.quotient * b + rd.remainder\free{a b rd }}
\indentrel{3}\begin{verbatim}
   (11)
      3 3       2        2         7
   15x D  + (51x  + 10x)D  + 29D +  =
                                   x
        3 3       2        2         7
     15x D  + (51x  + 10x)D  + 29D +
                                     x
Type: Equation LinearOrdinaryDifferentialOperator1 Fraction UnivariatePolynomial(x,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty11}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty11}{ugxLinearOrdinaryDifferent
\pastebutton{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty11}{\showpaste}
\tab{5}\spadcommand{a = rd.quotient * b + rd.remainder\free{a b rd }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorOneRatPagePatch12}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorOneRatPageFull12}{ugxLinearOrdinaryDifferenti
\pastebutton{ugxLinearOrdinaryDifferentialOperatorOneRatPageFull12}{\hidepaste}
\tab{5}\spadcommand{rightQuotient(a,b)\free{a b }}
\indentrel{3}\begin{verbatim}
   (12)  5x D + 7
Type: LinearOrdinaryDifferentialOperator1 Fraction UnivariatePolynomial(x,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty12}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty12}{ugxLinearOrdinaryDifferent
\pastebutton{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty12}{\showpaste}
\tab{5}\spadcommand{rightQuotient(a,b)\free{a b }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorOneRatPagePatch13}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorOneRatPageFull13}{ugxLinearOrdinaryDifferenti
\pastebutton{ugxLinearOrdinaryDifferentialOperatorOneRatPageFull13}{\hidepaste}
\tab{5}\spadcommand{rightRemainder(a,b)\free{a b }}
\indentrel{3}\begin{verbatim}
               5
   (13)  10D +
               x
Type: LinearOrdinaryDifferentialOperator1 Fraction UnivariatePolynomial(x,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty13}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty13}{ugxLinearOrdinaryDifferentialOperator(
\pastebutton{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty13}{\showpaste}
\tab{5}\spadcommand{rightRemainder(a,b)\free{a b }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorOneRatPagePatch14}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorOneRatPageFull14}{ugxLinearOrdinaryDifferentialOperatorOn
\pastebutton{ugxLinearOrdinaryDifferentialOperatorOneRatPageFull14}{\hidepaste}
\tab{5}\spadcommand{leftExactQuotient(a,b)\free{a b }}
\indentrel{3}\begin{verbatim}
   (14)  5x D + 7
Type: Union(LinearOrdinaryDifferentialOperator1 Fraction UnivariatePolynomial(x,Integer),...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty14}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty14}{ugxLinearOrdinaryDifferentialOperator(
\pastebutton{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty14}{\showpaste}
\tab{5}\spadcommand{leftExactQuotient(a,b)\free{a b }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorOneRatPagePatch15}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorOneRatPageFull15}{ugxLinearOrdinaryDifferentialOperatorOn
\pastebutton{ugxLinearOrdinaryDifferentialOperatorOneRatPageFull15}{\hidepaste}
\tab{5}\spadcommand{e := leftGcd(a,b)\bound{e }\free{a b }}
\indentrel{3}\begin{verbatim}
           2 2        1
   (15)  3x D  + 2D +
                      x
Type: LinearOrdinaryDifferentialOperator1 Fraction UnivariatePolynomial(x,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty15}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty15}{ugxLinearOrdinaryDifferentialOperator(
\pastebutton{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty15}{\showpaste}
\tab{5}\spadcommand{e := leftGcd(a,b)\bound{e }\free{a b }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorOneRatPagePatch16}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorOneRatPageFull16}{ugxLinearOrdinaryDifferentialOperatorOn
\pastebutton{ugxLinearOrdinaryDifferentialOperatorOneRatPageFull16}{\hidepaste}
\tab{5}\spadcommand{leftRemainder(a, e)\free{a e }}
\indentrel{3}\begin{verbatim}
   (16)  0
Type: LinearOrdinaryDifferentialOperator1 Fraction UnivariatePolynomial(x,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty16}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty16}{ugxLinearOrdinaryDifferent
\pastebutton{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty16}{\showpaste}
\tab{5}\spadcommand{leftRemainder(a, e)\free{a e }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorOneRatPagePatch17}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorOneRatPageFull17}{ugxLinearOrdinaryDifferenti
\pastebutton{ugxLinearOrdinaryDifferentialOperatorOneRatPageFull17}{\hidepaste}
\tab{5}\spadcommand{rightRemainder(a, e)\free{a e }}
\indentrel{3}\begin{verbatim}
              5
   (17)  10D +
              x
Type: LinearOrdinaryDifferentialOperator1 Fraction UnivariatePolynomial(x,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty17}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty17}{ugxLinearOrdinaryDifferent
\pastebutton{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty17}{\showpaste}
\tab{5}\spadcommand{rightRemainder(a, e)\free{a e }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorOneRatPagePatch18}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorOneRatPageFull18}{ugxLinearOrdinaryDifferenti
\pastebutton{ugxLinearOrdinaryDifferentialOperatorOneRatPageFull18}{\hidepaste}
\tab{5}\spadcommand{f := rightLcm(a,b)\bound{f }\free{a b }}
\indentrel{3}\begin{verbatim}
           3 3       2        2         7
   (18)  15x D  + (51x  + 10x)D  + 29D +
                                        x
Type: LinearOrdinaryDifferentialOperator1 Fraction UnivariatePolynomial(x,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty18}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty18}{ugxLinearOrdinaryDifferent
\pastebutton{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty18}{\showpaste}
\tab{5}\spadcommand{f := rightLcm(a,b)\bound{f }\free{a b }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorOneRatPagePatch19}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorOneRatPageFull19}{ugxLinearOrdinaryDifferenti
\pastebutton{ugxLinearOrdinaryDifferentialOperatorOneRatPageFull19}{\hidepaste}
\tab{5}\spadcommand{rightRemainder(f, b)\free{f b }}
\indentrel{3}\begin{verbatim}
              5
   (19)  10D +
```

```
              x
Type: LinearOrdinaryDifferentialOperator1 Fraction UnivariatePolynomial(x,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty19}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty19}{ugxLinearOrdinaryDifferentialOperator(
\pastebutton{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty19}{\showpaste}
\tab{5}\spadcommand{rightRemainder(f, b)\free{f b }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorOneRatPagePatch20}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorOneRatPageFull20}{ugxLinearOrdinaryDifferentialOperatorOn
\pastebutton{ugxLinearOrdinaryDifferentialOperatorOneRatPageFull20}{\hidepaste}
\tab{5}\spadcommand{leftRemainder(f, b)\free{f b }}
\indentrel{3}\begin{verbatim}
   (20)  0
Type: LinearOrdinaryDifferentialOperator1 Fraction UnivariatePolynomial(x,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty20}
\begin{paste}{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty20}{ugxLinearOrdinaryDifferentialOperator(
\pastebutton{ugxLinearOrdinaryDifferentialOperatorOneRatPageEmpty20}{\showpaste}
\tab{5}\spadcommand{leftRemainder(f, b)\free{f b }}
\end{paste}\end{patch}
```

———-

3.67 lodo2.ht

LinearOrdinaryDifferentialOperator2

⇒ “notitle” (ugxLinearODEOperatorTwoConstPage) 3.67 on page 906
⇒ “notitle” (ugxLinearODEOperatorTwoMatrixPage) 3.67 on page 911

— **lodo2.ht** —

```
\begin{page}{LinearODEOperatorTwoXmpPage}
{LinearOrdinaryDifferentialOperator2}
\beginscroll

\spadtype{LinearOrdinaryDifferentialOperator2(A, M)} is the domain of
linear ordinary differential operators with coefficients in the
differential ring \spad{A} and operating on \spad{M}, an
\spad{A}-module.  This includes the cases of operators which are
```

```
polynomials in \spad{D} acting upon scalar or vector expressions of a
single variable.  The coefficients of the operator polynomials can be
integers, rational functions, matrices or elements of other domains.
\showBlurb{LinearOrdinaryDifferentialOperator2}

\beginmenu
    \menudownlink{
{9.46.1. Differential Operators with Constant Coefficients}}
{ugxLinearODEOperatorTwoConstPage}
    \menudownlink{
{9.46.2.
Differential Operators with Matrix Coefficients Operating on Vectors}}
{ugxLinearODEOperatorTwoMatrixPage}
\endmenu
\endscroll
\autobuttons
\end{page}
```

———-

Differential Operators with Constant Coefficients

— lodo2.ht —

```
\begin{page}{ugxLinearODEOperatorTwoConstPage}
{Differential Operators with Constant Coefficients}
\beginscroll

This example shows differential operators with rational
number coefficients operating on univariate polynomials.

\labelSpace{3pc}
\xtc{
We begin by making type assignments so we can conveniently refer
to univariate polynomials in \spad{x} over the rationals.
}{
\spadpaste{Q  := Fraction Integer \bound{Q}}
}
\xtc{
}{
\spadpaste{PQ := UnivariatePolynomial('x, Q) \free{Q}\bound{PQ0}}
}
\xtc{
}{
\spadpaste{x: PQ := 'x \free{PQ0}\bound{x}}
}
\xtc{
```

```
Now we assign \spad{Dx} to be the differential operator
\spadfunFrom{D}{LinearOrdinaryDifferentialOperator2}
corresponding to \spad{d/dx}.
}{
\spadpaste{Dx: LODO2(Q, PQ) := D() \free{Q PQ0}\bound{Dx}}
}
\xtc{
New operators are created as polynomials in \spad{D()}.
}{
\spadpaste{a := Dx  + 1 \bound{a}\free{Dx}}
}
\xtc{
}{
\spadpaste{b := a + 1/2*Dx**2 - 1/2 \bound{b}\free{Dx}}
}
\xtc{
To apply the operator \spad{a} to the value \spad{p} the usual function
call syntax is used.
}{
\spadpaste{p := 4*x**2 + 2/3 \free{x}\bound{p}}
}
\xtc{
}{
\spadpaste{a p \free{a p}}
}
\xtc{
Operator multiplication is defined by the identity \spad{(a*b) p = a(b(p))}
}{
\spadpaste{(a * b) p = a b p \free{a b p}}
}
\xtc{
Exponentiation follows from multiplication.
}{
\spadpaste{c := (1/9)*b*(a + b)**2 \free{a b}\bound{c}}
}
\xtc{
Finally, note that operator expressions may be applied directly.
}{
\spadpaste{(a**2 - 3/4*b + c) (p + 1) \free{a b c p}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxLinearODEOperatorTwoConstPagePatch1}
\begin{paste}{ugxLinearODEOperatorTwoConstPageFull1}{ugxLinearODEOperatorTwoConstPageEmpty1}
\pastebutton{ugxLinearODEOperatorTwoConstPageFull1}{\hidepaste}
\tab{5}\spadcommand{Q := Fraction Integer\bound{Q }}
\indentrel{3}\begin{verbatim}
```

```
   (1)  Fraction Integer
                                            Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoConstPageEmpty1}
\begin{paste}{ugxLinearODEOperatorTwoConstPageEmpty1}{ugxLinearODEOperatorTwoConstPagePatch1}
\pastebutton{ugxLinearODEOperatorTwoConstPageEmpty1}{\showpaste}
\tab{5}\spadcommand{Q := Fraction Integer\bound{Q }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoConstPagePatch2}
\begin{paste}{ugxLinearODEOperatorTwoConstPageFull2}{ugxLinearODEOperatorTwoConstPageEmpty2}
\pastebutton{ugxLinearODEOperatorTwoConstPageFull2}{\hidepaste}
\tab{5}\spadcommand{PQ := UnivariatePolynomial('x, Q)\free{Q }\bound{PQ0 }}
\indentrel{3}\begin{verbatim}
   (2)  UnivariatePolynomial(x,Fraction Integer)
                                            Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoConstPageEmpty2}
\begin{paste}{ugxLinearODEOperatorTwoConstPageEmpty2}{ugxLinearODEOperatorTwoConstPagePatch2}
\pastebutton{ugxLinearODEOperatorTwoConstPageEmpty2}{\showpaste}
\tab{5}\spadcommand{PQ := UnivariatePolynomial('x, Q)\free{Q }\bound{PQ0 }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoConstPagePatch3}
\begin{paste}{ugxLinearODEOperatorTwoConstPageFull3}{ugxLinearODEOperatorTwoConstPageEmpty3}
\pastebutton{ugxLinearODEOperatorTwoConstPageFull3}{\hidepaste}
\tab{5}\spadcommand{x: PQ := 'x\free{PQ0 }\bound{x }}
\indentrel{3}\begin{verbatim}
   (3)  x
         Type: UnivariatePolynomial(x,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoConstPageEmpty3}
\begin{paste}{ugxLinearODEOperatorTwoConstPageEmpty3}{ugxLinearODEOperatorTwoConstPagePatch3}
\pastebutton{ugxLinearODEOperatorTwoConstPageEmpty3}{\showpaste}
\tab{5}\spadcommand{x: PQ := 'x\free{PQ0 }\bound{x }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoConstPagePatch4}
\begin{paste}{ugxLinearODEOperatorTwoConstPageFull4}{ugxLinearODEOperatorTwoConstPageEmpty4}
\pastebutton{ugxLinearODEOperatorTwoConstPageFull4}{\hidepaste}
\tab{5}\spadcommand{Dx: LODO2(Q, PQ) := D()\free{Q PQ0 }\bound{Dx }}
\indentrel{3}\begin{verbatim}
   (4)  D
Type: LinearOrdinaryDifferentialOperator2(Fraction Integer,UnivariatePolynomial(x,Fraction Integ
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoConstPageEmpty4}
\begin{paste}{ugxLinearODEOperatorTwoConstPageEmpty4}{ugxLinearODEOperatorTwoConstPagePatch4}
\pastebutton{ugxLinearODEOperatorTwoConstPageEmpty4}{\showpaste}
\tab{5}\spadcommand{Dx: LODO2(Q, PQ) := D()\free{Q PQ0 }\bound{Dx }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoConstPagePatch5}
\begin{paste}{ugxLinearODEOperatorTwoConstPageFull5}{ugxLinearODEOperatorTwoConstPageEmpty5}
\pastebutton{ugxLinearODEOperatorTwoConstPageFull5}{\hidepaste}
\tab{5}\spadcommand{a := Dx + 1\bound{a }\free{Dx }}
\indentrel{3}\begin{verbatim}
   (5)  D + 1
Type: LinearOrdinaryDifferentialOperator2(Fraction Integer,UnivariatePolynomial(x,Fraction Integer))
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoConstPageEmpty5}
\begin{paste}{ugxLinearODEOperatorTwoConstPageEmpty5}{ugxLinearODEOperatorTwoConstPagePatch5}
\pastebutton{ugxLinearODEOperatorTwoConstPageEmpty5}{\showpaste}
\tab{5}\spadcommand{a := Dx + 1\bound{a }\free{Dx }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoConstPagePatch6}
\begin{paste}{ugxLinearODEOperatorTwoConstPageFull6}{ugxLinearODEOperatorTwoConstPageEmpty6}
\pastebutton{ugxLinearODEOperatorTwoConstPageFull6}{\hidepaste}
\tab{5}\spadcommand{b := a + 1/2*Dx**2 - 1/2\bound{b }\free{Dx }}
\indentrel{3}\begin{verbatim}
        1  2       1
   (6)   D  + D +
        2          2
Type: LinearOrdinaryDifferentialOperator2(Fraction Integer,UnivariatePolynomial(x,Fraction Integer))
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoConstPageEmpty6}
\begin{paste}{ugxLinearODEOperatorTwoConstPageEmpty6}{ugxLinearODEOperatorTwoConstPagePatch6}
\pastebutton{ugxLinearODEOperatorTwoConstPageEmpty6}{\showpaste}
\tab{5}\spadcommand{b := a + 1/2*Dx**2 - 1/2\bound{b }\free{Dx }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoConstPagePatch7}
\begin{paste}{ugxLinearODEOperatorTwoConstPageFull7}{ugxLinearODEOperatorTwoConstPageEmpty7}
\pastebutton{ugxLinearODEOperatorTwoConstPageFull7}{\hidepaste}
\tab{5}\spadcommand{p := 4*x**2 + 2/3\free{x }\bound{p }}
\indentrel{3}\begin{verbatim}
          2   2
   (7)  4x  +
```

```
              3
         Type: UnivariatePolynomial(x,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoConstPageEmpty7}
\begin{paste}{ugxLinearODEOperatorTwoConstPageEmpty7}{ugxLinearODEOperatorTwoConstPagePatch7}
\pastebutton{ugxLinearODEOperatorTwoConstPageEmpty7}{\showpaste}
\tab{5}\spadcommand{p := 4*x**2 + 2/3\free{x }\bound{p }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoConstPagePatch8}
\begin{paste}{ugxLinearODEOperatorTwoConstPageFull8}{ugxLinearODEOperatorTwoConstPageEmpty8}
\pastebutton{ugxLinearODEOperatorTwoConstPageFull8}{\hidepaste}
\tab{5}\spadcommand{a p\free{a p }}
\indentrel{3}\begin{verbatim}
          2        2
   (8)  4x  + 8x +
                   3
         Type: UnivariatePolynomial(x,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoConstPageEmpty8}
\begin{paste}{ugxLinearODEOperatorTwoConstPageEmpty8}{ugxLinearODEOperatorTwoConstPagePatch8}
\pastebutton{ugxLinearODEOperatorTwoConstPageEmpty8}{\showpaste}
\tab{5}\spadcommand{a p\free{a p }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoConstPagePatch9}
\begin{paste}{ugxLinearODEOperatorTwoConstPageFull9}{ugxLinearODEOperatorTwoConstPageEmpty9}
\pastebutton{ugxLinearODEOperatorTwoConstPageFull9}{\hidepaste}
\tab{5}\spadcommand{(a * b) p = a b p\free{a b p }}
\indentrel{3}\begin{verbatim}
          2         37    2         37
   (9)  2x  + 12x + = 2x  + 12x +
                     3               3
Type: Equation UnivariatePolynomial(x,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoConstPageEmpty9}
\begin{paste}{ugxLinearODEOperatorTwoConstPageEmpty9}{ugxLinearODEOperatorTwoConstPagePatch9}
\pastebutton{ugxLinearODEOperatorTwoConstPageEmpty9}{\showpaste}
\tab{5}\spadcommand{(a * b) p = a b p\free{a b p }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoConstPagePatch10}
\begin{paste}{ugxLinearODEOperatorTwoConstPageFull10}{ugxLinearODEOperatorTwoConstPageEmpty10}
\pastebutton{ugxLinearODEOperatorTwoConstPageFull10}{\hidepaste}
```

```
\tab{5}\spadcommand{c := (1/9)*b*(a + b)**2\free{a b }\bound{c }}
\indentrel{3}\begin{verbatim}
   (10)
    1  6    5  5   13  4   19  3   79  2    7     1
    D  +  D  +  D  +  D  +  D  +  D +
   72      36      24      18      72      12     8
Type: LinearOrdinaryDifferentialOperator2(Fraction Integer,UnivariatePolynomial(x,Fraction Integer))
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoConstPageEmpty10}
\begin{paste}{ugxLinearODEOperatorTwoConstPageEmpty10}{ugxLinearODEOperatorTwoConstPagePatch10}
\pastebutton{ugxLinearODEOperatorTwoConstPageEmpty10}{\showpaste}
\tab{5}\spadcommand{c := (1/9)*b*(a + b)**2\free{a b }\bound{c }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoConstPagePatch11}
\begin{paste}{ugxLinearODEOperatorTwoConstPageFull11}{ugxLinearODEOperatorTwoConstPageEmpty11}
\pastebutton{ugxLinearODEOperatorTwoConstPageFull11}{\hidepaste}
\tab{5}\spadcommand{(a**2 - 3/4*b + c) (p + 1)\free{a b c p }}
\indentrel{3}\begin{verbatim}
          2   44     541
   (11)  3x  +  x +
               3      36
         Type: UnivariatePolynomial(x,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoConstPageEmpty11}
\begin{paste}{ugxLinearODEOperatorTwoConstPageEmpty11}{ugxLinearODEOperatorTwoConstPagePatch11}
\pastebutton{ugxLinearODEOperatorTwoConstPageEmpty11}{\showpaste}
\tab{5}\spadcommand{(a**2 - 3/4*b + c) (p + 1)\free{a b c p }}
\end{paste}\end{patch}
```

———-

Differential Operators with Matrix Coefficients Operating on Vectors

— lodo2.ht —

```
\begin{page}{ugxLinearODEOperatorTwoMatrixPage}
{Differential Operators with Matrix Coefficients Operating on Vectors}
\beginscroll

This is another example of linear ordinary differential
operators with non-commutative multiplication.
Unlike the rational function case, the differential ring of
```

```
square matrices (of a given dimension) with univariate
polynomial entries does not form a field.
Thus the number of operations available is more limited.

\labelSpace{1pc}
\xtc{
In this section, the operators have three by three
matrix coefficients with polynomial entries.
}{
\spadpaste{PZ   := UnivariatePolynomial(x,Integer)\bound{PZ0}}
}
\xtc{
}{
\spadpaste{x:PZ := 'x \free{PZ0}\bound{PZ}}
}
\xtc{
}{
\spadpaste{Mat  := SquareMatrix(3,PZ)\free{PZ}\bound{Mat}}
}
\xtc{
The operators act on the vectors considered as a \spad{Mat}-module.
}{
\spadpaste{Vect := DPMM(3, PZ, Mat, PZ);\free{PZ,Mat}\bound{Vect}}
}
\xtc{
}{
\spadpaste{Modo := LODO2(Mat, Vect);\free{Mat Vect}\bound{Modo}}
}
\xtc{
The matrix \spad{m} is used as a coefficient and the vectors \spad{p}
and \spad{q} are operated upon.
}{
\spadpaste{m:Mat := matrix [[x**2,1,0],[1,x**4,0],[0,0,4*x**2]]
\free{Mat}\bound{m}}
}
\xtc{
}{
\spadpaste{p:Vect := directProduct [3*x**2+1,2*x,7*x**3+2*x]
\free{Vect}\bound{p}}
}
\xtc{
}{
\spadpaste{q: Vect := m * p\free{m p Vect}\bound{q}}
}
\xtc{
Now form a few operators.
}{
\spadpaste{Dx : Modo := D()\bound{Dx}\free{Modo}}
}
\xtc{
```

```
}{
\spadpaste{a : Modo := Dx  + m\bound{a}\free{m Dx}}
}
\xtc{
}{
\spadpaste{b : Modo := m*Dx  + 1\bound{b}\free{m Dx}}
}
\xtc{
}{
\spadpaste{c := a*b \bound{c}\free{a b}}
}
\xtc{
These operators can be applied to vector values.
}{
\spadpaste{a p \free{p a}}
}
\xtc{
}{
\spadpaste{b p \free{p b}}
}
\xtc{
}{
\spadpaste{(a + b + c) (p + q) \free{a b c p q}}
}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxLinearODEOperatorTwoMatrixPagePatch1}
\begin{paste}{ugxLinearODEOperatorTwoMatrixPageFull1}{ugxLinearODEOperatorTwoMatrixPageEmpty1}
\pastebutton{ugxLinearODEOperatorTwoMatrixPageFull1}{\hidepaste}
\tab{5}\spadcommand{PZ := UnivariatePolynomial(x,Integer)\bound{PZ0 }}
\indentrel{3}\begin{verbatim}
   (1)  UnivariatePolynomial(x,Integer)
                                          Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoMatrixPageEmpty1}
\begin{paste}{ugxLinearODEOperatorTwoMatrixPageEmpty1}{ugxLinearODEOperatorTwoMatrixPagePatch1}
\pastebutton{ugxLinearODEOperatorTwoMatrixPageEmpty1}{\showpaste}
\tab{5}\spadcommand{PZ := UnivariatePolynomial(x,Integer)\bound{PZ0 }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoMatrixPagePatch2}
\begin{paste}{ugxLinearODEOperatorTwoMatrixPageFull2}{ugxLinearODEOperatorTwoMatrixPageEmpty2}
\pastebutton{ugxLinearODEOperatorTwoMatrixPageFull2}{\hidepaste}
\tab{5}\spadcommand{x:PZ := 'x\free{PZ0 }\bound{PZ }}
\indentrel{3}\begin{verbatim}
   (2)  x
```

```
                   Type: UnivariatePolynomial(x,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoMatrixPageEmpty2}
\begin{paste}{ugxLinearODEOperatorTwoMatrixPageEmpty2}{ugxLinearODEOperatorTwoMatrixPagePatch2}
\pastebutton{ugxLinearODEOperatorTwoMatrixPageEmpty2}{\showpaste}
\tab{5}\spadcommand{x:PZ := 'x\free{PZ0 }\bound{PZ }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoMatrixPagePatch3}
\begin{paste}{ugxLinearODEOperatorTwoMatrixPageFull3}{ugxLinearODEOperatorTwoMatrixPageEmpty3}
\pastebutton{ugxLinearODEOperatorTwoMatrixPageFull3}{\hidepaste}
\tab{5}\spadcommand{Mat := SquareMatrix(3,PZ)\free{PZ }\bound{Mat }}
\indentrel{3}\begin{verbatim}
   (3)  SquareMatrix(3,UnivariatePolynomial(x,Integer))
                                            Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoMatrixPageEmpty3}
\begin{paste}{ugxLinearODEOperatorTwoMatrixPageEmpty3}{ugxLinearODEOperatorTwoMatrixPagePatch3}
\pastebutton{ugxLinearODEOperatorTwoMatrixPageEmpty3}{\showpaste}
\tab{5}\spadcommand{Mat := SquareMatrix(3,PZ)\free{PZ }\bound{Mat }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoMatrixPagePatch4}
\begin{paste}{ugxLinearODEOperatorTwoMatrixPageFull4}{ugxLinearODEOperatorTwoMatrixPageEmpty4}
\pastebutton{ugxLinearODEOperatorTwoMatrixPageFull4}{\hidepaste}
\tab{5}\spadcommand{Vect := DPMM(3, PZ, Mat, PZ);\free{PZ Mat }\bound{Vect }}
\indentrel{3}\begin{verbatim}
                                            Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoMatrixPageEmpty4}
\begin{paste}{ugxLinearODEOperatorTwoMatrixPageEmpty4}{ugxLinearODEOperatorTwoMatrixPagePatch4}
\pastebutton{ugxLinearODEOperatorTwoMatrixPageEmpty4}{\showpaste}
\tab{5}\spadcommand{Vect := DPMM(3, PZ, Mat, PZ);\free{PZ Mat }\bound{Vect }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoMatrixPagePatch5}
\begin{paste}{ugxLinearODEOperatorTwoMatrixPageFull5}{ugxLinearODEOperatorTwoMatrixPageEmpty5}
\pastebutton{ugxLinearODEOperatorTwoMatrixPageFull5}{\hidepaste}
\tab{5}\spadcommand{Modo := LODO2(Mat, Vect);\free{Mat Vect }\bound{Modo }}
\indentrel{3}\begin{verbatim}
                                            Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugxLinearODEOperatorTwoMatrixPageEmpty5}
\begin{paste}{ugxLinearODEOperatorTwoMatrixPageEmpty5}{ugxLinearODEOperatorTwoMatrixPagePatch5}
\pastebutton{ugxLinearODEOperatorTwoMatrixPageEmpty5}{\showpaste}
\tab{5}\spadcommand{Modo := LODO2(Mat, Vect);\free{Mat Vect }\bound{Modo }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoMatrixPagePatch6}
\begin{paste}{ugxLinearODEOperatorTwoMatrixPageFull6}{ugxLinearODEOperatorTwoMatrixPageEmpty6}
\pastebutton{ugxLinearODEOperatorTwoMatrixPageFull6}{\hidepaste}
\tab{5}\spadcommand{m:Mat := matrix [[x**2,1,0],[1,x**4,0],[0,0,4*x**2]]\free{Mat }\bound{m }}
\indentrel{3}\begin{verbatim}
         2
        x   1    0

   (6)       4
        1   x    0

                  2
        0   0   4x
  Type: SquareMatrix(3,UnivariatePolynomial(x,Integer))
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoMatrixPageEmpty6}
\begin{paste}{ugxLinearODEOperatorTwoMatrixPageEmpty6}{ugxLinearODEOperatorTwoMatrixPagePatch6}
\pastebutton{ugxLinearODEOperatorTwoMatrixPageEmpty6}{\showpaste}
\tab{5}\spadcommand{m:Mat := matrix [[x**2,1,0],[1,x**4,0],[0,0,4*x**2]]\free{Mat }\bound{m }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoMatrixPagePatch7}
\begin{paste}{ugxLinearODEOperatorTwoMatrixPageFull7}{ugxLinearODEOperatorTwoMatrixPageEmpty7}
\pastebutton{ugxLinearODEOperatorTwoMatrixPageFull7}{\hidepaste}
\tab{5}\spadcommand{p:Vect := directProduct [3*x**2+1,2*x,7*x**3+2*x]\free{Vect }\bound{p }}
\indentrel{3}\begin{verbatim}
           2          3
   (7)  [3x  + 1,2x,7x  + 2x]
Type: DirectProductMatrixModule(3,UnivariatePolynomial(x,Integer),SquareMatrix(3,UnivariatePolynomial(x,Inte
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoMatrixPageEmpty7}
\begin{paste}{ugxLinearODEOperatorTwoMatrixPageEmpty7}{ugxLinearODEOperatorTwoMatrixPagePatch7}
\pastebutton{ugxLinearODEOperatorTwoMatrixPageEmpty7}{\showpaste}
\tab{5}\spadcommand{p:Vect := directProduct [3*x**2+1,2*x,7*x**3+2*x]\free{Vect }\bound{p }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoMatrixPagePatch8}
\begin{paste}{ugxLinearODEOperatorTwoMatrixPageFull8}{ugxLinearODEOperatorTwoMatrixPageEmpty8}
\pastebutton{ugxLinearODEOperatorTwoMatrixPageFull8}{\hidepaste}
\tab{5}\spadcommand{q: Vect := m * p\free{m p Vect }\bound{q }}
```

```
\indentrel{3}\begin{verbatim}
          4    2        5     2        5     3
   (8)  [3x  + x  + 2x,2x  + 3x  + 1,28x  + 8x ]
Type: DirectProductMatrixModule(3,UnivariatePolynomial(x,Integer),SquareMatrix(3,UnivariatePolyn
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoMatrixPageEmpty8}
\begin{paste}{ugxLinearODEOperatorTwoMatrixPageEmpty8}{ugxLinearODEOperatorTwoMatrixPagePatch8}
\pastebutton{ugxLinearODEOperatorTwoMatrixPageEmpty8}{\showpaste}
\tab{5}\spadcommand{q: Vect := m * p\free{m p Vect }\bound{q }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoMatrixPagePatch9}
\begin{paste}{ugxLinearODEOperatorTwoMatrixPageFull9}{ugxLinearODEOperatorTwoMatrixPageEmpty9}
\pastebutton{ugxLinearODEOperatorTwoMatrixPageFull9}{\hidepaste}
\tab{5}\spadcommand{Dx : Modo := D()\bound{Dx }\free{Modo }}
\indentrel{3}\begin{verbatim}
   (9)  D
Type: LinearOrdinaryDifferentialOperator2(SquareMatrix(3,UnivariatePolynomial(x,Integer)),Direct
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoMatrixPageEmpty9}
\begin{paste}{ugxLinearODEOperatorTwoMatrixPageEmpty9}{ugxLinearODEOperatorTwoMatrixPagePatch9}
\pastebutton{ugxLinearODEOperatorTwoMatrixPageEmpty9}{\showpaste}
\tab{5}\spadcommand{Dx : Modo := D()\bound{Dx }\free{Modo }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoMatrixPagePatch10}
\begin{paste}{ugxLinearODEOperatorTwoMatrixPageFull10}{ugxLinearODEOperatorTwoMatrixPageEmpty10}
\pastebutton{ugxLinearODEOperatorTwoMatrixPageFull10}{\hidepaste}
\tab{5}\spadcommand{a : Modo := Dx + m\bound{a }\free{m Dx }}
\indentrel{3}\begin{verbatim}
              2
             x   1    0

   (10)  D +      4
             1   x    0

                      2
             0   0   4x
Type: LinearOrdinaryDifferentialOperator2(SquareMatrix(3,UnivariatePolynomial(x,Integer)),Direct
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoMatrixPageEmpty10}
\begin{paste}{ugxLinearODEOperatorTwoMatrixPageEmpty10}{ugxLinearODEOperatorTwoMatrixPagePatch10
\pastebutton{ugxLinearODEOperatorTwoMatrixPageEmpty10}{\showpaste}
\tab{5}\spadcommand{a : Modo := Dx + m\bound{a }\free{m Dx }}
```

```
\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoMatrixPagePatch11}
\begin{paste}{ugxLinearODEOperatorTwoMatrixPageFull11}{ugxLinearODEOperatorTwoMatrixPageEmpty11}
\pastebutton{ugxLinearODEOperatorTwoMatrixPageFull11}{\hidepaste}
\tab{5}\spadcommand{b : Modo := m*Dx + 1\bound{b }\free{m Dx }}
\indentrel{3}\begin{verbatim}
         2
        x   1    0     1  0  0

   (11)       4     D + 0  1  0
        1   x    0
                       0  0  1
                 2
        0   0   4x
Type: LinearOrdinaryDifferentialOperator2(SquareMatrix(3,UnivariatePolynomial(x,Integer)),DirectProductMatr
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoMatrixPageEmpty11}
\begin{paste}{ugxLinearODEOperatorTwoMatrixPageEmpty11}{ugxLinearODEOperatorTwoMatrixPagePatch11}
\pastebutton{ugxLinearODEOperatorTwoMatrixPageEmpty11}{\showpaste}
\tab{5}\spadcommand{b : Modo := m*Dx + 1\bound{b }\free{m Dx }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoMatrixPagePatch12}
\begin{paste}{ugxLinearODEOperatorTwoMatrixPageFull12}{ugxLinearODEOperatorTwoMatrixPageEmpty12}
\pastebutton{ugxLinearODEOperatorTwoMatrixPageFull12}{\hidepaste}
\tab{5}\spadcommand{c := a*b\bound{c }\free{a b }}
\indentrel{3}\begin{verbatim}
   (12)
     2
    x   1    0
               2
         4     D
    1   x    0

             2
    0   0   4x
   +
     4              4    2
    x  + 2x + 2    x  + x            0

       4    2     8     3                   D
      x  + x     x  + 4x  + 2        0

                                    4
         0            0        16x  + 8x + 1
   +
     2
```

```
     x   1    0

          4
     1   x    0

               2
     0   0   4x
Type: LinearOrdinaryDifferentialOperator2(SquareMatrix(3,UnivariatePolynomial(x,Integer)),Direct
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoMatrixPageEmpty12}
\begin{paste}{ugxLinearODEOperatorTwoMatrixPageEmpty12}{ugxLinearODEOperatorTwoMatrixPagePatch12
\pastebutton{ugxLinearODEOperatorTwoMatrixPageEmpty12}{\showpaste}
\tab{5}\spadcommand{c := a*b\bound{c }\free{a b }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoMatrixPagePatch13}
\begin{paste}{ugxLinearODEOperatorTwoMatrixPageFull13}{ugxLinearODEOperatorTwoMatrixPageEmpty13}
\pastebutton{ugxLinearODEOperatorTwoMatrixPageFull13}{\hidepaste}
\tab{5}\spadcommand{a p\free{p a }}
\indentrel{3}\begin{verbatim}
   (13)
      4    2        5     2        5     3      2
   [3x  + x  + 8x,2x  + 3x  + 3,28x  + 8x  + 21x  + 2]
Type: DirectProductMatrixModule(3,UnivariatePolynomial(x,Integer),SquareMatrix(3,UnivariatePolyn
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoMatrixPageEmpty13}
\begin{paste}{ugxLinearODEOperatorTwoMatrixPageEmpty13}{ugxLinearODEOperatorTwoMatrixPagePatch13
\pastebutton{ugxLinearODEOperatorTwoMatrixPageEmpty13}{\showpaste}
\tab{5}\spadcommand{a p\free{p a }}
\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoMatrixPagePatch14}
\begin{paste}{ugxLinearODEOperatorTwoMatrixPageFull14}{ugxLinearODEOperatorTwoMatrixPageEmpty14}
\pastebutton{ugxLinearODEOperatorTwoMatrixPageFull14}{\hidepaste}
\tab{5}\spadcommand{b p\free{p b }}
\indentrel{3}\begin{verbatim}
            3     2       4         4     3     2
   (14)  [6x  + 3x  + 3,2x  + 8x,84x  + 7x  + 8x  + 2x]
Type: DirectProductMatrixModule(3,UnivariatePolynomial(x,Integer),SquareMatrix(3,UnivariatePolyn
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoMatrixPageEmpty14}
\begin{paste}{ugxLinearODEOperatorTwoMatrixPageEmpty14}{ugxLinearODEOperatorTwoMatrixPagePatch14
\pastebutton{ugxLinearODEOperatorTwoMatrixPageEmpty14}{\showpaste}
\tab{5}\spadcommand{b p\free{p b }}
```

```
\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoMatrixPagePatch15}
\begin{paste}{ugxLinearODEOperatorTwoMatrixPageFull15}{ugxLinearODEOperatorTwoMatrixPageEmpty15}
\pastebutton{ugxLinearODEOperatorTwoMatrixPageFull15}{\hidepaste}
\tab{5}\spadcommand{(a + b + c) (p + q)\free{a b c p q }}
\indentrel{3}\begin{verbatim}
   (15)
   [
         8      7      6      5      4      3      2
      10x  + 12x  + 16x  + 30x  + 85x  + 94x  + 40x
    +
      40x + 17
    ,

         12      9      8      7     6      5      4
      10x   + 10x  + 12x  + 92x  + 6x  + 32x  + 72x
    +
         3      2
      28x  + 49x  + 32x + 19
    ,

           8       7        6        5       4       3
      2240x  + 224x  + 1280x  + 3508x  + 492x  + 751x
    +
         2
      98x  + 18x + 4
    ]
Type: DirectProductMatrixModule(3,UnivariatePolynomial(x,Integer),SquareMatrix(3,UnivariatePolynomial(x,Inte
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxLinearODEOperatorTwoMatrixPageEmpty15}
\begin{paste}{ugxLinearODEOperatorTwoMatrixPageEmpty15}{ugxLinearODEOperatorTwoMatrixPagePatch15}
\pastebutton{ugxLinearODEOperatorTwoMatrixPageEmpty15}{\showpaste}
\tab{5}\spadcommand{(a + b + c) (p + q)\free{a b c p q }}
\end{paste}\end{patch}
```

——–

3.68 lpoly.ht

LiePolynomial

— lpoly.ht —

```
\begin{page}{LiePolynomialXmpPage}{LiePolynomial}
\beginscroll
Declaration of domains
\xtc{
}{
\spadpaste{RN    := Fraction Integer \bound{RN}}
}
\xtc{
}{
\spadpaste{Lpoly := LiePolynomial(Symbol,RN) \bound{Lpoly} \free{RN}}
}
\xtc{
}{
\spadpaste{Dpoly := XDPOLY(Symbol,RN) \bound{Dpoly} \free{RN}}
}
\xtc{
}{
\spadpaste{Lword := LyndonWord Symbol \bound{Lword}}
}

Initialisation
\xtc{
}{
\spadpaste{a:Symbol := 'a \bound{a}}
}
\xtc{
}{
\spadpaste{b:Symbol := 'b \bound{b}}
}
\xtc{
}{
\spadpaste{c:Symbol := 'c \bound{c}}
}
\xtc{
}{
\spadpaste{aa: Lpoly := a \bound{aa} \free{Lpoly} \free{a}}
}
\xtc{
}{
\spadpaste{bb: Lpoly := b \bound{bb} \free{Lpoly} \free{b}}
}
\xtc{
}{
\spadpaste{cc: Lpoly := c \bound{cc} \free{Lpoly} \free{c}}
}
\xtc{
}{
\spadpaste{p : Lpoly := [aa,bb] \bound{p} \free{aa} \free{bb} \free{Lpoly}}
}
\xtc{
```

```
}{
\spadpaste{q : Lpoly := [p,bb] \bound{q} \free{p} \free{bb} \free{Lpoly}}
}
\xtc{
All the Lyndon words of order 4
}{
\spadpaste{liste : List Lword := LyndonWordsList([a,b], 4) \free{a}
\free{b} \free{Lword} \bound{liste}}
}
\xtc{
}{
\spadpaste{r: Lpoly := p + q + 3*LiePoly(liste.4)$Lpoly \bound{r}
\free{Lpoly} \free{p} \free{q} \free{liste}}
}
\xtc{
}{
\spadpaste{s:Lpoly := [p,r] \bound{s} \free{Lpoly} \free{p} \free{r}}
}
\xtc{
}{
\spadpaste{t:Lpoly  := s  + 2*LiePoly(liste.3) - 5*LiePoly(liste.5)
\bound{t} \free{Lpoly} \free{s} \free{liste} }
}
\xtc{
}{
\spadpaste{degree t \free{t}}
}
\xtc{
}{
\spadpaste{mirror t \free{t}}
}

Jacobi Relation
\xtc{
}{
\spadpaste{Jacobi(p: Lpoly, q: Lpoly, r: Lpoly):
Lpoly == [[p,q]$Lpoly, r] + [[q,r]$Lpoly, p] +
[[r,p]$Lpoly, q] \free{Lpoly} \bound{J}}
}

Tests
\xtc{
}{
\spadpaste{test: Lpoly := Jacobi(a,b,b) \free{J Lpoly a b} \bound{test1}}
}
\xtc{
}{
\spadpaste{test: Lpoly := Jacobi(p,q,r) \free{J p q r Lpoly} \bound{test2}}
}
\xtc{
```

```
}{
\spadpaste{test: Lpoly := Jacobi(r,s,t) \free{J r s t Lpoly} \bound{test3}}
}

Evaluation
\xtc{
}{
\spadpaste{eval(p, a, p)$Lpoly}
}
\xtc{
}{
\spadpaste{eval(p, [a,b], [2*bb, 3*aa])$Lpoly \free{p a b bb aa Lpoly}}
}
\xtc{
}{
\spadpaste{r: Lpoly := [p,c] \free{p c Lpoly} \bound{rr}}
}
\xtc{
}{
\spadpaste{r1: Lpoly := eval(r, [a,b,c], [bb, cc, aa])$Lpoly
\free{rr a b c aa bb cc Lpoly} \bound{r1}}
}
\xtc{
}{
\spadpaste{r2: Lpoly := eval(r, [a,b,c], [cc, aa, bb])$Lpoly
\free{rr a b c cc bb aa Lpoly} \bound{r2}}
}
\xtc{
}{
\spadpaste{r + r1 + r2 \free{rr r1 r2}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{LiePolynomialXmpPagePatch1}
\begin{paste}{LiePolynomialXmpPageFull1}{LiePolynomialXmpPageEmpty1}
\pastebutton{LiePolynomialXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{RN := Fraction Integer\bound{RN }}
\indentrel{3}\begin{verbatim}
   (1)  Fraction Integer
                                          Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPageEmpty1}
\begin{paste}{LiePolynomialXmpPageEmpty1}{LiePolynomialXmpPagePatch1}
\pastebutton{LiePolynomialXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{RN := Fraction Integer\bound{RN }}
```

```
\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPagePatch2}
\begin{paste}{LiePolynomialXmpPageFull2}{LiePolynomialXmpPageEmpty2}
\pastebutton{LiePolynomialXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{Lpoly := LiePolynomial(Symbol,RN)\bound{Lpoly }\free{RN }}
\indentrel{3}\begin{verbatim}
   (2)  LiePolynomial(Symbol,Fraction Integer)
                                           Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPageEmpty2}
\begin{paste}{LiePolynomialXmpPageEmpty2}{LiePolynomialXmpPagePatch2}
\pastebutton{LiePolynomialXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{Lpoly := LiePolynomial(Symbol,RN)\bound{Lpoly }\free{RN }}
\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPagePatch3}
\begin{paste}{LiePolynomialXmpPageFull3}{LiePolynomialXmpPageEmpty3}
\pastebutton{LiePolynomialXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{Dpoly := XDPOLY(Symbol,RN)\bound{Dpoly }\free{RN }}
\indentrel{3}\begin{verbatim}
   (3)  XDistributedPolynomial(Symbol,Fraction Integer)
                                           Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPageEmpty3}
\begin{paste}{LiePolynomialXmpPageEmpty3}{LiePolynomialXmpPagePatch3}
\pastebutton{LiePolynomialXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{Dpoly := XDPOLY(Symbol,RN)\bound{Dpoly }\free{RN }}
\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPagePatch4}
\begin{paste}{LiePolynomialXmpPageFull4}{LiePolynomialXmpPageEmpty4}
\pastebutton{LiePolynomialXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{Lword := LyndonWord Symbol\bound{Lword }}
\indentrel{3}\begin{verbatim}
   (4)  LyndonWord Symbol
                                           Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPageEmpty4}
\begin{paste}{LiePolynomialXmpPageEmpty4}{LiePolynomialXmpPagePatch4}
\pastebutton{LiePolynomialXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{Lword := LyndonWord Symbol\bound{Lword }}
\end{paste}\end{patch}
```

```
\begin{patch}{LiePolynomialXmpPagePatch5}
\begin{paste}{LiePolynomialXmpPageFull5}{LiePolynomialXmpPageEmpty5}
\pastebutton{LiePolynomialXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{a:Symbol := 'a\bound{a }}
\indentrel{3}\begin{verbatim}
   (5)  a
                                          Type: Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPageEmpty5}
\begin{paste}{LiePolynomialXmpPageEmpty5}{LiePolynomialXmpPagePatch5}
\pastebutton{LiePolynomialXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{a:Symbol := 'a\bound{a }}
\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPagePatch6}
\begin{paste}{LiePolynomialXmpPageFull6}{LiePolynomialXmpPageEmpty6}
\pastebutton{LiePolynomialXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{b:Symbol := 'b\bound{b }}
\indentrel{3}\begin{verbatim}
   (6)  b
                                          Type: Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPageEmpty6}
\begin{paste}{LiePolynomialXmpPageEmpty6}{LiePolynomialXmpPagePatch6}
\pastebutton{LiePolynomialXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{b:Symbol := 'b\bound{b }}
\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPagePatch7}
\begin{paste}{LiePolynomialXmpPageFull7}{LiePolynomialXmpPageEmpty7}
\pastebutton{LiePolynomialXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{c:Symbol := 'c\bound{c }}
\indentrel{3}\begin{verbatim}
   (7)  c
                                          Type: Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPageEmpty7}
\begin{paste}{LiePolynomialXmpPageEmpty7}{LiePolynomialXmpPagePatch7}
\pastebutton{LiePolynomialXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{c:Symbol := 'c\bound{c }}
\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPagePatch8}
\begin{paste}{LiePolynomialXmpPageFull8}{LiePolynomialXmpPageEmpty8}
```

```
\pastebutton{LiePolynomialXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{aa: Lpoly := a\bound{aa }\free{Lpoly }\free{a }}
\indentrel{3}\begin{verbatim}
   (8)  [a]
           Type: LiePolynomial(Symbol,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPageEmpty8}
\begin{paste}{LiePolynomialXmpPageEmpty8}{LiePolynomialXmpPagePatch8}
\pastebutton{LiePolynomialXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{aa: Lpoly := a\bound{aa }\free{Lpoly }\free{a }}
\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPagePatch9}
\begin{paste}{LiePolynomialXmpPageFull9}{LiePolynomialXmpPageEmpty9}
\pastebutton{LiePolynomialXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{bb: Lpoly := b\bound{bb }\free{Lpoly }\free{b }}
\indentrel{3}\begin{verbatim}
   (9)  [b]
           Type: LiePolynomial(Symbol,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPageEmpty9}
\begin{paste}{LiePolynomialXmpPageEmpty9}{LiePolynomialXmpPagePatch9}
\pastebutton{LiePolynomialXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{bb: Lpoly := b\bound{bb }\free{Lpoly }\free{b }}
\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPagePatch10}
\begin{paste}{LiePolynomialXmpPageFull10}{LiePolynomialXmpPageEmpty10}
\pastebutton{LiePolynomialXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{cc: Lpoly := c\bound{cc }\free{Lpoly }\free{c }}
\indentrel{3}\begin{verbatim}
   (10)  [c]
           Type: LiePolynomial(Symbol,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPageEmpty10}
\begin{paste}{LiePolynomialXmpPageEmpty10}{LiePolynomialXmpPagePatch10}
\pastebutton{LiePolynomialXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{cc: Lpoly := c\bound{cc }\free{Lpoly }\free{c }}
\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPagePatch11}
\begin{paste}{LiePolynomialXmpPageFull11}{LiePolynomialXmpPageEmpty11}
\pastebutton{LiePolynomialXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{p : Lpoly := [aa,bb]\bound{p }\free{aa }\free{bb }\free{Lpoly }}
```

```
\indentrel{3}\begin{verbatim}
   (11)  [a b]
           Type: LiePolynomial(Symbol,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPageEmpty11}
\begin{paste}{LiePolynomialXmpPageEmpty11}{LiePolynomialXmpPagePatch11}
\pastebutton{LiePolynomialXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{p : Lpoly := [aa,bb]\bound{p }\free{aa }\free{bb }\free{Lpoly }}
\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPagePatch12}
\begin{paste}{LiePolynomialXmpPageFull12}{LiePolynomialXmpPageEmpty12}
\pastebutton{LiePolynomialXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{q : Lpoly := [p,bb]\bound{q }\free{p }\free{bb }\free{Lpoly }}
\indentrel{3}\begin{verbatim}
             2
   (12)  [a b ]
           Type: LiePolynomial(Symbol,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPageEmpty12}
\begin{paste}{LiePolynomialXmpPageEmpty12}{LiePolynomialXmpPagePatch12}
\pastebutton{LiePolynomialXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{q : Lpoly := [p,bb]\bound{q }\free{p }\free{bb }\free{Lpoly }}
\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPagePatch13}
\begin{paste}{LiePolynomialXmpPageFull13}{LiePolynomialXmpPageEmpty13}
\pastebutton{LiePolynomialXmpPageFull13}{\hidepaste}
\tab{5}\spadcommand{liste : List Lword := LyndonWordsList([a,b], 4)\free{a }\free{b }\free{Lword
\indentrel{3}\begin{verbatim}
   (13)
                    2       2    3     2 2      3
   [[a],[b],[a b],[a b],[a b ],[a b],[a b ],[a b ]]
                           Type: List LyndonWord Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPageEmpty13}
\begin{paste}{LiePolynomialXmpPageEmpty13}{LiePolynomialXmpPagePatch13}
\pastebutton{LiePolynomialXmpPageEmpty13}{\showpaste}
\tab{5}\spadcommand{liste : List Lword := LyndonWordsList([a,b], 4)\free{a }\free{b }\free{Lword
\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPagePatch14}
\begin{paste}{LiePolynomialXmpPageFull14}{LiePolynomialXmpPageEmpty14}
\pastebutton{LiePolynomialXmpPageFull14}{\hidepaste}
```

```
\tab{5}\spadcommand{r: Lpoly := p + q + 3*LiePoly(liste.4)$Lpoly\bound{r }\free{Lpoly }\free{p }\free{q }\f
\indentrel{3}\begin{verbatim}
                   2         2
   (14)  [a b] + 3[a b] + [a b ]
           Type: LiePolynomial(Symbol,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPageEmpty14}
\begin{paste}{LiePolynomialXmpPageEmpty14}{LiePolynomialXmpPagePatch14}
\pastebutton{LiePolynomialXmpPageEmpty14}{\showpaste}
\tab{5}\spadcommand{r: Lpoly := p + q + 3*LiePoly(liste.4)$Lpoly\bound{r }\free{Lpoly }\free{p }\free{q }\f
\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPagePatch15}
\begin{paste}{LiePolynomialXmpPageFull15}{LiePolynomialXmpPageEmpty15}
\pastebutton{LiePolynomialXmpPageFull15}{\hidepaste}
\tab{5}\spadcommand{s:Lpoly := [p,r]\bound{s }\free{Lpoly }\free{p }\free{r }}
\indentrel{3}\begin{verbatim}
             2                 2
   (15)  - 3[a b a b] + [a b a b ]
           Type: LiePolynomial(Symbol,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPageEmpty15}
\begin{paste}{LiePolynomialXmpPageEmpty15}{LiePolynomialXmpPagePatch15}
\pastebutton{LiePolynomialXmpPageEmpty15}{\showpaste}
\tab{5}\spadcommand{s:Lpoly := [p,r]\bound{s }\free{Lpoly }\free{p }\free{r }}
\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPagePatch16}
\begin{paste}{LiePolynomialXmpPageFull16}{LiePolynomialXmpPageEmpty16}
\pastebutton{LiePolynomialXmpPageFull16}{\hidepaste}
\tab{5}\spadcommand{t:Lpoly := s + 2*LiePoly(liste.3) - 5*LiePoly(liste.5)\bound{t }\free{Lpoly }\free{s }\
\indentrel{3}\begin{verbatim}
                     2       2                 2
   (16)  2[a b] - 5[a b ] - 3[a b a b] + [a b a b ]
           Type: LiePolynomial(Symbol,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPageEmpty16}
\begin{paste}{LiePolynomialXmpPageEmpty16}{LiePolynomialXmpPagePatch16}
\pastebutton{LiePolynomialXmpPageEmpty16}{\showpaste}
\tab{5}\spadcommand{t:Lpoly := s + 2*LiePoly(liste.3) - 5*LiePoly(liste.5)\bound{t }\free{Lpoly }\free{s }\
\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPagePatch17}
\begin{paste}{LiePolynomialXmpPageFull17}{LiePolynomialXmpPageEmpty17}
```

```
\pastebutton{LiePolynomialXmpPageFull17}{\hidepaste}
\tab{5}\spadcommand{degree t\free{t }}
\indentrel{3}\begin{verbatim}
   (17)  5
                                   Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPageEmpty17}
\begin{paste}{LiePolynomialXmpPageEmpty17}{LiePolynomialXmpPagePatch17}
\pastebutton{LiePolynomialXmpPageEmpty17}{\showpaste}
\tab{5}\spadcommand{degree t\free{t }}
\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPagePatch18}
\begin{paste}{LiePolynomialXmpPageFull18}{LiePolynomialXmpPageEmpty18}
\pastebutton{LiePolynomialXmpPageFull18}{\hidepaste}
\tab{5}\spadcommand{mirror t\free{t }}
\indentrel{3}\begin{verbatim}
                         2       2                 2
   (18)  - 2[a b] - 5[a b ] - 3[a b a b] + [a b a b ]
           Type: LiePolynomial(Symbol,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPageEmpty18}
\begin{paste}{LiePolynomialXmpPageEmpty18}{LiePolynomialXmpPagePatch18}
\pastebutton{LiePolynomialXmpPageEmpty18}{\showpaste}
\tab{5}\spadcommand{mirror t\free{t }}
\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPagePatch19}
\begin{paste}{LiePolynomialXmpPageFull19}{LiePolynomialXmpPageEmpty19}
\pastebutton{LiePolynomialXmpPageFull19}{\hidepaste}
\tab{5}\spadcommand{Jacobi(p: Lpoly, q: Lpoly, r: Lpoly): Lpoly == [[p,q]$Lpoly, r] + [[q,r]$Lpo
\indentrel{3}\begin{verbatim}
                                              Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPageEmpty19}
\begin{paste}{LiePolynomialXmpPageEmpty19}{LiePolynomialXmpPagePatch19}
\pastebutton{LiePolynomialXmpPageEmpty19}{\showpaste}
\tab{5}\spadcommand{Jacobi(p: Lpoly, q: Lpoly, r: Lpoly): Lpoly == [[p,q]$Lpoly, r] + [[q,r]$Lpo
\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPagePatch20}
\begin{paste}{LiePolynomialXmpPageFull20}{LiePolynomialXmpPageEmpty20}
\pastebutton{LiePolynomialXmpPageFull20}{\hidepaste}
\tab{5}\spadcommand{test: Lpoly := Jacobi(a,b,b)\free{J Lpoly a b }\bound{test1 }}
```

```
\indentrel{3}\begin{verbatim}
   (20)  0
           Type: LiePolynomial(Symbol,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPageEmpty20}
\begin{paste}{LiePolynomialXmpPageEmpty20}{LiePolynomialXmpPagePatch20}
\pastebutton{LiePolynomialXmpPageEmpty20}{\showpaste}
\tab{5}\spadcommand{test: Lpoly := Jacobi(a,b,b)\free{J Lpoly a b }\bound{test1 }}
\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPagePatch21}
\begin{paste}{LiePolynomialXmpPageFull21}{LiePolynomialXmpPageEmpty21}
\pastebutton{LiePolynomialXmpPageFull21}{\hidepaste}
\tab{5}\spadcommand{test: Lpoly := Jacobi(p,q,r)\free{J p q r Lpoly }\bound{test2 }}
\indentrel{3}\begin{verbatim}
   (21)  0
           Type: LiePolynomial(Symbol,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPageEmpty21}
\begin{paste}{LiePolynomialXmpPageEmpty21}{LiePolynomialXmpPagePatch21}
\pastebutton{LiePolynomialXmpPageEmpty21}{\showpaste}
\tab{5}\spadcommand{test: Lpoly := Jacobi(p,q,r)\free{J p q r Lpoly }\bound{test2 }}
\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPagePatch22}
\begin{paste}{LiePolynomialXmpPageFull22}{LiePolynomialXmpPageEmpty22}
\pastebutton{LiePolynomialXmpPageFull22}{\hidepaste}
\tab{5}\spadcommand{test: Lpoly := Jacobi(r,s,t)\free{J r s t Lpoly }\bound{test3 }}
\indentrel{3}\begin{verbatim}
   (22)  0
           Type: LiePolynomial(Symbol,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPageEmpty22}
\begin{paste}{LiePolynomialXmpPageEmpty22}{LiePolynomialXmpPagePatch22}
\pastebutton{LiePolynomialXmpPageEmpty22}{\showpaste}
\tab{5}\spadcommand{test: Lpoly := Jacobi(r,s,t)\free{J r s t Lpoly }\bound{test3 }}
\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPagePatch23}
\begin{paste}{LiePolynomialXmpPageFull23}{LiePolynomialXmpPageEmpty23}
\pastebutton{LiePolynomialXmpPageFull23}{\hidepaste}
\tab{5}\spadcommand{eval(p, a, p)$Lpoly}
\indentrel{3}\begin{verbatim}
             2
```

```
   (23)  [a b ]
            Type: LiePolynomial(Symbol,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPageEmpty23}
\begin{paste}{LiePolynomialXmpPageEmpty23}{LiePolynomialXmpPagePatch23}
\pastebutton{LiePolynomialXmpPageEmpty23}{\showpaste}
\tab{5}\spadcommand{eval(p, a, p)$Lpoly}
\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPagePatch24}
\begin{paste}{LiePolynomialXmpPageFull24}{LiePolynomialXmpPageEmpty24}
\pastebutton{LiePolynomialXmpPageFull24}{\hidepaste}
\tab{5}\spadcommand{eval(p, [a,b], [2*bb, 3*aa])$Lpoly\free{p a b bb aa Lpoly }}
\indentrel{3}\begin{verbatim}
   (24)  - 6[a b]
            Type: LiePolynomial(Symbol,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPageEmpty24}
\begin{paste}{LiePolynomialXmpPageEmpty24}{LiePolynomialXmpPagePatch24}
\pastebutton{LiePolynomialXmpPageEmpty24}{\showpaste}
\tab{5}\spadcommand{eval(p, [a,b], [2*bb, 3*aa])$Lpoly\free{p a b bb aa Lpoly }}
\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPagePatch25}
\begin{paste}{LiePolynomialXmpPageFull25}{LiePolynomialXmpPageEmpty25}
\pastebutton{LiePolynomialXmpPageFull25}{\hidepaste}
\tab{5}\spadcommand{r: Lpoly := [p,c]\free{p c Lpoly }\bound{rr }}
\indentrel{3}\begin{verbatim}
   (25)  [a b c] + [a c b]
            Type: LiePolynomial(Symbol,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPageEmpty25}
\begin{paste}{LiePolynomialXmpPageEmpty25}{LiePolynomialXmpPagePatch25}
\pastebutton{LiePolynomialXmpPageEmpty25}{\showpaste}
\tab{5}\spadcommand{r: Lpoly := [p,c]\free{p c Lpoly }\bound{rr }}
\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPagePatch26}
\begin{paste}{LiePolynomialXmpPageFull26}{LiePolynomialXmpPageEmpty26}
\pastebutton{LiePolynomialXmpPageFull26}{\hidepaste}
\tab{5}\spadcommand{r1: Lpoly := eval(r, [a,b,c], [bb, cc, aa])$Lpoly\free{rr a b c aa bb cc Lpo
\indentrel{3}\begin{verbatim}
   (26)  - [a b c]
            Type: LiePolynomial(Symbol,Fraction Integer)
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPageEmpty26}
\begin{paste}{LiePolynomialXmpPageEmpty26}{LiePolynomialXmpPagePatch26}
\pastebutton{LiePolynomialXmpPageEmpty26}{\showpaste}
\tab{5}\spadcommand{r1: Lpoly := eval(r, [a,b,c], [bb, cc, aa])$Lpoly\free{rr a b c aa bb cc Lpoly }\bound{r
\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPagePatch27}
\begin{paste}{LiePolynomialXmpPageFull27}{LiePolynomialXmpPageEmpty27}
\pastebutton{LiePolynomialXmpPageFull27}{\hidepaste}
\tab{5}\spadcommand{r2: Lpoly := eval(r, [a,b,c], [cc, aa, bb])$Lpoly\free{rr a b c cc bb aa Lpoly }\bound{r
\indentrel{3}\begin{verbatim}
   (27)  - [a c b]
           Type: LiePolynomial(Symbol,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPageEmpty27}
\begin{paste}{LiePolynomialXmpPageEmpty27}{LiePolynomialXmpPagePatch27}
\pastebutton{LiePolynomialXmpPageEmpty27}{\showpaste}
\tab{5}\spadcommand{r2: Lpoly := eval(r, [a,b,c], [cc, aa, bb])$Lpoly\free{rr a b c cc bb aa Lpoly }\bound{r
\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPagePatch28}
\begin{paste}{LiePolynomialXmpPageFull28}{LiePolynomialXmpPageEmpty28}
\pastebutton{LiePolynomialXmpPageFull28}{\hidepaste}
\tab{5}\spadcommand{r + r1 + r2\free{rr r1 r2 }}
\indentrel{3}\begin{verbatim}
   (28)  0
           Type: LiePolynomial(Symbol,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LiePolynomialXmpPageEmpty28}
\begin{paste}{LiePolynomialXmpPageEmpty28}{LiePolynomialXmpPagePatch28}
\pastebutton{LiePolynomialXmpPageEmpty28}{\showpaste}
\tab{5}\spadcommand{r + r1 + r2\free{rr r1 r2 }}
\end{paste}\end{patch}
```

———

3.69 lword.ht

LyndonWord

— lword.ht —

```
\begin{page}{LyndonWordXmpPage}{LyndonWord}
\beginscroll
Initialisations
\xtc{
}{
\spadpaste{a:Symbol :='a \bound{a}}
}
\xtc{
}{
\spadpaste{b:Symbol :='b \bound{b}}
}
\xtc{
}{
\spadpaste{c:Symbol :='c \bound{c}}
}
\xtc{
}{
\spadpaste{lword:= LyndonWord(Symbol) \bound{lword}}
}
\xtc{
}{
\spadpaste{magma := Magma(Symbol) \bound{magma}}
}
\xtc{
}{
\spadpaste{word   := OrderedFreeMonoid(Symbol) \bound{word}}
}
\xtc{
All Lyndon words of with a, b, c to order 3
}{
\spadpaste{LyndonWordsList1([a,b,c],3)$lword \free{lword} \free{a}
\free{b} \free{c} }
}
\xtc{
All Lyndon words of with a, b, c to order 3 in flat list
}{
\spadpaste{LyndonWordsList([a,b,c],3)$lword \free{a} \free{b}
\free{c} \free{lword}}
}
\xtc{
All Lyndon words of with a, b to order 5
}{
```

```
\spadpaste{lw := LyndonWordsList([a,b],5)$lword \free{a} \free{b}
\free{lword} \bound{lw}}
}
\xtc{
}{
\spadpaste{w1 : word := lw.4 :: word \free{word} \free{lw} \bound{w1}}
}
\xtc{
}{
\spadpaste{w2 : word := lw.5 :: word \free{word} \free{lw} \bound{w2}}
}

Let's try factoring
\xtc{
}{
\spadpaste{factor(a::word)$lword \free{a word lword}}
}
\xtc{
}{
\spadpaste{factor(w1*w2)$lword \free{ w1 w2 lword}}
}
\xtc{
}{
\spadpaste{factor(w2*w2)$lword \free{w2 lword}}
}
\xtc{
}{
\spadpaste{factor(w2*w1)$lword \free{w1 w2 lword}}
}

Checks and coercions
\xtc{
}{
\spadpaste{lyndon?(w1)$lword \free{w1 lword}}
}
\xtc{
}{
\spadpaste{lyndon?(w1*w2)$lword \free{w1 w2 lword}}
}
\xtc{
}{
\spadpaste{lyndon?(w2*w1)$lword \free{w1 w2 lword}}
}
\xtc{
}{
\spadpaste{lyndonIfCan(w1)$lword \free{w1 lword}}
}
\xtc{
}{
\spadpaste{lyndonIfCan(w2*w1)$lword \free{w1 w2 lword}}
```

```
}
\xtc{
}{
\spadpaste{lyndon(w1)$lword \free{w1 lword}}
}
\xtc{
}{
\spadpaste{lyndon(w1*w2)$lword \free{w1 w2 lword}}
}
\endscroll
\autobuttons
\end{page}

\begin{patch}{LyndonWordXmpPagePatch1}
\begin{paste}{LyndonWordXmpPageFull1}{LyndonWordXmpPageEmpty1}
\pastebutton{LyndonWordXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{a:Symbol :='a\bound{a }}
\indentrel{3}\begin{verbatim}
   (1)  a
                                            Type: Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LyndonWordXmpPageEmpty1}
\begin{paste}{LyndonWordXmpPageEmpty1}{LyndonWordXmpPagePatch1}
\pastebutton{LyndonWordXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{a:Symbol :='a\bound{a }}
\end{paste}\end{patch}

\begin{patch}{LyndonWordXmpPagePatch2}
\begin{paste}{LyndonWordXmpPageFull2}{LyndonWordXmpPageEmpty2}
\pastebutton{LyndonWordXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{b:Symbol :='b\bound{b }}
\indentrel{3}\begin{verbatim}
   (2)  b
                                            Type: Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LyndonWordXmpPageEmpty2}
\begin{paste}{LyndonWordXmpPageEmpty2}{LyndonWordXmpPagePatch2}
\pastebutton{LyndonWordXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{b:Symbol :='b\bound{b }}
\end{paste}\end{patch}

\begin{patch}{LyndonWordXmpPagePatch3}
\begin{paste}{LyndonWordXmpPageFull3}{LyndonWordXmpPageEmpty3}
\pastebutton{LyndonWordXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{c:Symbol :='c\bound{c }}
\indentrel{3}\begin{verbatim}
```

```
   (3)  c
                                           Type: Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LyndonWordXmpPageEmpty3}
\begin{paste}{LyndonWordXmpPageEmpty3}{LyndonWordXmpPagePatch3}
\pastebutton{LyndonWordXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{c:Symbol :='c\bound{c }}
\end{paste}\end{patch}

\begin{patch}{LyndonWordXmpPagePatch4}
\begin{paste}{LyndonWordXmpPageFull4}{LyndonWordXmpPageEmpty4}
\pastebutton{LyndonWordXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{lword:= LyndonWord(Symbol)\bound{lword }}
\indentrel{3}\begin{verbatim}
   (4)  LyndonWord Symbol
                                           Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LyndonWordXmpPageEmpty4}
\begin{paste}{LyndonWordXmpPageEmpty4}{LyndonWordXmpPagePatch4}
\pastebutton{LyndonWordXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{lword:= LyndonWord(Symbol)\bound{lword }}
\end{paste}\end{patch}

\begin{patch}{LyndonWordXmpPagePatch5}
\begin{paste}{LyndonWordXmpPageFull5}{LyndonWordXmpPageEmpty5}
\pastebutton{LyndonWordXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{magma := Magma(Symbol)\bound{magma }}
\indentrel{3}\begin{verbatim}
   (5)  Magma Symbol
                                           Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LyndonWordXmpPageEmpty5}
\begin{paste}{LyndonWordXmpPageEmpty5}{LyndonWordXmpPagePatch5}
\pastebutton{LyndonWordXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{magma := Magma(Symbol)\bound{magma }}
\end{paste}\end{patch}

\begin{patch}{LyndonWordXmpPagePatch6}
\begin{paste}{LyndonWordXmpPageFull6}{LyndonWordXmpPageEmpty6}
\pastebutton{LyndonWordXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{word := OrderedFreeMonoid(Symbol)\bound{word }}
\indentrel{3}\begin{verbatim}
   (6)  OrderedFreeMonoid Symbol
                                           Type: Domain
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LyndonWordXmpPageEmpty6}
\begin{paste}{LyndonWordXmpPageEmpty6}{LyndonWordXmpPagePatch6}
\pastebutton{LyndonWordXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{word := OrderedFreeMonoid(Symbol)\bound{word }}
\end{paste}\end{patch}

\begin{patch}{LyndonWordXmpPagePatch7}
\begin{paste}{LyndonWordXmpPageFull7}{LyndonWordXmpPageEmpty7}
\pastebutton{LyndonWordXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{LyndonWordsList1([a,b,c],3)$lword\free{lword }\free{a }\free{b }\free{c }}
\indentrel{3}\begin{verbatim}
   (7)
   [[[a],[b],[c]], [[a b],[a c],[b c]],

        2      2        2                         2
     [[a b], [a c], [a b ], [a b c], [a c b], [a c ],
        2        2
      [b c], [b c ]]
     ]
       Type: OneDimensionalArray List LyndonWord Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LyndonWordXmpPageEmpty7}
\begin{paste}{LyndonWordXmpPageEmpty7}{LyndonWordXmpPagePatch7}
\pastebutton{LyndonWordXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{LyndonWordsList1([a,b,c],3)$lword\free{lword }\free{a }\free{b }\free{c }}
\end{paste}\end{patch}

\begin{patch}{LyndonWordXmpPagePatch8}
\begin{paste}{LyndonWordXmpPageFull8}{LyndonWordXmpPageEmpty8}
\pastebutton{LyndonWordXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{LyndonWordsList([a,b,c],3)$lword\free{a }\free{b }\free{c }\free{lword }}
\indentrel{3}\begin{verbatim}
   (8)
                                          2      2
   [[a], [b], [c], [a b], [a c], [b c], [a b], [a c],
        2                       2     2        2
    [a b ], [a b c], [a c b], [a c ], [b c], [b c ]]
                           Type: List LyndonWord Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LyndonWordXmpPageEmpty8}
\begin{paste}{LyndonWordXmpPageEmpty8}{LyndonWordXmpPagePatch8}
\pastebutton{LyndonWordXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{LyndonWordsList([a,b,c],3)$lword\free{a }\free{b }\free{c }\free{lword }}
```

```
\end{paste}\end{patch}

\begin{patch}{LyndonWordXmpPagePatch9}
\begin{paste}{LyndonWordXmpPageFull9}{LyndonWordXmpPageEmpty9}
\pastebutton{LyndonWordXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{lw := LyndonWordsList([a,b],5)$lword\free{a }\free{b }\free{lword }\bound{lw }}
\indentrel{3}\begin{verbatim}
   (9)
                       2        2     3      2 2
   [[a], [b], [a b], [a b], [a b ], [a b], [a b ],
        3     4      3 2     2          2 3           2
    [a b ], [a b], [a b ], [a b a b], [a b ], [a b a b ],
        4
    [a b ]]
                           Type: List LyndonWord Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LyndonWordXmpPageEmpty9}
\begin{paste}{LyndonWordXmpPageEmpty9}{LyndonWordXmpPagePatch9}
\pastebutton{LyndonWordXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{lw := LyndonWordsList([a,b],5)$lword\free{a }\free{b }\free{lword }\bound{lw }}
\end{paste}\end{patch}

\begin{patch}{LyndonWordXmpPagePatch10}
\begin{paste}{LyndonWordXmpPageFull10}{LyndonWordXmpPageEmpty10}
\pastebutton{LyndonWordXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{w1 : word := lw.4 :: word\free{word }\free{lw }\bound{w1 }}
\indentrel{3}\begin{verbatim}
          2
   (10)  a b
                         Type: OrderedFreeMonoid Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LyndonWordXmpPageEmpty10}
\begin{paste}{LyndonWordXmpPageEmpty10}{LyndonWordXmpPagePatch10}
\pastebutton{LyndonWordXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{w1 : word := lw.4 :: word\free{word }\free{lw }\bound{w1 }}
\end{paste}\end{patch}

\begin{patch}{LyndonWordXmpPagePatch11}
\begin{paste}{LyndonWordXmpPageFull11}{LyndonWordXmpPageEmpty11}
\pastebutton{LyndonWordXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{w2 : word := lw.5 :: word\free{word }\free{lw }\bound{w2 }}
\indentrel{3}\begin{verbatim}
            2
   (11)  a b
                         Type: OrderedFreeMonoid Symbol
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LyndonWordXmpPageEmpty11}
\begin{paste}{LyndonWordXmpPageEmpty11}{LyndonWordXmpPagePatch11}
\pastebutton{LyndonWordXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{w2 : word := lw.5 :: word\free{word }\free{lw }\bound{w2 }}
\end{paste}\end{patch}

\begin{patch}{LyndonWordXmpPagePatch12}
\begin{paste}{LyndonWordXmpPageFull12}{LyndonWordXmpPageEmpty12}
\pastebutton{LyndonWordXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{factor(a::word)$lword\free{a word lword }}
\indentrel{3}\begin{verbatim}
   (12)  [[a]]
                           Type: List LyndonWord Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LyndonWordXmpPageEmpty12}
\begin{paste}{LyndonWordXmpPageEmpty12}{LyndonWordXmpPagePatch12}
\pastebutton{LyndonWordXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{factor(a::word)$lword\free{a word lword }}
\end{paste}\end{patch}

\begin{patch}{LyndonWordXmpPagePatch13}
\begin{paste}{LyndonWordXmpPageFull13}{LyndonWordXmpPageEmpty13}
\pastebutton{LyndonWordXmpPageFull13}{\hidepaste}
\tab{5}\spadcommand{factor(w1*w2)$lword\free{w1 w2 lword }}
\indentrel{3}\begin{verbatim}
            2     2
   (13)  [[a b a b ]]
                           Type: List LyndonWord Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LyndonWordXmpPageEmpty13}
\begin{paste}{LyndonWordXmpPageEmpty13}{LyndonWordXmpPagePatch13}
\pastebutton{LyndonWordXmpPageEmpty13}{\showpaste}
\tab{5}\spadcommand{factor(w1*w2)$lword\free{w1 w2 lword }}
\end{paste}\end{patch}

\begin{patch}{LyndonWordXmpPagePatch14}
\begin{paste}{LyndonWordXmpPageFull14}{LyndonWordXmpPageEmpty14}
\pastebutton{LyndonWordXmpPageFull14}{\hidepaste}
\tab{5}\spadcommand{factor(w2*w2)$lword\free{w2 lword }}
\indentrel{3}\begin{verbatim}
              2      2
   (14)  [[a b ],[a b ]]
                           Type: List LyndonWord Symbol
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LyndonWordXmpPageEmpty14}
\begin{paste}{LyndonWordXmpPageEmpty14}{LyndonWordXmpPagePatch14}
\pastebutton{LyndonWordXmpPageEmpty14}{\showpaste}
\tab{5}\spadcommand{factor(w2*w2)$lword\free{w2 lword }}
\end{paste}\end{patch}

\begin{patch}{LyndonWordXmpPagePatch15}
\begin{paste}{LyndonWordXmpPageFull15}{LyndonWordXmpPageEmpty15}
\pastebutton{LyndonWordXmpPageFull15}{\hidepaste}
\tab{5}\spadcommand{factor(w2*w1)$lword\free{w1 w2 lword }}
\indentrel{3}\begin{verbatim}
             2    2
   (15)  [[a b ],[a b]]
                         Type: List LyndonWord Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LyndonWordXmpPageEmpty15}
\begin{paste}{LyndonWordXmpPageEmpty15}{LyndonWordXmpPagePatch15}
\pastebutton{LyndonWordXmpPageEmpty15}{\showpaste}
\tab{5}\spadcommand{factor(w2*w1)$lword\free{w1 w2 lword }}
\end{paste}\end{patch}

\begin{patch}{LyndonWordXmpPagePatch16}
\begin{paste}{LyndonWordXmpPageFull16}{LyndonWordXmpPageEmpty16}
\pastebutton{LyndonWordXmpPageFull16}{\hidepaste}
\tab{5}\spadcommand{lyndon?(w1)$lword\free{w1 lword }}
\indentrel{3}\begin{verbatim}
   (16)  true
                                       Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LyndonWordXmpPageEmpty16}
\begin{paste}{LyndonWordXmpPageEmpty16}{LyndonWordXmpPagePatch16}
\pastebutton{LyndonWordXmpPageEmpty16}{\showpaste}
\tab{5}\spadcommand{lyndon?(w1)$lword\free{w1 lword }}
\end{paste}\end{patch}

\begin{patch}{LyndonWordXmpPagePatch17}
\begin{paste}{LyndonWordXmpPageFull17}{LyndonWordXmpPageEmpty17}
\pastebutton{LyndonWordXmpPageFull17}{\hidepaste}
\tab{5}\spadcommand{lyndon?(w1*w2)$lword\free{w1 w2 lword }}
\indentrel{3}\begin{verbatim}
   (17)  true
                                       Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{LyndonWordXmpPageEmpty17}
\begin{paste}{LyndonWordXmpPageEmpty17}{LyndonWordXmpPagePatch17}
\pastebutton{LyndonWordXmpPageEmpty17}{\showpaste}
\tab{5}\spadcommand{lyndon?(w1*w2)$lword\free{w1 w2 lword }}
\end{paste}\end{patch}

\begin{patch}{LyndonWordXmpPagePatch18}
\begin{paste}{LyndonWordXmpPageFull18}{LyndonWordXmpPageEmpty18}
\pastebutton{LyndonWordXmpPageFull18}{\hidepaste}
\tab{5}\spadcommand{lyndon?(w2*w1)$lword\free{w1 w2 lword }}
\indentrel{3}\begin{verbatim}
   (18)  false
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LyndonWordXmpPageEmpty18}
\begin{paste}{LyndonWordXmpPageEmpty18}{LyndonWordXmpPagePatch18}
\pastebutton{LyndonWordXmpPageEmpty18}{\showpaste}
\tab{5}\spadcommand{lyndon?(w2*w1)$lword\free{w1 w2 lword }}
\end{paste}\end{patch}

\begin{patch}{LyndonWordXmpPagePatch19}
\begin{paste}{LyndonWordXmpPageFull19}{LyndonWordXmpPageEmpty19}
\pastebutton{LyndonWordXmpPageFull19}{\hidepaste}
\tab{5}\spadcommand{lyndonIfCan(w1)$lword\free{w1 lword }}
\indentrel{3}\begin{verbatim}
           2
   (19)  [a b]
                     Type: Union(LyndonWord Symbol,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LyndonWordXmpPageEmpty19}
\begin{paste}{LyndonWordXmpPageEmpty19}{LyndonWordXmpPagePatch19}
\pastebutton{LyndonWordXmpPageEmpty19}{\showpaste}
\tab{5}\spadcommand{lyndonIfCan(w1)$lword\free{w1 lword }}
\end{paste}\end{patch}

\begin{patch}{LyndonWordXmpPagePatch20}
\begin{paste}{LyndonWordXmpPageFull20}{LyndonWordXmpPageEmpty20}
\pastebutton{LyndonWordXmpPageFull20}{\hidepaste}
\tab{5}\spadcommand{lyndonIfCan(w2*w1)$lword\free{w1 w2 lword }}
\indentrel{3}\begin{verbatim}
   (20)  "failed"
                              Type: Union("failed",...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{LyndonWordXmpPageEmpty20}
\begin{paste}{LyndonWordXmpPageEmpty20}{LyndonWordXmpPagePatch20}
\pastebutton{LyndonWordXmpPageEmpty20}{\showpaste}
\tab{5}\spadcommand{lyndonIfCan(w2*w1)$lword\free{w1 w2 lword }}
\end{paste}\end{patch}

\begin{patch}{LyndonWordXmpPagePatch21}
\begin{paste}{LyndonWordXmpPageFull21}{LyndonWordXmpPageEmpty21}
\pastebutton{LyndonWordXmpPageFull21}{\hidepaste}
\tab{5}\spadcommand{lyndon(w1)$lword\free{w1 lword }}
\indentrel{3}\begin{verbatim}
           2
   (21)  [a b]
                                Type: LyndonWord Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LyndonWordXmpPageEmpty21}
\begin{paste}{LyndonWordXmpPageEmpty21}{LyndonWordXmpPagePatch21}
\pastebutton{LyndonWordXmpPageEmpty21}{\showpaste}
\tab{5}\spadcommand{lyndon(w1)$lword\free{w1 lword }}
\end{paste}\end{patch}

\begin{patch}{LyndonWordXmpPagePatch22}
\begin{paste}{LyndonWordXmpPageFull22}{LyndonWordXmpPageEmpty22}
\pastebutton{LyndonWordXmpPageFull22}{\hidepaste}
\tab{5}\spadcommand{lyndon(w1*w2)$lword\free{w1 w2 lword }}
\indentrel{3}\begin{verbatim}
           2     2
   (22)  [a b a b ]
                                Type: LyndonWord Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{LyndonWordXmpPageEmpty22}
\begin{paste}{LyndonWordXmpPageEmpty22}{LyndonWordXmpPagePatch22}
\pastebutton{LyndonWordXmpPageEmpty22}{\showpaste}
\tab{5}\spadcommand{lyndon(w1*w2)$lword\free{w1 w2 lword }}
\end{paste}\end{patch}
```

3.70 magma.ht

Magma

— magma.ht —

```
\begin{page}{MagmaXmpPage}{Magma}
\beginscroll
Initialisations
\xtc{
}{
\spadpaste{x:Symbol :='x \bound{x}}
}
\xtc{
}{
\spadpaste{y:Symbol :='y \bound{y}}
}
\xtc{
}{
\spadpaste{z:Symbol :='z \bound{z}}
}
\xtc{
}{
\spadpaste{word := OrderedFreeMonoid(Symbol) \bound{word}}
}
\xtc{
}{
\spadpaste{tree := Magma(Symbol) \bound{tree}}
}

Let's make some trees
\xtc{
}{
\spadpaste{a:tree := x*x \free{x tree} \bound{a}}
}
\xtc{
}{
\spadpaste{b:tree := y*y \free{y tree} \bound{b}}
}
\xtc{
}{
\spadpaste{c:tree := a*b \free{a b tree} \bound{c}}
}

Query the trees
\xtc{
}{
\spadpaste{left c \free{c}}
```

```
}
\xtc{
}{
\spadpaste{right c \free{c}}
}
\xtc{
}{
\spadpaste{length c \free{c}}
}
\xtc{
Coerce to the monoid
}{
\spadpaste{c::word \free{c word}}
}

Check ordering
\xtc{
}{
\spadpaste{a < b \free{a b}}
}
\xtc{
}{
\spadpaste{a < c \free{a c}}
}
\xtc{
}{
\spadpaste{b < c \free{b c}}
}

Navigate the tree
\xtc{
}{
\spadpaste{first c \free{c}}
}
\xtc{
}{
\spadpaste{rest c \free{c}}
}
\xtc{
}{
\spadpaste{rest rest c  \free{c}}
}

Check ordering
\xtc{
}{
\spadpaste{ax:tree := a*x \free{a x tree} \bound{ax}}
}
\xtc{
}{
```

```
\spadpaste{xa:tree := x*a \free{a x tree} \bound{xa}}
}
\xtc{
}{
\spadpaste{xa < ax \free{ax xa}}
}
\xtc{
}{
\spadpaste{lexico(xa,ax) \free{ax xa}}
}
\endscroll
\autobuttons
\end{page}

\begin{patch}{MagmaXmpPagePatch1}
\begin{paste}{MagmaXmpPageFull1}{MagmaXmpPageEmpty1}
\pastebutton{MagmaXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{x:Symbol :='x\bound{x }}
\indentrel{3}\begin{verbatim}
   (1)  x
                                          Type: Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MagmaXmpPageEmpty1}
\begin{paste}{MagmaXmpPageEmpty1}{MagmaXmpPagePatch1}
\pastebutton{MagmaXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{x:Symbol :='x\bound{x }}
\end{paste}\end{patch}

\begin{patch}{MagmaXmpPagePatch2}
\begin{paste}{MagmaXmpPageFull2}{MagmaXmpPageEmpty2}
\pastebutton{MagmaXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{y:Symbol :='y\bound{y }}
\indentrel{3}\begin{verbatim}
   (2)  y
                                          Type: Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MagmaXmpPageEmpty2}
\begin{paste}{MagmaXmpPageEmpty2}{MagmaXmpPagePatch2}
\pastebutton{MagmaXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{y:Symbol :='y\bound{y }}
\end{paste}\end{patch}

\begin{patch}{MagmaXmpPagePatch3}
\begin{paste}{MagmaXmpPageFull3}{MagmaXmpPageEmpty3}
\pastebutton{MagmaXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{z:Symbol :='z\bound{z }}
```

```
\indentrel{3}\begin{verbatim}
   (3)  z
                                          Type: Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MagmaXmpPageEmpty3}
\begin{paste}{MagmaXmpPageEmpty3}{MagmaXmpPagePatch3}
\pastebutton{MagmaXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{z:Symbol :='z\bound{z }}
\end{paste}\end{patch}

\begin{patch}{MagmaXmpPagePatch4}
\begin{paste}{MagmaXmpPageFull4}{MagmaXmpPageEmpty4}
\pastebutton{MagmaXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{word := OrderedFreeMonoid(Symbol)\bound{word }}
\indentrel{3}\begin{verbatim}
   (4)  OrderedFreeMonoid Symbol
                                          Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MagmaXmpPageEmpty4}
\begin{paste}{MagmaXmpPageEmpty4}{MagmaXmpPagePatch4}
\pastebutton{MagmaXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{word := OrderedFreeMonoid(Symbol)\bound{word }}
\end{paste}\end{patch}

\begin{patch}{MagmaXmpPagePatch5}
\begin{paste}{MagmaXmpPageFull5}{MagmaXmpPageEmpty5}
\pastebutton{MagmaXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{tree := Magma(Symbol)\bound{tree }}
\indentrel{3}\begin{verbatim}
   (5)  Magma Symbol
                                          Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MagmaXmpPageEmpty5}
\begin{paste}{MagmaXmpPageEmpty5}{MagmaXmpPagePatch5}
\pastebutton{MagmaXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{tree := Magma(Symbol)\bound{tree }}
\end{paste}\end{patch}

\begin{patch}{MagmaXmpPagePatch6}
\begin{paste}{MagmaXmpPageFull6}{MagmaXmpPageEmpty6}
\pastebutton{MagmaXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{a:tree := x*x\free{x tree }\bound{a }}
\indentrel{3}\begin{verbatim}
   (6)  [x,x]
```

```
                                    Type: Magma Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MagmaXmpPageEmpty6}
\begin{paste}{MagmaXmpPageEmpty6}{MagmaXmpPagePatch6}
\pastebutton{MagmaXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{a:tree := x*x\free{x tree }\bound{a }}
\end{paste}\end{patch}

\begin{patch}{MagmaXmpPagePatch7}
\begin{paste}{MagmaXmpPageFull7}{MagmaXmpPageEmpty7}
\pastebutton{MagmaXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{b:tree := y*y\free{y tree }\bound{b }}
\indentrel{3}\begin{verbatim}
   (7)  [y,y]
                                    Type: Magma Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MagmaXmpPageEmpty7}
\begin{paste}{MagmaXmpPageEmpty7}{MagmaXmpPagePatch7}
\pastebutton{MagmaXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{b:tree := y*y\free{y tree }\bound{b }}
\end{paste}\end{patch}

\begin{patch}{MagmaXmpPagePatch8}
\begin{paste}{MagmaXmpPageFull8}{MagmaXmpPageEmpty8}
\pastebutton{MagmaXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{c:tree := a*b\free{a b tree }\bound{c }}
\indentrel{3}\begin{verbatim}
   (8)  [[x,x],[y,y]]
                                    Type: Magma Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MagmaXmpPageEmpty8}
\begin{paste}{MagmaXmpPageEmpty8}{MagmaXmpPagePatch8}
\pastebutton{MagmaXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{c:tree := a*b\free{a b tree }\bound{c }}
\end{paste}\end{patch}

\begin{patch}{MagmaXmpPagePatch9}
\begin{paste}{MagmaXmpPageFull9}{MagmaXmpPageEmpty9}
\pastebutton{MagmaXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{left c\free{c }}
\indentrel{3}\begin{verbatim}
   (9)  [x,x]
                                    Type: Magma Symbol
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MagmaXmpPageEmpty9}
\begin{paste}{MagmaXmpPageEmpty9}{MagmaXmpPagePatch9}
\pastebutton{MagmaXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{left c\free{c }}
\end{paste}\end{patch}

\begin{patch}{MagmaXmpPagePatch10}
\begin{paste}{MagmaXmpPageFull10}{MagmaXmpPageEmpty10}
\pastebutton{MagmaXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{right c\free{c }}
\indentrel{3}\begin{verbatim}
   (10)  [y,y]
                                     Type: Magma Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MagmaXmpPageEmpty10}
\begin{paste}{MagmaXmpPageEmpty10}{MagmaXmpPagePatch10}
\pastebutton{MagmaXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{right c\free{c }}
\end{paste}\end{patch}

\begin{patch}{MagmaXmpPagePatch11}
\begin{paste}{MagmaXmpPageFull11}{MagmaXmpPageEmpty11}
\pastebutton{MagmaXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{length c\free{c }}
\indentrel{3}\begin{verbatim}
   (11)  4
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MagmaXmpPageEmpty11}
\begin{paste}{MagmaXmpPageEmpty11}{MagmaXmpPagePatch11}
\pastebutton{MagmaXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{length c\free{c }}
\end{paste}\end{patch}

\begin{patch}{MagmaXmpPagePatch12}
\begin{paste}{MagmaXmpPageFull12}{MagmaXmpPageEmpty12}
\pastebutton{MagmaXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{c::word\free{c word }}
\indentrel{3}\begin{verbatim}
          2 2
   (12)  x y
                         Type: OrderedFreeMonoid Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{MagmaXmpPageEmpty12}
\begin{paste}{MagmaXmpPageEmpty12}{MagmaXmpPagePatch12}
\pastebutton{MagmaXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{c::word\free{c word }}
\end{paste}\end{patch}

\begin{patch}{MagmaXmpPagePatch13}
\begin{paste}{MagmaXmpPageFull13}{MagmaXmpPageEmpty13}
\pastebutton{MagmaXmpPageFull13}{\hidepaste}
\tab{5}\spadcommand{a < b\free{a b }}
\indentrel{3}\begin{verbatim}
   (13)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MagmaXmpPageEmpty13}
\begin{paste}{MagmaXmpPageEmpty13}{MagmaXmpPagePatch13}
\pastebutton{MagmaXmpPageEmpty13}{\showpaste}
\tab{5}\spadcommand{a < b\free{a b }}
\end{paste}\end{patch}

\begin{patch}{MagmaXmpPagePatch14}
\begin{paste}{MagmaXmpPageFull14}{MagmaXmpPageEmpty14}
\pastebutton{MagmaXmpPageFull14}{\hidepaste}
\tab{5}\spadcommand{a < c\free{a c }}
\indentrel{3}\begin{verbatim}
   (14)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MagmaXmpPageEmpty14}
\begin{paste}{MagmaXmpPageEmpty14}{MagmaXmpPagePatch14}
\pastebutton{MagmaXmpPageEmpty14}{\showpaste}
\tab{5}\spadcommand{a < c\free{a c }}
\end{paste}\end{patch}

\begin{patch}{MagmaXmpPagePatch15}
\begin{paste}{MagmaXmpPageFull15}{MagmaXmpPageEmpty15}
\pastebutton{MagmaXmpPageFull15}{\hidepaste}
\tab{5}\spadcommand{b < c\free{b c }}
\indentrel{3}\begin{verbatim}
   (15)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MagmaXmpPageEmpty15}
```

```
\begin{paste}{MagmaXmpPageEmpty15}{MagmaXmpPagePatch15}
\pastebutton{MagmaXmpPageEmpty15}{\showpaste}
\tab{5}\spadcommand{b < c\free{b c }}
\end{paste}\end{patch}

\begin{patch}{MagmaXmpPagePatch16}
\begin{paste}{MagmaXmpPageFull16}{MagmaXmpPageEmpty16}
\pastebutton{MagmaXmpPageFull16}{\hidepaste}
\tab{5}\spadcommand{first c\free{c }}
\indentrel{3}\begin{verbatim}
   (16)  x
                                          Type: Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MagmaXmpPageEmpty16}
\begin{paste}{MagmaXmpPageEmpty16}{MagmaXmpPagePatch16}
\pastebutton{MagmaXmpPageEmpty16}{\showpaste}
\tab{5}\spadcommand{first c\free{c }}
\end{paste}\end{patch}

\begin{patch}{MagmaXmpPagePatch17}
\begin{paste}{MagmaXmpPageFull17}{MagmaXmpPageEmpty17}
\pastebutton{MagmaXmpPageFull17}{\hidepaste}
\tab{5}\spadcommand{rest c\free{c }}
\indentrel{3}\begin{verbatim}
   (17)  [x,[y,y]]
                                    Type: Magma Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MagmaXmpPageEmpty17}
\begin{paste}{MagmaXmpPageEmpty17}{MagmaXmpPagePatch17}
\pastebutton{MagmaXmpPageEmpty17}{\showpaste}
\tab{5}\spadcommand{rest c\free{c }}
\end{paste}\end{patch}

\begin{patch}{MagmaXmpPagePatch18}
\begin{paste}{MagmaXmpPageFull18}{MagmaXmpPageEmpty18}
\pastebutton{MagmaXmpPageFull18}{\hidepaste}
\tab{5}\spadcommand{rest rest c\free{c }}
\indentrel{3}\begin{verbatim}
   (18)  [y,y]
                                    Type: Magma Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MagmaXmpPageEmpty18}
\begin{paste}{MagmaXmpPageEmpty18}{MagmaXmpPagePatch18}
\pastebutton{MagmaXmpPageEmpty18}{\showpaste}
```

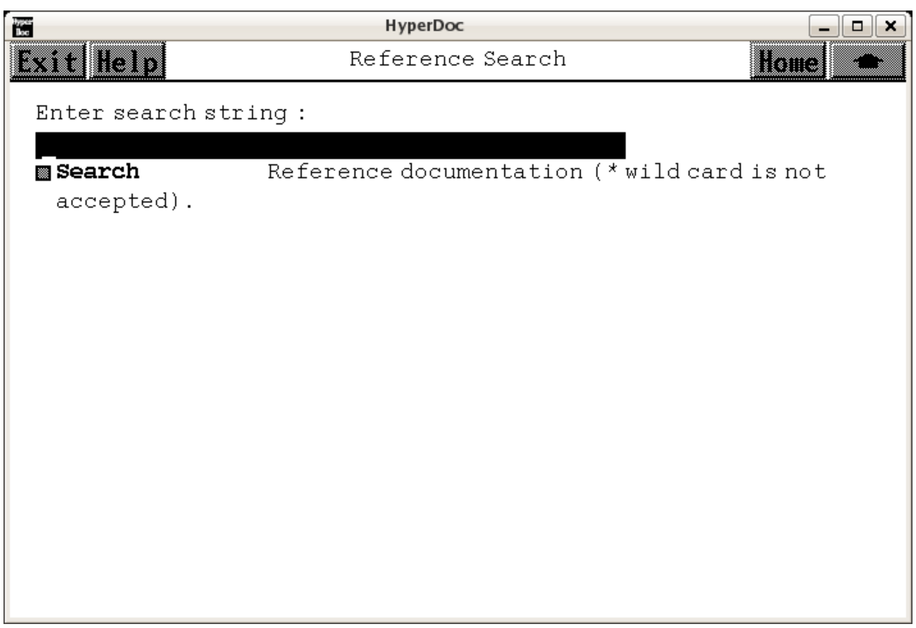
```
\tab{5}\spadcommand{rest rest c\free{c }}
\end{paste}\end{patch}

\begin{patch}{MagmaXmpPagePatch19}
\begin{paste}{MagmaXmpPageFull19}{MagmaXmpPageEmpty19}
\pastebutton{MagmaXmpPageFull19}{\hidepaste}
\tab{5}\spadcommand{ax:tree := a*x\free{a x tree }\bound{ax }}
\indentrel{3}\begin{verbatim}
   (19)  [[x,x],x]
                                      Type: Magma Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MagmaXmpPageEmpty19}
\begin{paste}{MagmaXmpPageEmpty19}{MagmaXmpPagePatch19}
\pastebutton{MagmaXmpPageEmpty19}{\showpaste}
\tab{5}\spadcommand{ax:tree := a*x\free{a x tree }\bound{ax }}
\end{paste}\end{patch}

\begin{patch}{MagmaXmpPagePatch20}
\begin{paste}{MagmaXmpPageFull20}{MagmaXmpPageEmpty20}
\pastebutton{MagmaXmpPageFull20}{\hidepaste}
\tab{5}\spadcommand{xa:tree := x*a\free{a x tree }\bound{xa }}
\indentrel{3}\begin{verbatim}
   (20)  [x,[x,x]]
                                      Type: Magma Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MagmaXmpPageEmpty20}
\begin{paste}{MagmaXmpPageEmpty20}{MagmaXmpPagePatch20}
\pastebutton{MagmaXmpPageEmpty20}{\showpaste}
\tab{5}\spadcommand{xa:tree := x*a\free{a x tree }\bound{xa }}
\end{paste}\end{patch}

\begin{patch}{MagmaXmpPagePatch21}
\begin{paste}{MagmaXmpPageFull21}{MagmaXmpPageEmpty21}
\pastebutton{MagmaXmpPageFull21}{\hidepaste}
\tab{5}\spadcommand{xa < ax\free{ax xa }}
\indentrel{3}\begin{verbatim}
   (21)  true
                                           Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MagmaXmpPageEmpty21}
\begin{paste}{MagmaXmpPageEmpty21}{MagmaXmpPagePatch21}
\pastebutton{MagmaXmpPageEmpty21}{\showpaste}
\tab{5}\spadcommand{xa < ax\free{ax xa }}
\end{paste}\end{patch}
```

```
\begin{patch}{MagmaXmpPagePatch22}
\begin{paste}{MagmaXmpPageFull22}{MagmaXmpPageEmpty22}
\pastebutton{MagmaXmpPageFull22}{\hidepaste}
\tab{5}\spadcommand{lexico(xa,ax)\free{ax xa }}
\indentrel{3}\begin{verbatim}
   (22)  false
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MagmaXmpPageEmpty22}
\begin{paste}{MagmaXmpPageEmpty22}{MagmaXmpPagePatch22}
\pastebutton{MagmaXmpPageEmpty22}{\showpaste}
\tab{5}\spadcommand{lexico(xa,ax)\free{ax xa }}
\end{paste}\end{patch}
```

———-

3.71 man0.ht

Reference Search



⇐ "Reference" (TopReferencePage) 3.1 on page 123

⇒ “Search” (ugSysCmdPage) 19 on page 2538

— man0.ht —

```
\begin{page}{RefSearchPage}{Reference Search}
\beginscroll
Enter search string :
\inputstring{pattern}{40}{}
\newline
\beginmenu
\menuunixlink{Search}
  {htsearch "\stringvalue{pattern}"}
  \tab{15} Reference documentation ({\em *} wild card is not accepted).
\endmenu
\endscroll
\end{page}
```

———-

Lisp Functions

⇐ “Root Page” (RootPage) 3.1 on page 117
⇐ “System Commands” (TopSettingsPage) 3.1 on page 120
⇐ “Axiom Browser” (Man0Page) 3.71 on page 962
⇐ “Axiom Reference” (TopReferencePage) 3.1 on page 123
⇐ “Basic Commands” (BasicCommand) 3.6 on page 155
⇐ “Calculus” (Calculus) 3.6 on page 156
⇐ “Glossary” (GlossaryPage) 3.49 on page 579
⇐ “The Axiom Link to NAG Software” (htxl) 3.63 on page 844
⇐ “C02 Zeros of Polynomials” (c02) 3.63 on page 846
⇐ “C02 Roots of One or More TranscendentalEquations” (c05) 3.63 on page 847
⇐ “C06 Summation of Series” (c06) 3.63 on page 847
⇐ “D03 Partial Differential Equations” (d03) 3.63 on page 852
⇐ “E01 Interpolation” (e01) 3.63 on page 853
⇐ “E02 Curve and Surface Fitting” (e02) 3.63 on page 854
⇐ “E04 Minimizing or Maximizing a Function” (e04) 3.63 on page 856
⇐ “F01 Matrix Operations - Including Inversion” (f01) 3.63 on page 857
⇐ “F02 Eigenvalues and Eigenvectors” (f02) 3.63 on page 858
⇐ “F04 Simultaneous Linear Equations” (f04) 3.63 on page 860
⇐ “F07 Linear Equations (LAPACK)” (f07) 3.63 on page 862
⇐ “S – Approximations of Special Functions” (s) 3.63 on page 863
⇐ “Domain Mapping” (DomainMapping) 3.72 on page 964
⇐ “Domain Record” (DomainRecord) 3.92 on page 1182
⇐ “Domain Union” (DomainUnion) 3.110 on page 1319
⇐ “HTXLinkPage4xPatch2” (HTXLinkPage4xPatch2) 21.24 on page 2686

⇐ “HTXLinkPage4xPatch4” (HTXLinkPage4xPatch4) 21.24 on page 2688
⇐ “Examples Using the Axiom/NAG Expert System” (UXANNAOdeEx) 24.1 on page 3981
⇐ “Examples Using the Axiom/NAG Expert System” (UXANNAIntEx) 24.1 on page 3979
⇐ “Examples Using the Axiom/NAG Expert System” (UXANNAOpt2Ex) 24.1 on page 3978

⇐ “Examples Using the Axiom/NAG Expert System” (UXANNAOptEx) 24.1 on page 3977
⇐ “Optimization” (UXANNAOpt) 24.1 on page 3975
⇐ “Ordinary Differential Equations” (UXANNAOde) 24.1 on page 3975
⇐ “Integration” (UXANNAInt) 24.1 on page 3974
⇐ “Axiom/NAG Expert System” (UXANNA) 24.1 on page 3973

There are lisp functions which exist behind certain pages and the page information is created dynamically by calling the function.

This is a list of the macros which use lisp functions

oPage	axiomFun
oPageFrom	axiomFunFrom
htGloss	gloss
htGloss	spadglos
htGloss	glossSee
conPage	aliascon
conPage	aliasdom
conPage	dom
conPage	con
conPage	conf
conOpPage	ops
htsn	keyword
htsn	op

This is a list of the lisp function, the link name, and the page name

- **annaOde** – Ordinary Differential Equations link from page Ordinary Differential Equations
- **annaOpt** – Optimization of a Single Multivariate Function link from page Optimization
- **annaOpt2** – Optimization of a set of observations of a data set link from page Optimization
- **annaPDESolve** – Second Order Elliptic Partial Differential Equation link from page Partial Differential Equations
- **annaOptDefaultSolve1** – Example 1 link from page Examples Using the Axiom/NAG Expert System
- **annaOptDefaultSolve2** – Example 2 link from page Examples Using the Axiom/NAG Expert System

- **annaOptDefaultSolve3** – Example 3 link from page Examples Using the Axiom/NAG Expert System
- **annaOptDefaultSolve4** – Example 4 link from page Examples Using the Axiom/NAG Expert System
- **annaOptDefaultSolve5** – Example 5 link from page Examples Using the Axiom/NAG Expert System
- **annaFoo** – Example 1 link from page Examples Using the Axiom/NAG Expert System
- **annaBar** – Example 2 link from page Examples Using the Axiom/NAG Expert System
- **annaJoe** – Example 3 link from page Examples Using the Axiom/NAG Expert System
- **annaSue** – Example 4 link from page Examples Using the Axiom/NAG Expert System
- **annaAnn** – Example 5 link from page Examples Using the Axiom/NAG Expert System
- **annaBab** – Example 6 link from page Examples Using the Axiom/NAG Expert System
- **annaFnar** – Example 7 link from page Examples Using the Axiom/NAG Expert System
- **annaDan** – Example 8 link from page Examples Using the Axiom/NAG Expert System
- **annaBlah** – Example 9 link from page Examples Using the Axiom/NAG Expert System
- **annaTub** – Example 10 link from page Examples Using the Axiom/NAG Expert System
- **annaRats** – Example 11 link from page Examples Using the Axiom/NAG Expert System
- **annaMInt** – Example 12 link from page Examples Using the Axiom/NAG Expert System
- **annaMInt** – Multiple Integration link from page Integration
- **annaInt** – Integration link from page Integration
- **annaOdeDefaultSolve1** – Example 1 link from page Examples Using the Axiom/NAG Expert System
- **annaOdeDefaultSolve2** – Example 2 link from page Examples Using the Axiom/NAG Expert System
- **annaOpt2DefaultSolve** – Example 1 link from page Examples Using the Axiom/NAG Expert System

- **aSearch** – Attributes link from page Axiom Browser
- **aokSearch** – General link from page Axiom Browser
- **bcDefiniteIntegrate** – Do an Definite Integral link from page Calculus
- **bcDifferentiate** – Differentiate link from page Calculus
- **bcDraw** – Draw link from page Basic Commands
- **bcIndefiniteIntegrate** – Do an Indefinite Integral link from page Calculus
- **bcLimit** – Find a Limit link from page Calculus
- **bcMatrix** – Matrix link from page Basic Commands
- **bcSeries** – Series link from page Basic Commands
- **bcSolve** – Solve link from page Basic Commands
- **bcSum** – Sum link from page Calculus
- **cSearch** – categories link from page Axiom Browser
- **dbSpecialDescription** – Description link from page Domain Mapping
- **dbSpecialDescription** – Description link from page Domain Record
- **dbSpecialDescription** – Description link from page Domain Union
- **dbSpecialDescription** – Description link from page Domain Untagged Union
- **dbSpecialExports** – Exports link from page Domain Union
- **dbSpecialOperations** – Operations link from page Domain Mapping
- **dbSpecialOperations** – Operations link from page Domain Record
- **dbSpecialOperations** – Operations link from page Domain Union
- **dbSpecialOperations** – Operations link from page Domain Untagged Union
- **dSearch** – domains link from page Axiom Browser
- **detailSearch** – domains link from page Axiom Browser
- **docSearch** – Documentation link from page Axiom Browser
- **genSearch** – Complete link from page Axiom Browser
- **htGloss** – Search link from page Glossary
- **HTSEARCH** – Reference link from page Axiom Browser
- **htsv** – System Variables link from page Axiom Reference

- **htSystemVariables** – Settings link from page System Commands
- **issueHT** – Definition link from page HTXLinkPage4xPatch2
- **kSearch** – Constructors link from page Axiom Browser
- **kSearch** – Browser pages for individual routines link from page htxl
- **kSearch** – C02 Zeros of Polynomials link from page c02
- **kSearch** – Browse link from page c05 Roots of One or More Transcendental Equations
- **kSearch** – Browse link from page c06 Summation of Series
- **kSearch** – Browse link from page d01 Quadrature
- **kSearch** – Browse link from page d02 Ordinary Differential Equations
- **kSearch** – Browse link from page d03 Partial Differential Equations
- **kSearch** – Browse link from page e01 Interpolation
- **kSearch** – Browse link from page e02 Curve and Surface Fitting
- **kSearch** – Browse link from page e04 Minimizing or Maximizing a Function
- **kSearch** – Browse link from page F01 Matrix Operations - Including Inversion
- **kSearch** – Browse link from page F02 Eigenvalues and Eigenvectors
- **kSearch** – Browse link from page F04 Simultaneous Linear Equations
- **linkToHTPage** – Interpret link from page HTXLinkPage4xPatch3
- **oSearch** – Operations link from page Axiom Browser
- **pSearch** – packages link from page Axiom Browser
- **startHTPage**– Interpret link from page HTXLinkPage4xPatch4

- **c02aff** – C02AFF link from page C02 Zeros of Polynomials
- **c02agf** – C02AGF link from page C02 Zeros of Polynomials

- **c05adf** – C05ADF link from page C05 Roots of One or More Transcendental Equations
- **c05ndf** – C05NDF link from page C05 Roots of One or More Transcendental Equations
- **c05pbf** – C05PBF link from page C05 Roots of One or More Transcendental Equations

- **c06eaf** – C06EAF link from page C06 Summation of Series
- **c06ebf** – C06EBF link from page C06 Summation of Series

- **c06ecf** – C06ECF link from page C06 Summation of Series
- **c06ekf** – C06EKF link from page C06 Summation of Series
- **c06fpf** – C06FPF link from page C06 Summation of Series
- **c06fqf** – C06FQF link from page C06 Summation of Series
- **c06frf** – C06FRF link from page C06 Summation of Series
- **c06fuf** – C06FUF link from page C06 Summation of Series
- **c06gbf** – C06GBF link from page C06 Summation of Series
- **c06gcf** – C06GCF link from page C06 Summation of Series
- **c06gqf** – C06GQF link from page C06 Summation of Series
- **c06gsf** – C06GSF link from page C06 Summation of Series

- **d01ajf** – D01AJF link from page D01 Quadrature
- **d01akf** – D01AKF link from page D01 Quadrature
- **d01alf** – D01ALF link from page D01 Quadrature
- **d01amf** – D01AMF link from page D01 Quadrature
- **d01anf** – D01ANF link from page D01 Quadrature
- **d01apf** – D01APF link from page D01 Quadrature
- **d01aqf** – D01AQF link from page D01 Quadrature
- **d01asf** – D01ASF link from page D01 Quadrature
- **d01bbf** – D01BBF link from page D01 Quadrature
- **d01fcf** – D01FCF link from page D01 Quadrature
- **d01gaf** – D01GAF link from page D01 Quadrature
- **d01gbf** – D01GBF link from page D01 Quadrature

- **d02bbf** – D02BBF link from page D02 Ordinary Differential Equations
- **d02bhf** – D02BHF link from page D02 Ordinary Differential Equations
- **d02cjf** – D02CJF link from page D02 Ordinary Differential Equations
- **d02ejf** – D02EJF link from page D02 Ordinary Differential Equations
- **d02gaf** – D02GAF link from page D02 Ordinary Differential Equations

- **d02gbf** – D02GBF link from page D02 Ordinary Differential Equations
- **d02kef** – D02KEF link from page D02 Ordinary Differential Equations
- **d02raf** – D02RAF link from page D02 Ordinary Differential Equations

- **d03edf** – D03EDF link from page D03 Partial Differential Equations
- **d03eef** – D03EEF link from page D03 Partial Differential Equations
- **d03faf** – D03FAF link from page D03 Partial Differential Equations

- **e01baf** – E01BAF link from page E01 Interpolation
- **e01bef** – E01BEF link from page E01 Interpolation
- **e01bff** – E01BFF link from page E01 Interpolation
- **e01bgf** – E01BGF link from page E01 Interpolation
- **e01bhf** – E01BHF link from page E01 Interpolation
- **e01daf** – E01DAF link from page E01 Interpolation
- **e01saf** – E01SAF link from page E01 Interpolation
- **e01sef** – E01SEF link from page E01 Interpolation

- **e02adf** – E02ADF link from page E02 Curve and Surface Fitting
- **e02aef** – E02AEF link from page E02 Curve and Surface Fitting
- **e02agf** – E02AGF link from page E02 Curve and Surface Fitting
- **e02ahf** – E02AHF link from page E02 Curve and Surface Fitting
- **e02ajf** – E02AJF link from page E02 Curve and Surface Fitting
- **e02akf** – E02AKF link from page E02 Curve and Surface Fitting
- **e02baf** – E02BAF link from page E02 Curve and Surface Fitting
- **e02bbf** – E02BBF link from page E02 Curve and Surface Fitting
- **e02bcf** – E02BCF link from page E02 Curve and Surface Fitting
- **e02bdf** – E02BDF link from page E02 Curve and Surface Fitting
- **e02bef** – E02BEF link from page E02 Curve and Surface Fitting
- **e02daf** – E02DAF link from page E02 Curve and Surface Fitting
- **e02dcf** – E02DCF link from page E02 Curve and Surface Fitting

- **e02ddf** – E02DDF link from page E02 Curve and Surface Fitting
- **e02def** – E02DEF link from page E02 Curve and Surface Fitting
- **e02dff** – E02DFF link from page E02 Curve and Surface Fitting
- **e02gaf** – E02GAF link from page E02 Curve and Surface Fitting
- **e02zaf** – E02ZAF link from page E02 Curve and Surface Fitting

- **e04dgf** – E04DGF link from page E04 Minimizing or Maximizing a Function
- **e04fdf** – E04FDF link from page E04 Minimizing or Maximizing a Function
- **e04gcf** – E04GCF link from page E04 Minimizing or Maximizing a Function
- **e04jaf** – E04JAF link from page E04 Minimizing or Maximizing a Function
- **e04mbf** – E04MBF link from page E04 Minimizing or Maximizing a Function
- **e04naf** – E04NAF link from page E04 Minimizing or Maximizing a Function
- **e04ucf** – E04UCF link from page E04 Minimizing or Maximizing a Function
- **e04ycf** – E04YCF link from page E04 Minimizing or Maximizing a Function

- **f01brf** – F01BRF link from page F01 Matrix Operations - Including Inversion
- **f01bsf** – F01BSF link from page F01 Matrix Operations - Including Inversion
- **f01maf** – F01MAF link from page F01 Matrix Operations - Including Inversion
- **f01mcf** – F01MCF link from page F01 Matrix Operations - Including Inversion
- **f01qcf** – F01QCF link from page F01 Matrix Operations - Including Inversion
- **f01qdf** – F01QDF link from page F01 Matrix Operations - Including Inversion
- **f01qef** – F01QEF link from page F01 Matrix Operations - Including Inversion
- **f01rcf** – F01RCF link from page F01 Matrix Operations - Including Inversion
- **f01rdf** – F01RDF link from page F01 Matrix Operations - Including Inversion
- **f01ref** – F01REF link from page F01 Matrix Operations - Including Inversion

- **f02aaf** – F02AAF link from page F02 Eigenvalues and Eigenvectors
- **f02abf** – F02ABF link from page F02 Eigenvalues and Eigenvectors
- **f02adf** – F02ADF link from page F02 Eigenvalues and Eigenvectors
- **f02aef** – F02AEF link from page F02 Eigenvalues and Eigenvectors

- **f02aff** – F02AFF link from page F02 Eigenvalues and Eigenvectors
- **f02agf** – F02AGF link from page F02 Eigenvalues and Eigenvectors
- **f02ajf** – F02AJF link from page F02 Eigenvalues and Eigenvectors
- **f02akf** – F02AKF link from page F02 Eigenvalues and Eigenvectors
- **f02awf** – F02AWF link from page F02 Eigenvalues and Eigenvectors
- **f02axf** – F02AXF link from page F02 Eigenvalues and Eigenvectors
- **f02bbf** – F02BBF link from page F02 Eigenvalues and Eigenvectors
- **f02bjf** – F02BJF link from page F02 Eigenvalues and Eigenvectors
- **f02fjf** – F02FJF link from page F02 Eigenvalues and Eigenvectors
- **f02wef** – F02WEF link from page F02 Eigenvalues and Eigenvectors
- **f02xef** – F02XEF link from page F02 Eigenvalues and Eigenvectors

- **f04adf** – F04ADF link from page F04 Simultaneous Linear Equations
- **f04arf** – F04ARF link from page F04 Simultaneous Linear Equations
- **f04asf** – F04ASF link from page F04 Simultaneous Linear Equations
- **f04atf** – F04ATF link from page F04 Simultaneous Linear Equations
- **f04axf** – F04AXF link from page F04 Simultaneous Linear Equations
- **f04faf** – F04FAF link from page F04 Simultaneous Linear Equations
- **f04jgf** – F04JGF link from page F04 Simultaneous Linear Equations
- **f04maf** – F04MAF link from page F04 Simultaneous Linear Equations
- **f04mbf** – F04MBF link from page F04 Simultaneous Linear Equations
- **f04mcf** – F04MCF link from page F04 Simultaneous Linear Equations
- **f04qaf** – F04QAF link from page F04 Simultaneous Linear Equations

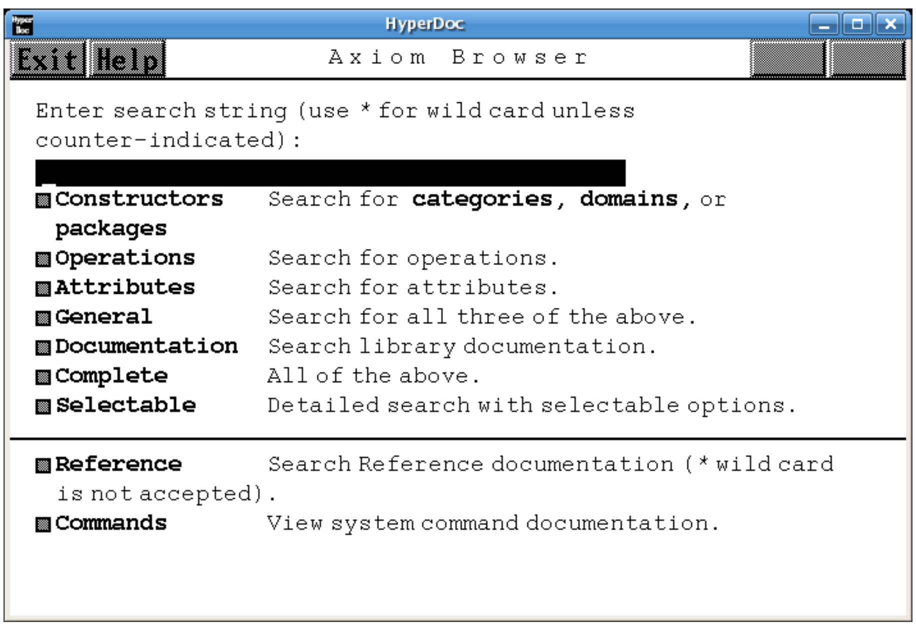
- **f07adf** – F07ADF link from page F07 Linear Equations (LAPACK)
- **f07aef** – F07AEF link from page F07 Linear Equations (LAPACK)
- **f07fdf** – F07FDF link from page F07 Linear Equations (LAPACK)
- **f07fef** – F07FEF link from page F07 Linear Equations (LAPACK)

- **s01eaf** – S01EAF link from page S – Approximations of Special Functions

- **s13aaf** – S13AAF link from page S – Approximations of Special Functions
- **s13acf** – S13ACF link from page S – Approximations of Special Functions
- **s13adf** – S13ADF link from page S – Approximations of Special Functions
- **s14aaf** – S14AAF link from page S – Approximations of Special Functions
- **s14abf** – S14ABF link from page S – Approximations of Special Functions
- **s14baf** – S14BAF link from page S – Approximations of Special Functions
- **s15adf** – S15ADF link from page S – Approximations of Special Functions
- **s15aef** – S15AEF link from page S – Approximations of Special Functions
- **s17acf** – S17ACF link from page S – Approximations of Special Functions
- **s17adf** – S17ADF link from page S – Approximations of Special Functions
- **s17aef** – S17AEF link from page S – Approximations of Special Functions
- **s17aff** – S17AFF link from page S – Approximations of Special Functions
- **s17agf** – S17AGF link from page S – Approximations of Special Functions
- **s17ahf** – S17AHF link from page S – Approximations of Special Functions
- **s17ajf** – S17AJF link from page S – Approximations of Special Functions
- **s17akf** – S17AKF link from page S – Approximations of Special Functions
- **s17dcf** – S17DCF link from page S – Approximations of Special Functions
- **s17def** – S17DEF link from page S – Approximations of Special Functions
- **s17dgf** – S17DGF link from page S – Approximations of Special Functions
- **s17dhf** – S17DHF link from page S – Approximations of Special Functions
- **s17dlf** – S17DLF link from page S – Approximations of Special Functions
- **s18acf** – S18ACF link from page S – Approximations of Special Functions
- **s18adf** – S18ADF link from page S – Approximations of Special Functions
- **s18aef** – S18AEF link from page S – Approximations of Special Functions
- **s18aff** – S18AFF link from page S – Approximations of Special Functions
- **s18dcf** – S18DCF link from page S – Approximations of Special Functions
- **s18def** – S18DEF link from page S – Approximations of Special Functions
- **s19aaf** – S19AAF link from page S – Approximations of Special Functions

- **s19abf** – S19ABF link from page S – Approximations of Special Functions
- **s19acf** – S19ACF link from page S – Approximations of Special Functions
- **s19adf** – S19ADF link from page S – Approximations of Special Functions
- **s20acf** – S20ACF link from page S – Approximations of Special Functions
- **s20adf** – S20ADF link from page S – Approximations of Special Functions
- **s21baf** – S21BAF link from page S – Approximations of Special Functions
- **s21bbf** – S21BBF link from page S – Approximations of Special Functions
- **s21bcf** – S21BCF link from page S – Approximations of Special Functions
- **s21bdf** – S21BDF link from page S – Approximations of Special Functions

Axiom Browser



⇐ "Root Page" (RootPage) 3.1 on page 117
⇒ "Commands" (ugSysCmdPage) 19 on page 2538
⇒ "Constructors" (LispFunctions) 3.71 on page 952

— **man0.ht** —

```
\begin{page}{Man0Page}{A x i o m \   B r o w s e r}

\beginscroll
```

```
Enter search string (use {\em *} for wild card unless counter-indicated):
\inputstring{pattern}{40}{}
\newline
\beginmenu
\menulispmemolink{Constructors}
  { (|kSearch| '|\stringvalue{pattern}|) }
  \tab{15} Search for
\lispmemolink{categories}{(|cSearch| '|\stringvalue{pattern}|)},
\lispmemolink{domains}{(|dSearch| '|\stringvalue{pattern}|)}, or
\lispmemolink{packages}{(|pSearch| '|\stringvalue{pattern}|)}
\menulispmemolink{Operations}
  { (|oSearch| '|\stringvalue{pattern}|) }
  \tab{15} Search for operations.
\menulispmemolink{Attributes}
  { (|aSearch| '|\stringvalue{pattern}|) }
  \tab{15} Search for attributes.
\menulispmemolink{General}
  { (|aokSearch| '|\stringvalue{pattern}|) }
  \tab{15} Search for all three of the above.
\menulispmemolink{Documentation}
  { (|docSearch| '|\stringvalue{pattern}|) }
  \tab{15} Search library documentation.
\menulispmemolink{Complete}
  { (|genSearch| '|\stringvalue{pattern}|) }
  \tab{15} All of the above.
\menulispmemolink{Selectable}
  { (|detailedSearch| '|\stringvalue{pattern}|) }
  \tab{15} Detailed search with selectable options.
\horizontalline
\menuunixlink{Reference}
  {htsearch "\stringvalue{pattern}"}
  \tab{15} Search Reference documentation ({\em *} wild card
is not accepted).
\menumemolink{Commands}{ugSysCmdPage}
\tab{15} View system command documentation.
\endmenu
\endscroll
\autobutt{BROWSEhelp}
\end{page}
```

———-

The Hyperdoc Browse Facility

— man0.ht —

```
\begin{page}{BROWSEhelp}{The Hyperdoc Browse Facility}
```

```
\beginscroll
\beginmenu
    \menudownlink{{The Front Page: Searching the Library}}
{ugBrowseStartPage}
    \menudownlink{{The Constructor Page}}{ugBrowseDomainPage}
    \menudownlink{{Miscellaneous Features of Browse}}
{ugBrowseMiscellaneousFeaturesPage}
\endmenu
\endscroll
\newline
\end{page}
```

——–

3.72 mapping.ht

Domain Mapping(T,S,...)

⇒ "Description" (LispFunctions) 3.71 on page 952
⇒ "Operations" (LispFunctions) 3.71 on page 952

— **mapping.ht** —

```
\begin{page}{DomainMapping}{Domain {\em Mapping(T,S,...)}}
\beginscroll
{\em Mapping} takes any number of arguments of the form:
\indentrel{2}
\newline \spad{T}, a domain of category \spadtype{SetCategory}
\newline \spad{S}, a domain of category \spadtype{SetCategory}
\newline\tab{10}...
\indentrel{-2}\newline
This constructor is a primitive in Axiom.
\newline
\beginmenu
\item\menulispdownlink{Description}
{(|dbSpecialDescription| '|Mapping|)}\tab{19}General description
\item\menulispdownlink{Operations}
{(|dbSpecialOperations| '|Mapping|)}
\tab{19}All exported operations of \spad{Mapping(T,S)}
%\item\menudownlink{Examples}   {MappingExamples}
\tab{19}Examples illustrating use
%\item\menudownlink{Exports}    {MappingExports}
\tab{19}Explicit categories and operations
\endmenu
\endscroll\end{page}
```

———-

Domain Constructor Mapping

— **mapping.ht** —

```
\begin{page}{MappingDescription}{Domain Constructor {\em Mapping}}
\beginscroll
\newline\menuitemstyle{}\tab{2}Mapping({\em T},{\em S,...})
\newline\tab{2}{\em Arguments:}\indent{17}\tab{-2}
{\em T}, a domain of category \spadtype{SetCategory}
\newline\tab{-2}
{\em S}, a domain of category \spadtype{SetCategory}
\newline\tab{10}...
\indent{0}\newline\tab{2}{\em Returns:}\indent{15}\tab{0}
the class of mappings from domain ({\em S,...})
into domain {\em T} as described below.
\indent{0}\newline\tab{2}{\em Description:}\indent{15}\tab{0}
{\em Mapping(T,S,...)} denotes the class of objects which are
mappings from a source domain ({\em S,...}) into a target domain {\em T}.
The {\em Mapping} constructor can take any number of arguments.
All but the first argument is regarded as part of a source tuple
for the mapping.
For example, {\em Mapping(T,A,B)} denotes the
class of mappings from {\em (A,B)} into {\em T}.
{\em Mapping} is a primitive domain of Axiom which cannot be
defined in the Axiom language.
\endscroll
\end{page}
```

———-

3.73 mappkg1.ht

MappingPackage1

— **mappkg1.ht** —

```
\begin{page}{MappingPackageOneXmpPage}{MappingPackage1}
\beginscroll

Function are objects of type \pspadtype{Mapping}.
In this section we demonstrate some library operations from the
```

```
packages \spadtype{MappingPackage1}, \spadtype{MappingPackage2}, and
\spadtype{MappingPackage3} that manipulate and create functions.
Some terminology: a {\it nullary} function takes no arguments,
a {\it unary} function takes one argument, and
a {\it binary} function takes two arguments.

\xtc{
We begin by creating an example function that raises a
rational number to an integer exponent.
}{
\spadpaste{power(q: FRAC INT, n: INT): FRAC INT == q**n \bound{power}}
}
\xtc{
}{
\spadpaste{power(2,3) \free{power}}
}
\xtc{
The \spadfunFrom{twist}{MappingPackage3} operation
transposes the arguments of a binary function.
Here \spad{rewop(a, b)} is \spad{power(b, a)}.
}{
\spadpaste{rewop := twist power \free{power}\bound{rewop}}
}
\xtc{
This is \texht{$2^3.$}{\spad{2**3.}}
}{
\spadpaste{rewop(3, 2) \free{rewop}}
}
\xtc{
Now we define \userfun{square} in terms of \userfun{power}.
}{
\spadpaste{square: FRAC INT -> FRAC INT \bound{squaredec}}
}
\xtc{
The \spadfunFrom{curryRight}{MappingPackage3} operation creates a unary
function from a binary one by providing a constant
argument on the right.
}{
\spadpaste{square:= curryRight(power, 2) \free{squaredec poswer}
\bound{square}}
}
\xtc{
Likewise, the
\spadfunFrom{curryLeft}{MappingPackage3} operation provides a constant
argument on the left.
}{
\spadpaste{square 4 \free{square}}
}
\xtc{
The \spadfunFrom{constantRight}{MappingPackage3} operation creates
```

```
(in a trivial way) a binary function from a unary one:
\spad{constantRight(f)} is the function \spad{g} such that
\spad{g(a,b)= f(a).}
}{
\spadpaste{squirrel:= constantRight(square)\$MAPPKG3(FRAC INT,
FRAC INT,FRAC INT) \free{square}\bound{squirrel}}
}
\xtc{
Likewise,
\spad{constantLeft(f)} is the function \spad{g} such that
\spad{g(a,b)= f(b).}
}{
\spadpaste{squirrel(1/2, 1/3) \free{squirrel}}
}
\xtc{
The \spadfunFrom{curry}{MappingPackage2} operation makes a
unary function nullary.
}{
\spadpaste{sixteen := curry(square, 4/1) \free{square}\bound{sixteen}}
}
\xtc{
}{
\spadpaste{sixteen() \free{sixteen}}
}
\xtc{
The \spadopFrom{*}{MappingPackage3} operation
constructs composed functions.
}{
\spadpaste{square2:=square*square \free{square}\bound{square2}}
}
\xtc{
}{
\spadpaste{square2  3 \free{square2}}
}
\xtc{
Use the \spadopFrom{**}{MappingPackage1} operation to create
functions that are \spad{n}-fold iterations of other functions.
}{
\spadpaste{sc(x: FRAC INT): FRAC INT == x + 1 \bound{sc}}
}
\xtc{
This is a list of \pspadtype{Mapping} objects.
}{
\spadpaste{incfns := [sc**i for i in 0..10] \free{sc}\bound{incfns}}
}
\xtc{
This is a list of applications of those functions.
}{
\spadpaste{[f 4 for f in incfns] \free{incfns}}
}
```

```
\xtc{
Use the \spadfunFrom{recur}{MappingPackage1}
operation for recursion:
\spad{g := recur f} means \spad{g(n,x) == f(n,f(n-1,...f(1,x))).}
}{
\spadpaste{times(n:NNI, i:INT):INT == n*i \bound{rdec}}
}
\xtc{
}{
\spadpaste{r := recur(times) \free{rdec}\bound{r}}
}
\xtc{
This is a factorial function.
}{
\spadpaste{fact := curryRight(r, 1) \free{r}\bound{fact}}
}
\xtc{
}{
\spadpaste{fact 4 \free{fact}}
}
\xtc{
Constructed functions can be used within other functions.
}{
\begin{spadsrc}[\free{square}\bound{mto2ton}]
mto2ton(m, n) ==
  raiser := square**n
  raiser m
\end{spadsrc}
}
\xtc{
This is \texht{$3^{2^3}.$}{\spad{3**(2**3).}}
}{
\spadpaste{mto2ton(3, 3) \free{mto2ton}}
}
\xtc{
Here \userfun{shiftfib} is a unary function that modifies its argument.
}{
\begin{spadsrc}[\bound{shiftfib}]
shiftfib(r: List INT) : INT ==
  t := r.1
  r.1 := r.2
  r.2 := r.2 + t
  t
\end{spadsrc}
}
\xtc{
By currying over the argument we get a function with private state.
}{
\spadpaste{fibinit: List INT := [0, 1] \bound{fibinitdec}}
}
```

```
\xtc{
}{
\spadpaste{fibs := curry(shiftfib, fibinit)
\free{shiftfib fibinit}\bound{fibs}}
}
\xtc{
}{
\spadpaste{[fibs() for i in 0..30] \free{fibs}}
}
\endscroll
\autobuttons
\end{page}

\begin{patch}{MappingPackageOneXmpPagePatch1}
\begin{paste}{MappingPackageOneXmpPageFull1}{MappingPackageOneXmpPageEmpty1}
\pastebutton{MappingPackageOneXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{power(q: FRAC INT, n: INT): FRAC INT == q**n\bound{power }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPageEmpty1}
\begin{paste}{MappingPackageOneXmpPageEmpty1}{MappingPackageOneXmpPagePatch1}
\pastebutton{MappingPackageOneXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{power(q: FRAC INT, n: INT): FRAC INT == q**n\bound{power }}
\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPagePatch2}
\begin{paste}{MappingPackageOneXmpPageFull2}{MappingPackageOneXmpPageEmpty2}
\pastebutton{MappingPackageOneXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{power(2,3)\free{power }}
\indentrel{3}\begin{verbatim}
   (2)  8
                                 Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPageEmpty2}
\begin{paste}{MappingPackageOneXmpPageEmpty2}{MappingPackageOneXmpPagePatch2}
\pastebutton{MappingPackageOneXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{power(2,3)\free{power }}
\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPagePatch3}
\begin{paste}{MappingPackageOneXmpPageFull3}{MappingPackageOneXmpPageEmpty3}
\pastebutton{MappingPackageOneXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{rewop := twist power\free{power }\bound{rewop }}
\indentrel{3}\begin{verbatim}
   (3)  theMap(MAPPKG3;twist;MM;5!0)
```

```
 Type: ((Integer,Fraction Integer) -> Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPageEmpty3}
\begin{paste}{MappingPackageOneXmpPageEmpty3}{MappingPackageOneXmpPagePatch3}
\pastebutton{MappingPackageOneXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{rewop := twist power\free{power }\bound{rewop }}
\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPagePatch4}
\begin{paste}{MappingPackageOneXmpPageFull4}{MappingPackageOneXmpPageEmpty4}
\pastebutton{MappingPackageOneXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{rewop(3, 2)\free{rewop }}
\indentrel{3}\begin{verbatim}
   (4)  8
                                 Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPageEmpty4}
\begin{paste}{MappingPackageOneXmpPageEmpty4}{MappingPackageOneXmpPagePatch4}
\pastebutton{MappingPackageOneXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{rewop(3, 2)\free{rewop }}
\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPagePatch5}
\begin{paste}{MappingPackageOneXmpPageFull5}{MappingPackageOneXmpPageEmpty5}
\pastebutton{MappingPackageOneXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{square: FRAC INT -> FRAC INT\bound{squaredec }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPageEmpty5}
\begin{paste}{MappingPackageOneXmpPageEmpty5}{MappingPackageOneXmpPagePatch5}
\pastebutton{MappingPackageOneXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{square: FRAC INT -> FRAC INT\bound{squaredec }}
\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPagePatch6}
\begin{paste}{MappingPackageOneXmpPageFull6}{MappingPackageOneXmpPageEmpty6}
\pastebutton{MappingPackageOneXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{square:= curryRight(power, 2)\free{squaredec poswer }\bound{square }}
\indentrel{3}\begin{verbatim}
   (6)  theMap(MAPPKG3;curryRight;MBM;1!0,430)
          Type: (Fraction Integer -> Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{MappingPackageOneXmpPageEmpty6}
\begin{paste}{MappingPackageOneXmpPageEmpty6}{MappingPackageOneXmpPagePatch6}
\pastebutton{MappingPackageOneXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{square:= curryRight(power, 2)\free{squaredec poswer }\bound{square }}
\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPagePatch7}
\begin{paste}{MappingPackageOneXmpPageFull7}{MappingPackageOneXmpPageEmpty7}
\pastebutton{MappingPackageOneXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{square 4\free{square }}
\indentrel{3}\begin{verbatim}
   (7)  16
                                 Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPageEmpty7}
\begin{paste}{MappingPackageOneXmpPageEmpty7}{MappingPackageOneXmpPagePatch7}
\pastebutton{MappingPackageOneXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{square 4\free{square }}
\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPagePatch8}
\begin{paste}{MappingPackageOneXmpPageFull8}{MappingPackageOneXmpPageEmpty8}
\pastebutton{MappingPackageOneXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{squirrel:= constantRight(square)$MAPPKG3(FRAC INT,FRAC INT,FRAC INT)\free{square }\bound
\indentrel{3}\begin{verbatim}
   (8)  theMap(MAPPKG3;constantRight;MM;3!0)
Type: ((Fraction Integer,Fraction Integer) -> Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPageEmpty8}
\begin{paste}{MappingPackageOneXmpPageEmpty8}{MappingPackageOneXmpPagePatch8}
\pastebutton{MappingPackageOneXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{squirrel:= constantRight(square)$MAPPKG3(FRAC INT,FRAC INT,FRAC INT)\free{square }\bound
\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPagePatch9}
\begin{paste}{MappingPackageOneXmpPageFull9}{MappingPackageOneXmpPageEmpty9}
\pastebutton{MappingPackageOneXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{squirrel(1/2, 1/3)\free{squirrel }}
\indentrel{3}\begin{verbatim}
        1
   (9)
        4
                                 Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{MappingPackageOneXmpPageEmpty9}
\begin{paste}{MappingPackageOneXmpPageEmpty9}{MappingPackageOneXmpPagePatch9}
\pastebutton{MappingPackageOneXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{squirrel(1/2, 1/3)\free{squirrel }}
\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPagePatch10}
\begin{paste}{MappingPackageOneXmpPageFull10}{MappingPackageOneXmpPageEmpty10}
\pastebutton{MappingPackageOneXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{sixteen := curry(square, 4/1)\free{square }\bound{sixteen }}
\indentrel{3}\begin{verbatim}
   (10)  theMap(MAPPKG2;curry;MAM;2!0,488)
                           Type: (() -> Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPageEmpty10}
\begin{paste}{MappingPackageOneXmpPageEmpty10}{MappingPackageOneXmpPagePatch10}
\pastebutton{MappingPackageOneXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{sixteen := curry(square, 4/1)\free{square }\bound{sixteen }}
\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPagePatch11}
\begin{paste}{MappingPackageOneXmpPageFull11}{MappingPackageOneXmpPageEmpty11}
\pastebutton{MappingPackageOneXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{sixteen()\free{sixteen }}
\indentrel{3}\begin{verbatim}
   (11)  16
                                   Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPageEmpty11}
\begin{paste}{MappingPackageOneXmpPageEmpty11}{MappingPackageOneXmpPagePatch11}
\pastebutton{MappingPackageOneXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{sixteen()\free{sixteen }}
\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPagePatch12}
\begin{paste}{MappingPackageOneXmpPageFull12}{MappingPackageOneXmpPageEmpty12}
\pastebutton{MappingPackageOneXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{square2:=square*square\free{square }\bound{square2 }}
\indentrel{3}\begin{verbatim}
   (12)  theMap(MAPPKG3;*;MMM;6!0,589)
            Type: (Fraction Integer -> Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPageEmpty12}
```

```
\begin{paste}{MappingPackageOneXmpPageEmpty12}{MappingPackageOneXmpPagePatch12}
\pastebutton{MappingPackageOneXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{square2:=square*square\free{square }\bound{square2 }}
\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPagePatch13}
\begin{paste}{MappingPackageOneXmpPageFull13}{MappingPackageOneXmpPageEmpty13}
\pastebutton{MappingPackageOneXmpPageFull13}{\hidepaste}
\tab{5}\spadcommand{square2 3\free{square2 }}
\indentrel{3}\begin{verbatim}
   (13)  81
                                 Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPageEmpty13}
\begin{paste}{MappingPackageOneXmpPageEmpty13}{MappingPackageOneXmpPagePatch13}
\pastebutton{MappingPackageOneXmpPageEmpty13}{\showpaste}
\tab{5}\spadcommand{square2 3\free{square2 }}
\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPagePatch14}
\begin{paste}{MappingPackageOneXmpPageFull14}{MappingPackageOneXmpPageEmpty14}
\pastebutton{MappingPackageOneXmpPageFull14}{\hidepaste}
\tab{5}\spadcommand{sc(x: FRAC INT): FRAC INT == x + 1\bound{sc }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPageEmpty14}
\begin{paste}{MappingPackageOneXmpPageEmpty14}{MappingPackageOneXmpPagePatch14}
\pastebutton{MappingPackageOneXmpPageEmpty14}{\showpaste}
\tab{5}\spadcommand{sc(x: FRAC INT): FRAC INT == x + 1\bound{sc }}
\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPagePatch15}
\begin{paste}{MappingPackageOneXmpPageFull15}{MappingPackageOneXmpPageEmpty15}
\pastebutton{MappingPackageOneXmpPageFull15}{\hidepaste}
\tab{5}\spadcommand{incfns := [sc**i for i in 0..10]\free{sc }\bound{incfns }}
\indentrel{3}\begin{verbatim}
   (15)
   [theMap(MAPPKG1;**;MNniM;6!0,314),
    theMap(MAPPKG1;**;MNniM;6!0,963),
    theMap(MAPPKG1;**;MNniM;6!0,810),
    theMap(MAPPKG1;**;MNniM;6!0,546),
    theMap(MAPPKG1;**;MNniM;6!0,338),
    theMap(MAPPKG1;**;MNniM;6!0,989),
    theMap(MAPPKG1;**;MNniM;6!0,218),
    theMap(MAPPKG1;**;MNniM;6!0,20),
```

```
    theMap(MAPPKG1;**;MNniM;6!0,533),
    theMap(MAPPKG1;**;MNniM;6!0,437),
    theMap(MAPPKG1;**;MNniM;6!0,900)]
      Type: List (Fraction Integer -> Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPageEmpty15}
\begin{paste}{MappingPackageOneXmpPageEmpty15}{MappingPackageOneXmpPagePatch15}
\pastebutton{MappingPackageOneXmpPageEmpty15}{\showpaste}
\tab{5}\spadcommand{incfns := [sc**i for i in 0..10]\free{sc }\bound{incfns }}
\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPagePatch16}
\begin{paste}{MappingPackageOneXmpPageFull16}{MappingPackageOneXmpPageEmpty16}
\pastebutton{MappingPackageOneXmpPageFull16}{\hidepaste}
\tab{5}\spadcommand{[f 4 for f in incfns]\free{incfns }}
\indentrel{3}\begin{verbatim}
   (16)  [4,5,6,7,8,9,10,11,12,13,14]
                              Type: List Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPageEmpty16}
\begin{paste}{MappingPackageOneXmpPageEmpty16}{MappingPackageOneXmpPagePatch16}
\pastebutton{MappingPackageOneXmpPageEmpty16}{\showpaste}
\tab{5}\spadcommand{[f 4 for f in incfns]\free{incfns }}
\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPagePatch17}
\begin{paste}{MappingPackageOneXmpPageFull17}{MappingPackageOneXmpPageEmpty17}
\pastebutton{MappingPackageOneXmpPageFull17}{\hidepaste}
\tab{5}\spadcommand{times(n:NNI, i:INT):INT == n*i\bound{rdec }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPageEmpty17}
\begin{paste}{MappingPackageOneXmpPageEmpty17}{MappingPackageOneXmpPagePatch17}
\pastebutton{MappingPackageOneXmpPageEmpty17}{\showpaste}
\tab{5}\spadcommand{times(n:NNI, i:INT):INT == n*i\bound{rdec }}
\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPagePatch18}
\begin{paste}{MappingPackageOneXmpPageFull18}{MappingPackageOneXmpPageEmpty18}
\pastebutton{MappingPackageOneXmpPageFull18}{\hidepaste}
\tab{5}\spadcommand{r := recur(times)\free{rdec }\bound{r }}
\indentrel{3}\begin{verbatim}
   (18)  theMap(MAPPKG1;recur;2M;7!0,161)
```

```
        Type: ((NonNegativeInteger,Integer) -> Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPageEmpty18}
\begin{paste}{MappingPackageOneXmpPageEmpty18}{MappingPackageOneXmpPagePatch18}
\pastebutton{MappingPackageOneXmpPageEmpty18}{\showpaste}
\tab{5}\spadcommand{r := recur(times)\free{rdec }\bound{r }}
\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPagePatch19}
\begin{paste}{MappingPackageOneXmpPageFull19}{MappingPackageOneXmpPageEmpty19}
\pastebutton{MappingPackageOneXmpPageFull19}{\hidepaste}
\tab{5}\spadcommand{fact := curryRight(r, 1)\free{r }\bound{fact }}
\indentrel{3}\begin{verbatim}
   (19)  theMap(MAPPKG3;curryRight;MBM;1!0,541)
                 Type: (NonNegativeInteger -> Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPageEmpty19}
\begin{paste}{MappingPackageOneXmpPageEmpty19}{MappingPackageOneXmpPagePatch19}
\pastebutton{MappingPackageOneXmpPageEmpty19}{\showpaste}
\tab{5}\spadcommand{fact := curryRight(r, 1)\free{r }\bound{fact }}
\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPagePatch20}
\begin{paste}{MappingPackageOneXmpPageFull20}{MappingPackageOneXmpPageEmpty20}
\pastebutton{MappingPackageOneXmpPageFull20}{\hidepaste}
\tab{5}\spadcommand{fact 4\free{fact }}
\indentrel{3}\begin{verbatim}
   (20)  24
                                 Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPageEmpty20}
\begin{paste}{MappingPackageOneXmpPageEmpty20}{MappingPackageOneXmpPagePatch20}
\pastebutton{MappingPackageOneXmpPageEmpty20}{\showpaste}
\tab{5}\spadcommand{fact 4\free{fact }}
\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPagePatch21}
\begin{paste}{MappingPackageOneXmpPageFull21}{MappingPackageOneXmpPageEmpty21}
\pastebutton{MappingPackageOneXmpPageFull21}{\hidepaste}
\tab{5}\spadcommand{mto2ton(m, n) ==
  raiser := square**n
  raiser m
\free{square }\bound{mto2ton }}
\indentrel{3}\begin{verbatim}
```

```
                                              Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPageEmpty21}
\begin{paste}{MappingPackageOneXmpPageEmpty21}{MappingPackageOneXmpPagePatch21}
\pastebutton{MappingPackageOneXmpPageEmpty21}{\showpaste}
\tab{5}\spadcommand{mto2ton(m, n) ==
  raiser := square**n
  raiser m
\free{square }\bound{mto2ton }}
\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPagePatch22}
\begin{paste}{MappingPackageOneXmpPageFull22}{MappingPackageOneXmpPageEmpty22}
\pastebutton{MappingPackageOneXmpPageFull22}{\hidepaste}
\tab{5}\spadcommand{mto2ton(3, 3)\free{mto2ton }}
\indentrel{3}\begin{verbatim}
   (22)  6561
                                  Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPageEmpty22}
\begin{paste}{MappingPackageOneXmpPageEmpty22}{MappingPackageOneXmpPagePatch22}
\pastebutton{MappingPackageOneXmpPageEmpty22}{\showpaste}
\tab{5}\spadcommand{mto2ton(3, 3)\free{mto2ton }}
\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPagePatch23}
\begin{paste}{MappingPackageOneXmpPageFull23}{MappingPackageOneXmpPageEmpty23}
\pastebutton{MappingPackageOneXmpPageFull23}{\hidepaste}
\tab{5}\spadcommand{shiftfib(r: List INT) : INT ==
  t := r.1
  r.1 := r.2
  r.2 := r.2 + t
  t
\bound{shiftfib }}
\indentrel{3}\begin{verbatim}
                                              Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPageEmpty23}
\begin{paste}{MappingPackageOneXmpPageEmpty23}{MappingPackageOneXmpPagePatch23}
\pastebutton{MappingPackageOneXmpPageEmpty23}{\showpaste}
\tab{5}\spadcommand{shiftfib(r: List INT) : INT ==
  t := r.1
  r.1 := r.2
  r.2 := r.2 + t
```

```
  t
\bound{shiftfib }}
\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPagePatch24}
\begin{paste}{MappingPackageOneXmpPageFull24}{MappingPackageOneXmpPageEmpty24}
\pastebutton{MappingPackageOneXmpPageFull24}{\hidepaste}
\tab{5}\spadcommand{fibinit: List INT := [0, 1]\bound{fibinitdec }}
\indentrel{3}\begin{verbatim}
   (24)  [0,1]
                                    Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPageEmpty24}
\begin{paste}{MappingPackageOneXmpPageEmpty24}{MappingPackageOneXmpPagePatch24}
\pastebutton{MappingPackageOneXmpPageEmpty24}{\showpaste}
\tab{5}\spadcommand{fibinit: List INT := [0, 1]\bound{fibinitdec }}
\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPagePatch25}
\begin{paste}{MappingPackageOneXmpPageFull25}{MappingPackageOneXmpPageEmpty25}
\pastebutton{MappingPackageOneXmpPageFull25}{\hidepaste}
\tab{5}\spadcommand{fibs := curry(shiftfib, fibinit)\free{shiftfib fibinit }\bound{fibs }}
\indentrel{3}\begin{verbatim}
   (25)  theMap(MAPPKG2;curry;MAM;2!0,91)
                                 Type: (() -> Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPageEmpty25}
\begin{paste}{MappingPackageOneXmpPageEmpty25}{MappingPackageOneXmpPagePatch25}
\pastebutton{MappingPackageOneXmpPageEmpty25}{\showpaste}
\tab{5}\spadcommand{fibs := curry(shiftfib, fibinit)\free{shiftfib fibinit }\bound{fibs }}
\end{paste}\end{patch}

\begin{patch}{MappingPackageOneXmpPagePatch26}
\begin{paste}{MappingPackageOneXmpPageFull26}{MappingPackageOneXmpPageEmpty26}
\pastebutton{MappingPackageOneXmpPageFull26}{\hidepaste}
\tab{5}\spadcommand{[fibs() for i in 0..30]\free{fibs }}
\indentrel{3}\begin{verbatim}
   (26)
   [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233,
    377, 610, 987, 1597, 2584, 4181, 6765, 10946, 17711,
    28657, 46368, 75025, 121393, 196418, 317811, 514229,
    832040]
                                    Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{MappingPackageOneXmpPageEmpty26}
\begin{paste}{MappingPackageOneXmpPageEmpty26}{MappingPackageOneXmpPagePatch26}
\pastebutton{MappingPackageOneXmpPageEmpty26}{\showpaste}
\tab{5}\spadcommand{[fibs() for i in 0..30]\free{fibs }}
\end{paste}\end{patch}
```

———-

3.74 mset.ht

MultiSet

— mset.ht —

```
\begin{page}{MultiSetXmpPage}{MultiSet}
\beginscroll
The domain \spadtype{Multiset(R)} is similar to \spadtype{Set(R)}
except that multiplicities
(counts of duplications) are maintained and displayed.
Use the operation \spadfunFrom{multiset}{Multiset} to create
multisets from lists.
All the standard operations from sets are available for
multisets.
An element with multiplicity greater than one has the
multiplicity displayed first, then a colon, and then the element.

\xtc{
Create a multiset of integers.
}{
\spadpaste{s := multiset [1,2,3,4,5,4,3,2,3,4,5,6,7,4,10]\bound{s}}
}
\xtc{
The operation \spadfunX{insert} adds an element to a multiset.
}{
\spadpaste{insert!(3,s)\bound{s1}\free{s}}
}
\xtc{
Use \spadfunX{remove} to remove an element.
If a third argument is present, it specifies how many instances
to remove. Otherwise all instances of the element are removed.
Display the resulting multiset.
}{
\spadpaste{remove!(3,s,1); s\bound{s2}\free{s1}}
}
\xtc{
}{
```

```
\spadpaste{remove!(5,s); s\bound{s2}\free{s1}}
}
\xtc{
The operation \spadfun{count} returns the number of copies
of a given value.
}{
\spadpaste{count(5,s)\free{s2}}
}
\xtc{
A second multiset.
}{
\spadpaste{t := multiset [2,2,2,-9]\bound{t}}
}
\xtc{
The \spadfun{union} of two multisets is additive.
}{
\spadpaste{U := union(s,t)\bound{U}}
}
\xtc{
The \spadfun{intersect} operation gives the elements that are in
common, with additive multiplicity.
}{
\spadpaste{I := intersect(s,t)\bound{I}}
}
\xtc{
The \spadfun{difference} of \spad{s} and \spad{t} consists of
the elements that \spad{s} has but \spad{t} does not.
Elements are regarded as indistinguishable, so that if \spad{s}
and \spad{t} have any element in common, the \spadfun{difference}
does not contain that element.
}{
\spadpaste{difference(s,t)\free{s2 t}}
}
\xtc{
The \spadfun{symmetricDifference} is the \spadfun{union}
of \spad{difference(s, t)} and \spad{difference(t, s)}.
}{
\spadpaste{S := symmetricDifference(s,t)\bound{S}\free{s2 t}}
}
\xtc{
Check that the \spadfun{union} of the \spadfun{symmetricDifference} and
the \spadfun{intersect} equals the \spadfun{union} of the elements.
}{
\spadpaste{(U = union(S,I))@Boolean\free{S I U}}
}
\xtc{
Check some inclusion relations.
}{
\spadpaste{t1 := multiset [1,2,2,3]; [t1 < t, t1 < s, t < s, t1 <= s]
\free{t s2}}
```

```
}
\endscroll
\autobuttons
\end{page}

\begin{patch}{MultiSetXmpPagePatch1}
\begin{paste}{MultiSetXmpPageFull1}{MultiSetXmpPageEmpty1}
\pastebutton{MultiSetXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{s := multiset [1,2,3,4,5,4,3,2,3,4,5,6,7,4,10]\bound{s }}
\indentrel{3}\begin{verbatim}
   (1)  {7,2: 5,3: 3,1,10,6,4: 4,2: 2}
                          Type: Multiset PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MultiSetXmpPageEmpty1}
\begin{paste}{MultiSetXmpPageEmpty1}{MultiSetXmpPagePatch1}
\pastebutton{MultiSetXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{s := multiset [1,2,3,4,5,4,3,2,3,4,5,6,7,4,10]\bound{s }}
\end{paste}\end{patch}

\begin{patch}{MultiSetXmpPagePatch2}
\begin{paste}{MultiSetXmpPageFull2}{MultiSetXmpPageEmpty2}
\pastebutton{MultiSetXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{insert!(3,s)\bound{s1 }\free{s }}
\indentrel{3}\begin{verbatim}
   (2)  {7,2: 5,4: 3,1,10,6,4: 4,2: 2}
                          Type: Multiset PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MultiSetXmpPageEmpty2}
\begin{paste}{MultiSetXmpPageEmpty2}{MultiSetXmpPagePatch2}
\pastebutton{MultiSetXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{insert!(3,s)\bound{s1 }\free{s }}
\end{paste}\end{patch}

\begin{patch}{MultiSetXmpPagePatch3}
\begin{paste}{MultiSetXmpPageFull3}{MultiSetXmpPageEmpty3}
\pastebutton{MultiSetXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{remove!(3,s,1); s\bound{s2 }\free{s1 }}
\indentrel{3}\begin{verbatim}
   (3)  {7,2: 5,3: 3,1,10,6,4: 4,2: 2}
                          Type: Multiset PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MultiSetXmpPageEmpty3}
\begin{paste}{MultiSetXmpPageEmpty3}{MultiSetXmpPagePatch3}
\pastebutton{MultiSetXmpPageEmpty3}{\showpaste}
```

```
\tab{5}\spadcommand{remove!(3,s,1); s\bound{s2 }\free{s1 }}
\end{paste}\end{patch}

\begin{patch}{MultiSetXmpPagePatch4}
\begin{paste}{MultiSetXmpPageFull4}{MultiSetXmpPageEmpty4}
\pastebutton{MultiSetXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{remove!(5,s); s\bound{s2 }\free{s1 }}
\indentrel{3}\begin{verbatim}
   (4)  {7,3: 3,1,10,6,4: 4,2: 2}
                         Type: Multiset PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MultiSetXmpPageEmpty4}
\begin{paste}{MultiSetXmpPageEmpty4}{MultiSetXmpPagePatch4}
\pastebutton{MultiSetXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{remove!(5,s); s\bound{s2 }\free{s1 }}
\end{paste}\end{patch}

\begin{patch}{MultiSetXmpPagePatch5}
\begin{paste}{MultiSetXmpPageFull5}{MultiSetXmpPageEmpty5}
\pastebutton{MultiSetXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{count(5,s)\free{s2 }}
\indentrel{3}\begin{verbatim}
   (5)  0
                               Type: NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MultiSetXmpPageEmpty5}
\begin{paste}{MultiSetXmpPageEmpty5}{MultiSetXmpPagePatch5}
\pastebutton{MultiSetXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{count(5,s)\free{s2 }}
\end{paste}\end{patch}

\begin{patch}{MultiSetXmpPagePatch6}
\begin{paste}{MultiSetXmpPageFull6}{MultiSetXmpPageEmpty6}
\pastebutton{MultiSetXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{t := multiset [2,2,2,-9]\bound{t }}
\indentrel{3}\begin{verbatim}
   (6)  {- 9,3: 2}
                                 Type: Multiset Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MultiSetXmpPageEmpty6}
\begin{paste}{MultiSetXmpPageEmpty6}{MultiSetXmpPagePatch6}
\pastebutton{MultiSetXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{t := multiset [2,2,2,-9]\bound{t }}
\end{paste}\end{patch}
```

```
\begin{patch}{MultiSetXmpPagePatch7}
\begin{paste}{MultiSetXmpPageFull7}{MultiSetXmpPageEmpty7}
\pastebutton{MultiSetXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{U := union(s,t)\bound{U }}
\indentrel{3}\begin{verbatim}
   (7)  {7,3: 3,1,- 9,10,6,4: 4,5: 2}
                                 Type: Multiset Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MultiSetXmpPageEmpty7}
\begin{paste}{MultiSetXmpPageEmpty7}{MultiSetXmpPagePatch7}
\pastebutton{MultiSetXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{U := union(s,t)\bound{U }}
\end{paste}\end{patch}

\begin{patch}{MultiSetXmpPagePatch8}
\begin{paste}{MultiSetXmpPageFull8}{MultiSetXmpPageEmpty8}
\pastebutton{MultiSetXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{I := intersect(s,t)\bound{I }}
\indentrel{3}\begin{verbatim}
   (8)  {5: 2}
                                 Type: Multiset Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MultiSetXmpPageEmpty8}
\begin{paste}{MultiSetXmpPageEmpty8}{MultiSetXmpPagePatch8}
\pastebutton{MultiSetXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{I := intersect(s,t)\bound{I }}
\end{paste}\end{patch}

\begin{patch}{MultiSetXmpPagePatch9}
\begin{paste}{MultiSetXmpPageFull9}{MultiSetXmpPageEmpty9}
\pastebutton{MultiSetXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{difference(s,t)\free{s2 t }}
\indentrel{3}\begin{verbatim}
   (9)  {7,3: 3,1,10,6,4: 4}
                                 Type: Multiset Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MultiSetXmpPageEmpty9}
\begin{paste}{MultiSetXmpPageEmpty9}{MultiSetXmpPagePatch9}
\pastebutton{MultiSetXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{difference(s,t)\free{s2 t }}
\end{paste}\end{patch}

\begin{patch}{MultiSetXmpPagePatch10}
```

```
\begin{paste}{MultiSetXmpPageFull10}{MultiSetXmpPageEmpty10}
\pastebutton{MultiSetXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{S := symmetricDifference(s,t)\bound{S }\free{s2 t }}
\indentrel{3}\begin{verbatim}
   (10)  {7,3: 3,1,- 9,10,6,4: 4}
                                 Type: Multiset Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MultiSetXmpPageEmpty10}
\begin{paste}{MultiSetXmpPageEmpty10}{MultiSetXmpPagePatch10}
\pastebutton{MultiSetXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{S := symmetricDifference(s,t)\bound{S }\free{s2 t }}
\end{paste}\end{patch}

\begin{patch}{MultiSetXmpPagePatch11}
\begin{paste}{MultiSetXmpPageFull11}{MultiSetXmpPageEmpty11}
\pastebutton{MultiSetXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{(U = union(S,I))@Boolean\free{S I U }}
\indentrel{3}\begin{verbatim}
   (11)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MultiSetXmpPageEmpty11}
\begin{paste}{MultiSetXmpPageEmpty11}{MultiSetXmpPagePatch11}
\pastebutton{MultiSetXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{(U = union(S,I))@Boolean\free{S I U }}
\end{paste}\end{patch}

\begin{patch}{MultiSetXmpPagePatch12}
\begin{paste}{MultiSetXmpPageFull12}{MultiSetXmpPageEmpty12}
\pastebutton{MultiSetXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{t1 := multiset [1,2,2,3]; [t1 < t, t1 < s, t < s, t1 <= s]\free{t s2 }}
\indentrel{3}\begin{verbatim}
   (12)  [false,true,false,true]
                                     Type: List Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MultiSetXmpPageEmpty12}
\begin{paste}{MultiSetXmpPageEmpty12}{MultiSetXmpPagePatch12}
\pastebutton{MultiSetXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{t1 := multiset [1,2,2,3]; [t1 < t, t1 < s, t < s, t1 <= s]\free{t s2 }}
\end{paste}\end{patch}
```

3.75 matrix.ht

Matrix

⇒ "notitle" (ugxMatrixCreatePage) 3.75 on page 984
⇒ "notitle" (ugxMatrixOpsPage) 3.75 on page 997

— **matrix.ht** —

```
\begin{page}{MatrixXmpPage}{Matrix}
\beginscroll

The \spadtype{Matrix} domain provides arithmetic operations on matrices
and standard functions from linear algebra.
This domain is similar to the \spadtype{TwoDimensionalArray} domain, except
that the entries for \spadtype{Matrix} must belong to a \spadtype{Ring}.

\beginmenu
    \menudownlink{{9.52.1. Creating Matrices}}{ugxMatrixCreatePage}
    \menudownlink{{9.52.2. Operations on Matrices}}{ugxMatrixOpsPage}
\endmenu
\endscroll
\autobuttons
\end{page}
```

————

Creating Matrices

— **matrix.ht** —

```
\begin{page}{ugxMatrixCreatePage}{Creating Matrices}
\beginscroll

There are many ways to create a matrix from a collection of
values or from existing matrices.

\xtc{
If the matrix has almost all items equal to the same value,
use \spadfunFrom{new}{Matrix} to create a matrix filled with that
value and then reset the entries that are different.
}{
\spadpaste{m : Matrix(Integer) := new(3,3,0) \bound{m}}
}
\xtc{
To change the entry in the second row, third column to \spad{5}, use
```

```
\spadfunFrom{setelt}{Matrix}.
}{
\spadpaste{setelt(m,2,3,5) \free{m}\bound{m1}}
}
\xtc{
An alternative syntax is to use assignment.
}{
\spadpaste{m(1,2) := 10 \free{m1}\bound{m2}}
}
\xtc{
The matrix was {\it destructively modified}.
}{
\spadpaste{m \free{m2}}
}

\xtc{
If you already have the matrix entries as a list of lists,
use \spadfunFrom{matrix}{Matrix}.
}{
\spadpaste{matrix [[1,2,3,4],[0,9,8,7]]}
}

\xtc{
If the matrix is diagonal, use
\spadfunFrom{diagonalMatrix}{Matrix}.
}{
\spadpaste{dm := diagonalMatrix [1,x**2,x**3,x**4,x**5] \bound{dm}}
}
\xtc{
Use \spadfunFromX{setRow}{Matrix} and
\spadfunFromX{setColumn}{Matrix} to change a row or column of a matrix.
}{
\spadpaste{setRow!(dm,5,vector [1,1,1,1,1]) \free{dm}\bound{dm1}}
}
\xtc{
}{
\spadpaste{setColumn!(dm,2,vector [y,y,y,y,y]) \free{dm1}\bound{dm2}}
}

%
\xtc{
Use \spadfunFrom{copy}{Matrix} to make a copy of a matrix.
}{
\spadpaste{cdm := copy(dm) \free{dm2}\bound{cdm}}
}
\xtc{
This is useful if you intend to modify a matrix destructively but
want a copy of the original.
}{
\spadpaste{setelt(dm,4,1,1-x**7) \free{dm2}\bound{setdm}}
```

```
}
\xtc{
}{
\spadpaste{[dm,cdm] \free{setdm cdm}}
}

%
\xtc{
Use \spadfunFrom{subMatrix}{Matrix} to extract part of an
existing matrix.
The syntax is \spad{subMatrix({\it m, firstrow, lastrow, firstcol,
lastcol})}.
}{
\spadpaste{subMatrix(dm,2,3,2,4) \free{setdm}}
}

%
\xtc{
To change a submatrix, use \spadfunFromX{setsubMatrix}{Matrix}.
}{
\spadpaste{d := diagonalMatrix [1.2,-1.3,1.4,-1.5] \bound{d}}
}
\xtc{
If \spad{e} is too big to fit where you specify, an error message
is displayed.
Use \spadfunFrom{subMatrix}{Matrix} to extract part of \spad{e}, if
necessary.
}{
\spadpaste{e := matrix [[6.7,9.11],[-31.33,67.19]] \bound{e}}
}
\xtc{
This changes the submatrix of \spad{d} whose upper left corner is
at the first row and second column and whose size is that of \spad{e}.
}{
\spadpaste{setsubMatrix!(d,1,2,e) \free{d e}\bound{d1}}
}
\xtc{
}{
\spadpaste{d \free{d1}}
}
%

%
\xtc{
Matrices can be joined either horizontally or vertically to make
new matrices.
}{
\spadpaste{a := matrix [[1/2,1/3,1/4],[1/5,1/6,1/7]] \bound{a}}
}
\xtc{
```

```
}{
\spadpaste{b := matrix [[3/5,3/7,3/11],[3/13,3/17,3/19]] \bound{b}}
}
\xtc{
Use \spadfunFrom{horizConcat}{Matrix} to append them side to side.
The two matrices must have the same number of rows.
}{
\spadpaste{horizConcat(a,b) \free{a b}}
}
\xtc{
Use \spadfunFrom{vertConcat}{Matrix} to stack one upon the other.
The two matrices must have the same number of columns.
}{
\spadpaste{vab := vertConcat(a,b) \free{a b}\bound{vab}}
}

%
\xtc{
The operation
\spadfunFrom{transpose}{Matrix} is used to create a new matrix by
reflection across the main diagonal.
}{
\spadpaste{transpose vab \free{vab}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxMatrixCreatePagePatch1}
\begin{paste}{ugxMatrixCreatePageFull1}{ugxMatrixCreatePageEmpty1}
\pastebutton{ugxMatrixCreatePageFull1}{\hidepaste}
\tab{5}\spadcommand{m : Matrix(Integer) := new(3,3,0)\bound{m }}
\indentrel{3}\begin{verbatim}
        0  0  0

   (1)  0  0  0

        0  0  0
                                   Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxMatrixCreatePageEmpty1}
\begin{paste}{ugxMatrixCreatePageEmpty1}{ugxMatrixCreatePagePatch1}
\pastebutton{ugxMatrixCreatePageEmpty1}{\showpaste}
\tab{5}\spadcommand{m : Matrix(Integer) := new(3,3,0)\bound{m }}
\end{paste}\end{patch}

\begin{patch}{ugxMatrixCreatePagePatch2}
```

```
\begin{paste}{ugxMatrixCreatePageFull2}{ugxMatrixCreatePageEmpty2}
\pastebutton{ugxMatrixCreatePageFull2}{\hidepaste}
\tab{5}\spadcommand{setelt(m,2,3,5)\free{m }\bound{m1 }}
\indentrel{3}\begin{verbatim}
   (2)  5
                                 Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxMatrixCreatePageEmpty2}
\begin{paste}{ugxMatrixCreatePageEmpty2}{ugxMatrixCreatePagePatch2}
\pastebutton{ugxMatrixCreatePageEmpty2}{\showpaste}
\tab{5}\spadcommand{setelt(m,2,3,5)\free{m }\bound{m1 }}
\end{paste}\end{patch}

\begin{patch}{ugxMatrixCreatePagePatch3}
\begin{paste}{ugxMatrixCreatePageFull3}{ugxMatrixCreatePageEmpty3}
\pastebutton{ugxMatrixCreatePageFull3}{\hidepaste}
\tab{5}\spadcommand{m(1,2) := 10\free{m1 }\bound{m2 }}
\indentrel{3}\begin{verbatim}
   (3)  10
                                 Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxMatrixCreatePageEmpty3}
\begin{paste}{ugxMatrixCreatePageEmpty3}{ugxMatrixCreatePagePatch3}
\pastebutton{ugxMatrixCreatePageEmpty3}{\showpaste}
\tab{5}\spadcommand{m(1,2) := 10\free{m1 }\bound{m2 }}
\end{paste}\end{patch}

\begin{patch}{ugxMatrixCreatePagePatch4}
\begin{paste}{ugxMatrixCreatePageFull4}{ugxMatrixCreatePageEmpty4}
\pastebutton{ugxMatrixCreatePageFull4}{\hidepaste}
\tab{5}\spadcommand{m\free{m2 }}
\indentrel{3}\begin{verbatim}
        0  10  0

   (4)  0  0   5

        0  0   0
                                  Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxMatrixCreatePageEmpty4}
\begin{paste}{ugxMatrixCreatePageEmpty4}{ugxMatrixCreatePagePatch4}
\pastebutton{ugxMatrixCreatePageEmpty4}{\showpaste}
\tab{5}\spadcommand{m\free{m2 }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugxMatrixCreatePagePatch5}
\begin{paste}{ugxMatrixCreatePageFull5}{ugxMatrixCreatePageEmpty5}
\pastebutton{ugxMatrixCreatePageFull5}{\hidepaste}
\tab{5}\spadcommand{matrix [[1,2,3,4],[0,9,8,7]]}
\indentrel{3}\begin{verbatim}
        1  2  3  4
   (5)
        0  9  8  7
                                   Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxMatrixCreatePageEmpty5}
\begin{paste}{ugxMatrixCreatePageEmpty5}{ugxMatrixCreatePagePatch5}
\pastebutton{ugxMatrixCreatePageEmpty5}{\showpaste}
\tab{5}\spadcommand{matrix [[1,2,3,4],[0,9,8,7]]}
\end{paste}\end{patch}

\begin{patch}{ugxMatrixCreatePagePatch6}
\begin{paste}{ugxMatrixCreatePageFull6}{ugxMatrixCreatePageEmpty6}
\pastebutton{ugxMatrixCreatePageFull6}{\hidepaste}
\tab{5}\spadcommand{dm := diagonalMatrix [1,x**2,x**3,x**4,x**5]\bound{dm }}
\indentrel{3}\begin{verbatim}
        1  0   0   0   0

            2
        0  x   0   0   0

                3
   (6)  0  0   x   0   0

                    4
        0  0   0   x   0

                        5
        0  0   0   0   x
                        Type: Matrix Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxMatrixCreatePageEmpty6}
\begin{paste}{ugxMatrixCreatePageEmpty6}{ugxMatrixCreatePagePatch6}
\pastebutton{ugxMatrixCreatePageEmpty6}{\showpaste}
\tab{5}\spadcommand{dm := diagonalMatrix [1,x**2,x**3,x**4,x**5]\bound{dm }}
\end{paste}\end{patch}

\begin{patch}{ugxMatrixCreatePagePatch7}
\begin{paste}{ugxMatrixCreatePageFull7}{ugxMatrixCreatePageEmpty7}
\pastebutton{ugxMatrixCreatePageFull7}{\hidepaste}
```

```
\tab{5}\spadcommand{setRow!(dm,5,vector [1,1,1,1,1])\free{dm }\bound{dm1 }}
\indentrel{3}\begin{verbatim}
       1  0   0   0   0

           2
       0  x   0   0   0

   (7)         3
       0  0   x   0   0

                   4
       0  0   0   x   0

       1  1   1   1   1
                      Type: Matrix Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxMatrixCreatePageEmpty7}
\begin{paste}{ugxMatrixCreatePageEmpty7}{ugxMatrixCreatePagePatch7}
\pastebutton{ugxMatrixCreatePageEmpty7}{\showpaste}
\tab{5}\spadcommand{setRow!(dm,5,vector [1,1,1,1,1])\free{dm }\bound{dm1 }}
\end{paste}\end{patch}

\begin{patch}{ugxMatrixCreatePagePatch8}
\begin{paste}{ugxMatrixCreatePageFull8}{ugxMatrixCreatePageEmpty8}
\pastebutton{ugxMatrixCreatePageFull8}{\hidepaste}
\tab{5}\spadcommand{setColumn!(dm,2,vector [y,y,y,y,y])\free{dm1 }\bound{dm2 }}
\indentrel{3}\begin{verbatim}
       1  y  0   0   0

       0  y  0   0   0

              3
   (8) 0  y  x   0   0

                  4
       0  y  0   x   0

       1  y  1   1   1
                      Type: Matrix Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxMatrixCreatePageEmpty8}
\begin{paste}{ugxMatrixCreatePageEmpty8}{ugxMatrixCreatePagePatch8}
\pastebutton{ugxMatrixCreatePageEmpty8}{\showpaste}
\tab{5}\spadcommand{setColumn!(dm,2,vector [y,y,y,y,y])\free{dm1 }\bound{dm2 }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugxMatrixCreatePagePatch9}
\begin{paste}{ugxMatrixCreatePageFull9}{ugxMatrixCreatePageEmpty9}
\pastebutton{ugxMatrixCreatePageFull9}{\hidepaste}
\tab{5}\spadcommand{cdm := copy(dm)\free{dm2 }\bound{cdm }}
\indentrel{3}\begin{verbatim}
       1  y  0   0   0

       0  y  0   0   0

             3
   (9) 0  y  x   0   0

                 4
       0  y  0   x   0

       1  y  1   1   1
                    Type: Matrix Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxMatrixCreatePageEmpty9}
\begin{paste}{ugxMatrixCreatePageEmpty9}{ugxMatrixCreatePagePatch9}
\pastebutton{ugxMatrixCreatePageEmpty9}{\showpaste}
\tab{5}\spadcommand{cdm := copy(dm)\free{dm2 }\bound{cdm }}
\end{paste}\end{patch}

\begin{patch}{ugxMatrixCreatePagePatch10}
\begin{paste}{ugxMatrixCreatePageFull10}{ugxMatrixCreatePageEmpty10}
\pastebutton{ugxMatrixCreatePageFull10}{\hidepaste}
\tab{5}\spadcommand{setelt(dm,4,1,1-x**7)\free{dm2 }\bound{setdm }}
\indentrel{3}\begin{verbatim}
          7
   (10)  - x  + 1
                          Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxMatrixCreatePageEmpty10}
\begin{paste}{ugxMatrixCreatePageEmpty10}{ugxMatrixCreatePagePatch10}
\pastebutton{ugxMatrixCreatePageEmpty10}{\showpaste}
\tab{5}\spadcommand{setelt(dm,4,1,1-x**7)\free{dm2 }\bound{setdm }}
\end{paste}\end{patch}

\begin{patch}{ugxMatrixCreatePagePatch11}
\begin{paste}{ugxMatrixCreatePageFull11}{ugxMatrixCreatePageEmpty11}
\pastebutton{ugxMatrixCreatePageFull11}{\hidepaste}
\tab{5}\spadcommand{[dm,cdm]\free{setdm cdm }}
\indentrel{3}\begin{verbatim}
            1      y  0   0   0 1  y  0   0   0
```

```
            0       y  0   0   0 0  y  0   0   0

                       3               3
   (11)  [   0       y  x   0   0,0  y  x   0   0]

            7              4              4
         - x  + 1  y  0   x   0 0  y  0   x   0

            1       y  1   1   1 1  y  1   1   1
                  Type: List Matrix Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxMatrixCreatePageEmpty11}
\begin{paste}{ugxMatrixCreatePageEmpty11}{ugxMatrixCreatePagePatch11}
\pastebutton{ugxMatrixCreatePageEmpty11}{\showpaste}
\tab{5}\spadcommand{[dm,cdm]\free{setdm cdm }}
\end{paste}\end{patch}

\begin{patch}{ugxMatrixCreatePagePatch12}
\begin{paste}{ugxMatrixCreatePageFull12}{ugxMatrixCreatePageEmpty12}
\pastebutton{ugxMatrixCreatePageFull12}{\hidepaste}
\tab{5}\spadcommand{subMatrix(dm,2,3,2,4)\free{setdm }}
\indentrel{3}\begin{verbatim}
         y  0   0
   (12)
            3
         y  x   0
                       Type: Matrix Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxMatrixCreatePageEmpty12}
\begin{paste}{ugxMatrixCreatePageEmpty12}{ugxMatrixCreatePagePatch12}
\pastebutton{ugxMatrixCreatePageEmpty12}{\showpaste}
\tab{5}\spadcommand{subMatrix(dm,2,3,2,4)\free{setdm }}
\end{paste}\end{patch}

\begin{patch}{ugxMatrixCreatePagePatch13}
\begin{paste}{ugxMatrixCreatePageFull13}{ugxMatrixCreatePageEmpty13}
\pastebutton{ugxMatrixCreatePageFull13}{\hidepaste}
\tab{5}\spadcommand{d := diagonalMatrix [1.2,-1.3,1.4,-1.5]\bound{d }}
\indentrel{3}\begin{verbatim}
         1.2   0.0   0.0   0.0

         0.0  - 1.3  0.0   0.0
   (13)
         0.0   0.0   1.4   0.0

         0.0   0.0   0.0  - 1.5
```

```
                                     Type: Matrix Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxMatrixCreatePageEmpty13}
\begin{paste}{ugxMatrixCreatePageEmpty13}{ugxMatrixCreatePagePatch13}
\pastebutton{ugxMatrixCreatePageEmpty13}{\showpaste}
\tab{5}\spadcommand{d := diagonalMatrix [1.2,-1.3,1.4,-1.5]\bound{d }}
\end{paste}\end{patch}

\begin{patch}{ugxMatrixCreatePagePatch14}
\begin{paste}{ugxMatrixCreatePageFull14}{ugxMatrixCreatePageEmpty14}
\pastebutton{ugxMatrixCreatePageFull14}{\hidepaste}
\tab{5}\spadcommand{e := matrix [[6.7,9.11],[-31.33,67.19]]\bound{e }}
\indentrel{3}\begin{verbatim}
           6.7    9.11
   (14)
         - 31.33  67.19
                                     Type: Matrix Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxMatrixCreatePageEmpty14}
\begin{paste}{ugxMatrixCreatePageEmpty14}{ugxMatrixCreatePagePatch14}
\pastebutton{ugxMatrixCreatePageEmpty14}{\showpaste}
\tab{5}\spadcommand{e := matrix [[6.7,9.11],[-31.33,67.19]]\bound{e }}
\end{paste}\end{patch}

\begin{patch}{ugxMatrixCreatePagePatch15}
\begin{paste}{ugxMatrixCreatePageFull15}{ugxMatrixCreatePageEmpty15}
\pastebutton{ugxMatrixCreatePageFull15}{\hidepaste}
\tab{5}\spadcommand{setsubMatrix!(d,1,2,e)\free{d e }\bound{d1 }}
\indentrel{3}\begin{verbatim}
         1.2    6.7    9.11    0.0

         0.0  - 31.33  67.19   0.0
   (15)
         0.0    0.0     1.4    0.0

         0.0    0.0     0.0   - 1.5
                                     Type: Matrix Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxMatrixCreatePageEmpty15}
\begin{paste}{ugxMatrixCreatePageEmpty15}{ugxMatrixCreatePagePatch15}
\pastebutton{ugxMatrixCreatePageEmpty15}{\showpaste}
\tab{5}\spadcommand{setsubMatrix!(d,1,2,e)\free{d e }\bound{d1 }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugxMatrixCreatePagePatch16}
\begin{paste}{ugxMatrixCreatePageFull16}{ugxMatrixCreatePageEmpty16}
\pastebutton{ugxMatrixCreatePageFull16}{\hidepaste}
\tab{5}\spadcommand{d\free{d1 }}
\indentrel{3}\begin{verbatim}
         1.2    6.7    9.11    0.0

         0.0  - 31.33  67.19   0.0
   (16)
         0.0    0.0     1.4    0.0

         0.0    0.0     0.0   - 1.5
                                     Type: Matrix Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxMatrixCreatePageEmpty16}
\begin{paste}{ugxMatrixCreatePageEmpty16}{ugxMatrixCreatePagePatch16}
\pastebutton{ugxMatrixCreatePageEmpty16}{\showpaste}
\tab{5}\spadcommand{d\free{d1 }}
\end{paste}\end{patch}

\begin{patch}{ugxMatrixCreatePagePatch17}
\begin{paste}{ugxMatrixCreatePageFull17}{ugxMatrixCreatePageEmpty17}
\pastebutton{ugxMatrixCreatePageFull17}{\hidepaste}
\tab{5}\spadcommand{a := matrix [[1/2,1/3,1/4],[1/5,1/6,1/7]]\bound{a }}
\indentrel{3}\begin{verbatim}
         1  1  1

         2  3  4
   (17)
         1  1  1

         5  6  7
                          Type: Matrix Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxMatrixCreatePageEmpty17}
\begin{paste}{ugxMatrixCreatePageEmpty17}{ugxMatrixCreatePagePatch17}
\pastebutton{ugxMatrixCreatePageEmpty17}{\showpaste}
\tab{5}\spadcommand{a := matrix [[1/2,1/3,1/4],[1/5,1/6,1/7]]\bound{a }}
\end{paste}\end{patch}

\begin{patch}{ugxMatrixCreatePagePatch18}
\begin{paste}{ugxMatrixCreatePageFull18}{ugxMatrixCreatePageEmpty18}
\pastebutton{ugxMatrixCreatePageFull18}{\hidepaste}
\tab{5}\spadcommand{b := matrix [[3/5,3/7,3/11],[3/13,3/17,3/19]]\bound{b }}
\indentrel{3}\begin{verbatim}
         3   3    3
```

```
        5   7   11
   (18)
         3   3   3

        13  17  19
                         Type: Matrix Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxMatrixCreatePageEmpty18}
\begin{paste}{ugxMatrixCreatePageEmpty18}{ugxMatrixCreatePagePatch18}
\pastebutton{ugxMatrixCreatePageEmpty18}{\showpaste}
\tab{5}\spadcommand{b := matrix [[3/5,3/7,3/11],[3/13,3/17,3/19]]\bound{b }}
\end{paste}\end{patch}

\begin{patch}{ugxMatrixCreatePagePatch19}
\begin{paste}{ugxMatrixCreatePageFull19}{ugxMatrixCreatePageEmpty19}
\pastebutton{ugxMatrixCreatePageFull19}{\hidepaste}
\tab{5}\spadcommand{horizConcat(a,b)\free{a b }}
\indentrel{3}\begin{verbatim}
        1  1  1  3   3    3

        2  3  4  5   7   11
   (19)
        1  1  1   3   3   3

        5  6  7  13  17  19
                         Type: Matrix Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxMatrixCreatePageEmpty19}
\begin{paste}{ugxMatrixCreatePageEmpty19}{ugxMatrixCreatePagePatch19}
\pastebutton{ugxMatrixCreatePageEmpty19}{\showpaste}
\tab{5}\spadcommand{horizConcat(a,b)\free{a b }}
\end{paste}\end{patch}

\begin{patch}{ugxMatrixCreatePagePatch20}
\begin{paste}{ugxMatrixCreatePageFull20}{ugxMatrixCreatePageEmpty20}
\pastebutton{ugxMatrixCreatePageFull20}{\hidepaste}
\tab{5}\spadcommand{vab := vertConcat(a,b)\free{a b }\bound{vab }}
\indentrel{3}\begin{verbatim}
        1   1   1

        2   3   4

        1   1   1

        5   6   7
```

```
   (20)
         3   3    3
         -   -   --
         5   7   11

          3    3   3
         --   --  --
         13   17  19
                             Type: Matrix Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxMatrixCreatePageEmpty20}
\begin{paste}{ugxMatrixCreatePageEmpty20}{ugxMatrixCreatePagePatch20}
\pastebutton{ugxMatrixCreatePageEmpty20}{\showpaste}
\tab{5}\spadcommand{vab := vertConcat(a,b)\free{a b }\bound{vab }}
\end{paste}\end{patch}

\begin{patch}{ugxMatrixCreatePagePatch21}
\begin{paste}{ugxMatrixCreatePageFull21}{ugxMatrixCreatePageEmpty21}
\pastebutton{ugxMatrixCreatePageFull21}{\hidepaste}
\tab{5}\spadcommand{transpose vab\free{vab }}
\indentrel{3}\begin{verbatim}
         1  1  3    3

         2  5  5   13

         1  1  3    3
   (21)
         3  6  7   17

         1  1   3   3

         4  7  11  19
                             Type: Matrix Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxMatrixCreatePageEmpty21}
\begin{paste}{ugxMatrixCreatePageEmpty21}{ugxMatrixCreatePagePatch21}
\pastebutton{ugxMatrixCreatePageEmpty21}{\showpaste}
\tab{5}\spadcommand{transpose vab\free{vab }}
\end{paste}\end{patch}
```

———

Operations on Matrices

⇒ "notitle" (ugIntroTwoDimPage) 6 on page 1548
⇒ "notitle" (ugProblemEigenPage) 12 on page 2123
⇒ "notitle" (ugxFloatHilbertPage) 3.41 on page 502
⇒ "notitle" (PermanentXmpPage) 3.85 on page 1085
⇒ "notitle" (VectorXmpPage) 3.114 on page 1351
⇒ "notitle" (OneDimensionalArrayXmpPage) 3.4 on page 138
⇒ "notitle" (TwoDimensionalArrayXmpPage) 3.5 on page 143

— **matrix.ht** —

```
\begin{page}{ugxMatrixOpsPage}{Operations on Matrices}
\beginscroll

\labelSpace{3pc}
\xtc{
Axiom provides both left and right scalar multiplication.
}{
\spadpaste{m := matrix [[1,2],[3,4]] \bound{m}}
}
\xtc{
}{
\spadpaste{4 * m * (-5)\free{m}}
}
\xtc{
You can add, subtract, and multiply matrices provided, of course, that
the matrices have compatible dimensions.
If not, an error message is displayed.
}{
\spadpaste{n := matrix([[1,0,-2],[-3,5,1]]) \bound{n}}
}
\xtc{
This following product is defined but \spad{n * m} is not.
}{
\spadpaste{m * n \free{m n}}
}

The operations \spadfunFrom{nrows}{Matrix} and \spadfunFrom{ncols}{Matrix}
return the number of rows and columns of a matrix.
You can extract a row or a column of a matrix using the operations
\spadfunFrom{row}{Matrix} and \spadfunFrom{column}{Matrix}.
The object returned is a \spadtype{Vector}.
\xtc{
Here is the third column of the matrix \spad{n}.
}{
\spadpaste{vec := column(n,3) \free{n} \bound{vec}}
}
\xtc{
```

```
You can multiply a matrix on the left by a ``row vector'' and on the right
by a ``column vector.''
}{
\spadpaste{vec * m \free{vec m}}
}
\xtc{
Of course, the dimensions of the vector and the matrix must be compatible
or an error message is returned.
}{
\spadpaste{m * vec \free{vec m}}
}

The operation \spadfunFrom{inverse}{Matrix} computes the inverse of a
matrix if
{Operations on Matrices}
the matrix is invertible, and returns \spad{"failed"} if not.
\xtc{
This Hilbert matrix is invertible.
}{
\spadpaste{hilb := matrix([[1/(i + j) for i in 1..3] for j in 1..3])
\bound{hilb}}
}
\xtc{
}{
\spadpaste{inverse(hilb) \free{hilb}}
}
\xtc{
This matrix is not invertible.
}{
\spadpaste{mm := matrix([[1,2,3,4], [5,6,7,8], [9,10,11,12],
[13,14,15,16]]) \bound{mm}}
}
\xtc{
}{
\spadpaste{inverse(mm) \free{mm}}
}

The operation
\spadfunFrom{determinant}{Matrix} computes the determinant of a matrix
{Operations on Matrices}
provided that the entries of the matrix belong to a
\spadtype{CommutativeRing}.
\xtc{
The above matrix \spad{mm} is not invertible and, hence, must have
determinant \spad{0}.
}{
\spadpaste{determinant(mm) \free{mm}}
}
\xtc{
The operation
```

```
\spadfunFrom{trace}{SquareMatrix} computes the trace of a {\em square}
matrix.
{Operations on Matrices}
}{
\spadpaste{trace(mm) \free{mm}}
}

\xtc{
The operation \spadfunFrom{rank}{Matrix} computes the {\it rank} of a
matrix:
the maximal number of linearly independent rows or columns.
}{
\spadpaste{rank(mm) \free{mm}}
}
\xtc{
The operation \spadfunFrom{nullity}{Matrix} computes the {\it nullity} of
a matrix: the dimension of its null space.
}{
\spadpaste{nullity(mm) \free{mm}}
}
\xtc{
The operation \spadfunFrom{nullSpace}{Matrix} returns a list containing
a basis for the null space of a matrix.
Note that the nullity is the number of elements in a basis for the null
space.
}{
\spadpaste{nullSpace(mm) \free{mm}}
}
\xtc{
The operation
\spadfunFrom{rowEchelon}{Matrix} returns the row echelon form of a
{Operations on Matrices}
matrix.
It is easy to see that the rank of this matrix is two and that its nullity
is also two.
}{
\spadpaste{rowEchelon(mm) \free{mm}}
}

For more information on related topics, see
\downlink{``Expanding to Higher Dimensions''}{ugIntroTwoDimPage} in
Section 1.7\ignore{ugIntroTwoDim},
\downlink{``Computation of Eigenvalues and Eigenvectors''}
{ugProblemEigenPage} in Section 8.4\ignore{ugProblemEigen},
\downlink{``Determinant of a Hilbert Matrix''}
{ugxFloatHilbertPage}\ignore{ugxFloatHilbert},
\downlink{`Permanent'}{PermanentXmpPage}\ignore{Permanent},
\downlink{`Vector'}{VectorXmpPage}\ignore{Vector},
\downlink{`OneDimensionalArray'}{OneDimensionalArrayXmpPage}
\ignore{OneDimensionalArray}, and
```

```
\downlink{`TwoDimensionalArray'}{TwoDimensionalArrayXmpPage}
\ignore{TwoDimensionalArray}.
%
\showBlurb{Matrix}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxMatrixOpsPagePatch1}
\begin{paste}{ugxMatrixOpsPageFull1}{ugxMatrixOpsPageEmpty1}
\pastebutton{ugxMatrixOpsPageFull1}{\hidepaste}
\tab{5}\spadcommand{m := matrix [[1,2],[3,4]]\bound{m }}
\indentrel{3}\begin{verbatim}
        1  2
   (1)
        3  4
                                   Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxMatrixOpsPageEmpty1}
\begin{paste}{ugxMatrixOpsPageEmpty1}{ugxMatrixOpsPagePatch1}
\pastebutton{ugxMatrixOpsPageEmpty1}{\showpaste}
\tab{5}\spadcommand{m := matrix [[1,2],[3,4]]\bound{m }}
\end{paste}\end{patch}

\begin{patch}{ugxMatrixOpsPagePatch2}
\begin{paste}{ugxMatrixOpsPageFull2}{ugxMatrixOpsPageEmpty2}
\pastebutton{ugxMatrixOpsPageFull2}{\hidepaste}
\tab{5}\spadcommand{4 * m * (-5)\free{m }}
\indentrel{3}\begin{verbatim}
        - 20  - 40
   (2)
        - 60  - 80
                                   Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxMatrixOpsPageEmpty2}
\begin{paste}{ugxMatrixOpsPageEmpty2}{ugxMatrixOpsPagePatch2}
\pastebutton{ugxMatrixOpsPageEmpty2}{\showpaste}
\tab{5}\spadcommand{4 * m * (-5)\free{m }}
\end{paste}\end{patch}

\begin{patch}{ugxMatrixOpsPagePatch3}
\begin{paste}{ugxMatrixOpsPageFull3}{ugxMatrixOpsPageEmpty3}
\pastebutton{ugxMatrixOpsPageFull3}{\hidepaste}
\tab{5}\spadcommand{n := matrix([[1,0,-2],[-3,5,1]])\bound{n }}
\indentrel{3}\begin{verbatim}
         1   0  - 2
```

```
   (3)
        - 3  5   1
                                   Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxMatrixOpsPageEmpty3}
\begin{paste}{ugxMatrixOpsPageEmpty3}{ugxMatrixOpsPagePatch3}
\pastebutton{ugxMatrixOpsPageEmpty3}{\showpaste}
\tab{5}\spadcommand{n := matrix([[1,0,-2],[-3,5,1]])\bound{n }}
\end{paste}\end{patch}

\begin{patch}{ugxMatrixOpsPagePatch4}
\begin{paste}{ugxMatrixOpsPageFull4}{ugxMatrixOpsPageEmpty4}
\pastebutton{ugxMatrixOpsPageFull4}{\hidepaste}
\tab{5}\spadcommand{m * n\free{m n }}
\indentrel{3}\begin{verbatim}
        - 5  10   0
   (4)
        - 9  20  - 2
                                   Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxMatrixOpsPageEmpty4}
\begin{paste}{ugxMatrixOpsPageEmpty4}{ugxMatrixOpsPagePatch4}
\pastebutton{ugxMatrixOpsPageEmpty4}{\showpaste}
\tab{5}\spadcommand{m * n\free{m n }}
\end{paste}\end{patch}

\begin{patch}{ugxMatrixOpsPagePatch5}
\begin{paste}{ugxMatrixOpsPageFull5}{ugxMatrixOpsPageEmpty5}
\pastebutton{ugxMatrixOpsPageFull5}{\hidepaste}
\tab{5}\spadcommand{vec := column(n,3)\free{n }\bound{vec }}
\indentrel{3}\begin{verbatim}
   (5)  [- 2,1]
                                   Type: Vector Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxMatrixOpsPageEmpty5}
\begin{paste}{ugxMatrixOpsPageEmpty5}{ugxMatrixOpsPagePatch5}
\pastebutton{ugxMatrixOpsPageEmpty5}{\showpaste}
\tab{5}\spadcommand{vec := column(n,3)\free{n }\bound{vec }}
\end{paste}\end{patch}

\begin{patch}{ugxMatrixOpsPagePatch6}
\begin{paste}{ugxMatrixOpsPageFull6}{ugxMatrixOpsPageEmpty6}
\pastebutton{ugxMatrixOpsPageFull6}{\hidepaste}
\tab{5}\spadcommand{vec * m\free{vec m }}
```

```
\indentrel{3}\begin{verbatim}
   (6)  [1,0]
                                  Type: Vector Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxMatrixOpsPageEmpty6}
\begin{paste}{ugxMatrixOpsPageEmpty6}{ugxMatrixOpsPagePatch6}
\pastebutton{ugxMatrixOpsPageEmpty6}{\showpaste}
\tab{5}\spadcommand{vec * m\free{vec m }}
\end{paste}\end{patch}

\begin{patch}{ugxMatrixOpsPagePatch7}
\begin{paste}{ugxMatrixOpsPageFull7}{ugxMatrixOpsPageEmpty7}
\pastebutton{ugxMatrixOpsPageFull7}{\hidepaste}
\tab{5}\spadcommand{m * vec\free{vec m }}
\indentrel{3}\begin{verbatim}
   (7)  [0,- 2]
                                  Type: Vector Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxMatrixOpsPageEmpty7}
\begin{paste}{ugxMatrixOpsPageEmpty7}{ugxMatrixOpsPagePatch7}
\pastebutton{ugxMatrixOpsPageEmpty7}{\showpaste}
\tab{5}\spadcommand{m * vec\free{vec m }}
\end{paste}\end{patch}

\begin{patch}{ugxMatrixOpsPagePatch8}
\begin{paste}{ugxMatrixOpsPageFull8}{ugxMatrixOpsPageEmpty8}
\pastebutton{ugxMatrixOpsPageFull8}{\hidepaste}
\tab{5}\spadcommand{hilb := matrix([[1/(i + j) for i in 1..3] for j in 1..3])\bound{hilb }}
\indentrel{3}\begin{verbatim}
        1  1  1

        2  3  4

        1  1  1
   (8)
        3  4  5

        1  1  1

        4  5  6
                         Type: Matrix Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxMatrixOpsPageEmpty8}
\begin{paste}{ugxMatrixOpsPageEmpty8}{ugxMatrixOpsPagePatch8}
```

```
\pastebutton{ugxMatrixOpsPageEmpty8}{\showpaste}
\tab{5}\spadcommand{hilb := matrix([[1/(i + j) for i in 1..3] for j in 1..3])\bound{hilb }}
\end{paste}\end{patch}

\begin{patch}{ugxMatrixOpsPagePatch9}
\begin{paste}{ugxMatrixOpsPageFull9}{ugxMatrixOpsPageEmpty9}
\pastebutton{ugxMatrixOpsPageFull9}{\hidepaste}
\tab{5}\spadcommand{inverse(hilb)\free{hilb }}
\indentrel{3}\begin{verbatim}
         72    - 240   180

   (9)  - 240   900   - 720

         180   - 720   600
               Type: Union(Matrix Fraction Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxMatrixOpsPageEmpty9}
\begin{paste}{ugxMatrixOpsPageEmpty9}{ugxMatrixOpsPagePatch9}
\pastebutton{ugxMatrixOpsPageEmpty9}{\showpaste}
\tab{5}\spadcommand{inverse(hilb)\free{hilb }}
\end{paste}\end{patch}

\begin{patch}{ugxMatrixOpsPagePatch10}
\begin{paste}{ugxMatrixOpsPageFull10}{ugxMatrixOpsPageEmpty10}
\pastebutton{ugxMatrixOpsPageFull10}{\hidepaste}
\tab{5}\spadcommand{mm := matrix([[1,2,3,4], [5,6,7,8], [9,10,11,12], [13,14,15,16]])\bound{mm }}
\indentrel{3}\begin{verbatim}
         1   2   3   4

         5   6   7   8
   (10)
         9   10  11  12

         13  14  15  16
                                   Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxMatrixOpsPageEmpty10}
\begin{paste}{ugxMatrixOpsPageEmpty10}{ugxMatrixOpsPagePatch10}
\pastebutton{ugxMatrixOpsPageEmpty10}{\showpaste}
\tab{5}\spadcommand{mm := matrix([[1,2,3,4], [5,6,7,8], [9,10,11,12], [13,14,15,16]])\bound{mm }}
\end{paste}\end{patch}

\begin{patch}{ugxMatrixOpsPagePatch11}
\begin{paste}{ugxMatrixOpsPageFull11}{ugxMatrixOpsPageEmpty11}
\pastebutton{ugxMatrixOpsPageFull11}{\hidepaste}
\tab{5}\spadcommand{inverse(mm)\free{mm }}
```

```
\indentrel{3}\begin{verbatim}
   (11)  "failed"
                              Type: Union("failed",...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxMatrixOpsPageEmpty11}
\begin{paste}{ugxMatrixOpsPageEmpty11}{ugxMatrixOpsPagePatch11}
\pastebutton{ugxMatrixOpsPageEmpty11}{\showpaste}
\tab{5}\spadcommand{inverse(mm)\free{mm }}
\end{paste}\end{patch}

\begin{patch}{ugxMatrixOpsPagePatch12}
\begin{paste}{ugxMatrixOpsPageFull12}{ugxMatrixOpsPageEmpty12}
\pastebutton{ugxMatrixOpsPageFull12}{\hidepaste}
\tab{5}\spadcommand{determinant(mm)\free{mm }}
\indentrel{3}\begin{verbatim}
   (12)  0
                               Type: NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxMatrixOpsPageEmpty12}
\begin{paste}{ugxMatrixOpsPageEmpty12}{ugxMatrixOpsPagePatch12}
\pastebutton{ugxMatrixOpsPageEmpty12}{\showpaste}
\tab{5}\spadcommand{determinant(mm)\free{mm }}
\end{paste}\end{patch}

\begin{patch}{ugxMatrixOpsPagePatch13}
\begin{paste}{ugxMatrixOpsPageFull13}{ugxMatrixOpsPageEmpty13}
\pastebutton{ugxMatrixOpsPageFull13}{\hidepaste}
\tab{5}\spadcommand{trace(mm)\free{mm }}
\indentrel{3}\begin{verbatim}
   (13)  34
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxMatrixOpsPageEmpty13}
\begin{paste}{ugxMatrixOpsPageEmpty13}{ugxMatrixOpsPagePatch13}
\pastebutton{ugxMatrixOpsPageEmpty13}{\showpaste}
\tab{5}\spadcommand{trace(mm)\free{mm }}
\end{paste}\end{patch}

\begin{patch}{ugxMatrixOpsPagePatch14}
\begin{paste}{ugxMatrixOpsPageFull14}{ugxMatrixOpsPageEmpty14}
\pastebutton{ugxMatrixOpsPageFull14}{\hidepaste}
\tab{5}\spadcommand{rank(mm)\free{mm }}
\indentrel{3}\begin{verbatim}
   (14)  2
```

```
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxMatrixOpsPageEmpty14}
\begin{paste}{ugxMatrixOpsPageEmpty14}{ugxMatrixOpsPagePatch14}
\pastebutton{ugxMatrixOpsPageEmpty14}{\showpaste}
\tab{5}\spadcommand{rank(mm)\free{mm }}
\end{paste}\end{patch}

\begin{patch}{ugxMatrixOpsPagePatch15}
\begin{paste}{ugxMatrixOpsPageFull15}{ugxMatrixOpsPageEmpty15}
\pastebutton{ugxMatrixOpsPageFull15}{\hidepaste}
\tab{5}\spadcommand{nullity(mm)\free{mm }}
\indentrel{3}\begin{verbatim}
   (15)  2
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxMatrixOpsPageEmpty15}
\begin{paste}{ugxMatrixOpsPageEmpty15}{ugxMatrixOpsPagePatch15}
\pastebutton{ugxMatrixOpsPageEmpty15}{\showpaste}
\tab{5}\spadcommand{nullity(mm)\free{mm }}
\end{paste}\end{patch}

\begin{patch}{ugxMatrixOpsPagePatch16}
\begin{paste}{ugxMatrixOpsPageFull16}{ugxMatrixOpsPageEmpty16}
\pastebutton{ugxMatrixOpsPageFull16}{\hidepaste}
\tab{5}\spadcommand{nullSpace(mm)\free{mm }}
\indentrel{3}\begin{verbatim}
   (16)  [[1,- 2,1,0],[2,- 3,0,1]]
                              Type: List Vector Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxMatrixOpsPageEmpty16}
\begin{paste}{ugxMatrixOpsPageEmpty16}{ugxMatrixOpsPagePatch16}
\pastebutton{ugxMatrixOpsPageEmpty16}{\showpaste}
\tab{5}\spadcommand{nullSpace(mm)\free{mm }}
\end{paste}\end{patch}

\begin{patch}{ugxMatrixOpsPagePatch17}
\begin{paste}{ugxMatrixOpsPageFull17}{ugxMatrixOpsPageEmpty17}
\pastebutton{ugxMatrixOpsPageFull17}{\hidepaste}
\tab{5}\spadcommand{rowEchelon(mm)\free{mm }}
\indentrel{3}\begin{verbatim}
         1  2  3  4

         0  4  8  12
```

```
  (17)
       0  0  0  0

       0  0  0  0
                                         Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxMatrixOpsPageEmpty17}
\begin{paste}{ugxMatrixOpsPageEmpty17}{ugxMatrixOpsPagePatch17}
\pastebutton{ugxMatrixOpsPageEmpty17}{\showpaste}
\tab{5}\spadcommand{rowEchelon(mm)\free{mm }}
\end{paste}\end{patch}
```

3.76 mkfunc.ht

MakeFunction

⇒ "notitle" (ugUserMakePage) 10 on page 1889

— **mkfunc.ht** —

```
\begin{page}{MakeFunctionXmpPage}{MakeFunction}
\beginscroll
It is sometimes useful to be able to define a function given by
the result of a calculation.
%
\xtc{
Suppose that you have obtained the following expression
after several computations
and that you now want to tabulate the numerical values of \spad{f}
for \spad{x} between \spad{-1} and \spad{+1} with increment
\spad{0.1}.
}{
\spadpaste{expr := (x - exp x + 1)**2 * (sin(x**2) * x + 1)**3
\bound{expr}}
}
%
You could, of course, use the function
\spadfunFrom{eval}{Expression} within a loop and evaluate
\spad{expr} twenty-one times, but this would be quite slow.
A better way is to create a numerical function \spad{f} such that
\spad{f(x)} is defined by the expression \spad{expr} above,
but without retyping \spad{expr}!
The package \spadtype{MakeFunction} provides the operation
```

```
\spadfunFrom{function}{MakeFunction} which does exactly this.
%
\xtc{
Issue this to create the function \spad{f(x)} given by \spad{expr}.
}{
\spadpaste{function(expr, f, x) \bound{f}\free{expr}}
}
\xtc{
To tabulate \spad{expr}, we can now quickly evaluate \spad{f} 21 times.
}{
\spadpaste{tbl := [f(0.1 * i - 1) for i in 0..20]; \free{f}\bound{tbl}}
}
%
%
\xtc{
Use the list \spad{[x1,...,xn]} as the
third argument to \spadfunFrom{function}{MakeFunction}
to create a multivariate function \spad{f(x1,...,xn)}.
}{
\spadpaste{e := (x - y + 1)**2 * (x**2 * y + 1)**2 \bound{e}}
}
\xtc{
}{
\spadpaste{function(e, g, [x, y]) \free{e}}
}
%
%
\xtc{
In the case of just two
variables, they can be given as arguments without making them into a list.
}{
\spadpaste{function(e, h, x, y) \free{e}\bound{h}}
}
%
%
\xtc{
Note that the functions created by \spadfunFrom{function}{MakeFunction}
are not limited to floating point numbers, but can be applied to any type
for which they are defined.
}{
\spadpaste{m1 := squareMatrix [[1, 2], [3, 4]] \bound{m1}}
}
\xtc{
}{
\spadpaste{m2 := squareMatrix [[1, 0], [-1, 1]] \bound{m2}}
}
\xtc{
}{
\spadpaste{h(m1, m2) \free{h m1 m2}}
}
```

```
%
For more information, see
\downlink{``Making Functions from Objects''}{ugUserMakePage}
in Section 6.14\ignore{ugUserMake}.
\showBlurb{MakeFunction}
\endscroll
\autobuttons
\end{page}

\begin{patch}{MakeFunctionXmpPagePatch1}
\begin{paste}{MakeFunctionXmpPageFull1}{MakeFunctionXmpPageEmpty1}
\pastebutton{MakeFunctionXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{expr := (x - exp x + 1)**2 * (sin(x**2) * x + 1)**3\bound{expr }}
\indentrel{3}\begin{verbatim}
   (1)
       3   x 2        4     3   x    5     4    3      2 3
     (x (%e )  + (- 2x  - 2x )%e  + x  + 2x  + x )sin(x )
   +
          2   x 2        3     2   x     4     3     2
       (3x (%e )  + (- 6x  - 6x )%e  + 3x  + 6x  + 3x )
    *
            2 2
       sin(x )
   +
            x 2        2        x     3     2           2
     (3x (%e )  + (- 6x  - 6x)%e  + 3x  + 6x  + 3x)sin(x )
   +
        x 2               x    2
     (%e )  + (- 2x - 2)%e  + x  + 2x + 1
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MakeFunctionXmpPageEmpty1}
\begin{paste}{MakeFunctionXmpPageEmpty1}{MakeFunctionXmpPagePatch1}
\pastebutton{MakeFunctionXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{expr := (x - exp x + 1)**2 * (sin(x**2) * x + 1)**3\bound{expr }}
\end{paste}\end{patch}

\begin{patch}{MakeFunctionXmpPagePatch2}
\begin{paste}{MakeFunctionXmpPageFull2}{MakeFunctionXmpPageEmpty2}
\pastebutton{MakeFunctionXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{function(expr, f, x)\bound{f }\free{expr }}
\indentrel{3}\begin{verbatim}
   (2)  f
                                           Type: Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MakeFunctionXmpPageEmpty2}
```

```
\begin{paste}{MakeFunctionXmpPageEmpty2}{MakeFunctionXmpPagePatch2}
\pastebutton{MakeFunctionXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{function(expr, f, x)\bound{f }\free{expr }}
\end{paste}\end{patch}

\begin{patch}{MakeFunctionXmpPagePatch3}
\begin{paste}{MakeFunctionXmpPageFull3}{MakeFunctionXmpPageEmpty3}
\pastebutton{MakeFunctionXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{tbl := [f(0.1 * i - 1) for i in 0..20];\free{f }\bound{tbl }}
\indentrel{3}\begin{verbatim}
                                   Type: List Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MakeFunctionXmpPageEmpty3}
\begin{paste}{MakeFunctionXmpPageEmpty3}{MakeFunctionXmpPagePatch3}
\pastebutton{MakeFunctionXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{tbl := [f(0.1 * i - 1) for i in 0..20];\free{f }\bound{tbl }}
\end{paste}\end{patch}

\begin{patch}{MakeFunctionXmpPagePatch4}
\begin{paste}{MakeFunctionXmpPageFull4}{MakeFunctionXmpPageEmpty4}
\pastebutton{MakeFunctionXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{e := (x - y + 1)**2 * (x**2 * y + 1)**2\bound{e }}
\indentrel{3}\begin{verbatim}
   (4)
     4 4        5     4     2  3
    x y  + (- 2x  - 2x  + 2x )y
   +
      6     5    4     3     2      2
    (x  + 2x  + x  - 4x  - 4x  + 1)y
   +
       4     3     2              2
    (2x  + 4x  + 2x  - 2x - 2)y + x  + 2x + 1
                            Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MakeFunctionXmpPageEmpty4}
\begin{paste}{MakeFunctionXmpPageEmpty4}{MakeFunctionXmpPagePatch4}
\pastebutton{MakeFunctionXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{e := (x - y + 1)**2 * (x**2 * y + 1)**2\bound{e }}
\end{paste}\end{patch}

\begin{patch}{MakeFunctionXmpPagePatch5}
\begin{paste}{MakeFunctionXmpPageFull5}{MakeFunctionXmpPageEmpty5}
\pastebutton{MakeFunctionXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{function(e, g, [x, y])\free{e }}
\indentrel{3}\begin{verbatim}
   (5)  g
```

```
                                            Type: Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MakeFunctionXmpPageEmpty5}
\begin{paste}{MakeFunctionXmpPageEmpty5}{MakeFunctionXmpPagePatch5}
\pastebutton{MakeFunctionXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{function(e, g, [x, y])\free{e }}
\end{paste}\end{patch}

\begin{patch}{MakeFunctionXmpPagePatch6}
\begin{paste}{MakeFunctionXmpPageFull6}{MakeFunctionXmpPageEmpty6}
\pastebutton{MakeFunctionXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{function(e, h, x, y)\free{e }\bound{h }}
\indentrel{3}\begin{verbatim}
   (6)  h
                                            Type: Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MakeFunctionXmpPageEmpty6}
\begin{paste}{MakeFunctionXmpPageEmpty6}{MakeFunctionXmpPagePatch6}
\pastebutton{MakeFunctionXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{function(e, h, x, y)\free{e }\bound{h }}
\end{paste}\end{patch}

\begin{patch}{MakeFunctionXmpPagePatch7}
\begin{paste}{MakeFunctionXmpPageFull7}{MakeFunctionXmpPageEmpty7}
\pastebutton{MakeFunctionXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{m1 := squareMatrix [[1, 2], [3, 4]]\bound{m1 }}
\indentrel{3}\begin{verbatim}
        1  2
   (7)
        3  4
                          Type: SquareMatrix(2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MakeFunctionXmpPageEmpty7}
\begin{paste}{MakeFunctionXmpPageEmpty7}{MakeFunctionXmpPagePatch7}
\pastebutton{MakeFunctionXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{m1 := squareMatrix [[1, 2], [3, 4]]\bound{m1 }}
\end{paste}\end{patch}

\begin{patch}{MakeFunctionXmpPagePatch8}
\begin{paste}{MakeFunctionXmpPageFull8}{MakeFunctionXmpPageEmpty8}
\pastebutton{MakeFunctionXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{m2 := squareMatrix [[1, 0], [-1, 1]]\bound{m2 }}
\indentrel{3}\begin{verbatim}
         1   0
```

```
   (8)
        - 1  1
                          Type: SquareMatrix(2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MakeFunctionXmpPageEmpty8}
\begin{paste}{MakeFunctionXmpPageEmpty8}{MakeFunctionXmpPagePatch8}
\pastebutton{MakeFunctionXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{m2 := squareMatrix [[1, 0], [-1, 1]]\bound{m2 }}
\end{paste}\end{patch}

\begin{patch}{MakeFunctionXmpPagePatch9}
\begin{paste}{MakeFunctionXmpPageFull9}{MakeFunctionXmpPageEmpty9}
\pastebutton{MakeFunctionXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{h(m1, m2)\free{h m1 m2 }}
\indentrel{3}\begin{verbatim}
        - 7836   8960
   (9)
        - 17132  19588
                          Type: SquareMatrix(2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MakeFunctionXmpPageEmpty9}
\begin{paste}{MakeFunctionXmpPageEmpty9}{MakeFunctionXmpPagePatch9}
\pastebutton{MakeFunctionXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{h(m1, m2)\free{h m1 m2 }}
\end{paste}\end{patch}
```

——–

3.77 mpoly.ht

MultivariatePolynomial

⇒ "notitle" (PolynomialXmpPage) 3.88 on page 1110
⇒ "notitle" (UnivariatePolyXmpPage) 3.112 on page 1327
⇒ "notitle" (DistributedMultivariatePolyXmpPage) 3.24 on page 375

— **mpoly.ht** —

```
\begin{page}{MultivariatePolyXmpPage}{MultivariatePolynomial}
\beginscroll

The domain constructor \spadtype{MultivariatePolynomial} is similar to
```

```
\spadtype{Polynomial} except that it specifies the variables to be used.
\spadtype{Polynomial} are available for \spadtype{MultivariatePolynomial}.
The abbreviation for \spadtype{MultivariatePolynomial} is
\spadtype{MPOLY}.
The type expressions
\centerline{{\spadtype{MultivariatePolynomial([x,y],Integer)}}}
and
\centerline{{\spadtype{MPOLY([x,y],INT)}}}
refer to the domain of multivariate polynomials in the variables
\spad{x} and \spad{y} where the coefficients are restricted to be integers.
The first
variable specified is the main variable and the display of the
polynomial reflects this.
\xtc{
This polynomial appears with
terms in descending powers of the variable \spad{x}.
}{
\spadpaste{m : MPOLY([x,y],INT) := (x**2 - x*y**3 +3*y)**2 \bound{m}}
}
\xtc{
It is easy to see a different variable ordering by doing a conversion.
}{
\spadpaste{m :: MPOLY([y,x],INT) \free{m}}
}
\xtc{
You can use other, unspecified variables, by using
\spadtype{Polynomial} in the coefficient type of \spadtype{MPOLY}.
}{
\spadpaste{p : MPOLY([x,y],POLY INT) \bound{pdec}}
}
\xtc{
}{
\spadpaste{p := (a**2*x - b*y**2 + 1)**2 \free{pdec}\bound{p}}
}
\xtc{
Conversions can be used to re-express such polynomials in terms of
the other variables.  For example, you can first push all the
variables into a polynomial with integer coefficients.
}{
\spadpaste{p :: POLY INT \free{p}\bound{prev}}
}
\xtc{
Now pull out the variables of interest.
}{
\spadpaste{\% :: MPOLY([a,b],POLY INT) \free{prev}}
}

\beginImportant
\noindent {\bf Restriction:}
\texht{\begin{quotation}\noindent}{\newline\indent{5}}
```

```
Axiom does not allow you to create types where
\spadtype{MultivariatePolynomial} is contained in the coefficient type of
\spadtype{Polynomial}. Therefore,
\spad{MPOLY([x,y],POLY INT)} is legal but
\spad{POLY MPOLY([x,y],INT)} is not.
\texht{\end{quotation}}{\indent{0}}
\endImportant

\xtc{
Multivariate polynomials may be combined with univariate polynomials
to create types with special structures.
}{
\spadpaste{q : UP(x, FRAC MPOLY([y,z],INT)) \bound{qdec}}
}
\xtc{
This is a polynomial in \spad{x} whose coefficients are
quotients of polynomials in \spad{y} and \spad{z}.
}{
\spadpaste{q := (x**2 - x*(z+1)/y +2)**2 \free{qdec}\bound{q}}
}
\xtc{
Use conversions for structural rearrangements.
\spad{z} does not appear in a denominator and so it can be made
the main variable.
}{
\spadpaste{q :: UP(z, FRAC MPOLY([x,y],INT)) \free{q}}
}
\xtc{
Or you can make a multivariate polynomial in \spad{x} and \spad{z}
whose coefficients are fractions in polynomials in \spad{y}.
}{
\spadpaste{q :: MPOLY([x,z], FRAC UP(y,INT)) \free{q}}
}
A conversion like \spad{q :: MPOLY([x,y], FRAC UP(z,INT))} is not
possible in this example
because \spad{y} appears in the denominator of a fraction.
As you can see, Axiom provides extraordinary flexibility in the
manipulation and display of expressions via its conversion facility.

For more information on related topics, see
\downlink{`Polynomial'}{PolynomialXmpPage}\ignore{Polynomial},
\downlink{`UnivariatePolynomial'}{UnivariatePolyXmpPage}
\ignore{UnivariatePolynomial}, and
\downlink{`DistributedMultivariatePoly'}
{DistributedMultivariatePolyXmpPage}
\ignore{DistributedMultivariatePoly}.
\showBlurb{MultivariatePolynomial}
\endscroll
\autobuttons
\end{page}
```

```
\begin{patch}{MultivariatePolyXmpPagePatch1}
\begin{paste}{MultivariatePolyXmpPageFull1}{MultivariatePolyXmpPageEmpty1}
\pastebutton{MultivariatePolyXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{m : MPOLY([x,y],INT) := (x**2 - x*y**3 +3*y)**2\bound{m }}
\indentrel{3}\begin{verbatim}
        4     3 3     6       2     4      2
   (1)  x  - 2y x  + (y  + 6y)x  - 6y x + 9y
            Type: MultivariatePolynomial([x,y],Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MultivariatePolyXmpPageEmpty1}
\begin{paste}{MultivariatePolyXmpPageEmpty1}{MultivariatePolyXmpPagePatch1}
\pastebutton{MultivariatePolyXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{m : MPOLY([x,y],INT) := (x**2 - x*y**3 +3*y)**2\bound{m }}
\end{paste}\end{patch}

\begin{patch}{MultivariatePolyXmpPagePatch2}
\begin{paste}{MultivariatePolyXmpPageFull2}{MultivariatePolyXmpPageEmpty2}
\pastebutton{MultivariatePolyXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{m :: MPOLY([y,x],INT)\free{m }}
\indentrel{3}\begin{verbatim}
        2 6       4     3 3     2     2     4
   (2)  x y  - 6x y  - 2x y  + 9y  + 6x y + x
            Type: MultivariatePolynomial([y,x],Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MultivariatePolyXmpPageEmpty2}
\begin{paste}{MultivariatePolyXmpPageEmpty2}{MultivariatePolyXmpPagePatch2}
\pastebutton{MultivariatePolyXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{m :: MPOLY([y,x],INT)\free{m }}
\end{paste}\end{patch}

\begin{patch}{MultivariatePolyXmpPagePatch3}
\begin{paste}{MultivariatePolyXmpPageFull3}{MultivariatePolyXmpPageEmpty3}
\pastebutton{MultivariatePolyXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{p : MPOLY([x,y],POLY INT)\bound{pdec }}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MultivariatePolyXmpPageEmpty3}
\begin{paste}{MultivariatePolyXmpPageEmpty3}{MultivariatePolyXmpPagePatch3}
\pastebutton{MultivariatePolyXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{p : MPOLY([x,y],POLY INT)\bound{pdec }}
\end{paste}\end{patch}
```

```
\begin{patch}{MultivariatePolyXmpPagePatch4}
\begin{paste}{MultivariatePolyXmpPageFull4}{MultivariatePolyXmpPageEmpty4}
\pastebutton{MultivariatePolyXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{p := (a**2*x - b*y**2 + 1)**2\free{pdec }\bound{p }}
\indentrel{3}\begin{verbatim}
         4 2        2   2     2      2 4       2
   (4)  a x  + (- 2a b y  + 2a )x + b y  - 2b y  + 1
 Type: MultivariatePolynomial([x,y],Polynomial Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MultivariatePolyXmpPageEmpty4}
\begin{paste}{MultivariatePolyXmpPageEmpty4}{MultivariatePolyXmpPagePatch4}
\pastebutton{MultivariatePolyXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{p := (a**2*x - b*y**2 + 1)**2\free{pdec }\bound{p }}
\end{paste}\end{patch}

\begin{patch}{MultivariatePolyXmpPagePatch5}
\begin{paste}{MultivariatePolyXmpPageFull5}{MultivariatePolyXmpPageEmpty5}
\pastebutton{MultivariatePolyXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{p :: POLY INT\free{p }\bound{prev }}
\indentrel{3}\begin{verbatim}
         2 4        2          2    4 2     2
   (5)  b y  + (- 2a b x - 2b)y  + a x  + 2a x + 1
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MultivariatePolyXmpPageEmpty5}
\begin{paste}{MultivariatePolyXmpPageEmpty5}{MultivariatePolyXmpPagePatch5}
\pastebutton{MultivariatePolyXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{p :: POLY INT\free{p }\bound{prev }}
\end{paste}\end{patch}

\begin{patch}{MultivariatePolyXmpPagePatch6}
\begin{paste}{MultivariatePolyXmpPageFull6}{MultivariatePolyXmpPageEmpty6}
\pastebutton{MultivariatePolyXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{\% :: MPOLY([a,b],POLY INT)\free{prev }}
\indentrel{3}\begin{verbatim}
         2 4          2        2    4 2     2
   (6)  x a  + (- 2x y b + 2x)a  + y b  - 2y b + 1
 Type: MultivariatePolynomial([a,b],Polynomial Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MultivariatePolyXmpPageEmpty6}
\begin{paste}{MultivariatePolyXmpPageEmpty6}{MultivariatePolyXmpPagePatch6}
\pastebutton{MultivariatePolyXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{\% :: MPOLY([a,b],POLY INT)\free{prev }}
\end{paste}\end{patch}
```

```
\begin{patch}{MultivariatePolyXmpPagePatch7}
\begin{paste}{MultivariatePolyXmpPageFull7}{MultivariatePolyXmpPageEmpty7}
\pastebutton{MultivariatePolyXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{q : UP(x, FRAC MPOLY([y,z],INT))\bound{qdec }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MultivariatePolyXmpPageEmpty7}
\begin{paste}{MultivariatePolyXmpPageEmpty7}{MultivariatePolyXmpPagePatch7}
\pastebutton{MultivariatePolyXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{q : UP(x, FRAC MPOLY([y,z],INT))\bound{qdec }}
\end{paste}\end{patch}

\begin{patch}{MultivariatePolyXmpPagePatch8}
\begin{paste}{MultivariatePolyXmpPageFull8}{MultivariatePolyXmpPageEmpty8}
\pastebutton{MultivariatePolyXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{q := (x**2 - x*(z+1)/y +2)**2\free{qdec }\bound{q }}
\indentrel{3}\begin{verbatim}
   (8)
                         2    2
      4   - 2z - 2  3   4y  + z  + 2z + 1  2   - 4z - 4
     x  +  x  +  x  +  x
              y                 2                 y
                               y
   +
     4
Type: UnivariatePolynomial(x,Fraction MultivariatePolynomial([y,z],Integer))
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MultivariatePolyXmpPageEmpty8}
\begin{paste}{MultivariatePolyXmpPageEmpty8}{MultivariatePolyXmpPagePatch8}
\pastebutton{MultivariatePolyXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{q := (x**2 - x*(z+1)/y +2)**2\free{qdec }\bound{q }}
\end{paste}\end{patch}

\begin{patch}{MultivariatePolyXmpPagePatch9}
\begin{paste}{MultivariatePolyXmpPageFull9}{MultivariatePolyXmpPageEmpty9}
\pastebutton{MultivariatePolyXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{q :: UP(z, FRAC MPOLY([x,y],INT))\free{q }}
\indentrel{3}\begin{verbatim}
   (9)
      2            3     2
     x   2   - 2y x  + 2x  - 4y x
      z  +  z
      2             2
     y             y
```

```
   +
      2 4       3      2      2            2
     y x  - 2y x  + (4y  + 1)x  - 4y x + 4y

                          2
                         y
Type: UnivariatePolynomial(z,Fraction MultivariatePolynomial([x,y],Integer))
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MultivariatePolyXmpPageEmpty9}
\begin{paste}{MultivariatePolyXmpPageEmpty9}{MultivariatePolyXmpPagePatch9}
\pastebutton{MultivariatePolyXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{q :: UP(z, FRAC MPOLY([x,y],INT))\free{q }}
\end{paste}\end{patch}

\begin{patch}{MultivariatePolyXmpPagePatch10}
\begin{paste}{MultivariatePolyXmpPageFull10}{MultivariatePolyXmpPageEmpty10}
\pastebutton{MultivariatePolyXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{q :: MPOLY([x,z], FRAC UP(y,INT))\free{q }}
\indentrel{3}\begin{verbatim}
   (10)
                                            2
      4      2     2  3     1  2    2     4y  + 1  2
     x  + (-  z - )x  + ( z  +  z + )x
             y     y        2       2         2
                           y       y         y
   +
        4     4
     (-  z - )x + 4
        y     y
Type: MultivariatePolynomial([x,z],Fraction UnivariatePolynomial(y,Integer))
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{MultivariatePolyXmpPageEmpty10}
\begin{paste}{MultivariatePolyXmpPageEmpty10}{MultivariatePolyXmpPagePatch10}
\pastebutton{MultivariatePolyXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{q :: MPOLY([x,z], FRAC UP(y,INT))\free{q }}
\end{paste}\end{patch}
```

———-

3.78 newuser.ht

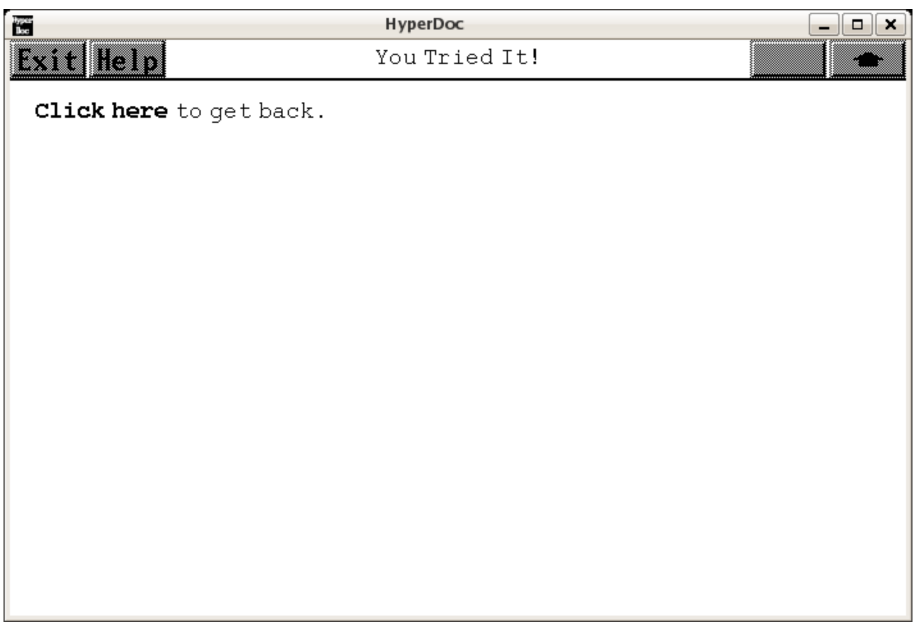
No More Help :-(

— newuser.ht —

```
\begin{page}{NoMoreHelpPage}{No More Help :-(}
\beginscroll\vspace{2}
\centerline{No additional or specific help information is available.}
\centerline{Click on \ \ExitButton{QuitPage} \  to get back.}
\endscroll
\end{page}
```

———-

You Tried It!



⇐ "Reference" (TopReferencePage) 3.1 on page 123

— newuser.ht —

```
\begin{page}{YouTriedIt}{You Tried It!}
\beginscroll
\upbutton{Click here}{UpPage} to get back.
\endscroll
```

```
\end{page}
```

———-

3.79 none.ht

None

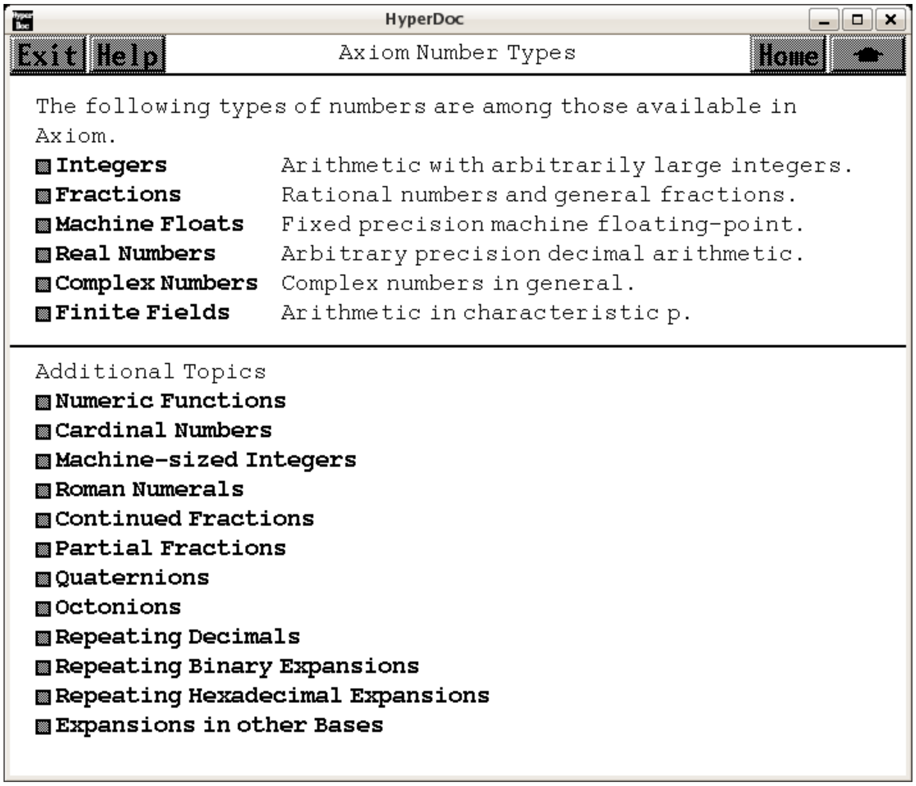
— none.ht —

```
\begin{page}{NoneXmpPage}{None}
\beginscroll
The \spadtype{None} domain is not very useful for interactive
work but it is provided nevertheless for completeness of the
Axiom type system.
\xtc{
Probably the only place you will ever see it is if you enter an
empty list with no type information.
}{
\spadpaste{[]}
}
\xtc{
Such an empty list can be converted into an empty list
of any other type.
}{
\spadpaste{[] :: List Float}
}
\xtc{
If you wish to produce an empty list of a particular
type directly, such as \spadtype{List NonNegativeInteger}, do it this way.
}{
\spadpaste{[]\$List(NonNegativeInteger)}
}
\endscroll
\autobuttons
\end{page}

\begin{patch}{NoneXmpPagePatch1}
\begin{paste}{NoneXmpPageFull1}{NoneXmpPageEmpty1}
\pastebutton{NoneXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{[]}
\indentrel{3}\begin{verbatim}
   (1)  []
                                        Type: List None
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{NoneXmpPageEmpty1}
\begin{paste}{NoneXmpPageEmpty1}{NoneXmpPagePatch1}
\pastebutton{NoneXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{[]}
\end{paste}\end{patch}

\begin{patch}{NoneXmpPagePatch2}
\begin{paste}{NoneXmpPageFull2}{NoneXmpPageEmpty2}
\pastebutton{NoneXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{[] :: List Float}
\indentrel{3}\begin{verbatim}
   (2)  []
                                       Type: List Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{NoneXmpPageEmpty2}
\begin{paste}{NoneXmpPageEmpty2}{NoneXmpPagePatch2}
\pastebutton{NoneXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{[] :: List Float}
\end{paste}\end{patch}

\begin{patch}{NoneXmpPagePatch3}
\begin{paste}{NoneXmpPageFull3}{NoneXmpPageEmpty3}
\pastebutton{NoneXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{[]$List(NonNegativeInteger)}
\indentrel{3}\begin{verbatim}
   (3)  []
                          Type: List NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{NoneXmpPageEmpty3}
\begin{paste}{NoneXmpPageEmpty3}{NoneXmpPagePatch3}
\pastebutton{NoneXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{[]$List(NonNegativeInteger)}
\end{paste}\end{patch}
```

3.80 numbers.ht

Axiom Number Types



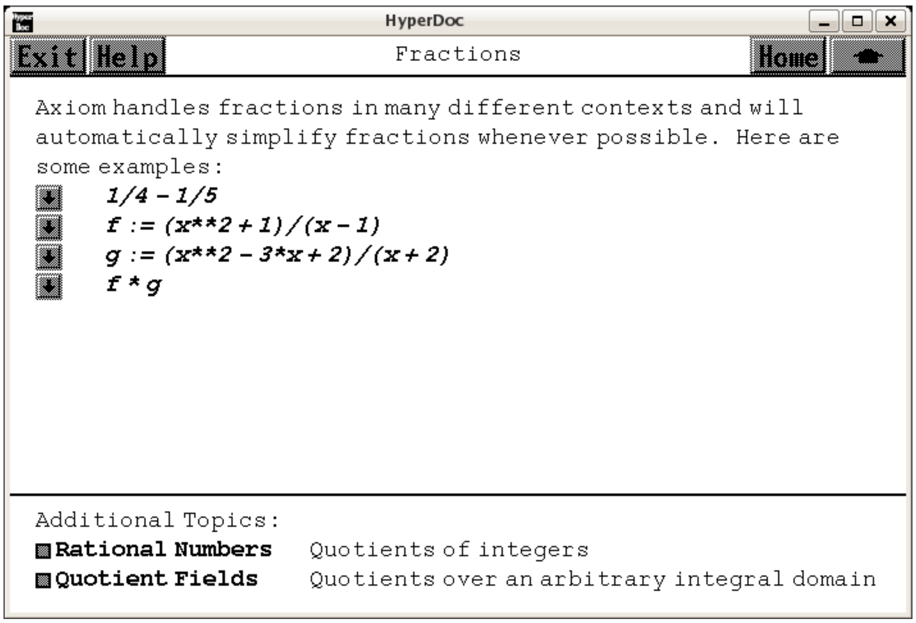
⇐ “Axiom Topics” (TopicPage) 3.108 on page 1313
⇒ “Integers” (IntegerPage) 3.80 on page 1029
⇒ “Fractions” (FractionPage) 3.80 on page 1023
⇒ “Machine Floats” (DoubleFloatXmpPage) 3.23 on page 369
⇒ “Real Numbers” (FloatXmpPage) 3.41 on page 487
⇒ “Complex Numbers” (ComplexXmpPage) 3.16 on page 254
⇒ “Finite Fields” (ugProblemFinitePage) 12 on page 2235
⇒ “Numeric Functions” (ugProblemNumericPage) 12 on page 2079
⇒ “Cardinal Numbers” (CardinalNumberXmpPage) 3.12 on page 185
⇒ “Machine-sized Integers” (SingleIntegerXmpPage) 3.98 on page 1237
⇒ “Roman Numerals” (RomanNumeralXmpPage) 3.94 on page 1213
⇒ “Continued Fractions” (ContinuedFractionXmpPage) 3.17 on page 262
⇒ “Partial Fractions” (PartialFractionXmpPage) 3.86 on page 1088
⇒ “Quaternions” (QuaternionXmpPage) 3.89 on page 1134
⇒ “Octonions” (OctonionXmpPage) 3.81 on page 1044
⇒ “Repeating Decimals” (DecimalExpansionXmpPage) 3.21 on page 348

⇒ "Repeating Binary Expansions" (BinaryExpansionXmpPage) 3.8 on page 163
⇒ "Repeating Hexadecimal Expansions" (HexExpansionXmpPage) 3.54 on page 692
⇒ "Expansions in other Bases" (RadixExpansionXmpPage) 3.90 on page 1140

— **numbers.ht** —

```
\begin{page}{NumberPage}{Axiom Number Types}
\beginscroll
The following types of numbers are among those available in Axiom.
\beginmenu

\menulink{Integers}{IntegerPage}\tab{16}
Arithmetic with arbitrarily large integers.

\menulink{Fractions}{FractionPage} \tab{16}
Rational numbers and general fractions.

\menulink{Machine Floats}{DoubleFloatXmpPage} \tab{16}
Fixed precision machine floating-point.

\menulink{Real Numbers}{FloatXmpPage} \tab{16}
Arbitrary precision decimal arithmetic.

\menulink{Complex Numbers}{ComplexXmpPage} \tab{16}
Complex numbers in general.

\menulink{Finite Fields}{ugProblemFinitePage} \tab{16}
Arithmetic in characteristic \spad{p}.
\endmenu
\horizontalline\newline
Additional Topics
\beginmenu

\menulink{Numeric Functions}{ugProblemNumericPage}
\menulink{Cardinal Numbers}{CardinalNumberXmpPage}
\menulink{Machine-sized Integers}{SingleIntegerXmpPage}
\menulink{Roman Numerals}{RomanNumeralXmpPage}
\menulink{Continued Fractions}{ContinuedFractionXmpPage}
\menulink{Partial Fractions}{PartialFractionXmpPage}
\menulink{Quaternions}{QuaternionXmpPage}
\menulink{Octonions}{OctonionXmpPage}
\menulink{Repeating Decimals}{DecimalExpansionXmpPage}
\menulink{Repeating Binary Expansions}{BinaryExpansionXmpPage}
\menulink{Repeating Hexadecimal Expansions}{HexExpansionXmpPage}
\menulink{Expansions in other Bases}{RadixExpansionXmpPage}

\endmenu

\endscroll
\autobuttons
\end{page}
```

Fraction



⇐ "Axiom Number Types" (NumberPage) 3.80 on page 1021
⇒ "Rational Numbers" (RationalNumberPage) 3.80 on page 1025
⇒ "Quotient Fields" (FractionXmpPage) 3.45 on page 543

— **numbers.ht** —

```
\begin{page}{FractionPage}{Fractions}
\beginscroll
Axiom handles fractions in many different contexts
and will automatically simplify fractions whenever possible.
Here are some examples:
\spadpaste{1/4 - 1/5}
\spadpaste{f := (x**2 + 1)/(x - 1) \bound{f}}
\spadpaste{g := (x**2 - 3*x + 2)/(x + 2) \bound{g}}
\spadpaste{f * g \free{f g}}
\endscroll
Additional Topics:
\beginmenu
\menulink{Rational Numbers}{RationalNumberPage} \tab{18}
Quotients of integers
\menulink{Quotient Fields}{FractionXmpPage}  \tab{18}
```

```
Quotients over an arbitrary integral domain
\endmenu
\autobuttons
\end{page}

\begin{patch}{FractionPagePatch1}
\begin{paste}{FractionPageFull1}{FractionPageEmpty1}
\pastebutton{FractionPageFull1}{\hidepaste}
\tab{5}\spadcommand{1/4 - 1/5}
\indentrel{3}\begin{verbatim}
         1
   (1)
        20
                                 Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FractionPageEmpty1}
\begin{paste}{FractionPageEmpty1}{FractionPagePatch1}
\pastebutton{FractionPageEmpty1}{\showpaste}
\tab{5}\spadcommand{1/4 - 1/5}
\end{paste}\end{patch}

\begin{patch}{FractionPagePatch2}
\begin{paste}{FractionPageFull2}{FractionPageEmpty2}
\pastebutton{FractionPageFull2}{\hidepaste}
\tab{5}\spadcommand{f := (x**2 + 1)/(x - 1)\bound{f }}
\indentrel{3}\begin{verbatim}
         2
        x  + 1
   (2)
         x - 1
                      Type: Fraction Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FractionPageEmpty2}
\begin{paste}{FractionPageEmpty2}{FractionPagePatch2}
\pastebutton{FractionPageEmpty2}{\showpaste}
\tab{5}\spadcommand{f := (x**2 + 1)/(x - 1)\bound{f }}
\end{paste}\end{patch}

\begin{patch}{FractionPagePatch3}
\begin{paste}{FractionPageFull3}{FractionPageEmpty3}
\pastebutton{FractionPageFull3}{\hidepaste}
\tab{5}\spadcommand{g := (x**2 - 3*x + 2)/(x + 2)\bound{g }}
\indentrel{3}\begin{verbatim}
         2
        x  - 3x + 2
   (3)
```

```
          x + 2
                     Type: Fraction Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FractionPageEmpty3}
\begin{paste}{FractionPageEmpty3}{FractionPagePatch3}
\pastebutton{FractionPageEmpty3}{\showpaste}
\tab{5}\spadcommand{g := (x**2 - 3*x + 2)/(x + 2)\bound{g }}
\end{paste}\end{patch}

\begin{patch}{FractionPagePatch4}
\begin{paste}{FractionPageFull4}{FractionPageEmpty4}
\pastebutton{FractionPageFull4}{\hidepaste}
\tab{5}\spadcommand{f * g\free{f g }}
\indentrel{3}\begin{verbatim}
         3     2
        x  - 2x  + x - 2
   (4)
             x + 2
                     Type: Fraction Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{FractionPageEmpty4}
\begin{paste}{FractionPageEmpty4}{FractionPagePatch4}
\pastebutton{FractionPageEmpty4}{\showpaste}
\tab{5}\spadcommand{f * g\free{f g }}
\end{paste}\end{patch}
```

———-

Rational Number

— numbers.ht —

```
\begin{page}{RationalNumberPage}{Rational Numbers}
\beginscroll
Like integers, rational numbers can be arbitrarily large.
For example:
\spadpaste{61657 ** 10 / 999983 ** 12}
Rational numbers will not be converted to decimals unless you explicitly
ask Axiom to do so.
To convert a rational number to a decimal, use the function
\spadfun{numeric}.
Here's an example:
\spadpaste{x := 104348/33215 \bound{x}}
```

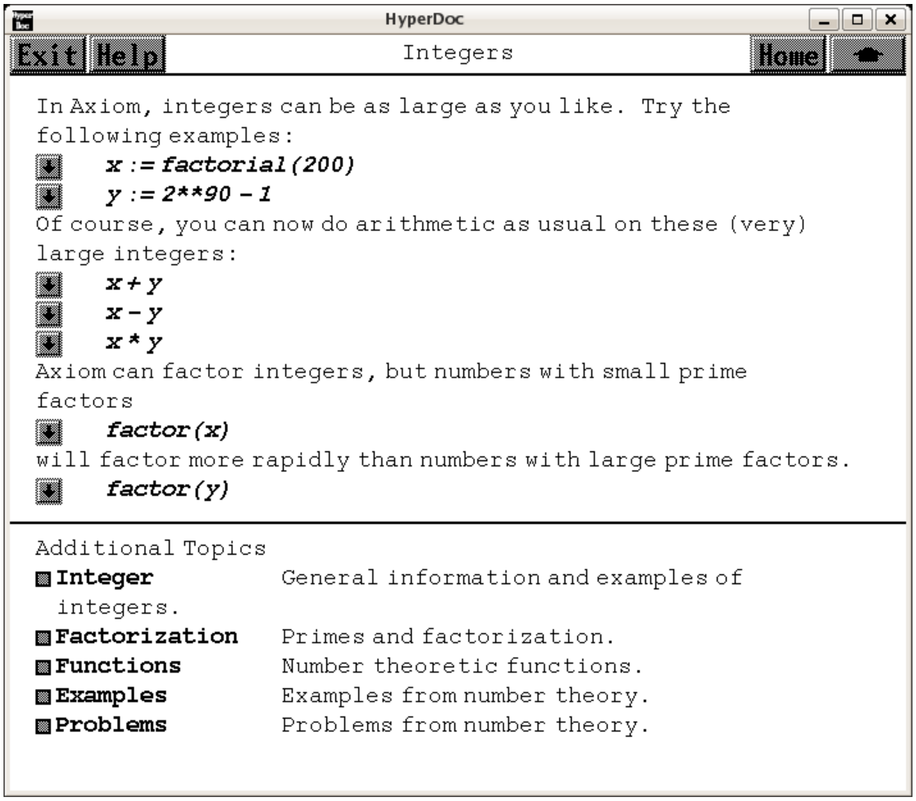
```
\spadpaste{numeric x \free{x}}
You can find the numerator and denominator of rational numbers using
the functions \spadfun{numer} and \spadfun{denom}, respectively.
\spadpaste{numer(x) \free{x}}
\spadpaste{denom(x) \free{x}}
To factor the numerator and denominator of a fraction, use the following
command:
\spadpaste{factor(numer x) / factor(denom x) \free{x}}
\endscroll
\autobuttons
\end{page}

\begin{patch}{RationalNumberPagePatch1}
\begin{paste}{RationalNumberPageFull1}{RationalNumberPageEmpty1}
\pastebutton{RationalNumberPageFull1}{\hidepaste}
\tab{5}\spadcommand{61657 ** 10 / 999983 ** 12}
\indentrel{3}\begin{verbatim}
   (1)
     794006207119672937688869745365148806136551203249
  /
    999796019072919181341770495558788771223957844095225 8_
     46167460930641229761
                                  Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RationalNumberPageEmpty1}
\begin{paste}{RationalNumberPageEmpty1}{RationalNumberPagePatch1}
\pastebutton{RationalNumberPageEmpty1}{\showpaste}
\tab{5}\spadcommand{61657 ** 10 / 999983 ** 12}
\end{paste}\end{patch}

\begin{patch}{RationalNumberPagePatch2}
\begin{paste}{RationalNumberPageFull2}{RationalNumberPageEmpty2}
\pastebutton{RationalNumberPageFull2}{\hidepaste}
\tab{5}\spadcommand{x := 104348/33215\bound{x }}
\indentrel{3}\begin{verbatim}
        104348
   (2)
         33215
                                  Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RationalNumberPageEmpty2}
\begin{paste}{RationalNumberPageEmpty2}{RationalNumberPagePatch2}
\pastebutton{RationalNumberPageEmpty2}{\showpaste}
\tab{5}\spadcommand{x := 104348/33215\bound{x }}
\end{paste}\end{patch}
```

```
\begin{patch}{RationalNumberPagePatch3}
\begin{paste}{RationalNumberPageFull3}{RationalNumberPageEmpty3}
\pastebutton{RationalNumberPageFull3}{\hidepaste}
\tab{5}\spadcommand{numeric x\free{x }}
\indentrel{3}\begin{verbatim}
   (3)  3.1415926539 214210447
                                            Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RationalNumberPageEmpty3}
\begin{paste}{RationalNumberPageEmpty3}{RationalNumberPagePatch3}
\pastebutton{RationalNumberPageEmpty3}{\showpaste}
\tab{5}\spadcommand{numeric x\free{x }}
\end{paste}\end{patch}

\begin{patch}{RationalNumberPagePatch4}
\begin{paste}{RationalNumberPageFull4}{RationalNumberPageEmpty4}
\pastebutton{RationalNumberPageFull4}{\hidepaste}
\tab{5}\spadcommand{numer(x)\free{x }}
\indentrel{3}\begin{verbatim}
   (4)  104348
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RationalNumberPageEmpty4}
\begin{paste}{RationalNumberPageEmpty4}{RationalNumberPagePatch4}
\pastebutton{RationalNumberPageEmpty4}{\showpaste}
\tab{5}\spadcommand{numer(x)\free{x }}
\end{paste}\end{patch}

\begin{patch}{RationalNumberPagePatch5}
\begin{paste}{RationalNumberPageFull5}{RationalNumberPageEmpty5}
\pastebutton{RationalNumberPageFull5}{\hidepaste}
\tab{5}\spadcommand{denom(x)\free{x }}
\indentrel{3}\begin{verbatim}
   (5)  33215
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RationalNumberPageEmpty5}
\begin{paste}{RationalNumberPageEmpty5}{RationalNumberPagePatch5}
\pastebutton{RationalNumberPageEmpty5}{\showpaste}
\tab{5}\spadcommand{denom(x)\free{x }}
\end{paste}\end{patch}

\begin{patch}{RationalNumberPagePatch6}
\begin{paste}{RationalNumberPageFull6}{RationalNumberPageEmpty6}
```

```
\pastebutton{RationalNumberPageFull6}{\hidepaste}
\tab{5}\spadcommand{factor(numer x) / factor(denom x)\free{x }}
\indentrel{3}\begin{verbatim}
         2
        2 19 1373
   (6)
        5 7 13 73
                        Type: Fraction Factored Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RationalNumberPageEmpty6}
\begin{paste}{RationalNumberPageEmpty6}{RationalNumberPagePatch6}
\pastebutton{RationalNumberPageEmpty6}{\showpaste}
\tab{5}\spadcommand{factor(numer x) / factor(denom x)\free{x }}
\end{paste}\end{patch}
```

———-

Integers



⇐ “Axiom Number Types” (NumberPage) 3.80 on page 1021
⇒ “Integer” (IntegerXmpPage) 3.55 on page 696
⇒ “Factorization” (ugxIntegerPrimesPage) 3.55 on page 712
⇒ “Functions” (IntNumberTheoryFnsXmpPage) 3.56 on page 722
⇒ “Examples” (IntegerExamplePage) 3.80 on page 1034
⇒ “Problems” (IntegerProblemPage) 3.80 on page 1037

— **numbers.ht** —

```
\begin{page}{IntegerPage}{Integers}
\beginscroll
In Axiom, integers can be as large as you like.
Try the following examples:
\spadpaste{x := factorial(200) \bound{x}}
\spadpaste{y := 2**90 - 1 \bound{y}}
Of course, you can now do arithmetic as usual on these (very)
large integers:
\spadpaste{x + y \free{x y}}
\spadpaste{x - y \free{x y}}
\spadpaste{x * y \free{x y}}
```

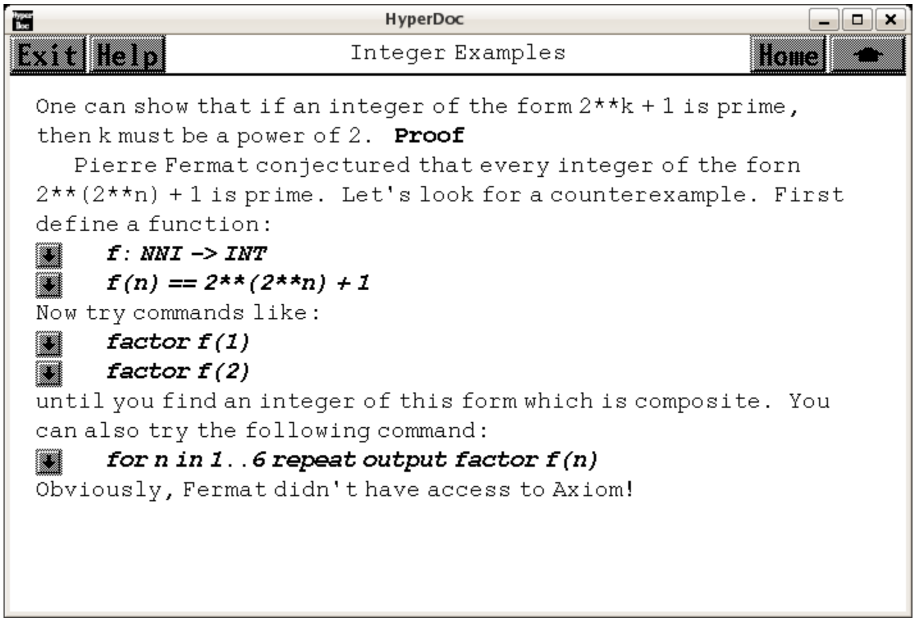
```
Axiom can factor integers, but numbers with small prime factors
\spadpaste{factor(x) \free{x}}
will factor more rapidly than numbers with large prime factors.
\spadpaste{factor(y) \free{y}}
\horizontalline
Additional Topics
\beginmenu

\menulink{Integer}{IntegerXmpPage} \tab{16}
General information and examples of integers.

\menulink{Factorization}{ugxIntegerPrimesPage} \tab{16}
Primes and factorization.

\menulink{Functions}{IntNumberTheoryFnsXmpPage} \tab{16}
Number theoretic functions.

\menulink{Examples}{IntegerExamplePage} \tab{16}
Examples from number theory.

\menulink{Problems}{IntegerProblemPage} \tab{16}
Problems from number theory.

\endmenu
\endscroll
\autobuttons
\end{page}

\begin{patch}{IntegerPagePatch1}
\begin{paste}{IntegerPageFull1}{IntegerPageEmpty1}
\pastebutton{IntegerPageFull1}{\hidepaste}
\tab{5}\spadcommand{x := factorial(200)\bound{x }}
\indentrel{3}\begin{verbatim}
   (1)
  788657867364790503552363213932185062295135977687173263_
   294742533244359449963403342920304284011984623904177212_
   213891963883025764279024263710506192662495282993111346_
   628572707633172373969889439224456214516642402540332918_
   864131227428294853277524242407573903240321257405579568_
   866022603190417032406235170085879617892222278962370389_
   737472000000000000000000000000000000000000000000000000_
   000
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntegerPageEmpty1}
\begin{paste}{IntegerPageEmpty1}{IntegerPagePatch1}
\pastebutton{IntegerPageEmpty1}{\showpaste}
\tab{5}\spadcommand{x := factorial(200)\bound{x }}
```

```
\end{paste}\end{patch}

\begin{patch}{IntegerPagePatch2}
\begin{paste}{IntegerPageFull2}{IntegerPageEmpty2}
\pastebutton{IntegerPageFull2}{\hidepaste}
\tab{5}\spadcommand{y := 2**90 - 1\bound{y }}
\indentrel{3}\begin{verbatim}
   (2)  1237940039285380274899124223
                                Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntegerPageEmpty2}
\begin{paste}{IntegerPageEmpty2}{IntegerPagePatch2}
\pastebutton{IntegerPageEmpty2}{\showpaste}
\tab{5}\spadcommand{y := 2**90 - 1\bound{y }}
\end{paste}\end{patch}

\begin{patch}{IntegerPagePatch3}
\begin{paste}{IntegerPageFull3}{IntegerPageEmpty3}
\pastebutton{IntegerPageFull3}{\hidepaste}
\tab{5}\spadcommand{x + y\free{x y }}
\indentrel{3}\begin{verbatim}
   (3)
  7886578673647905035523632139321850622951359776871732_
   2947425332443594499634033429203042840119846239041772_
   2138919638830257642790242637105061926624952829931113_
   6285727076331723739698894392244562145166424025403329_
   8641312274282948532775242424075739032403212574055795_
   8660226031904170324062351700858796178922222789623703_
   9737472000000000000000000001237940039285380274899124_
   223
                                Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntegerPageEmpty3}
\begin{paste}{IntegerPageEmpty3}{IntegerPagePatch3}
\pastebutton{IntegerPageEmpty3}{\showpaste}
\tab{5}\spadcommand{x + y\free{x y }}
\end{paste}\end{patch}

\begin{patch}{IntegerPagePatch4}
\begin{paste}{IntegerPageFull4}{IntegerPageEmpty4}
\pastebutton{IntegerPageFull4}{\hidepaste}
\tab{5}\spadcommand{x - y\free{x y }}
\indentrel{3}\begin{verbatim}
   (4)
  7886578673647905035523632139321850622951359776871732_
   2947425332443594499634033429203042840119846239041772_
```

```
   21389196388302576427902426371050619266249528299311134_
   62857270763317237396988943922445621451664240254033291_
   86413122742829485327752424240757390324032125740557956_
   86602260319041703240623517008587961789222227896237038_
   9737471999999999999999999999998762059960714619725100875_
   777
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntegerPageEmpty4}
\begin{paste}{IntegerPageEmpty4}{IntegerPagePatch4}
\pastebutton{IntegerPageEmpty4}{\showpaste}
\tab{5}\spadcommand{x - y\free{x y }}
\end{paste}\end{patch}

\begin{patch}{IntegerPagePatch5}
\begin{paste}{IntegerPageFull5}{IntegerPageEmpty5}
\pastebutton{IntegerPageFull5}{\hidepaste}
\tab{5}\spadcommand{x * y\free{x y }}
\indentrel{3}\begin{verbatim}
   (5)
  97631115130829298218436311966095022577664296676541404 2_
   37079496488133389834070329188092355479782812765687260_
   17975734913119466356078732929100728088106228471338396_
   75509315106953260921744797014165125163884859138819053_
   52475858689630194698878995048210905618067437176553811_
   33973032509524956986554360537566475497856969235918273_
   09521182392695050703382396859842560000000000000000000_
   000000000000000000000000000000
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntegerPageEmpty5}
\begin{paste}{IntegerPageEmpty5}{IntegerPagePatch5}
\pastebutton{IntegerPageEmpty5}{\showpaste}
\tab{5}\spadcommand{x * y\free{x y }}
\end{paste}\end{patch}

\begin{patch}{IntegerPagePatch6}
\begin{paste}{IntegerPageFull6}{IntegerPageEmpty6}
\pastebutton{IntegerPageFull6}{\hidepaste}
\tab{5}\spadcommand{factor(x)\free{x }}
\indentrel{3}\begin{verbatim}
   (6)
      197 97 49 32  19  16  11  10  8  6  6  5  4  4  4  3
     2   3  5  7  11  13  17  19  23 29 31 37 41 43 47 53
  *
       3  3  2  2  2  2  2  2  2
```

```
     59 61 67 71 73 79 83 89 97 101 103 107 109 113 127
  *
     131 137 139 149 151 157 163 167 173 179 181 191 193
  *
     197 199
                                 Type: Factored Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntegerPageEmpty6}
\begin{paste}{IntegerPageEmpty6}{IntegerPagePatch6}
\pastebutton{IntegerPageEmpty6}{\showpaste}
\tab{5}\spadcommand{factor(x)\free{x }}
\end{paste}\end{patch}

\begin{patch}{IntegerPagePatch7}
\begin{paste}{IntegerPageFull7}{IntegerPageEmpty7}
\pastebutton{IntegerPageFull7}{\hidepaste}
\tab{5}\spadcommand{factor(y)\free{y }}
\indentrel{3}\begin{verbatim}
        3
   (7)  3 7 11 19 31 73 151 331 631 23311 18837001
                                 Type: Factored Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntegerPageEmpty7}
\begin{paste}{IntegerPageEmpty7}{IntegerPagePatch7}
\pastebutton{IntegerPageEmpty7}{\showpaste}
\tab{5}\spadcommand{factor(y)\free{y }}
\end{paste}\end{patch}
```

Integer Examples



⇐ "Integers" (IntegerPage) 3.80 on page 1029
⇒ "Proof" (IntegerExampleProofPage) 3.80 on page 1036

— **numbers.ht** —

```
\begin{page}{IntegerExamplePage}{Integer Examples}
\beginscroll
One can show that if an integer of the form 2**k + 1 is prime, then
k must be a power of 2.
\downlink{Proof}{IntegerExampleProofPage}
\par
Pierre Fermat conjectured that every integer of the forn 2**(2**n) + 1
is prime.
Let's look for a counterexample.
First define a function:
\spadpaste{f: NNI -> INT \bound{f1}}
\spadpaste{f(n) == 2**(2**n) + 1 \bound{f} \free{f1}}
Now try commands like:
\spadpaste{factor f(1) \free{f}}
\spadpaste{factor f(2) \free{f}}
until you find an integer of this form which is composite.
You can also try the following command:
\spadpaste{for n in 1..6 repeat output factor f(n) \free{f}}
Obviously, Fermat didn't have access to Axiom!
\endscroll
\autobuttons
```

```
\end{page}

\begin{patch}{IntegerExamplePagePatch1}
\begin{paste}{IntegerExamplePageFull1}{IntegerExamplePageEmpty1}
\pastebutton{IntegerExamplePageFull1}{\hidepaste}
\tab{5}\spadcommand{f: NNI -> INT\bound{f1 }}
\indentrel{3}\begin{verbatim}
                                              Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntegerExamplePageEmpty1}
\begin{paste}{IntegerExamplePageEmpty1}{IntegerExamplePagePatch1}
\pastebutton{IntegerExamplePageEmpty1}{\showpaste}
\tab{5}\spadcommand{f: NNI -> INT\bound{f1 }}
\end{paste}\end{patch}

\begin{patch}{IntegerExamplePagePatch2}
\begin{paste}{IntegerExamplePageFull2}{IntegerExamplePageEmpty2}
\pastebutton{IntegerExamplePageFull2}{\hidepaste}
\tab{5}\spadcommand{f(n) == 2**(2**n) + 1\bound{f }\free{f1 }}
\indentrel{3}\begin{verbatim}
                                              Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntegerExamplePageEmpty2}
\begin{paste}{IntegerExamplePageEmpty2}{IntegerExamplePagePatch2}
\pastebutton{IntegerExamplePageEmpty2}{\showpaste}
\tab{5}\spadcommand{f(n) == 2**(2**n) + 1\bound{f }\free{f1 }}
\end{paste}\end{patch}

\begin{patch}{IntegerExamplePagePatch3}
\begin{paste}{IntegerExamplePageFull3}{IntegerExamplePageEmpty3}
\pastebutton{IntegerExamplePageFull3}{\hidepaste}
\tab{5}\spadcommand{factor f(1)\free{f }}
\indentrel{3}\begin{verbatim}
   (3)  5
                                  Type: Factored Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntegerExamplePageEmpty3}
\begin{paste}{IntegerExamplePageEmpty3}{IntegerExamplePagePatch3}
\pastebutton{IntegerExamplePageEmpty3}{\showpaste}
\tab{5}\spadcommand{factor f(1)\free{f }}
\end{paste}\end{patch}

\begin{patch}{IntegerExamplePagePatch4}
\begin{paste}{IntegerExamplePageFull4}{IntegerExamplePageEmpty4}
```

```
\pastebutton{IntegerExamplePageFull4}{\hidepaste}
\tab{5}\spadcommand{factor f(2)\free{f }}
\indentrel{3}\begin{verbatim}
   (4)  17
                                 Type: Factored Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntegerExamplePageEmpty4}
\begin{paste}{IntegerExamplePageEmpty4}{IntegerExamplePagePatch4}
\pastebutton{IntegerExamplePageEmpty4}{\showpaste}
\tab{5}\spadcommand{factor f(2)\free{f }}
\end{paste}\end{patch}

\begin{patch}{IntegerExamplePagePatch5}
\begin{paste}{IntegerExamplePageFull5}{IntegerExamplePageEmpty5}
\pastebutton{IntegerExamplePageFull5}{\hidepaste}
\tab{5}\spadcommand{for n in 1..6 repeat output factor f(n)\free{f }}
\indentrel{3}\begin{verbatim}
   5
   17
   257
   65537
   641 6700417
   274177 67280421310721
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntegerExamplePageEmpty5}
\begin{paste}{IntegerExamplePageEmpty5}{IntegerExamplePagePatch5}
\pastebutton{IntegerExamplePageEmpty5}{\showpaste}
\tab{5}\spadcommand{for n in 1..6 repeat output factor f(n)\free{f }}
\end{paste}\end{patch}
```

————

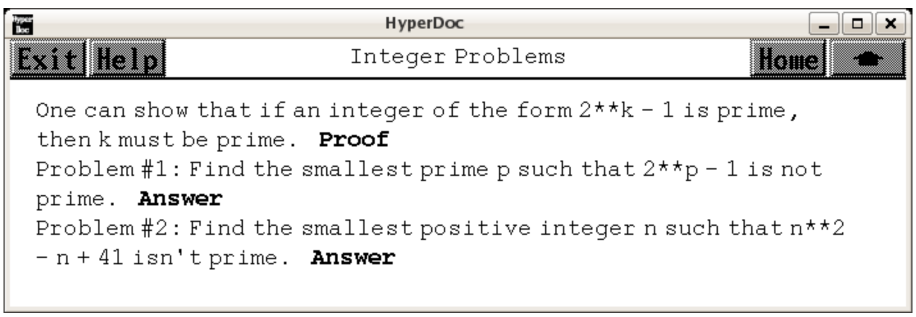
Integer Example Proof

— **numbers.ht** —

```
\begin{page}{IntegerExampleProofPage}{Integer Example Proof}
\beginscroll
Proposition.  If 2**k + 1 is prime, then k is a power of 2.
\newline
Proof.  Suppose that k = m * n with m > 1 odd.  Then
%
```

```
\centerline{2**n = -1 (mod (2**n + 1))}
\centerline{2**(n * m) = (-1)**m = -1 (mod (2**n + 1))}
\centerline{2**k + 1 = 0 (mod (2**n + 1))}
%
Therefore, 2**k + 1 is divisible by 2**n + 1.
Now 1 < 2**n + 1 and since m > 1, 2**n + 1 < 2**k + 1.
Hence, 2**k + 1 has a non-trivial factor.
\newline
QED
\endscroll
\autobuttons
\end{page}
```

Integer Problems



$\Leftarrow$ "Integers" (IntegerPage) 3.80 on page 1029
$\Rightarrow$ "Proof" (IntegerProblemProofPage) 3.80 on page 1038
$\Rightarrow$ "Answer" (IntegerProblemAnswerPage1) 3.80 on page 1038
$\Rightarrow$ "Answer" (IntegerProblemAnswerPage2) 3.80 on page 1042

— **numbers.ht** —

```
\begin{page}{IntegerProblemPage}{Integer Problems}
\beginscroll
One can show that if an integer of the form 2**k - 1 is prime, then
k must be prime.
\downlink{Proof}{IntegerProblemProofPage}
\newline
Problem \#1:  Find the smallest prime p such that \spad{2**p - 1} is not prime.
\downlink{Answer}{IntegerProblemAnswerPage1}
\newline
Problem \#2:  Find the smallest positive integer \spad{n} such that
\spad{n**2 - n + 41} isn't prime.
\downlink{Answer}{IntegerProblemAnswerPage2}
\endscroll
```

```
\autobuttons
\end{page}
```

———

Integer Problem Proof

— numbers.ht —

```
\begin{page}{IntegerProblemProofPage}{Integer Problem Proof}
\beginscroll
Proposition.  If \spad{2**k - 1} is prime, then \spad{k} is prime.
\newline
Proof.  Suppose that k = m * n is a non-trivial factorization.
Then
%
\centerline{2**m = 1 (mod (2**m - 1))}
\centerline{2**(m * n) = 1 (mod (2**m - 1))}
\newline
and 2**m - 1 is a non-trivial factor of 2**k - 1.
\newline
QED
\endscroll
\autobuttons
\end{page}
```

———

Solution to Problem #1

— numbers.ht —

```
\begin{page}{IntegerProblemAnswerPage1}{Solution to Problem \#1}
\beginscroll
Problem \#1:  Find the smallest prime p such that \spad{2**p - 1}
is not prime.
\newline
First, define a function:
\spadpaste{f: NNI -> INT \bound{f1}}
\spadpaste{f(n) == 2**n - 1 \bound{f} \free{f1}}
You can try factoring f(p) as p ranges through the set of primes.
For example,
\spadpaste{factor f(7) \free{f}}
```

```
This gets tedious after a while, so let's use Axiom's stream
facility.  (A stream is essentially an infinite sequence.)
\newline
First, we create a stream consisting of the positive integers:
\spadpaste{ints := [n for n in 1..] \bound{ints}}
Now, we create a stream consisting of the primes:
\spadpaste{primes := [x for x in ints | prime? x] \bound{primes}
\free{ints}}
Here's the 25th prime:
\spadpaste{primes.25 \free{primes}}
Next, create the stream of numbers of the form 2**p - 1 with p prime:
\spadpaste{numbers := [f(n) for n in primes] \bound{numbers} \free{primes f}}
Finally, form
the stream of factorizations of the elements of \spad{numbers}:
\spadpaste{factors := [factor n for n in numbers] \bound{factors}
\free{numbers}}
You can see that the fifth number in the stream (2047 = 23*89)
is the first one that has a non-trivial factorization.
Since 2**11 = 2048, the solution to the problem is 11.
\newline
Here's another way to see that 2047 is the first number in the stream that
is composite:
\spadpaste{nums := [x for x in numbers | not prime? x] \bound{nums}
\free{numbers}}
\endscroll
\autobuttons
\end{page}

\begin{patch}{IntegerProblemAnswerPage1Patch1}
\begin{paste}{IntegerProblemAnswerPage1Full1}{IntegerProblemAnswerPage1Empty1}
\pastebutton{IntegerProblemAnswerPage1Full1}{\hidepaste}
\tab{5}\spadcommand{f: NNI -> INT\bound{f1 }}
\indentrel{3}\begin{verbatim}
                                              Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntegerProblemAnswerPage1Empty1}
\begin{paste}{IntegerProblemAnswerPage1Empty1}{IntegerProblemAnswerPage1Patch1}
\pastebutton{IntegerProblemAnswerPage1Empty1}{\showpaste}
\tab{5}\spadcommand{f: NNI -> INT\bound{f1 }}
\end{paste}\end{patch}

\begin{patch}{IntegerProblemAnswerPage1Patch2}
\begin{paste}{IntegerProblemAnswerPage1Full2}{IntegerProblemAnswerPage1Empty2}
\pastebutton{IntegerProblemAnswerPage1Full2}{\hidepaste}
\tab{5}\spadcommand{f(n) == 2**n - 1\bound{f }\free{f1 }}
\indentrel{3}\begin{verbatim}
                                              Type: Void
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntegerProblemAnswerPage1Empty2}
\begin{paste}{IntegerProblemAnswerPage1Empty2}{IntegerProblemAnswerPage1Patch2}
\pastebutton{IntegerProblemAnswerPage1Empty2}{\showpaste}
\tab{5}\spadcommand{f(n) == 2**n - 1\bound{f }\free{f1 }}
\end{paste}\end{patch}

\begin{patch}{IntegerProblemAnswerPage1Patch3}
\begin{paste}{IntegerProblemAnswerPage1Full3}{IntegerProblemAnswerPage1Empty3}
\pastebutton{IntegerProblemAnswerPage1Full3}{\hidepaste}
\tab{5}\spadcommand{factor f(7)\free{f }}
\indentrel{3}\begin{verbatim}
   (3)  127
                                 Type: Factored Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntegerProblemAnswerPage1Empty3}
\begin{paste}{IntegerProblemAnswerPage1Empty3}{IntegerProblemAnswerPage1Patch3}
\pastebutton{IntegerProblemAnswerPage1Empty3}{\showpaste}
\tab{5}\spadcommand{factor f(7)\free{f }}
\end{paste}\end{patch}

\begin{patch}{IntegerProblemAnswerPage1Patch4}
\begin{paste}{IntegerProblemAnswerPage1Full4}{IntegerProblemAnswerPage1Empty4}
\pastebutton{IntegerProblemAnswerPage1Full4}{\hidepaste}
\tab{5}\spadcommand{ints := [n for n in 1..]\bound{ints }}
\indentrel{3}\begin{verbatim}
   (4)  [1,2,3,4,5,6,7,8,9,10,...]
                           Type: Stream PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntegerProblemAnswerPage1Empty4}
\begin{paste}{IntegerProblemAnswerPage1Empty4}{IntegerProblemAnswerPage1Patch4}
\pastebutton{IntegerProblemAnswerPage1Empty4}{\showpaste}
\tab{5}\spadcommand{ints := [n for n in 1..]\bound{ints }}
\end{paste}\end{patch}

\begin{patch}{IntegerProblemAnswerPage1Patch5}
\begin{paste}{IntegerProblemAnswerPage1Full5}{IntegerProblemAnswerPage1Empty5}
\pastebutton{IntegerProblemAnswerPage1Full5}{\hidepaste}
\tab{5}\spadcommand{primes := [x for x in ints | prime? x]\bound{primes }\free{ints }}
\indentrel{3}\begin{verbatim}
   (5)  [2,3,5,7,11,13,17,19,23,29,...]
                           Type: Stream PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{IntegerProblemAnswerPage1Empty5}
\begin{paste}{IntegerProblemAnswerPage1Empty5}{IntegerProblemAnswerPage1Patch5}
\pastebutton{IntegerProblemAnswerPage1Empty5}{\showpaste}
\tab{5}\spadcommand{primes := [x for x in ints | prime? x]\bound{primes }\free{ints }}
\end{paste}\end{patch}

\begin{patch}{IntegerProblemAnswerPage1Patch6}
\begin{paste}{IntegerProblemAnswerPage1Full6}{IntegerProblemAnswerPage1Empty6}
\pastebutton{IntegerProblemAnswerPage1Full6}{\hidepaste}
\tab{5}\spadcommand{primes.25\free{primes }}
\indentrel{3}\begin{verbatim}
   (6)  97
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntegerProblemAnswerPage1Empty6}
\begin{paste}{IntegerProblemAnswerPage1Empty6}{IntegerProblemAnswerPage1Patch6}
\pastebutton{IntegerProblemAnswerPage1Empty6}{\showpaste}
\tab{5}\spadcommand{primes.25\free{primes }}
\end{paste}\end{patch}

\begin{patch}{IntegerProblemAnswerPage1Patch7}
\begin{paste}{IntegerProblemAnswerPage1Full7}{IntegerProblemAnswerPage1Empty7}
\pastebutton{IntegerProblemAnswerPage1Full7}{\hidepaste}
\tab{5}\spadcommand{numbers := [f(n) for n in primes]\bound{numbers }\free{primes f }}
\indentrel{3}\begin{verbatim}
   (7)
   [3, 7, 31, 127, 2047, 8191, 131071, 524287, 8388607,
    536870911, ...]
                                   Type: Stream Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntegerProblemAnswerPage1Empty7}
\begin{paste}{IntegerProblemAnswerPage1Empty7}{IntegerProblemAnswerPage1Patch7}
\pastebutton{IntegerProblemAnswerPage1Empty7}{\showpaste}
\tab{5}\spadcommand{numbers := [f(n) for n in primes]\bound{numbers }\free{primes f }}
\end{paste}\end{patch}

\begin{patch}{IntegerProblemAnswerPage1Patch8}
\begin{paste}{IntegerProblemAnswerPage1Full8}{IntegerProblemAnswerPage1Empty8}
\pastebutton{IntegerProblemAnswerPage1Full8}{\hidepaste}
\tab{5}\spadcommand{factors := [factor n for n in numbers]\bound{factors }\free{numbers }}
\indentrel{3}\begin{verbatim}
   (8)
   [3, 7, 31, 127, 23 89, 8191, 131071, 524287,
    47 178481, 233 1103 2089, ...]
                          Type: Stream Factored Integer
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntegerProblemAnswerPage1Empty8}
\begin{paste}{IntegerProblemAnswerPage1Empty8}{IntegerProblemAnswerPage1Patch8}
\pastebutton{IntegerProblemAnswerPage1Empty8}{\showpaste}
\tab{5}\spadcommand{factors := [factor n for n in numbers]\bound{factors }\free{numbers }}
\end{paste}\end{patch}

\begin{patch}{IntegerProblemAnswerPage1Patch9}
\begin{paste}{IntegerProblemAnswerPage1Full9}{IntegerProblemAnswerPage1Empty9}
\pastebutton{IntegerProblemAnswerPage1Full9}{\hidepaste}
\tab{5}\spadcommand{nums := [x for x in numbers | not prime? x]\bound{nums }\free{numbers }}
\indentrel{3}\begin{verbatim}
   (9)
   [2047, 8388607, 536870911, 137438953471,
    2199023255551, 8796093022207, 140737488355327,
    9007199254740991, 576460752303423487,
    147573952589676412927, ...]
                                   Type: Stream Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntegerProblemAnswerPage1Empty9}
\begin{paste}{IntegerProblemAnswerPage1Empty9}{IntegerProblemAnswerPage1Patch9}
\pastebutton{IntegerProblemAnswerPage1Empty9}{\showpaste}
\tab{5}\spadcommand{nums := [x for x in numbers | not prime? x]\bound{nums }\free{numbers }}
\end{paste}\end{patch}
```

———-

Solution to Problem #2

— numbers.ht —

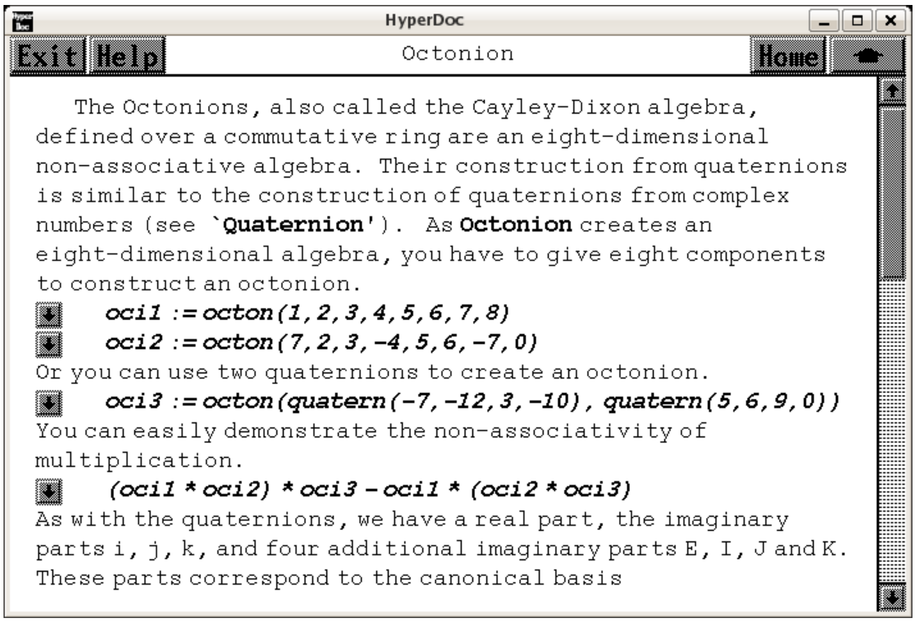
```
\begin{page}{IntegerProblemAnswerPage2}{Solution to Problem \#2}
\beginscroll
Problem \#2:  Find the smallest positive integer n such that
\spad{n**2 - n + 41} is not prime.
\newline
When n = 41, n**2 - n + 41 = 41**2, which certainly isn't prime.
Let's see if any smaller integer works.
Here are the first 40 values:
\spadpaste{numbers := [n**2 - n + 41 for n in 0..40] \bound{numbers}}
Now have Axiom factor the numbers on this list:
\spadpaste{[factor n for n in numbers] \free{numbers}}
You can see that 41 is the smallest positive integer n such that
n**n - n + 41 is not prime.
```

```
\endscroll
\autobuttons
\end{page}

\begin{patch}{IntegerProblemAnswerPage2Patch1}
\begin{paste}{IntegerProblemAnswerPage2Full1}{IntegerProblemAnswerPage2Empty1}
\pastebutton{IntegerProblemAnswerPage2Full1}{\hidepaste}
\tab{5}\spadcommand{numbers := [n**2 - n + 41 for n in 0..40]\bound{numbers }}
\indentrel{3}\begin{verbatim}
   (1)
   [41, 41, 43, 47, 53, 61, 71, 83, 97, 113, 131, 151,
    173, 197, 223, 251, 281, 313, 347, 383, 421, 461,
    503, 547, 593, 641, 691, 743, 797, 853, 911, 971,
    1033, 1097, 1163, 1231, 1301, 1373, 1447, 1523, 1601]
                                     Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntegerProblemAnswerPage2Empty1}
\begin{paste}{IntegerProblemAnswerPage2Empty1}{IntegerProblemAnswerPage2Patch1}
\pastebutton{IntegerProblemAnswerPage2Empty1}{\showpaste}
\tab{5}\spadcommand{numbers := [n**2 - n + 41 for n in 0..40]\bound{numbers }}
\end{paste}\end{patch}

\begin{patch}{IntegerProblemAnswerPage2Patch2}
\begin{paste}{IntegerProblemAnswerPage2Full2}{IntegerProblemAnswerPage2Empty2}
\pastebutton{IntegerProblemAnswerPage2Full2}{\hidepaste}
\tab{5}\spadcommand{[factor n for n in numbers]\free{numbers }}
\indentrel{3}\begin{verbatim}
   (2)
   [41, 41, 43, 47, 53, 61, 71, 83, 97, 113, 131, 151,
    173, 197, 223, 251, 281, 313, 347, 383, 421, 461,
    503, 547, 593, 641, 691, 743, 797, 853, 911, 971,
    1033, 1097, 1163, 1231, 1301, 1373, 1447, 1523, 1601]
                            Type: List Factored Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntegerProblemAnswerPage2Empty2}
\begin{paste}{IntegerProblemAnswerPage2Empty2}{IntegerProblemAnswerPage2Patch2}
\pastebutton{IntegerProblemAnswerPage2Empty2}{\showpaste}
\tab{5}\spadcommand{[factor n for n in numbers]\free{numbers }}
\end{paste}\end{patch}
```

———-

3.81 oct.ht

Octonion



⇐ "Axiom Number Types" (NumberPage) 3.80 on page 1021
⇒ "Quaternion" (QuaternionXmpPage) 3.89 on page 1134

— **oct.ht** —

```
\begin{page}{OctonionXmpPage}{Octonion}
\beginscroll
The Octonions, also called the Cayley-Dixon algebra, defined over a
commutative ring are an eight-dimensional non-associative algebra.
Their construction from quaternions is similar to the construction
of quaternions from complex numbers
(see \downlink{‘Quaternion’}{QuaternionXmpPage}\ignore{Quaternion}).
%
\xtc{
As \spadtype{Octonion} creates an eight-dimensional algebra, you have to
give eight components to construct an octonion.
}{
\spadpaste{oci1 := octon(1,2,3,4,5,6,7,8) \bound{oci1}}
}
\xtc{
}{
\spadpaste{oci2 := octon(7,2,3,-4,5,6,-7,0) \bound{oci2}}
}
%
```

```
%
\xtc{
Or you can use two quaternions to create an octonion.
}{
\spadpaste{oci3 := octon(quatern(-7,-12,3,-10), quatern(5,6,9,0))
\bound{oci3}}
}
%
%
\xtc{
You can easily demonstrate the non-associativity of multiplication.
}{
\spadpaste{(oci1 * oci2) * oci3 - oci1 * (oci2 * oci3)
\free{oci1 oci2 oci3}}
}
%
As with the quaternions, we have a real part, the imaginary
parts \spad{i}, \spad{j}, \spad{k}, and four
additional imaginary parts \spad{E}, \spad{I}, \spad{J} and \spad{K}.
These parts correspond to the canonical basis
\spad{(1,i,j,k,E,I,J,K)}.
\xtc{
For each basis element there is a component operation to extract
the coefficient of the basis element for a given octonion.
%\spadfunFrom{real}{Octonion},
%\spadfunFrom{imagi}{Octonion},
%\spadfunFrom{imagj}{Octonion},
%\spadfunFrom{imagk}{Octonion},
%\spadfunFrom{imagE}{Octonion},
%\spadfunFrom{imagI}{Octonion},
%\spadfunFrom{imagJ}{Octonion}, and
%\spadfunFrom{imagK}{Octonion}.
}{
\spadpaste{[real oci1, imagi oci1, imagj oci1, imagk oci1, imagE oci1,
imagI oci1, imagJ oci1, imagK oci1] \free{oci1}}
}
%
A basis with respect to the
quaternions is given by \spad{(1,E)}.
However, you might ask, what then are the commuting rules?
To answer this, we create some generic elements.
%
\xtc{
We do this in Axiom
by simply changing the ground ring from \spadtype{Integer} to
\spadtype{Polynomial Integer}.
}{
\spadpaste{q : Quaternion Polynomial Integer := quatern(q1, qi, qj, qk)
\bound{q}}
}
```

```
\xtc{
}{
\spadpaste{E : Octonion Polynomial Integer:= octon(0,0,0,0,1,0,0,0)
\bound{E}}
}
%
\xtc{
Note that quaternions are automatically converted to octonions in the
obvious way.
}{
\spadpaste{q * E \free{q E}}
}
\xtc{
}{
\spadpaste{E * q \free{E q}}
}
\xtc{
}{
\spadpaste{q * 1\$(Octonion Polynomial Integer) \free{q}}
}
\xtc{
}{
\spadpaste{1\$(Octonion Polynomial Integer) * q \free{q}}
}
\xtc{
Finally, we check that the \spadfunFrom{norm}{Octonion},
defined as the sum of the squares of the coefficients,
is a multiplicative map.
}{
\spadpaste{o : Octonion Polynomial Integer := octon(o1, oi, oj, ok, oE,
oI, oJ, oK) \bound{o}}
}
\xtc{
}{
\spadpaste{norm o \free{o}}
}
\xtc{
}{
\spadpaste{p : Octonion Polynomial Integer := octon(p1, pi, pj, pk, pE,
pI, pJ, pK) \bound{p}}
}
\xtc{
Since the result is \spad{0}, the norm is multiplicative.
}{
\spadpaste{norm(o*p)-norm(p)*norm(p)\free{o p} }
}
\showBlurb{Octonion}
\endscroll
\autobuttons
\end{page}
```

```
\begin{patch}{OctonionXmpPagePatch1}
\begin{paste}{OctonionXmpPageFull1}{OctonionXmpPageEmpty1}
\pastebutton{OctonionXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{oci1 := octon(1,2,3,4,5,6,7,8)\bound{oci1 }}
\indentrel{3}\begin{verbatim}
   (1)  1 + 2i + 3j + 4k + 5E + 6I + 7J + 8K
                                 Type: Octonion Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OctonionXmpPageEmpty1}
\begin{paste}{OctonionXmpPageEmpty1}{OctonionXmpPagePatch1}
\pastebutton{OctonionXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{oci1 := octon(1,2,3,4,5,6,7,8)\bound{oci1 }}
\end{paste}\end{patch}

\begin{patch}{OctonionXmpPagePatch2}
\begin{paste}{OctonionXmpPageFull2}{OctonionXmpPageEmpty2}
\pastebutton{OctonionXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{oci2 := octon(7,2,3,-4,5,6,-7,0)\bound{oci2 }}
\indentrel{3}\begin{verbatim}
   (2)  7 + 2i + 3j - 4k + 5E + 6I - 7J
                                 Type: Octonion Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OctonionXmpPageEmpty2}
\begin{paste}{OctonionXmpPageEmpty2}{OctonionXmpPagePatch2}
\pastebutton{OctonionXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{oci2 := octon(7,2,3,-4,5,6,-7,0)\bound{oci2 }}
\end{paste}\end{patch}

\begin{patch}{OctonionXmpPagePatch3}
\begin{paste}{OctonionXmpPageFull3}{OctonionXmpPageEmpty3}
\pastebutton{OctonionXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{oci3 := octon(quatern(-7,-12,3,-10), quatern(5,6,9,0))\bound{oci3 }}
\indentrel{3}\begin{verbatim}
   (3)  - 7 - 12i + 3j - 10k + 5E + 6I + 9J
                                 Type: Octonion Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OctonionXmpPageEmpty3}
\begin{paste}{OctonionXmpPageEmpty3}{OctonionXmpPagePatch3}
\pastebutton{OctonionXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{oci3 := octon(quatern(-7,-12,3,-10), quatern(5,6,9,0))\bound{oci3 }}
\end{paste}\end{patch}

\begin{patch}{OctonionXmpPagePatch4}
```

```
\begin{paste}{OctonionXmpPageFull4}{OctonionXmpPageEmpty4}
\pastebutton{OctonionXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{(oci1 * oci2) * oci3 - oci1 * (oci2 * oci3)\free{oci1 oci2 oci3 }}
\indentrel{3}\begin{verbatim}
   (4)
   2696i - 2928j - 4072k + 16E - 1192I + 832J + 2616K
                                  Type: Octonion Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OctonionXmpPageEmpty4}
\begin{paste}{OctonionXmpPageEmpty4}{OctonionXmpPagePatch4}
\pastebutton{OctonionXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{(oci1 * oci2) * oci3 - oci1 * (oci2 * oci3)\free{oci1 oci2 oci3 }}
\end{paste}\end{patch}

\begin{patch}{OctonionXmpPagePatch5}
\begin{paste}{OctonionXmpPageFull5}{OctonionXmpPageEmpty5}
\pastebutton{OctonionXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{[real oci1, imagi oci1, imagj oci1, imagk oci1, imagE oci1, imagI oci1, imag
\indentrel{3}\begin{verbatim}
   (5)  [1,2,3,4,5,6,7,8]
                              Type: List PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OctonionXmpPageEmpty5}
\begin{paste}{OctonionXmpPageEmpty5}{OctonionXmpPagePatch5}
\pastebutton{OctonionXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{[real oci1, imagi oci1, imagj oci1, imagk oci1, imagE oci1, imagI oci1, imag
\end{paste}\end{patch}

\begin{patch}{OctonionXmpPagePatch6}
\begin{paste}{OctonionXmpPageFull6}{OctonionXmpPageEmpty6}
\pastebutton{OctonionXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{q : Quaternion Polynomial Integer := quatern(q1, qi, qj, qk)\bound{q }}
\indentrel{3}\begin{verbatim}
   (6)  q1 + qi i + qj j + qk k
                    Type: Quaternion Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OctonionXmpPageEmpty6}
\begin{paste}{OctonionXmpPageEmpty6}{OctonionXmpPagePatch6}
\pastebutton{OctonionXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{q : Quaternion Polynomial Integer := quatern(q1, qi, qj, qk)\bound{q }}
\end{paste}\end{patch}

\begin{patch}{OctonionXmpPagePatch7}
\begin{paste}{OctonionXmpPageFull7}{OctonionXmpPageEmpty7}
```

```
\pastebutton{OctonionXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{E : Octonion Polynomial Integer:= octon(0,0,0,0,1,0,0,0)\bound{E }}
\indentrel{3}\begin{verbatim}
   (7)  E
                     Type: Octonion Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OctonionXmpPageEmpty7}
\begin{paste}{OctonionXmpPageEmpty7}{OctonionXmpPagePatch7}
\pastebutton{OctonionXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{E : Octonion Polynomial Integer:= octon(0,0,0,0,1,0,0,0)\bound{E }}
\end{paste}\end{patch}

\begin{patch}{OctonionXmpPagePatch8}
\begin{paste}{OctonionXmpPageFull8}{OctonionXmpPageEmpty8}
\pastebutton{OctonionXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{q * E\free{q E }}
\indentrel{3}\begin{verbatim}
   (8)  q1 E + qi I + qj J + qk K
                     Type: Octonion Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OctonionXmpPageEmpty8}
\begin{paste}{OctonionXmpPageEmpty8}{OctonionXmpPagePatch8}
\pastebutton{OctonionXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{q * E\free{q E }}
\end{paste}\end{patch}

\begin{patch}{OctonionXmpPagePatch9}
\begin{paste}{OctonionXmpPageFull9}{OctonionXmpPageEmpty9}
\pastebutton{OctonionXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{E * q\free{E q }}
\indentrel{3}\begin{verbatim}
   (9)  q1 E - qi I - qj J - qk K
                     Type: Octonion Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OctonionXmpPageEmpty9}
\begin{paste}{OctonionXmpPageEmpty9}{OctonionXmpPagePatch9}
\pastebutton{OctonionXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{E * q\free{E q }}
\end{paste}\end{patch}

\begin{patch}{OctonionXmpPagePatch10}
\begin{paste}{OctonionXmpPageFull10}{OctonionXmpPageEmpty10}
\pastebutton{OctonionXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{q * 1$(Octonion Polynomial Integer)\free{q }}
```

```
\indentrel{3}\begin{verbatim}
   (10)  q1 + qi i + qj j + qk k
                      Type: Octonion Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OctonionXmpPageEmpty10}
\begin{paste}{OctonionXmpPageEmpty10}{OctonionXmpPagePatch10}
\pastebutton{OctonionXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{q * 1$(Octonion Polynomial Integer)\free{q }}
\end{paste}\end{patch}

\begin{patch}{OctonionXmpPagePatch11}
\begin{paste}{OctonionXmpPageFull11}{OctonionXmpPageEmpty11}
\pastebutton{OctonionXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{1$(Octonion Polynomial Integer) * q\free{q }}
\indentrel{3}\begin{verbatim}
   (11)  q1 + qi i + qj j + qk k
                      Type: Octonion Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OctonionXmpPageEmpty11}
\begin{paste}{OctonionXmpPageEmpty11}{OctonionXmpPagePatch11}
\pastebutton{OctonionXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{1$(Octonion Polynomial Integer) * q\free{q }}
\end{paste}\end{patch}

\begin{patch}{OctonionXmpPagePatch12}
\begin{paste}{OctonionXmpPageFull12}{OctonionXmpPageEmpty12}
\pastebutton{OctonionXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{o : Octonion Polynomial Integer := octon(o1, oi, oj, ok, oE, oI, oJ, oK)\bou
\indentrel{3}\begin{verbatim}
   (12)
   o1 + oi i + oj j + ok k + oE E + oI I + oJ J + oK K
                      Type: Octonion Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OctonionXmpPageEmpty12}
\begin{paste}{OctonionXmpPageEmpty12}{OctonionXmpPagePatch12}
\pastebutton{OctonionXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{o : Octonion Polynomial Integer := octon(o1, oi, oj, ok, oE, oI, oJ, oK)\bou
\end{paste}\end{patch}

\begin{patch}{OctonionXmpPagePatch13}
\begin{paste}{OctonionXmpPageFull13}{OctonionXmpPageEmpty13}
\pastebutton{OctonionXmpPageFull13}{\hidepaste}
\tab{5}\spadcommand{norm o\free{o }}
\indentrel{3}\begin{verbatim}
```

```
          2     2     2     2     2     2     2     2
   (13)  ok  + oj  + oi  + oK  + oJ  + oI  + oE  + o1
                              Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OctonionXmpPageEmpty13}
\begin{paste}{OctonionXmpPageEmpty13}{OctonionXmpPagePatch13}
\pastebutton{OctonionXmpPageEmpty13}{\showpaste}
\tab{5}\spadcommand{norm o\free{o }}
\end{paste}\end{patch}

\begin{patch}{OctonionXmpPagePatch14}
\begin{paste}{OctonionXmpPageFull14}{OctonionXmpPageEmpty14}
\pastebutton{OctonionXmpPageFull14}{\hidepaste}
\tab{5}\spadcommand{p : Octonion Polynomial Integer := octon(p1, pi, pj, pk, pE, pI, pJ, pK)\bound{p }}
\indentrel{3}\begin{verbatim}
   (14)
   p1 + pi i + pj j + pk k + pE E + pI I + pJ J + pK K
                      Type: Octonion Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OctonionXmpPageEmpty14}
\begin{paste}{OctonionXmpPageEmpty14}{OctonionXmpPagePatch14}
\pastebutton{OctonionXmpPageEmpty14}{\showpaste}
\tab{5}\spadcommand{p : Octonion Polynomial Integer := octon(p1, pi, pj, pk, pE, pI, pJ, pK)\bound{p }}
\end{paste}\end{patch}

\begin{patch}{OctonionXmpPagePatch15}
\begin{paste}{OctonionXmpPageFull15}{OctonionXmpPageEmpty15}
\pastebutton{OctonionXmpPageFull15}{\hidepaste}
\tab{5}\spadcommand{norm(o*p)-norm(p)*norm(p)\free{o p }}
\indentrel{3}\begin{verbatim}
   (15)
         4
     - pk
   +
             2      2      2      2      2      2      2
         - 2pj  - 2pi  - 2pK  - 2pJ  - 2pI  - 2pE  - 2p1
       +
           2     2     2     2     2     2     2     2
         ok  + oj  + oi  + oK  + oJ  + oI  + oE  + o1
    *
         2
       pk
   +
         4
     - pj
   +
```

```
              2      2      2      2      2      2     2
       - 2pi  - 2pK  - 2pJ  - 2pI  - 2pE  - 2p1  + ok
     +
         2     2     2     2     2     2     2
       oj  + oi  + oK  + oJ  + oI  + oE  + o1
  *
       2
     pj
 +
       4
   - pi
 +
              2      2      2      2      2     2     2
       - 2pK  - 2pJ  - 2pI  - 2pE  - 2p1  + ok  + oj
     +
         2     2     2     2     2     2
       oi  + oK  + oJ  + oI  + oE  + o1
  *
       2
     pi
 +
       4
   - pK
 +
              2      2      2      2     2     2     2
       - 2pJ  - 2pI  - 2pE  - 2p1  + ok  + oj  + oi
     +
         2     2     2     2     2
       oK  + oJ  + oI  + oE  + o1
  *
       2
     pK
 +
       4
   - pJ
 +
              2      2      2     2     2     2     2
       - 2pI  - 2pE  - 2p1  + ok  + oj  + oi  + oK
     +
         2     2     2     2
       oJ  + oI  + oE  + o1
  *
       2
     pJ
 +
       4
   - pI
 +
              2      2     2     2     2     2     2     2
       - 2pE  - 2p1  + ok  + oj  + oi  + oK  + oJ  + oI
```

```
      +
           2      2
         oE  + o1
   *
         2
       pI
  +
         4
     - pE
  +
              2      2      2      2      2      2      2      2
         - 2p1  + ok  + oj  + oi  + oK  + oJ  + oI  + oE
       +
           2
         o1
   *
         2
       pE
  +
         4
     - p1
  +
        2      2      2      2      2      2      2      2   2
     (ok  + oj  + oi  + oK  + oJ  + oI  + oE  + o1 )p1
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OctonionXmpPageEmpty15}
\begin{paste}{OctonionXmpPageEmpty15}{OctonionXmpPagePatch15}
\pastebutton{OctonionXmpPageEmpty15}{\showpaste}
\tab{5}\spadcommand{norm(o*p)-norm(p)*norm(p)\free{o p }}
\end{paste}\end{patch}
```

———-

3.82 odpol.ht

OrderlyDifferentialPolynomial

— odpol.ht —

```
\begin{page}{OrderlyDifferentialPolyXmpPage}
{OrderlyDifferentialPolynomial}
\beginscroll
```

```
Many systems of differential equations may be transformed to
equivalent systems of ordinary differential equations where the
equations are expressed polynomially in terms of the unknown
functions.  In Axiom, the domain constructors
\spadtype{OrderlyDifferentialPolynomial} (abbreviated
\spadtype{ODPOL}) and \spadtype{SequentialDifferentialPolynomial}
(abbreviation \spadtype{SDPOL}) implement two domains of ordinary
differential polynomials over any differential ring.  In the simplest
case, this differential ring is usually either the ring of integers,
or the field of rational numbers.  However, Axiom can handle ordinary
differential polynomials over a field of rational functions in a
single indeterminate.

The two domains \spadtype{ODPOL} and \spadtype{SDPOL} are almost
identical, the only difference being the choice of a different
ranking, which is an ordering of the derivatives of the
indeterminates.  The first domain uses an orderly ranking, that is,
derivatives of higher order are ranked higher, and derivatives of the
same order are ranked alphabetically.  The second domain uses a
sequential ranking, where derivatives are ordered first alphabetically
by the differential indeterminates, and then by order.  A more general
domain constructor,
\spadtype{DifferentialSparseMultivariatePolynomial} (abbreviation
\spadtype{DSMP}) allows both a user-provided list of differential
indeterminates as well as a user-defined ranking.  We shall illustrate
\spadtype{ODPOL(FRAC INT)}, which constructs a domain of ordinary
differential polynomials in an arbitrary number of differential
indeterminates with rational numbers as coefficients.  \xtc{ }{
\spadpaste{dpol:= ODPOL(FRAC INT) \bound{dpol}} }

\xtc{
A differential indeterminate \spad{w} may be viewed as an infinite
sequence of algebraic indeterminates, which are the derivatives of
\spad{w}.
To facilitate referencing these, Axiom provides the operation
\spadfunFrom{makeVariable}{OrderlyDifferentialPolynomial} to convert an
element of type \spadtype{Symbol} to a map from the natural numbers to the
differential polynomial ring.
}{
\spadpaste{w := makeVariable('w)\$dpol \free{dpol}\bound{w}}
}
\xtc{
}{
\spadpaste{z := makeVariable('z)\$dpol \free{dpol}\bound{z}}
}
\xtc{
The fifth derivative of \spad{w} can be obtained by applying the map
\spad{w} to the number \spad{5.}
Note that the order of differentiation is given as a subscript (except
when the order is 0).
```

```
}{
\spadpaste{w.5 \free{w}}
}
\xtc{
}{
\spadpaste{w 0 \free{w}}
}
\xtc{
The first five derivatives of \spad{z} can be generated by a list.
}{
\spadpaste{[z.i for i in 1..5] \free{z}}
}
\xtc{
The usual arithmetic can be used to form a differential polynomial from
the derivatives.
}{
\spadpaste{f:= w.4 - w.1 * w.1 * z.3 \free{w}\free{z}\bound{f}}
}
\xtc{
}{
\spadpaste{g:=(z.1)**3 * (z.2)**2 - w.2 \free{z}\free{w}\bound{g}}
}
\xtc{
The operation \spadfunFrom{D}{OrderlyDifferentialPolynomial}
computes the derivative of any differential polynomial.
}{
\spadpaste{D(f) \free{f}}
}
\xtc{
The same operation can compute higher derivatives, like the
fourth derivative.
}{
\spadpaste{D(f,4) \free{f}}
}
\xtc{
The operation \spadfunFrom{makeVariable}{OrderlyDifferentialPolynomial}
creates a map to facilitate referencing the derivatives of \spad{f},
similar to the map \spad{w}.
}{
\spadpaste{df:=makeVariable(f)\$dpol \free{f}\bound{df}}
}
\xtc{
The fourth derivative of f may be referenced easily.
}{
\spadpaste{df.4 \free{df}}
}
\xtc{
The operation \spadfunFrom{order}{OrderlyDifferentialPolynomial}
returns the order of a differential polynomial, or the order
in a specified differential indeterminate.
```

```
}{
\spadpaste{order(g)  \free{g}}
}
\xtc{
}{
\spadpaste{order(g, 'w)  \free{g}}
}
\xtc{
The operation
\spadfunFrom{differentialVariables}{OrderlyDifferentialPolynomial}
returns a list of differential indeterminates occurring in a
differential polynomial.
}{
\spadpaste{differentialVariables(g)  \free{g}}
}
\xtc{
The operation \spadfunFrom{degree}{OrderlyDifferentialPolynomial}
returns the degree, or the degree in the differential indeterminate
specified.
}{
\spadpaste{degree(g) \free{g}}
}
\xtc{
}{
\spadpaste{degree(g, 'w)  \free{g}}
}
\xtc{
The operation \spadfunFrom{weights}{OrderlyDifferentialPolynomial}
returns a list of weights of differential monomials appearing in
differential polynomial, or a list of weights in a specified
differential indeterminate.
}{
\spadpaste{weights(g)  \free{g}}
}
\xtc{
}{
\spadpaste{weights(g,'w) \free{g}}
}
\xtc{
The operation \spadfunFrom{weight}{OrderlyDifferentialPolynomial} returns
the maximum weight of all differential monomials appearing in the
differential polynomial.
}{
\spadpaste{weight(g)  \free{g}}
}
\xtc{
A differential polynomial is {\em isobaric} if the weights of all
differential monomials appearing in it are equal.
}{
\spadpaste{isobaric?(g) \free{g}}
```

```
}
\xtc{
To substitute {\em differentially}, use
\spadfunFrom{eval}{OrderlyDifferentialPolynomial}.  Note that we must
coerce \spad{'w} to \spadtype{Symbol}, since in \spadtype{ODPOL},
differential indeterminates belong to the domain \spadtype{Symbol}.
Compare this result to the next, which substitutes {\em algebraically}
(no substitution is done since \spad{w.0} does not appear in \spad{g}).
}{
\spadpaste{eval(g,['w::Symbol],[f]) \free{f}\free{g}}
}
\xtc{
}{
\spadpaste{eval(g,variables(w.0),[f]) \free{f}\free{g}}
}
\xtc{
Since \spadtype{OrderlyDifferentialPolynomial} belongs to
\spadtype{PolynomialCategory}, all the operations defined in the latter
category, or in packages for the latter category, are available.
}{
\spadpaste{monomials(g) \free{g}}
}
\xtc{
}{
\spadpaste{variables(g) \free{g}}
}
\xtc{
}{
\spadpaste{gcd(f,g) \free{f}\free{g}}
}
\xtc{
}{
\spadpaste{groebner([f,g]) \free{f}\free{g}}
}
\xtc{
The next three operations are essential for elimination procedures in
differential polynomial rings.
The operation \spadfunFrom{leader}{OrderlyDifferentialPolynomial} returns
the leader of a differential polynomial, which is the highest ranked
derivative of the differential indeterminates that occurs.
}{
\spadpaste{lg:=leader(g)  \free{g}\bound{lg}}
}
\xtc{
The operation \spadfunFrom{separant}{OrderlyDifferentialPolynomial}
returns the separant of a differential polynomial, which is the
partial derivative with respect to the leader.
}{
\spadpaste{sg:=separant(g)  \free{g}\bound{sg}}
}
```

```
\xtc{
The operation \spadfunFrom{initial}{OrderlyDifferentialPolynomial}
returns the initial, which is the leading coefficient when the given
differential polynomial is expressed as a polynomial in the leader.
}{
\spadpaste{ig:=initial(g)  \free{g}\bound{ig}}
}
\xtc{
Using these three operations, it is possible to reduce \spad{f} modulo
the differential ideal generated by \spad{g}.  The general scheme is
to first reduce the order, then reduce the degree in the leader.
First, eliminate \spad{z.3} using the derivative of \spad{g}.
}{
\spadpaste{g1 := D g \free{g}\bound{g1}}
}
\xtc{
Find its leader.
}{
\spadpaste{lg1:= leader g1 \free{g1}\bound{lg1}}
}
\xtc{
Differentiate \spad{f} partially with respect to this leader.
}{
\spadpaste{pdf:=D(f, lg1) \free{f}\free{lg1}\bound{pdf}}
}
\xtc{
Compute the partial remainder of \spad{f} with respect to \spad{g}.
}{
\spadpaste{prf:=sg * f- pdf * g1 \free{f}\free{sg}\free{pdf}
\free{g1}\bound{prf}}
}
\xtc{
Note that high powers of \spad{lg} still appear in \spad{prf}.
Compute the leading coefficient of \spad{prf}
as a polynomial in the leader of \spad{g}.
}{
\spadpaste{lcf:=leadingCoefficient univariate(prf, lg)
\free{prf}\free{lg}\bound{lcf}}
}
\xtc{
Finally, continue eliminating the high powers of \spad{lg} appearing in
\spad{prf} to obtain the (pseudo) remainder of \spad{f} modulo \spad{g}
and its derivatives.
}{
\spadpaste{ig * prf - lcf * g * lg \free{ig}\free{prf}
\free{lcf}\free{g}\free{lg}}
}
\showBlurb{OrderlyDifferentialPolyomial}
\showBlurb{SequentialDifferentialPolynomial}
\endscroll
```

```
\autobuttons
\end{page}

\begin{patch}{OrderlyDifferentialPolynomialXmpPagePatch1}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageFull1}{OrderlyDifferentialPolynomialXmpPageEmpty1}
\pastebutton{OrderlyDifferentialPolynomialXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{dpol:= ODPOL(FRAC INT)\bound{dpol }}
\indentrel{3}\begin{verbatim}
   (1)  OrderlyDifferentialPolynomial Fraction Integer
                                            Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPageEmpty1}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageEmpty1}{OrderlyDifferentialPolynomialXmpPagePatch1}
\pastebutton{OrderlyDifferentialPolynomialXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{dpol:= ODPOL(FRAC INT)\bound{dpol }}
\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPagePatch2}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageFull2}{OrderlyDifferentialPolynomialXmpPageEmpty2}
\pastebutton{OrderlyDifferentialPolynomialXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{w := makeVariable('w)$dpol\free{dpol }\bound{w }}
\indentrel{3}\begin{verbatim}
   (2)  theMap(DPOLCAT-;makeVariable;AM;17!0,62)
Type: (NonNegativeInteger -> OrderlyDifferentialPolynomial Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPageEmpty2}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageEmpty2}{OrderlyDifferentialPolynomialXmpPagePatch2}
\pastebutton{OrderlyDifferentialPolynomialXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{w := makeVariable('w)$dpol\free{dpol }\bound{w }}
\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPagePatch3}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageFull3}{OrderlyDifferentialPolynomialXmpPageEmpty3}
\pastebutton{OrderlyDifferentialPolynomialXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{z := makeVariable('z)$dpol\free{dpol }\bound{z }}
\indentrel{3}\begin{verbatim}
   (3)  theMap(DPOLCAT-;makeVariable;AM;17!0,187)
Type: (NonNegativeInteger -> OrderlyDifferentialPolynomial Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPageEmpty3}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageEmpty3}{OrderlyDifferentialPolynomialXmpPagePatch3}
\pastebutton{OrderlyDifferentialPolynomialXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{z := makeVariable('z)$dpol\free{dpol }\bound{z }}
\end{paste}\end{patch}
```

```
\begin{patch}{OrderlyDifferentialPolynomialXmpPagePatch4}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageFull4}{OrderlyDifferentialPolynomialXmpPageEmp
\pastebutton{OrderlyDifferentialPolynomialXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{w.5\free{w }}
\indentrel{3}\begin{verbatim}
   (4)  w
         5
   Type: OrderlyDifferentialPolynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPageEmpty4}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageEmpty4}{OrderlyDifferentialPolynomialXmpPagePa
\pastebutton{OrderlyDifferentialPolynomialXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{w.5\free{w }}
\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPagePatch5}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageFull5}{OrderlyDifferentialPolynomialXmpPageEmp
\pastebutton{OrderlyDifferentialPolynomialXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{w 0\free{w }}
\indentrel{3}\begin{verbatim}
   (5)  w
   Type: OrderlyDifferentialPolynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPageEmpty5}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageEmpty5}{OrderlyDifferentialPolynomialXmpPagePa
\pastebutton{OrderlyDifferentialPolynomialXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{w 0\free{w }}
\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPagePatch6}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageFull6}{OrderlyDifferentialPolynomialXmpPageEmp
\pastebutton{OrderlyDifferentialPolynomialXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{[z.i for i in 1..5]\free{z }}
\indentrel{3}\begin{verbatim}
   (6)  [z ,z ,z ,z ,z ]
          1  2  3  4  5
Type: List OrderlyDifferentialPolynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPageEmpty6}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageEmpty6}{OrderlyDifferentialPolynomialXmpPagePa
\pastebutton{OrderlyDifferentialPolynomialXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{[z.i for i in 1..5]\free{z }}
\end{paste}\end{patch}
```

```
\begin{patch}{OrderlyDifferentialPolynomialXmpPagePatch7}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageFull7}{OrderlyDifferentialPolynomialXmpPageEmpty7}
\pastebutton{OrderlyDifferentialPolynomialXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{f:= w.4 - w.1 * w.1 * z.3\free{w }\free{z }\bound{f }}
\indentrel{3}\begin{verbatim}
               2
   (7)  w  - w  z
         4    1  3
   Type: OrderlyDifferentialPolynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPageEmpty7}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageEmpty7}{OrderlyDifferentialPolynomialXmpPagePatch7}
\pastebutton{OrderlyDifferentialPolynomialXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{f:= w.4 - w.1 * w.1 * z.3\free{w }\free{z }\bound{f }}
\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPagePatch8}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageFull8}{OrderlyDifferentialPolynomialXmpPageEmpty8}
\pastebutton{OrderlyDifferentialPolynomialXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{g:=(z.1)**3 * (z.2)**2 - w.2\free{z }\free{w }\bound{g }}
\indentrel{3}\begin{verbatim}
          3  2
   (8)  z  z   - w
         1  2     2
   Type: OrderlyDifferentialPolynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPageEmpty8}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageEmpty8}{OrderlyDifferentialPolynomialXmpPagePatch8}
\pastebutton{OrderlyDifferentialPolynomialXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{g:=(z.1)**3 * (z.2)**2 - w.2\free{z }\free{w }\bound{g }}
\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPagePatch9}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageFull9}{OrderlyDifferentialPolynomialXmpPageEmpty9}
\pastebutton{OrderlyDifferentialPolynomialXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{D(f)\free{f }}
\indentrel{3}\begin{verbatim}
               2
   (9)  w  - w  z  - 2w w z
         5    1  4     1 2 3
   Type: OrderlyDifferentialPolynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPageEmpty9}
```

```
\begin{paste}{OrderlyDifferentialPolynomialXmpPageEmpty9}{OrderlyDifferentialPolynomialXmpPagePa
\pastebutton{OrderlyDifferentialPolynomialXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{D(f)\free{f }}
\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPagePatch10}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageFull10}{OrderlyDifferentialPolynomialXmpPageEm
\pastebutton{OrderlyDifferentialPolynomialXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{D(f,4)\free{f }}
\indentrel{3}\begin{verbatim}
   (10)
            2                              2
     w  - w  z  - 8w w z  + (- 12w w  - 12w  )z  - 2w z w
      8    1  7     1 2 6        1 3      2   5     1 3 5
   +
                                        2
     (- 8w w  - 24w w )z  - 8w z w  - 6w  z
          1 4      2 3  4    2 3 4     3  3
   Type: OrderlyDifferentialPolynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPageEmpty10}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageEmpty10}{OrderlyDifferentialPolynomialXmpPageP
\pastebutton{OrderlyDifferentialPolynomialXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{D(f,4)\free{f }}
\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPagePatch11}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageFull11}{OrderlyDifferentialPolynomialXmpPageEm
\pastebutton{OrderlyDifferentialPolynomialXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{df:=makeVariable(f)$dpol\free{f }\bound{df }}
\indentrel{3}\begin{verbatim}
   (11)  theMap(DPOLCAT-;makeVariable;AM;17!0,488)
Type: (NonNegativeInteger -> OrderlyDifferentialPolynomial Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPageEmpty11}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageEmpty11}{OrderlyDifferentialPolynomialXmpPageP
\pastebutton{OrderlyDifferentialPolynomialXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{df:=makeVariable(f)$dpol\free{f }\bound{df }}
\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPagePatch12}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageFull12}{OrderlyDifferentialPolynomialXmpPageEm
\pastebutton{OrderlyDifferentialPolynomialXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{df.4\free{df }}
\indentrel{3}\begin{verbatim}
   (12)
```

```
            2                               2
     w  - w  z  - 8w w z  + (- 12w w  - 12w  )z  - 2w z w
      8    1  7     1 2 6         1 3      2   5     1 3 5
   +
                                         2
     (- 8w w  - 24w w )z  - 8w z w  - 6w  z
          1 4      2 3  4     2 3 4     3  3
   Type: OrderlyDifferentialPolynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPageEmpty12}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageEmpty12}{OrderlyDifferentialPolynomialXmpPagePatch12}
\pastebutton{OrderlyDifferentialPolynomialXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{df.4\free{df }}
\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPagePatch13}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageFull13}{OrderlyDifferentialPolynomialXmpPageEmpty13}
\pastebutton{OrderlyDifferentialPolynomialXmpPageFull13}{\hidepaste}
\tab{5}\spadcommand{order(g)\free{g }}
\indentrel{3}\begin{verbatim}
   (13)  2
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPageEmpty13}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageEmpty13}{OrderlyDifferentialPolynomialXmpPagePatch13}
\pastebutton{OrderlyDifferentialPolynomialXmpPageEmpty13}{\showpaste}
\tab{5}\spadcommand{order(g)\free{g }}
\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPagePatch14}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageFull14}{OrderlyDifferentialPolynomialXmpPageEmpty14}
\pastebutton{OrderlyDifferentialPolynomialXmpPageFull14}{\hidepaste}
\tab{5}\spadcommand{order(g, 'w)\free{g }}
\indentrel{3}\begin{verbatim}
   (14)  2
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPageEmpty14}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageEmpty14}{OrderlyDifferentialPolynomialXmpPagePatch14}
\pastebutton{OrderlyDifferentialPolynomialXmpPageEmpty14}{\showpaste}
\tab{5}\spadcommand{order(g, 'w)\free{g }}
\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPagePatch15}
```

```
\begin{paste}{OrderlyDifferentialPolynomialXmpPageFull15}{OrderlyDifferentialPolynomialXmpPageEm
\pastebutton{OrderlyDifferentialPolynomialXmpPageFull15}{\hidepaste}
\tab{5}\spadcommand{differentialVariables(g)\free{g }}
\indentrel{3}\begin{verbatim}
   (15)  [z,w]
                                       Type: List Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPageEmpty15}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageEmpty15}{OrderlyDifferentialPolynomialXmpPageP
\pastebutton{OrderlyDifferentialPolynomialXmpPageEmpty15}{\showpaste}
\tab{5}\spadcommand{differentialVariables(g)\free{g }}
\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPagePatch16}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageFull16}{OrderlyDifferentialPolynomialXmpPageEm
\pastebutton{OrderlyDifferentialPolynomialXmpPageFull16}{\hidepaste}
\tab{5}\spadcommand{degree(g)\free{g }}
\indentrel{3}\begin{verbatim}
          2  3
   (16)  z  z
          2  1
Type: IndexedExponents OrderlyDifferentialVariable Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPageEmpty16}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageEmpty16}{OrderlyDifferentialPolynomialXmpPageP
\pastebutton{OrderlyDifferentialPolynomialXmpPageEmpty16}{\showpaste}
\tab{5}\spadcommand{degree(g)\free{g }}
\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPagePatch17}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageFull17}{OrderlyDifferentialPolynomialXmpPageEm
\pastebutton{OrderlyDifferentialPolynomialXmpPageFull17}{\hidepaste}
\tab{5}\spadcommand{degree(g, 'w)\free{g }}
\indentrel{3}\begin{verbatim}
   (17)  1
                                   Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPageEmpty17}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageEmpty17}{OrderlyDifferentialPolynomialXmpPageP
\pastebutton{OrderlyDifferentialPolynomialXmpPageEmpty17}{\showpaste}
\tab{5}\spadcommand{degree(g, 'w)\free{g }}
\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPagePatch18}
```

```
\begin{paste}{OrderlyDifferentialPolynomialXmpPageFull18}{OrderlyDifferentialPolynomialXmpPageEmpty18}
\pastebutton{OrderlyDifferentialPolynomialXmpPageFull18}{\hidepaste}
\tab{5}\spadcommand{weights(g)\free{g }}
\indentrel{3}\begin{verbatim}
   (18)  [7,2]
                          Type: List NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPageEmpty18}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageEmpty18}{OrderlyDifferentialPolynomialXmpPagePatch18}
\pastebutton{OrderlyDifferentialPolynomialXmpPageEmpty18}{\showpaste}
\tab{5}\spadcommand{weights(g)\free{g }}
\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPagePatch19}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageFull19}{OrderlyDifferentialPolynomialXmpPageEmpty19}
\pastebutton{OrderlyDifferentialPolynomialXmpPageFull19}{\hidepaste}
\tab{5}\spadcommand{weights(g,'w)\free{g }}
\indentrel{3}\begin{verbatim}
   (19)  [2]
                          Type: List NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPageEmpty19}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageEmpty19}{OrderlyDifferentialPolynomialXmpPagePatch19}
\pastebutton{OrderlyDifferentialPolynomialXmpPageEmpty19}{\showpaste}
\tab{5}\spadcommand{weights(g,'w)\free{g }}
\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPagePatch20}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageFull20}{OrderlyDifferentialPolynomialXmpPageEmpty20}
\pastebutton{OrderlyDifferentialPolynomialXmpPageFull20}{\hidepaste}
\tab{5}\spadcommand{weight(g)\free{g }}
\indentrel{3}\begin{verbatim}
   (20)  7
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPageEmpty20}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageEmpty20}{OrderlyDifferentialPolynomialXmpPagePatch20}
\pastebutton{OrderlyDifferentialPolynomialXmpPageEmpty20}{\showpaste}
\tab{5}\spadcommand{weight(g)\free{g }}
\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPagePatch21}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageFull21}{OrderlyDifferentialPolynomialXmpPageEmpty21}
\pastebutton{OrderlyDifferentialPolynomialXmpPageFull21}{\hidepaste}
```

```
\tab{5}\spadcommand{isobaric?(g)\free{g }}
\indentrel{3}\begin{verbatim}
   (21)  false
                                               Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPageEmpty21}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageEmpty21}{OrderlyDifferentialPolynomialXmpPageP
\pastebutton{OrderlyDifferentialPolynomialXmpPageEmpty21}{\showpaste}
\tab{5}\spadcommand{isobaric?(g)\free{g }}
\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPagePatch22}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageFull22}{OrderlyDifferentialPolynomialXmpPageEm
\pastebutton{OrderlyDifferentialPolynomialXmpPageFull22}{\hidepaste}
\tab{5}\spadcommand{eval(g,['w::Symbol],[f])\free{f }\free{g }}
\indentrel{3}\begin{verbatim}
   (22)
           2                            2        3  2
   - w  + w  z  + 4w w z  + (2w w  + 2w  )z  + z  z
      6    1  5     1 2 4      1 3     2   3    1  2
   Type: OrderlyDifferentialPolynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPageEmpty22}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageEmpty22}{OrderlyDifferentialPolynomialXmpPageP
\pastebutton{OrderlyDifferentialPolynomialXmpPageEmpty22}{\showpaste}
\tab{5}\spadcommand{eval(g,['w::Symbol],[f])\free{f }\free{g }}
\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPagePatch23}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageFull23}{OrderlyDifferentialPolynomialXmpPageEm
\pastebutton{OrderlyDifferentialPolynomialXmpPageFull23}{\hidepaste}
\tab{5}\spadcommand{eval(g,variables(w.0),[f])\free{f }\free{g }}
\indentrel{3}\begin{verbatim}
          3  2
   (23)  z  z   - w
          1  2     2
   Type: OrderlyDifferentialPolynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPageEmpty23}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageEmpty23}{OrderlyDifferentialPolynomialXmpPageP
\pastebutton{OrderlyDifferentialPolynomialXmpPageEmpty23}{\showpaste}
\tab{5}\spadcommand{eval(g,variables(w.0),[f])\free{f }\free{g }}
\end{paste}\end{patch}
```

```
\begin{patch}{OrderlyDifferentialPolynomialXmpPagePatch24}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageFull24}{OrderlyDifferentialPolynomialXmpPageEmpty24}
\pastebutton{OrderlyDifferentialPolynomialXmpPageFull24}{\hidepaste}
\tab{5}\spadcommand{monomials(g)\free{g }}
\indentrel{3}\begin{verbatim}
           3  2
   (24)  [z  z  ,- w ]
          1  2     2
Type: List OrderlyDifferentialPolynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPageEmpty24}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageEmpty24}{OrderlyDifferentialPolynomialXmpPagePatch24}
\pastebutton{OrderlyDifferentialPolynomialXmpPageEmpty24}{\showpaste}
\tab{5}\spadcommand{monomials(g)\free{g }}
\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPagePatch25}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageFull25}{OrderlyDifferentialPolynomialXmpPageEmpty25}
\pastebutton{OrderlyDifferentialPolynomialXmpPageFull25}{\hidepaste}
\tab{5}\spadcommand{variables(g)\free{g }}
\indentrel{3}\begin{verbatim}
   (25)  [z ,w ,z ]
          2  2  1
         Type: List OrderlyDifferentialVariable Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPageEmpty25}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageEmpty25}{OrderlyDifferentialPolynomialXmpPagePatch25}
\pastebutton{OrderlyDifferentialPolynomialXmpPageEmpty25}{\showpaste}
\tab{5}\spadcommand{variables(g)\free{g }}
\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPagePatch26}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageFull26}{OrderlyDifferentialPolynomialXmpPageEmpty26}
\pastebutton{OrderlyDifferentialPolynomialXmpPageFull26}{\hidepaste}
\tab{5}\spadcommand{gcd(f,g)\free{f }\free{g }}
\indentrel{3}\begin{verbatim}
   (26)  1
   Type: OrderlyDifferentialPolynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPageEmpty26}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageEmpty26}{OrderlyDifferentialPolynomialXmpPagePatch26}
\pastebutton{OrderlyDifferentialPolynomialXmpPageEmpty26}{\showpaste}
\tab{5}\spadcommand{gcd(f,g)\free{f }\free{g }}
\end{paste}\end{patch}
```

```
\begin{patch}{OrderlyDifferentialPolynomialXmpPagePatch27}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageFull27}{OrderlyDifferentialPolynomialXmpPageEm
\pastebutton{OrderlyDifferentialPolynomialXmpPageFull27}{\hidepaste}
\tab{5}\spadcommand{groebner([f,g])\free{f }\free{g }}
\indentrel{3}\begin{verbatim}
                 2     3  2
   (27)  [w  - w  z ,z  z   - w ]
           4    1  3  1  2     2
Type: List OrderlyDifferentialPolynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPageEmpty27}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageEmpty27}{OrderlyDifferentialPolynomialXmpPageP
\pastebutton{OrderlyDifferentialPolynomialXmpPageEmpty27}{\showpaste}
\tab{5}\spadcommand{groebner([f,g])\free{f }\free{g }}
\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPagePatch28}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageFull28}{OrderlyDifferentialPolynomialXmpPageEm
\pastebutton{OrderlyDifferentialPolynomialXmpPageFull28}{\hidepaste}
\tab{5}\spadcommand{lg:=leader(g)\free{g }\bound{lg }}
\indentrel{3}\begin{verbatim}
   (28)  z
          2
               Type: OrderlyDifferentialVariable Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPageEmpty28}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageEmpty28}{OrderlyDifferentialPolynomialXmpPageP
\pastebutton{OrderlyDifferentialPolynomialXmpPageEmpty28}{\showpaste}
\tab{5}\spadcommand{lg:=leader(g)\free{g }\bound{lg }}
\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPagePatch29}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageFull29}{OrderlyDifferentialPolynomialXmpPageEm
\pastebutton{OrderlyDifferentialPolynomialXmpPageFull29}{\hidepaste}
\tab{5}\spadcommand{sg:=separant(g)\free{g }\bound{sg }}
\indentrel{3}\begin{verbatim}
            3
   (29)  2z  z
           1  2
   Type: OrderlyDifferentialPolynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPageEmpty29}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageEmpty29}{OrderlyDifferentialPolynomialXmpPageP
```

```
\pastebutton{OrderlyDifferentialPolynomialXmpPageEmpty29}{\showpaste}
\tab{5}\spadcommand{sg:=separant(g)\free{g }\bound{sg }}
\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPagePatch30}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageFull30}{OrderlyDifferentialPolynomialXmpPageEmpty30}
\pastebutton{OrderlyDifferentialPolynomialXmpPageFull30}{\hidepaste}
\tab{5}\spadcommand{ig:=initial(g)\free{g }\bound{ig }}
\indentrel{3}\begin{verbatim}
          3
   (30)  z
         1
   Type: OrderlyDifferentialPolynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPageEmpty30}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageEmpty30}{OrderlyDifferentialPolynomialXmpPagePatch30}
\pastebutton{OrderlyDifferentialPolynomialXmpPageEmpty30}{\showpaste}
\tab{5}\spadcommand{ig:=initial(g)\free{g }\bound{ig }}
\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPagePatch31}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageFull31}{OrderlyDifferentialPolynomialXmpPageEmpty31}
\pastebutton{OrderlyDifferentialPolynomialXmpPageFull31}{\hidepaste}
\tab{5}\spadcommand{g1 := D g\free{g }\bound{g1 }}
\indentrel{3}\begin{verbatim}
           3               2  3
   (31)  2z  z z  - w  + 3z  z
           1  2 3    3     1  2
   Type: OrderlyDifferentialPolynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPageEmpty31}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageEmpty31}{OrderlyDifferentialPolynomialXmpPagePatch31}
\pastebutton{OrderlyDifferentialPolynomialXmpPageEmpty31}{\showpaste}
\tab{5}\spadcommand{g1 := D g\free{g }\bound{g1 }}
\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPagePatch32}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageFull32}{OrderlyDifferentialPolynomialXmpPageEmpty32}
\pastebutton{OrderlyDifferentialPolynomialXmpPageFull32}{\hidepaste}
\tab{5}\spadcommand{lg1:= leader g1\free{g1 }\bound{lg1 }}
\indentrel{3}\begin{verbatim}
   (32)  z
         3
              Type: OrderlyDifferentialVariable Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{OrderlyDifferentialPolynomialXmpPageEmpty32}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageEmpty32}{OrderlyDifferentialPolynomialXmpPageP
\pastebutton{OrderlyDifferentialPolynomialXmpPageEmpty32}{\showpaste}
\tab{5}\spadcommand{lg1:= leader g1\free{g1 }\bound{lg1 }}
\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPagePatch33}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageFull33}{OrderlyDifferentialPolynomialXmpPageEm
\pastebutton{OrderlyDifferentialPolynomialXmpPageFull33}{\hidepaste}
\tab{5}\spadcommand{pdf:=D(f, lg1)\free{f }\free{lg1 }\bound{pdf }}
\indentrel{3}\begin{verbatim}
             2
   (33)  - w
            1
   Type: OrderlyDifferentialPolynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPageEmpty33}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageEmpty33}{OrderlyDifferentialPolynomialXmpPageP
\pastebutton{OrderlyDifferentialPolynomialXmpPageEmpty33}{\showpaste}
\tab{5}\spadcommand{pdf:=D(f, lg1)\free{f }\free{lg1 }\bound{pdf }}
\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPagePatch34}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageFull34}{OrderlyDifferentialPolynomialXmpPageEm
\pastebutton{OrderlyDifferentialPolynomialXmpPageFull34}{\hidepaste}
\tab{5}\spadcommand{prf:=sg * f- pdf * g1\free{f }\free{sg }\free{pdf }\free{g1 }\bound{prf }}
\indentrel{3}\begin{verbatim}
            3         2         2  2  3
   (34)  2z  z w  - w  w  + 3w  z  z
           1  2 4    1  3     1  1  2
   Type: OrderlyDifferentialPolynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPageEmpty34}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageEmpty34}{OrderlyDifferentialPolynomialXmpPageP
\pastebutton{OrderlyDifferentialPolynomialXmpPageEmpty34}{\showpaste}
\tab{5}\spadcommand{prf:=sg * f- pdf * g1\free{f }\free{sg }\free{pdf }\free{g1 }\bound{prf }}
\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPagePatch35}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageFull35}{OrderlyDifferentialPolynomialXmpPageEm
\pastebutton{OrderlyDifferentialPolynomialXmpPageFull35}{\hidepaste}
\tab{5}\spadcommand{lcf:=leadingCoefficient univariate(prf, lg)\free{prf }\free{lg }\bound{lcf }
\indentrel{3}\begin{verbatim}
            2  2
   (35)  3w  z
```

```
          1  1
   Type: OrderlyDifferentialPolynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPageEmpty35}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageEmpty35}{OrderlyDifferentialPolynomialXmpPagePatch35}
\pastebutton{OrderlyDifferentialPolynomialXmpPageEmpty35}{\showpaste}
\tab{5}\spadcommand{lcf:=leadingCoefficient univariate(prf, lg)\free{prf }\free{lg }\bound{lcf }}
\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPagePatch36}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageFull36}{OrderlyDifferentialPolynomialXmpPageEmpty36}
\pastebutton{OrderlyDifferentialPolynomialXmpPageFull36}{\hidepaste}
\tab{5}\spadcommand{ig * prf - lcf * g * lg\free{ig }\free{prf }\free{lcf }\free{g }\free{lg }}
\indentrel{3}\begin{verbatim}
           6         2  3        2  2
   (36)  2z  z w  - w  z  w  + 3w  z  w z
          1  2 4    1  1  3     1  1  2 2
   Type: OrderlyDifferentialPolynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OrderlyDifferentialPolynomialXmpPageEmpty36}
\begin{paste}{OrderlyDifferentialPolynomialXmpPageEmpty36}{OrderlyDifferentialPolynomialXmpPagePatch36}
\pastebutton{OrderlyDifferentialPolynomialXmpPageEmpty36}{\showpaste}
\tab{5}\spadcommand{ig * prf - lcf * g * lg\free{ig }\free{prf }\free{lcf }\free{g }\free{lg }}
\end{paste}\end{patch}
```

——-

3.83 op.ht

Operator

— op.ht —

```
\begin{page}{OperatorXmpPage}{Operator}
\beginscroll
Given any ring \spad{R}, the ring of the \spadtype{Integer}-linear
operators over \spad{R} is called \spadtype{Operator(R)}.  To create
an operator over \spad{R}, first create a basic operator using the
operation \spadfun{operator}, and then convert it to
\spadtype{Operator(R)} for the \spad{R} you want.
%
\xtc{
```

```
We choose \spad{R} to be the two by two matrices over the integers.
}{
\spadpaste{R := SQMATRIX(2, INT)\bound{r}}
}
\xtc{
Create the operator \spad{tilde} on \spad{R}.
}{
\spadpaste{t := operator("tilde") :: OP(R) \free{r}\bound{t}}
}
%
Since \spadtype{Operator} is unexposed we must either package-call
operations from it, or expose it explicitly.  For convenience we will
do the latter.
%
\noOutputXtc{
Expose \spad{Operator}.
}{
\spadpaste{)set expose add constructor Operator \free{t}\bound{expose}}
}
%
To attach an evaluation function (from \spad{R} to \spad{R}) to an
operator over \spad{R}, use \spad{evaluate(op, f)} where \spad{op}
is an operator over \spad{R} and \spad{f} is a function \spad{R ->
R}.
This needs to be done only once when the operator is defined.
Note that \spad{f} must be \spadtype{Integer}-linear (that is,
\spad{f(ax+y) = a f(x) + f(y)} for any integer \spad{a}, and any
\spad{x} and \spad{y} in \spad{R}).
%
\xtc{
We now attach the transpose map to the above operator \spad{t}.
}{
\spadpaste{evaluate(t, m +-> transpose m)\free{expose}\free{t}\bound{evt}}
}
%
Operators can be manipulated formally as in any ring: \spadop{+} is the
pointwise addition and \spadop{*} is composition.
Any element \spad{x} of \spad{R} can be converted to an operator
\subscriptText{\tt op}{\tt x}
over \spad{R}, and the evaluation function of
\subscriptText{\tt op}{\tt x}
is left-multiplication by \spad{x}.
%
\xtc{
Multiplying on the
left by this matrix swaps the two rows.
}{
\spadpaste{s : R := matrix [[0, 1], [1, 0]]\bound{s}}
}
%
```

```
\xtc{
Can you guess what is the action of the following operator?
}{
\spadpaste{rho := t * s\free{evt s}\bound{rho}}
}
%
%
\xtc{
Hint: applying \spad{rho} four times gives the identity, so
\spad{rho**4-1} should return 0 when applied to any two by two matrix.
}{
\spadpaste{z := rho**4 - 1\free{rho}\bound{z}}
}
%
%
\xtc{
Now check with this matrix.
}{
\spadpaste{m:R := matrix [[1, 2], [3, 4]]\bound{m}}
}
\xtc{
}{
\spadpaste{z m\free{z m}}
}
%
%
\xtc{
As you have probably guessed by now, \spad{rho} acts on matrices
by rotating the elements clockwise.
}{
\spadpaste{rho m\free{rho m}}
}
\xtc{
}{
\spadpaste{rho rho m\free{rho m}}
}
\xtc{
}{
\spadpaste{(rho**3) m\free{rho m}}
}
%
%
\xtc{
Do the swapping of rows and transposition commute?
We can check by computing their bracket.
}{
\spadpaste{b := t * s - s * t\free{s evt}\bound{b}}
}
%
%
```

```
\xtc{
Now apply it to \spad{m}.
}{
\spadpaste{b m \free{b m}}
}
%

Next we demonstrate how to define a differential operator
on a polynomial ring.
\xtc{
This is the recursive definition of the \spad{n}-th Legendre polynomial.
}{
\begin{spadsrc}[\bound{l}]
L n ==
  n = 0 => 1
  n = 1 => x
  (2*n-1)/n * x * L(n-1) - (n-1)/n * L(n-2)
\end{spadsrc}
}
\xtc{
Create the differential operator \texht{$d \over {dx}$}{\spad{d/dx}} on
polynomials in \spad{x} over the rational numbers.
}{
\spadpaste{dx := operator("D") :: OP(POLY FRAC INT) \bound{dx}}
}
\xtc{
Now attach the map to it.
}{
\spadpaste{evaluate(dx, p +-> D(p, 'x)) \free{dx}\bound{edx}}
}
\xtc{
This is the differential equation satisfied by the \spad{n}-th
Legendre polynomial.
}{
\spadpaste{E n == (1 - x**2) * dx**2 - 2 * x * dx + n*(n+1)
\free{edx}\bound{E}}
}
\xtc{
Now we verify this for \spad{n = 15}.
Here is the polynomial.
}{
\spadpaste{L 15 \free{L}}
}
\xtc{
Here is the operator.
}{
\spadpaste{E 15 \free{E}}
}
\xtc{
Here is the evaluation.
```

```
}{
\spadpaste{(E 15)(L 15) \free{L E}}
}
\endscroll
\autobuttons
\end{page}

\begin{patch}{OperatorXmpPagePatch1}
\begin{paste}{OperatorXmpPageFull1}{OperatorXmpPageEmpty1}
\pastebutton{OperatorXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{R := SQMATRIX(2, INT)\bound{r }}
\indentrel{3}\begin{verbatim}
   (1)  SquareMatrix(2,Integer)
                                          Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OperatorXmpPageEmpty1}
\begin{paste}{OperatorXmpPageEmpty1}{OperatorXmpPagePatch1}
\pastebutton{OperatorXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{R := SQMATRIX(2, INT)\bound{r }}
\end{paste}\end{patch}

\begin{patch}{OperatorXmpPagePatch2}
\begin{paste}{OperatorXmpPageFull2}{OperatorXmpPageEmpty2}
\pastebutton{OperatorXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{t := operator("tilde") :: OP(R)\free{r }\bound{t }}
\indentrel{3}\begin{verbatim}
   (2)  tilde
                 Type: Operator SquareMatrix(2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OperatorXmpPageEmpty2}
\begin{paste}{OperatorXmpPageEmpty2}{OperatorXmpPagePatch2}
\pastebutton{OperatorXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{t := operator("tilde") :: OP(R)\free{r }\bound{t }}
\end{paste}\end{patch}

\begin{patch}{OperatorXmpPagePatch3}
\begin{paste}{OperatorXmpPageFull3}{OperatorXmpPageEmpty3}
\pastebutton{OperatorXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{)set expose add constructor Operator\free{t }\bound{expose }}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OperatorXmpPageEmpty3}
\begin{paste}{OperatorXmpPageEmpty3}{OperatorXmpPagePatch3}
\pastebutton{OperatorXmpPageEmpty3}{\showpaste}
```

```
\tab{5}\spadcommand{)set expose add constructor Operator\free{t }\bound{expose }}
\end{paste}\end{patch}

\begin{patch}{OperatorXmpPagePatch4}
\begin{paste}{OperatorXmpPageFull4}{OperatorXmpPageEmpty4}
\pastebutton{OperatorXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{evaluate(t, m +-> transpose m)\free{expose }\free{t }\bound{evt }}
\indentrel{3}\begin{verbatim}
   (3)  tilde
                  Type: Operator SquareMatrix(2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OperatorXmpPageEmpty4}
\begin{paste}{OperatorXmpPageEmpty4}{OperatorXmpPagePatch4}
\pastebutton{OperatorXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{evaluate(t, m +-> transpose m)\free{expose }\free{t }\bound{evt }}
\end{paste}\end{patch}

\begin{patch}{OperatorXmpPagePatch5}
\begin{paste}{OperatorXmpPageFull5}{OperatorXmpPageEmpty5}
\pastebutton{OperatorXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{s : R := matrix [[0, 1], [1, 0]]\bound{s }}
\indentrel{3}\begin{verbatim}
        0  1
   (4)
        1  0
                          Type: SquareMatrix(2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OperatorXmpPageEmpty5}
\begin{paste}{OperatorXmpPageEmpty5}{OperatorXmpPagePatch5}
\pastebutton{OperatorXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{s : R := matrix [[0, 1], [1, 0]]\bound{s }}
\end{paste}\end{patch}

\begin{patch}{OperatorXmpPagePatch6}
\begin{paste}{OperatorXmpPageFull6}{OperatorXmpPageEmpty6}
\pastebutton{OperatorXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{rho := t * s\free{evt s }\bound{rho }}
\indentrel{3}\begin{verbatim}
            0  1
   (5)  tilde
            1  0
                  Type: Operator SquareMatrix(2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OperatorXmpPageEmpty6}
```

```
\begin{paste}{OperatorXmpPageEmpty6}{OperatorXmpPagePatch6}
\pastebutton{OperatorXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{rho := t * s\free{evt s }\bound{rho }}
\end{paste}\end{patch}

\begin{patch}{OperatorXmpPagePatch7}
\begin{paste}{OperatorXmpPageFull7}{OperatorXmpPageEmpty7}
\pastebutton{OperatorXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{z := rho**4 - 1\free{rho }\bound{z }}
\indentrel{3}\begin{verbatim}
   (6)
              0  1     0  1     0  1     0  1
   - 1 + tilde    tilde    tilde    tilde
              1  0     1  0     1  0     1  0
                 Type: Operator SquareMatrix(2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OperatorXmpPageEmpty7}
\begin{paste}{OperatorXmpPageEmpty7}{OperatorXmpPagePatch7}
\pastebutton{OperatorXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{z := rho**4 - 1\free{rho }\bound{z }}
\end{paste}\end{patch}

\begin{patch}{OperatorXmpPagePatch8}
\begin{paste}{OperatorXmpPageFull8}{OperatorXmpPageEmpty8}
\pastebutton{OperatorXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{m:R := matrix [[1, 2], [3, 4]]\bound{m }}
\indentrel{3}\begin{verbatim}
        1  2
   (7)
        3  4
                          Type: SquareMatrix(2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OperatorXmpPageEmpty8}
\begin{paste}{OperatorXmpPageEmpty8}{OperatorXmpPagePatch8}
\pastebutton{OperatorXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{m:R := matrix [[1, 2], [3, 4]]\bound{m }}
\end{paste}\end{patch}

\begin{patch}{OperatorXmpPagePatch9}
\begin{paste}{OperatorXmpPageFull9}{OperatorXmpPageEmpty9}
\pastebutton{OperatorXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{z m\free{z m }}
\indentrel{3}\begin{verbatim}
        0  0
   (8)
        0  0
```

```
                          Type: SquareMatrix(2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OperatorXmpPageEmpty9}
\begin{paste}{OperatorXmpPageEmpty9}{OperatorXmpPagePatch9}
\pastebutton{OperatorXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{z m\free{z m }}
\end{paste}\end{patch}

\begin{patch}{OperatorXmpPagePatch10}
\begin{paste}{OperatorXmpPageFull10}{OperatorXmpPageEmpty10}
\pastebutton{OperatorXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{rho m\free{rho m }}
\indentrel{3}\begin{verbatim}
        3  1
   (9)
        4  2
                          Type: SquareMatrix(2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OperatorXmpPageEmpty10}
\begin{paste}{OperatorXmpPageEmpty10}{OperatorXmpPagePatch10}
\pastebutton{OperatorXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{rho m\free{rho m }}
\end{paste}\end{patch}

\begin{patch}{OperatorXmpPagePatch11}
\begin{paste}{OperatorXmpPageFull11}{OperatorXmpPageEmpty11}
\pastebutton{OperatorXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{rho rho m\free{rho m }}
\indentrel{3}\begin{verbatim}
         4  3
   (10)
         2  1
                          Type: SquareMatrix(2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OperatorXmpPageEmpty11}
\begin{paste}{OperatorXmpPageEmpty11}{OperatorXmpPagePatch11}
\pastebutton{OperatorXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{rho rho m\free{rho m }}
\end{paste}\end{patch}

\begin{patch}{OperatorXmpPagePatch12}
\begin{paste}{OperatorXmpPageFull12}{OperatorXmpPageEmpty12}
\pastebutton{OperatorXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{(rho**3) m\free{rho m }}
```

```
\indentrel{3}\begin{verbatim}
        2  4
   (11)
        1  3
                          Type: SquareMatrix(2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OperatorXmpPageEmpty12}
\begin{paste}{OperatorXmpPageEmpty12}{OperatorXmpPagePatch12}
\pastebutton{OperatorXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{(rho**3) m\free{rho m }}
\end{paste}\end{patch}

\begin{patch}{OperatorXmpPagePatch13}
\begin{paste}{OperatorXmpPageFull13}{OperatorXmpPageEmpty13}
\pastebutton{OperatorXmpPageFull13}{\hidepaste}
\tab{5}\spadcommand{b := t * s - s * t\free{s evt }\bound{b }}
\indentrel{3}\begin{verbatim}
          0  1            0  1
   (12)  -     tilde + tilde
          1  0            1  0
                 Type: Operator SquareMatrix(2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OperatorXmpPageEmpty13}
\begin{paste}{OperatorXmpPageEmpty13}{OperatorXmpPagePatch13}
\pastebutton{OperatorXmpPageEmpty13}{\showpaste}
\tab{5}\spadcommand{b := t * s - s * t\free{s evt }\bound{b }}
\end{paste}\end{patch}

\begin{patch}{OperatorXmpPagePatch14}
\begin{paste}{OperatorXmpPageFull14}{OperatorXmpPageEmpty14}
\pastebutton{OperatorXmpPageFull14}{\hidepaste}
\tab{5}\spadcommand{b m\free{b m }}
\indentrel{3}\begin{verbatim}
        1  - 3
   (13)
        3  - 1
                          Type: SquareMatrix(2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OperatorXmpPageEmpty14}
\begin{paste}{OperatorXmpPageEmpty14}{OperatorXmpPagePatch14}
\pastebutton{OperatorXmpPageEmpty14}{\showpaste}
\tab{5}\spadcommand{b m\free{b m }}
\end{paste}\end{patch}
```

```
\begin{patch}{OperatorXmpPagePatch15}
\begin{paste}{OperatorXmpPageFull15}{OperatorXmpPageEmpty15}
\pastebutton{OperatorXmpPageFull15}{\hidepaste}
\tab{5}\spadcommand{L n ==
  n = 0 => 1
  n = 1 => x
  (2*n-1)/n * x * L(n-1) - (n-1)/n * L(n-2)
\bound{l }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OperatorXmpPageEmpty15}
\begin{paste}{OperatorXmpPageEmpty15}{OperatorXmpPagePatch15}
\pastebutton{OperatorXmpPageEmpty15}{\showpaste}
\tab{5}\spadcommand{L n ==
  n = 0 => 1
  n = 1 => x
  (2*n-1)/n * x * L(n-1) - (n-1)/n * L(n-2)
\bound{l }}
\end{paste}\end{patch}

\begin{patch}{OperatorXmpPagePatch16}
\begin{paste}{OperatorXmpPageFull16}{OperatorXmpPageEmpty16}
\pastebutton{OperatorXmpPageFull16}{\hidepaste}
\tab{5}\spadcommand{dx := operator("D") :: OP(POLY FRAC INT)\bound{dx }}
\indentrel{3}\begin{verbatim}
   (15)  D
             Type: Operator Polynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OperatorXmpPageEmpty16}
\begin{paste}{OperatorXmpPageEmpty16}{OperatorXmpPagePatch16}
\pastebutton{OperatorXmpPageEmpty16}{\showpaste}
\tab{5}\spadcommand{dx := operator("D") :: OP(POLY FRAC INT)\bound{dx }}
\end{paste}\end{patch}

\begin{patch}{OperatorXmpPagePatch17}
\begin{paste}{OperatorXmpPageFull17}{OperatorXmpPageEmpty17}
\pastebutton{OperatorXmpPageFull17}{\hidepaste}
\tab{5}\spadcommand{evaluate(dx, p +-> D(p, 'x))\free{dx }\bound{edx }}
\indentrel{3}\begin{verbatim}
   (16)  D
             Type: Operator Polynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OperatorXmpPageEmpty17}
```

```
\begin{paste}{OperatorXmpPageEmpty17}{OperatorXmpPagePatch17}
\pastebutton{OperatorXmpPageEmpty17}{\showpaste}
\tab{5}\spadcommand{evaluate(dx, p +-> D(p, 'x))\free{dx }\bound{edx }}
\end{paste}\end{patch}

\begin{patch}{OperatorXmpPagePatch18}
\begin{paste}{OperatorXmpPageFull18}{OperatorXmpPageEmpty18}
\pastebutton{OperatorXmpPageFull18}{\hidepaste}
\tab{5}\spadcommand{E n == (1 - x**2) * dx**2 - 2 * x * dx + n*(n+1)\free{edx }\bound{E }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OperatorXmpPageEmpty18}
\begin{paste}{OperatorXmpPageEmpty18}{OperatorXmpPagePatch18}
\pastebutton{OperatorXmpPageEmpty18}{\showpaste}
\tab{5}\spadcommand{E n == (1 - x**2) * dx**2 - 2 * x * dx + n*(n+1)\free{edx }\bound{E }}
\end{paste}\end{patch}

\begin{patch}{OperatorXmpPagePatch19}
\begin{paste}{OperatorXmpPageFull19}{OperatorXmpPageEmpty19}
\pastebutton{OperatorXmpPageFull19}{\hidepaste}
\tab{5}\spadcommand{L 15\free{L }}
\indentrel{3}\begin{verbatim}
   (18)
     9694845  15   35102025  13   50702925  11
      x   -  x   +  x
       2048          2048           2048
   +
       37182145  9   14549535  7   2909907  5   255255  3
     -  x  +  x  -  x  +  x
         2048          2048          2048        2048
   +
       6435
     -  x
       2048
                       Type: Polynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OperatorXmpPageEmpty19}
\begin{paste}{OperatorXmpPageEmpty19}{OperatorXmpPagePatch19}
\pastebutton{OperatorXmpPageEmpty19}{\showpaste}
\tab{5}\spadcommand{L 15\free{L }}
\end{paste}\end{patch}

\begin{patch}{OperatorXmpPagePatch20}
\begin{paste}{OperatorXmpPageFull20}{OperatorXmpPageEmpty20}
\pastebutton{OperatorXmpPageFull20}{\hidepaste}
```

```
\tab{5}\spadcommand{E 15\free{E }}
\indentrel{3}\begin{verbatim}
                        2      2
   (19)  240 - 2x D - (x  - 1)D
             Type: Operator Polynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OperatorXmpPageEmpty20}
\begin{paste}{OperatorXmpPageEmpty20}{OperatorXmpPagePatch20}
\pastebutton{OperatorXmpPageEmpty20}{\showpaste}
\tab{5}\spadcommand{E 15\free{E }}
\end{paste}\end{patch}

\begin{patch}{OperatorXmpPagePatch21}
\begin{paste}{OperatorXmpPageFull21}{OperatorXmpPageEmpty21}
\pastebutton{OperatorXmpPageFull21}{\hidepaste}
\tab{5}\spadcommand{(E 15)(L 15)\free{L E }}
\indentrel{3}\begin{verbatim}
   (20)  0
                      Type: Polynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OperatorXmpPageEmpty21}
\begin{paste}{OperatorXmpPageEmpty21}{OperatorXmpPagePatch21}
\pastebutton{OperatorXmpPageEmpty21}{\showpaste}
\tab{5}\spadcommand{(E 15)(L 15)\free{L E }}
\end{paste}\end{patch}
```

————

3.84 ovar.ht

OrderedVariableList

— ovar.ht —

```
\begin{page}{OrderedVariableListXmpPage}{OrderedVariableList}
\beginscroll

The domain \spadtype{OrderedVariableList} provides symbols
which are restricted to a particular list and have a definite
ordering. Those two features are specified by a \spadtype{List Symbol}
object that is the argument to the domain.
\xtc{
```

```
This is a sample ordering of three symbols.
}{
\spadpaste{ls:List Symbol:=['x,'a,'z] \bound{ls}}
}
\xtc{
Let's build the domain
}{
\spadpaste{Z:=OVAR ls \bound{Z} \free{ls}}
}
\xtc{
How many variables does it have?
}{
\spadpaste{size()$Z \free{Z}}
}
\xtc{
They are (in the imposed order)
}{
\spadpaste{lv:=[index(i::PI)$Z for i in 1..size()$Z] \bound{lv}\free{Z}}
}
\xtc{
Check that the ordering is right
}{
\spadpaste{sorted?(>,lv) \free{lv}}
}
\endscroll
\autobuttons
\end{page}

\begin{patch}{OrderedVariableListXmpPagePatch1}
\begin{paste}{OrderedVariableListXmpPageFull1}{OrderedVariableListXmpPageEmpty1}
\pastebutton{OrderedVariableListXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{ls:List Symbol:=['x,'a,'z]\bound{ls }}
\indentrel{3}\begin{verbatim}
   (1)  [x,a,z]
                                      Type: List Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OrderedVariableListXmpPageEmpty1}
\begin{paste}{OrderedVariableListXmpPageEmpty1}{OrderedVariableListXmpPagePatch1}
\pastebutton{OrderedVariableListXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{ls:List Symbol:=['x,'a,'z]\bound{ls }}
\end{paste}\end{patch}

\begin{patch}{OrderedVariableListXmpPagePatch2}
\begin{paste}{OrderedVariableListXmpPageFull2}{OrderedVariableListXmpPageEmpty2}
\pastebutton{OrderedVariableListXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{Z:=OVAR ls\bound{Z }\free{ls }}
\indentrel{3}\begin{verbatim}
   (2)  OrderedVariableList [x,a,z]
```

```
                                           Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OrderedVariableListXmpPageEmpty2}
\begin{paste}{OrderedVariableListXmpPageEmpty2}{OrderedVariableListXmpPagePatch2}
\pastebutton{OrderedVariableListXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{Z:=OVAR ls\bound{Z }\free{ls }}
\end{paste}\end{patch}

\begin{patch}{OrderedVariableListXmpPagePatch3}
\begin{paste}{OrderedVariableListXmpPageFull3}{OrderedVariableListXmpPageEmpty3}
\pastebutton{OrderedVariableListXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{size()$Z\free{Z }}
\indentrel{3}\begin{verbatim}
   (3)  3
                                 Type: NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OrderedVariableListXmpPageEmpty3}
\begin{paste}{OrderedVariableListXmpPageEmpty3}{OrderedVariableListXmpPagePatch3}
\pastebutton{OrderedVariableListXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{size()$Z\free{Z }}
\end{paste}\end{patch}

\begin{patch}{OrderedVariableListXmpPagePatch4}
\begin{paste}{OrderedVariableListXmpPageFull4}{OrderedVariableListXmpPageEmpty4}
\pastebutton{OrderedVariableListXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{lv:=[index(i::PI)$Z for i in 1..size()$Z]\bound{lv }\free{Z }}
\indentrel{3}\begin{verbatim}
   (4)  [x,a,z]
                  Type: List OrderedVariableList [x,a,z]
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OrderedVariableListXmpPageEmpty4}
\begin{paste}{OrderedVariableListXmpPageEmpty4}{OrderedVariableListXmpPagePatch4}
\pastebutton{OrderedVariableListXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{lv:=[index(i::PI)$Z for i in 1..size()$Z]\bound{lv }\free{Z }}
\end{paste}\end{patch}

\begin{patch}{OrderedVariableListXmpPagePatch5}
\begin{paste}{OrderedVariableListXmpPageFull5}{OrderedVariableListXmpPageEmpty5}
\pastebutton{OrderedVariableListXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{sorted?(>,lv)\free{lv }}
\indentrel{3}\begin{verbatim}
   (5)  true
                                          Type: Boolean
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{OrderedVariableListXmpPageEmpty5}
\begin{paste}{OrderedVariableListXmpPageEmpty5}{OrderedVariableListXmpPagePatch5}
\pastebutton{OrderedVariableListXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{sorted?(>,lv)\free{lv }}
\end{paste}\end{patch}
```

———-

3.85 perman.ht

Permanent

— perman.ht —

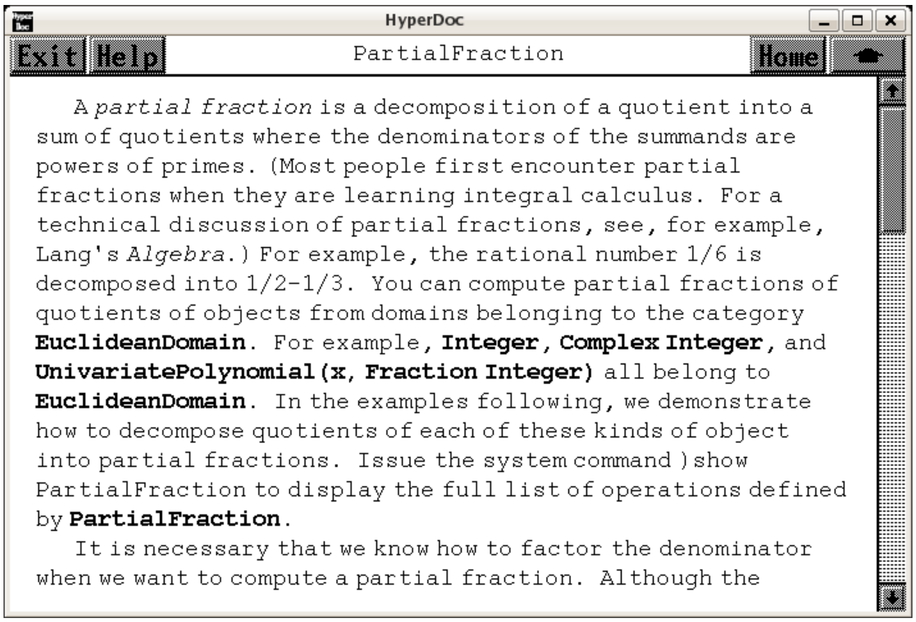
```
\begin{page}{PermanentXmpPage}{Permanent}
\beginscroll
The package \spadtype{Permanent} provides the function
\spadfunFrom{permanent}{Permanent} for square matrices.
The \spadfunFrom{permanent}{Permanent} of a square matrix can be computed
in the same way as the determinant by expansion of minors except that for
the permanent the sign for each element is \spad{1}, rather than being
\spad{1} if the row plus column indices is positive and \spad{-1}
otherwise.
This function is much more difficult to compute efficiently than the
\spadfunFrom{determinant}{Matrix}.
An example of the use of \spadfunFrom{permanent}{Permanent} is the
calculation of the \eth{\spad{n}} derangement number, defined to be
the number of different possibilities for \spad{n} couples to dance
but never with their own spouse.
\xtc{
Consider an \spad{n} by \spad{n} matrix with entries
\spad{0} on the diagonal and
\spad{1} elsewhere.
Think of the rows as one-half of each couple (for example, the males)
and the columns the other half.  The permanent of such a matrix gives
the desired derangement number.
}{
\begin{spadsrc}[\bound{kn}]
kn n ==
  r : MATRIX INT := new(n,n,1)
  for i in 1..n repeat
    r.i.i := 0
  r
\end{spadsrc}
```

```
}
\xtc{
Here are some derangement numbers, which you see grow quite fast.
}{
\spadpaste{permanent(kn(5) :: SQMATRIX(5,INT)) \free{kn}}
}
\xtc{
}{
\spadpaste{[permanent(kn(n) :: SQMATRIX(n,INT)) for n in 1..13] \free{kn}}
}
\endscroll
\autobuttons
\end{page}

\begin{patch}{PermanentXmpPagePatch1}
\begin{paste}{PermanentXmpPageFull1}{PermanentXmpPageEmpty1}
\pastebutton{PermanentXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{kn n ==
  r : MATRIX INT := new(n,n,1)
  for i in 1..n repeat
    r.i.i := 0
  r
\bound{kn }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PermanentXmpPageEmpty1}
\begin{paste}{PermanentXmpPageEmpty1}{PermanentXmpPagePatch1}
\pastebutton{PermanentXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{kn n ==
  r : MATRIX INT := new(n,n,1)
  for i in 1..n repeat
    r.i.i := 0
  r
\bound{kn }}
\end{paste}\end{patch}

\begin{patch}{PermanentXmpPagePatch2}
\begin{paste}{PermanentXmpPageFull2}{PermanentXmpPageEmpty2}
\pastebutton{PermanentXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{permanent(kn(5) :: SQMATRIX(5,INT))\free{kn }}
\indentrel{3}\begin{verbatim}
   (2)  44
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PermanentXmpPageEmpty2}
```

```
\begin{paste}{PermanentXmpPageEmpty2}{PermanentXmpPagePatch2}
\pastebutton{PermanentXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{permanent(kn(5) :: SQMATRIX(5,INT))\free{kn }}
\end{paste}\end{patch}

\begin{patch}{PermanentXmpPagePatch3}
\begin{paste}{PermanentXmpPageFull3}{PermanentXmpPageEmpty3}
\pastebutton{PermanentXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{[permanent(kn(n) :: SQMATRIX(n,INT)) for n in 1..13]\free{kn }}
\indentrel{3}\begin{verbatim}
   (3)
   [0, 1, 2, 9, 44, 265, 1854, 14833, 133496, 1334961,
    14684570, 176214841, 2290792932]
                            Type: List NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PermanentXmpPageEmpty3}
\begin{paste}{PermanentXmpPageEmpty3}{PermanentXmpPagePatch3}
\pastebutton{PermanentXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{[permanent(kn(n) :: SQMATRIX(n,INT)) for n in 1..13]\free{kn }}
\end{paste}\end{patch}
```

3.86 pfr.ht

PartialFraction



⇐ "Axiom Number Types" (NumberPage) 3.80 on page 1021
⇒ "FullPartialFracExpansion" (FullPartialFracExpansionXmpPage) 3.46 on page 549

— **pfr.ht** —

```
\begin{page}{PartialFractionXmpPage}{PartialFraction}
\beginscroll
A {\it partial fraction} is a decomposition of a quotient into a sum
of quotients where the denominators of the summands are powers of
primes.\footnote{Most people first encounter partial fractions when
they are learning integral calculus.  For a technical discussion of
partial fractions, see, for example, Lang's {\it Algebra.}} For
example, the rational number \spad{1/6} is decomposed into
\spad{1/2-1/3}.  You can compute partial fractions of quotients of
objects from domains belonging to the category
\spadtype{EuclideanDomain}.  For
example, \spadtype{Integer}, \spadtype{Complex Integer}, and
\spadtype{UnivariatePolynomial(x, Fraction Integer)} all belong to
\spadtype{EuclideanDomain}.  In the examples following, we demonstrate
how to decompose quotients of each of these kinds of object into
partial fractions.  Issue the system command \spadcmd{)show
PartialFraction} to display the full list of operations defined by
\spadtype{PartialFraction}.
```

```
It is necessary that we know how to factor the denominator when we
want to compute a partial fraction.  Although the interpreter can
often do this automatically, it may be necessary for you to include a
call to \spadfun{factor}.  In these examples, it is not necessary to
factor the denominators explicitly.
%
\xtc{
The main operation for computing partial fractions is called
\spadfunFrom{partialFraction}{PartialFraction} and we use this to
compute a decomposition of \spad{1 / 10!}.
The first argument to \spadfunFrom{partialFraction}{PartialFraction} is
the numerator of the quotient and the second argument is the factored
denominator.
}{
\spadpaste{partialFraction(1,factorial 10) \bound{prev1}}
}
\xtc{
Since the denominators are powers of primes, it may be possible
to expand the numerators further with respect to those primes. Use the
operation \spadfunFrom{padicFraction}{PartialFraction} to do this.
}{
\spadpaste{f := padicFraction(\%) \free{prev1}\bound{f}}
}
%
%
\xtc{
The operation \spadfunFrom{compactFraction}{PartialFraction} returns an
expanded fraction into the usual form.
The compacted version is used internally for computational efficiency.
}{
\spadpaste{compactFraction(f) \free{f}}
}
%
\xtc{
You can add, subtract, multiply
and divide partial fractions.  In addition, you can extract the parts
of the decomposition.
\spadfunFrom{numberOfFractionalTerms}{PartialFraction} computes
the number of terms in the fractional part.
This does not include the whole part of the fraction,
which you get by calling \spadfunFrom{wholePart}{PartialFraction}.
In this example, the whole part is just \spad{0}.
}{
\spadpaste{numberOfFractionalTerms(f) \free{f}}
}
\xtc{
The operation \spadfunFrom{nthFractionalTerm}{PartialFraction} returns
the individual terms in the decomposition.  Notice that the object
returned is a partial fraction itself.
\spadfunFrom{firstNumer}{PartialFraction} and
```

```
\spadfunFrom{firstDenom}{PartialFraction} extract the numerator and
denominator of the first term of the fraction.
}{
\spadpaste{nthFractionalTerm(f,3) \free{f}}
}
%

%
\xtc{
Given two gaussian integers (see
\downlink{`Complex'}{ComplexXmpPage}\ignore{Complex}), you can
decompose their quotient into a partial fraction.
}{
\spadpaste{partialFraction(1,- 13 + 14 * \%i) \bound{prev2}}
}
%
\xtc{
To convert back to a quotient, simply use a conversion.
}{
\spadpaste{\% :: Fraction Complex Integer \free{prev2}}
}

To conclude this section, we compute the decomposition of
\texht{\narrowDisplay{1 \over {{(x + 1)}{(x + 2)}^2{(x + 3)}^3{(x + 4)}^4}}}{
\begin{verbatim}
                   1
     -------------------------------
                   2       3       4
     (x + 1)(x + 2) (x + 3) (x + 4)
\end{verbatim}
}
The polynomials in this object have type
\spadtype{UnivariatePolynomial(x, Fraction Integer)}.
%
\xtc{
We use the \spadfunFrom{primeFactor}{Factored} operation (see
\downlink{`Factored'}{FactoredXmpPage}
\ignore{Factored}) to create the denominator in factored form directly.
}{
\spadpaste{u : FR UP(x, FRAC INT) := reduce(*,[primeFactor(x+i,i)
for i in 1..4]) \bound{u}}
}
%
%
\xtc{
These are the compact and expanded partial fractions for the quotient.
}{
\spadpaste{partialFraction(1,u) \free{u}\bound{prev3}}
}
\xtc{
```

```
}{
\spadpaste{padicFraction \% \free{prev3}}
}

All see \downlink{‘FullPartialFracExpansion’}
{FullPartialFracExpansionXmpPage}
\ignore{FullPartialFracExpansion} for examples of
factor-free conversion of quotients to full partial fractions.
\endscroll
\autobuttons
\end{page}

\begin{patch}{PartialFractionXmpPagePatch1}
\begin{paste}{PartialFractionXmpPageFull1}{PartialFractionXmpPageEmpty1}
\pastebutton{PartialFractionXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{partialFraction(1,factorial 10)\bound{prev1 }}
\indentrel{3}\begin{verbatim}
        159   23   12   1
   (1)   -  -  +
          8    4    2   7
         2    3    5
                          Type: PartialFraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PartialFractionXmpPageEmpty1}
\begin{paste}{PartialFractionXmpPageEmpty1}{PartialFractionXmpPagePatch1}
\pastebutton{PartialFractionXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{partialFraction(1,factorial 10)\bound{prev1 }}
\end{paste}\end{patch}

\begin{patch}{PartialFractionXmpPagePatch2}
\begin{paste}{PartialFractionXmpPageFull2}{PartialFractionXmpPageEmpty2}
\pastebutton{PartialFractionXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{f := padicFraction(\%)\free{prev1 }\bound{f }}
\indentrel{3}\begin{verbatim}
   (2)
   1    1    1    1    1    1    2    1    2   2    2   1
    +  +  +  +  +  -  -  -  -  -  +
   2    4    5    6    7    8    2    3    4   5    2   7
       2    2    2    2    2    3    3    3        5
                          Type: PartialFraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PartialFractionXmpPageEmpty2}
\begin{paste}{PartialFractionXmpPageEmpty2}{PartialFractionXmpPagePatch2}
\pastebutton{PartialFractionXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{f := padicFraction(\%)\free{prev1 }\bound{f }}
\end{paste}\end{patch}
```

```
\begin{patch}{PartialFractionXmpPagePatch3}
\begin{paste}{PartialFractionXmpPageFull3}{PartialFractionXmpPageEmpty3}
\pastebutton{PartialFractionXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{compactFraction(f)\free{f }}
\indentrel{3}\begin{verbatim}
        159   23   12   1
   (3)   -  -  +
          8    4    2   7
         2    3    5
                          Type: PartialFraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PartialFractionXmpPageEmpty3}
\begin{paste}{PartialFractionXmpPageEmpty3}{PartialFractionXmpPagePatch3}
\pastebutton{PartialFractionXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{compactFraction(f)\free{f }}
\end{paste}\end{patch}

\begin{patch}{PartialFractionXmpPagePatch4}
\begin{paste}{PartialFractionXmpPageFull4}{PartialFractionXmpPageEmpty4}
\pastebutton{PartialFractionXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{numberOfFractionalTerms(f)\free{f }}
\indentrel{3}\begin{verbatim}
   (4)  12
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PartialFractionXmpPageEmpty4}
\begin{paste}{PartialFractionXmpPageEmpty4}{PartialFractionXmpPagePatch4}
\pastebutton{PartialFractionXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{numberOfFractionalTerms(f)\free{f }}
\end{paste}\end{patch}

\begin{patch}{PartialFractionXmpPagePatch5}
\begin{paste}{PartialFractionXmpPageFull5}{PartialFractionXmpPageEmpty5}
\pastebutton{PartialFractionXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{nthFractionalTerm(f,3)\free{f }}
\indentrel{3}\begin{verbatim}
         1
   (5)
         5
        2
                          Type: PartialFraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PartialFractionXmpPageEmpty5}
```

```
\begin{paste}{PartialFractionXmpPageEmpty5}{PartialFractionXmpPagePatch5}
\pastebutton{PartialFractionXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{nthFractionalTerm(f,3)\free{f }}
\end{paste}\end{patch}

\begin{patch}{PartialFractionXmpPagePatch6}
\begin{paste}{PartialFractionXmpPageFull6}{PartialFractionXmpPageEmpty6}
\pastebutton{PartialFractionXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{partialFraction(1,- 13 + 14 * \%i)\bound{prev2 }}
\indentrel{3}\begin{verbatim}
             1        4
   (6)  -  +
         1 + 2%i   3 + 8%i
                 Type: PartialFraction Complex Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PartialFractionXmpPageEmpty6}
\begin{paste}{PartialFractionXmpPageEmpty6}{PartialFractionXmpPagePatch6}
\pastebutton{PartialFractionXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{partialFraction(1,- 13 + 14 * \%i)\bound{prev2 }}
\end{paste}\end{patch}

\begin{patch}{PartialFractionXmpPagePatch7}
\begin{paste}{PartialFractionXmpPageFull7}{PartialFractionXmpPageEmpty7}
\pastebutton{PartialFractionXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{\% :: Fraction Complex Integer\free{prev2 }}
\indentrel{3}\begin{verbatim}
              %i
   (7)  -
         14 + 13%i
                        Type: Fraction Complex Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PartialFractionXmpPageEmpty7}
\begin{paste}{PartialFractionXmpPageEmpty7}{PartialFractionXmpPagePatch7}
\pastebutton{PartialFractionXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{\% :: Fraction Complex Integer\free{prev2 }}
\end{paste}\end{patch}

\begin{patch}{PartialFractionXmpPagePatch8}
\begin{paste}{PartialFractionXmpPageFull8}{PartialFractionXmpPageEmpty8}
\pastebutton{PartialFractionXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{u : FR UP(x, FRAC INT) := reduce(*,[primeFactor(x+i,i) for i in 1..4])\bound{u }}
\indentrel{3}\begin{verbatim}
                      2       3       4
   (8)  (x + 1)(x + 2) (x + 3) (x + 4)
Type: Factored UnivariatePolynomial(x,Fraction Integer)
\end{verbatim}
```

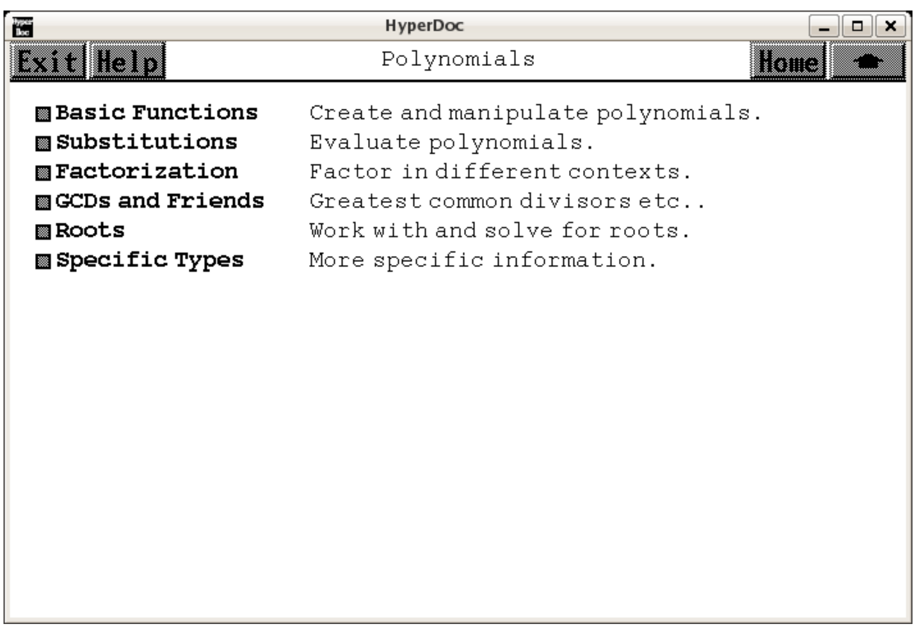
```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PartialFractionXmpPageEmpty8}
\begin{paste}{PartialFractionXmpPageEmpty8}{PartialFractionXmpPagePatch8}
\pastebutton{PartialFractionXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{u : FR UP(x, FRAC INT) := reduce(*,[primeFactor(x+i,i) for i in 1..4])\bound
\end{paste}\end{patch}

\begin{patch}{PartialFractionXmpPagePatch9}
\begin{paste}{PartialFractionXmpPageFull9}{PartialFractionXmpPageEmpty9}
\pastebutton{PartialFractionXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{partialFraction(1,u)\free{u }\bound{prev3 }}
\indentrel{3}\begin{verbatim}
   (9)
       1     1      7     17  2         139
           x +    -  x  - 12x -
      648    4     16      8             8
      +  +
     x + 1          2                3
             (x + 2)          (x + 3)
   +
     607  3   10115  2   391     44179
      x  +  x  +  x +
     324       432        4       324

                          4
                  (x + 4)
Type: PartialFraction UnivariatePolynomial(x,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PartialFractionXmpPageEmpty9}
\begin{paste}{PartialFractionXmpPageEmpty9}{PartialFractionXmpPagePatch9}
\pastebutton{PartialFractionXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{partialFraction(1,u)\free{u }\bound{prev3 }}
\end{paste}\end{patch}

\begin{patch}{PartialFractionXmpPagePatch10}
\begin{paste}{PartialFractionXmpPageFull10}{PartialFractionXmpPageEmpty10}
\pastebutton{PartialFractionXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{padicFraction \%\free{prev3 }}
\indentrel{3}\begin{verbatim}
   (10)
       1       1         1        17        3

      648      4        16         8        4
      +  -   -  +
     x + 1   x + 2          2   x + 3          2
                     (x + 2)            (x + 3)
   +
```

```
           1       607       403        13          1

           2       324       432        36          12
     -  +  +  +  +
              3   x + 4           2          3          4
       (x + 3)            (x + 4)    (x + 4)    (x + 4)
Type: PartialFraction UnivariatePolynomial(x,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PartialFractionXmpPageEmpty10}
\begin{paste}{PartialFractionXmpPageEmpty10}{PartialFractionXmpPagePatch10}
\pastebutton{PartialFractionXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{padicFraction \%\free{prev3 }}
\end{paste}\end{patch}
```

———

3.87 poly.ht

Polynomials



⇐ "Topics" (TopicPage) 3.108 on page 1313
⇒ "Basic Functions" (PolynomialBasicPage) 3.87 on page 1097

⇒ “Substitutions” (PolynomialSubstitutionPage) 3.87 on page 1104
⇒ “Factorizations” (ugProblemFactorPage) 12 on page 2101
⇒ “GCDs and Friends” (PolynomialGCDPage) 3.87 on page 1108
⇒ “Roots” (PolynomialRootPage) 3.87 on page 1110
⇒ “Specific Types” (PolynomialTypesPage) 3.87 on page 1096

— poly.ht —

```
\begin{page}{PolynomialPage}{Polynomials}
\beginscroll
\beginmenu
\menulink{Basic Functions}{PolynomialBasicPage} \tab{18}
Create and manipulate polynomials.
\menulink{Substitutions}{PolynomialSubstitutionPage} \tab{18}
Evaluate polynomials.
\menulink{Factorization}{ugProblemFactorPage} \tab{18}
Factor in different contexts.
\menulink{GCDs and Friends}{PolynomialGCDPage} \tab{18}
Greatest common divisors etc..
\menulink{Roots}{PolynomialRootPage}\tab{18}
Work with and solve for roots.
\menulink{Specific Types}{PolynomialTypesPage}\tab{18}
More specific information.
\endmenu
\endscroll
\autobuttons
\end{page}
```

———-

The Specific Polynomial Types

— poly.ht —

```
\begin{page}{PolynomialTypesPage}{The Specific Polynomial Types}
\beginscroll
\beginmenu
\menulink{Polynomial}{PolynomialXmpPage} \newline
The general type.
\menulink{UnivariatePolynomial}{UnivariatePolyXmpPage} \newline
One variable polynomials.
\menulink{MultivariatePolynomial}{MultivariatePolyXmpPage}
\newline
Multiple variable polynomials, recursive structure.
\menulink{DistributedMultivariatePoly}
{DistributedMultivariatePolyXmpPage}
\newline
```

```
Multiple variable polynomials, non-recursive structure.
\menulink{UnivariateSkewPolynomial}
{UnivariateSkewPolynomial}
\newline
Skew or Ore polynomials
\endmenu
\endscroll
\autobuttons
\end{page}
```

———-

Basic Operations On Polynomials

— poly.ht —

```
\begin{page}{PolynomialBasicPage}{Basic Operations On Polynomials}
\beginscroll
You create
polynomials using the usual operations of \spadopFrom{+}{Polynomial},
\spadopFrom{-}{Polynomial}, \spadopFrom{*}{Polynomial}
(for multiplication), and
\spadopFrom{**}{Polynomial} (for exponentiation).
Here are two examples: \newline
\spadpaste{p := a*x**2 + b*x*y + c*y**2 \bound{p}}
\spadpaste{q := 13*x**2 + 3*z \bound{q}}
These operations can also be used to combine polynomials.
Try the following:
\spadpaste{p + q \free{p q}}
\spadpaste{p - 3*q \free{p q}}
\spadpaste{p**2 + p*q \free{p q}}
\spadpaste{r := (p + q)**2 \bound{r} \free{p q}}
As you can see from the above examples, the variables are ordered
by defaults \spad{z > y > x > c > b > a},
that is, \spad{z} is the main variable, then
\spad{y} and so on in reverse alphabetical order.
You can redefine this
ordering (for display purposes only) with the \spadfun{setVariableOrder}
command.
For example, the following
makes \spad{a} the main variable, then \spad{b}, and so on:
\spadpaste{setVariableOrder [a,b,c,x,y,z] \bound{vord}}
Now compare the way polynomials are displayed:
\spadpaste{p \free{p vord}}
\spadpaste{q \free{q vord}}
\spadpaste{r \free{r vord}}
To return to the system's default ordering,
```

```
use \spadfun{resetVariableOrder}.
\spadpaste{resetVariableOrder() \bound{rvord}}
\spadpaste{p \free{p rvord}}
Polynomial coefficients can be pulled out
using the function \spadfun{coefficient}. \newline
For example:
\spadpaste{coefficient(q,x,2) \free{q}}
will give you the coefficient of \spad{x**2} in the polynomial \spad{q}.
\newline
Try these commands:
\spadpaste{coefficient(r,x,3) \free{r}}
\spadpaste{c := coefficient(r,z,1) \free{r} \bound{c}}
\spadpaste{coefficient(c,x,2) \free{c}}
Coefficients of monomials can be obtained as follows:
\spadpaste{coefficient(q**2, [x,z], [2,1]) \free{q}}
This will return the coefficient of x**2 * z in the polynomial q**2.
Also,
\spadpaste{coefficient(r, [x,y], [2,2]) \free{r}}
will return the coefficient of \spad{x**2 * y**2}
in the polynomial \spad{r(x,y)}.
\endscroll
\autobuttons
\end{page}

\begin{patch}{PolynomialBasicPagePatch1}
\begin{paste}{PolynomialBasicPageFull1}{PolynomialBasicPageEmpty1}
\pastebutton{PolynomialBasicPageFull1}{\hidepaste}
\tab{5}\spadcommand{p := a*x**2 + b*x*y + c*y**2\bound{p }}
\indentrel{3}\begin{verbatim}
          2            2
   (1)  c y  + b x y + a x
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialBasicPageEmpty1}
\begin{paste}{PolynomialBasicPageEmpty1}{PolynomialBasicPagePatch1}
\pastebutton{PolynomialBasicPageEmpty1}{\showpaste}
\tab{5}\spadcommand{p := a*x**2 + b*x*y + c*y**2\bound{p }}
\end{paste}\end{patch}

\begin{patch}{PolynomialBasicPagePatch2}
\begin{paste}{PolynomialBasicPageFull2}{PolynomialBasicPageEmpty2}
\pastebutton{PolynomialBasicPageFull2}{\hidepaste}
\tab{5}\spadcommand{q := 13*x**2 + 3*z\bound{q }}
\indentrel{3}\begin{verbatim}
               2
   (2)  3z + 13x
                               Type: Polynomial Integer
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialBasicPageEmpty2}
\begin{paste}{PolynomialBasicPageEmpty2}{PolynomialBasicPagePatch2}
\pastebutton{PolynomialBasicPageEmpty2}{\showpaste}
\tab{5}\spadcommand{q := 13*x**2 + 3*z\bound{q }}
\end{paste}\end{patch}

\begin{patch}{PolynomialBasicPagePatch3}
\begin{paste}{PolynomialBasicPageFull3}{PolynomialBasicPageEmpty3}
\pastebutton{PolynomialBasicPageFull3}{\hidepaste}
\tab{5}\spadcommand{p + q\free{p q }}
\indentrel{3}\begin{verbatim}
                2                   2
   (3)  3z + c y  + b x y + (a + 13)x
                              Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialBasicPageEmpty3}
\begin{paste}{PolynomialBasicPageEmpty3}{PolynomialBasicPagePatch3}
\pastebutton{PolynomialBasicPageEmpty3}{\showpaste}
\tab{5}\spadcommand{p + q\free{p q }}
\end{paste}\end{patch}

\begin{patch}{PolynomialBasicPagePatch4}
\begin{paste}{PolynomialBasicPageFull4}{PolynomialBasicPageEmpty4}
\pastebutton{PolynomialBasicPageFull4}{\hidepaste}
\tab{5}\spadcommand{p - 3*q\free{p q }}
\indentrel{3}\begin{verbatim}
                  2                   2
   (4)  - 9z + c y  + b x y + (a - 39)x
                              Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialBasicPageEmpty4}
\begin{paste}{PolynomialBasicPageEmpty4}{PolynomialBasicPagePatch4}
\pastebutton{PolynomialBasicPageEmpty4}{\showpaste}
\tab{5}\spadcommand{p - 3*q\free{p q }}
\end{paste}\end{patch}

\begin{patch}{PolynomialBasicPagePatch5}
\begin{paste}{PolynomialBasicPageFull5}{PolynomialBasicPageEmpty5}
\pastebutton{PolynomialBasicPageFull5}{\hidepaste}
\tab{5}\spadcommand{p**2 + p*q\free{p q }}
\indentrel{3}\begin{verbatim}
   (5)
          2                2      2 4           3
     (3c y  + 3b x y + 3a x )z + c y  + 2b c x y
```

```
   +
                 2  2 2               3      2        4
   ((2a + 13)c + b )x y  + (2a + 13)b x y + (a  + 13a)x
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialBasicPageEmpty5}
\begin{paste}{PolynomialBasicPageEmpty5}{PolynomialBasicPagePatch5}
\pastebutton{PolynomialBasicPageEmpty5}{\showpaste}
\tab{5}\spadcommand{p**2 + p*q\free{p q }}
\end{paste}\end{patch}

\begin{patch}{PolynomialBasicPagePatch6}
\begin{paste}{PolynomialBasicPageFull6}{PolynomialBasicPageEmpty6}
\pastebutton{PolynomialBasicPageFull6}{\hidepaste}
\tab{5}\spadcommand{r := (p + q)**2\bound{r }\free{p q }}
\indentrel{3}\begin{verbatim}
   (6)
      2        2                       2      2 4
     9z  + (6c y  + 6b x y + (6a + 78)x )z + c y
   +
           3                  2  2 2               3
     2b c x y  + ((2a + 26)c + b )x y  + (2a + 26)b x y
   +
      2             4
     (a  + 26a + 169)x
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialBasicPageEmpty6}
\begin{paste}{PolynomialBasicPageEmpty6}{PolynomialBasicPagePatch6}
\pastebutton{PolynomialBasicPageEmpty6}{\showpaste}
\tab{5}\spadcommand{r := (p + q)**2\bound{r }\free{p q }}
\end{paste}\end{patch}

\begin{patch}{PolynomialBasicPagePatch7}
\begin{paste}{PolynomialBasicPageFull7}{PolynomialBasicPageEmpty7}
\pastebutton{PolynomialBasicPageFull7}{\hidepaste}
\tab{5}\spadcommand{setVariableOrder [a,b,c,x,y,z]\bound{vord }}
\indentrel{3}\begin{verbatim}
                                              Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialBasicPageEmpty7}
\begin{paste}{PolynomialBasicPageEmpty7}{PolynomialBasicPagePatch7}
\pastebutton{PolynomialBasicPageEmpty7}{\showpaste}
\tab{5}\spadcommand{setVariableOrder [a,b,c,x,y,z]\bound{vord }}
```

```
\end{paste}\end{patch}

\begin{patch}{PolynomialBasicPagePatch8}
\begin{paste}{PolynomialBasicPageFull8}{PolynomialBasicPageEmpty8}
\pastebutton{PolynomialBasicPageFull8}{\hidepaste}
\tab{5}\spadcommand{p\free{p vord }}
\indentrel{3}\begin{verbatim}
         2             2
   (8)  x a + y x b + y c
                              Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialBasicPageEmpty8}
\begin{paste}{PolynomialBasicPageEmpty8}{PolynomialBasicPagePatch8}
\pastebutton{PolynomialBasicPageEmpty8}{\showpaste}
\tab{5}\spadcommand{p\free{p vord }}
\end{paste}\end{patch}

\begin{patch}{PolynomialBasicPagePatch9}
\begin{paste}{PolynomialBasicPageFull9}{PolynomialBasicPageEmpty9}
\pastebutton{PolynomialBasicPageFull9}{\hidepaste}
\tab{5}\spadcommand{q\free{q vord }}
\indentrel{3}\begin{verbatim}
           2
   (9)  13x  + 3z
                              Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialBasicPageEmpty9}
\begin{paste}{PolynomialBasicPageEmpty9}{PolynomialBasicPagePatch9}
\pastebutton{PolynomialBasicPageEmpty9}{\showpaste}
\tab{5}\spadcommand{q\free{q vord }}
\end{paste}\end{patch}

\begin{patch}{PolynomialBasicPagePatch10}
\begin{paste}{PolynomialBasicPageFull10}{PolynomialBasicPageEmpty10}
\pastebutton{PolynomialBasicPageFull10}{\hidepaste}
\tab{5}\spadcommand{r\free{r vord }}
\indentrel{3}\begin{verbatim}
   (10)
      4 2        3      2 2       4       2      2 2 2
     x a  + (2y x b + 2y x c + 26x  + 6z x )a + y x b
   +
        3          3               4 2
     (2y x c + 26y x  + 6z y x)b + y c
   +
         2 2       2         4        2     2
     (26y x  + 6z y )c + 169x  + 78z x  + 9z
```

```
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialBasicPageEmpty10}
\begin{paste}{PolynomialBasicPageEmpty10}{PolynomialBasicPagePatch10}
\pastebutton{PolynomialBasicPageEmpty10}{\showpaste}
\tab{5}\spadcommand{r\free{r vord }}
\end{paste}\end{patch}

\begin{patch}{PolynomialBasicPagePatch11}
\begin{paste}{PolynomialBasicPageFull11}{PolynomialBasicPageEmpty11}
\pastebutton{PolynomialBasicPageFull11}{\hidepaste}
\tab{5}\spadcommand{resetVariableOrder()\bound{rvord }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialBasicPageEmpty11}
\begin{paste}{PolynomialBasicPageEmpty11}{PolynomialBasicPagePatch11}
\pastebutton{PolynomialBasicPageEmpty11}{\showpaste}
\tab{5}\spadcommand{resetVariableOrder()\bound{rvord }}
\end{paste}\end{patch}

\begin{patch}{PolynomialBasicPagePatch12}
\begin{paste}{PolynomialBasicPageFull12}{PolynomialBasicPageEmpty12}
\pastebutton{PolynomialBasicPageFull12}{\hidepaste}
\tab{5}\spadcommand{p\free{p rvord }}
\indentrel{3}\begin{verbatim}
            2              2
   (12)  c y  + b x y + a x
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialBasicPageEmpty12}
\begin{paste}{PolynomialBasicPageEmpty12}{PolynomialBasicPagePatch12}
\pastebutton{PolynomialBasicPageEmpty12}{\showpaste}
\tab{5}\spadcommand{p\free{p rvord }}
\end{paste}\end{patch}

\begin{patch}{PolynomialBasicPagePatch13}
\begin{paste}{PolynomialBasicPageFull13}{PolynomialBasicPageEmpty13}
\pastebutton{PolynomialBasicPageFull13}{\hidepaste}
\tab{5}\spadcommand{coefficient(q,x,2)\free{q }}
\indentrel{3}\begin{verbatim}
   (13)  13
                               Type: Polynomial Integer
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialBasicPageEmpty13}
\begin{paste}{PolynomialBasicPageEmpty13}{PolynomialBasicPagePatch13}
\pastebutton{PolynomialBasicPageEmpty13}{\showpaste}
\tab{5}\spadcommand{coefficient(q,x,2)\free{q }}
\end{paste}\end{patch}

\begin{patch}{PolynomialBasicPagePatch14}
\begin{paste}{PolynomialBasicPageFull14}{PolynomialBasicPageEmpty14}
\pastebutton{PolynomialBasicPageFull14}{\hidepaste}
\tab{5}\spadcommand{coefficient(r,x,3)\free{r }}
\indentrel{3}\begin{verbatim}
   (14)  (2a + 26)b y
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialBasicPageEmpty14}
\begin{paste}{PolynomialBasicPageEmpty14}{PolynomialBasicPagePatch14}
\pastebutton{PolynomialBasicPageEmpty14}{\showpaste}
\tab{5}\spadcommand{coefficient(r,x,3)\free{r }}
\end{paste}\end{patch}

\begin{patch}{PolynomialBasicPagePatch15}
\begin{paste}{PolynomialBasicPageFull15}{PolynomialBasicPageEmpty15}
\pastebutton{PolynomialBasicPageFull15}{\hidepaste}
\tab{5}\spadcommand{c := coefficient(r,z,1)\free{r }\bound{c }}
\indentrel{3}\begin{verbatim}
             2                       2
   (15)  6c y  + 6b x y + (6a + 78)x
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialBasicPageEmpty15}
\begin{paste}{PolynomialBasicPageEmpty15}{PolynomialBasicPagePatch15}
\pastebutton{PolynomialBasicPageEmpty15}{\showpaste}
\tab{5}\spadcommand{c := coefficient(r,z,1)\free{r }\bound{c }}
\end{paste}\end{patch}

\begin{patch}{PolynomialBasicPagePatch16}
\begin{paste}{PolynomialBasicPageFull16}{PolynomialBasicPageEmpty16}
\pastebutton{PolynomialBasicPageFull16}{\hidepaste}
\tab{5}\spadcommand{coefficient(c,x,2)\free{c }}
\indentrel{3}\begin{verbatim}
   (16)  6a + 78
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{PolynomialBasicPageEmpty16}
\begin{paste}{PolynomialBasicPageEmpty16}{PolynomialBasicPagePatch16}
\pastebutton{PolynomialBasicPageEmpty16}{\showpaste}
\tab{5}\spadcommand{coefficient(c,x,2)\free{c }}
\end{paste}\end{patch}

\begin{patch}{PolynomialBasicPagePatch17}
\begin{paste}{PolynomialBasicPageFull17}{PolynomialBasicPageEmpty17}
\pastebutton{PolynomialBasicPageFull17}{\hidepaste}
\tab{5}\spadcommand{coefficient(q**2, [x,z], [2,1])\free{q }}
\indentrel{3}\begin{verbatim}
   (17)  78
                              Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialBasicPageEmpty17}
\begin{paste}{PolynomialBasicPageEmpty17}{PolynomialBasicPagePatch17}
\pastebutton{PolynomialBasicPageEmpty17}{\showpaste}
\tab{5}\spadcommand{coefficient(q**2, [x,z], [2,1])\free{q }}
\end{paste}\end{patch}

\begin{patch}{PolynomialBasicPagePatch18}
\begin{paste}{PolynomialBasicPageFull18}{PolynomialBasicPageEmpty18}
\pastebutton{PolynomialBasicPageFull18}{\hidepaste}
\tab{5}\spadcommand{coefficient(r, [x,y], [2,2])\free{r }}
\indentrel{3}\begin{verbatim}
                      2
   (18)  (2a + 26)c + b
                              Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialBasicPageEmpty18}
\begin{paste}{PolynomialBasicPageEmpty18}{PolynomialBasicPagePatch18}
\pastebutton{PolynomialBasicPageEmpty18}{\showpaste}
\tab{5}\spadcommand{coefficient(r, [x,y], [2,2])\free{r }}
\end{paste}\end{patch}
```

———-

Polynomial Evaluation and Substitution

— poly.ht —

```
\begin{page}{PolynomialSubstitutionPage}
```

```
{Polynomial Evaluation and Substitution}
\beginscroll
The function \spadfun{eval} is used to substitute values into polynomials.
Here's an example of how to use it:
\spadpaste{p := x**2 + y**2 \bound{p}}
\spadpaste{eval(p,x=5) \free{p}}
\newline
This example would give you the value of the polynomial \spad{p} at 5.
You can also substitute into polynomials with
several variables.  First, specify the polynomial, then give
a list of bindings of the form \spad{variable = value}. For example:
\spadpaste{eval(p,[x = a + b,y = c + d]) \free{p}}
Here \spad{x} was replaced by \spad{a + b},
and \spad{y} was replaced by \spad{c + d}.
Here's another example:
\spadpaste{q := x**3 + 5*x - y**4 \bound{q}}
\spadpaste{eval(q,[x=y,y=x]) \free{q}}
Substitution is done ``in parallel.''
That is, Axiom takes
\spad{q(x,y)} and returns \spad{q(y,x)}.
\newline
You can also substitute numerical values for some or all of the
variables:
\spadpaste{px := eval(p, y = sin(2.0)) \bound{px} \free{p}}
\spadpaste{eval(px, x = cos(2.0)) \free{px}}
\endscroll
\autobuttons
\end{page}

\begin{patch}{PolynomialSubstitutionPagePatch1}
\begin{paste}{PolynomialSubstitutionPageFull1}{PolynomialSubstitutionPageEmpty1}
\pastebutton{PolynomialSubstitutionPageFull1}{\hidepaste}
\tab{5}\spadcommand{p := x**2 + y**2\bound{p }}
\indentrel{3}\begin{verbatim}
         2    2
   (1)  y  + x
                              Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialSubstitutionPageEmpty1}
\begin{paste}{PolynomialSubstitutionPageEmpty1}{PolynomialSubstitutionPagePatch1}
\pastebutton{PolynomialSubstitutionPageEmpty1}{\showpaste}
\tab{5}\spadcommand{p := x**2 + y**2\bound{p }}
\end{paste}\end{patch}

\begin{patch}{PolynomialSubstitutionPagePatch2}
\begin{paste}{PolynomialSubstitutionPageFull2}{PolynomialSubstitutionPageEmpty2}
\pastebutton{PolynomialSubstitutionPageFull2}{\hidepaste}
\tab{5}\spadcommand{eval(p,x=5)\free{p }}
```

```
\indentrel{3}\begin{verbatim}
         2
   (2)  y  + 25
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialSubstitutionPageEmpty2}
\begin{paste}{PolynomialSubstitutionPageEmpty2}{PolynomialSubstitutionPagePatch2}
\pastebutton{PolynomialSubstitutionPageEmpty2}{\showpaste}
\tab{5}\spadcommand{eval(p,x=5)\free{p }}
\end{paste}\end{patch}

\begin{patch}{PolynomialSubstitutionPagePatch3}
\begin{paste}{PolynomialSubstitutionPageFull3}{PolynomialSubstitutionPageEmpty3}
\pastebutton{PolynomialSubstitutionPageFull3}{\hidepaste}
\tab{5}\spadcommand{eval(p,[x = a + b,y = c + d])\free{p }}
\indentrel{3}\begin{verbatim}
         2           2    2           2
   (3)  d  + 2c d + c  + b  + 2a b + a
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialSubstitutionPageEmpty3}
\begin{paste}{PolynomialSubstitutionPageEmpty3}{PolynomialSubstitutionPagePatch3}
\pastebutton{PolynomialSubstitutionPageEmpty3}{\showpaste}
\tab{5}\spadcommand{eval(p,[x = a + b,y = c + d])\free{p }}
\end{paste}\end{patch}

\begin{patch}{PolynomialSubstitutionPagePatch4}
\begin{paste}{PolynomialSubstitutionPageFull4}{PolynomialSubstitutionPageEmpty4}
\pastebutton{PolynomialSubstitutionPageFull4}{\hidepaste}
\tab{5}\spadcommand{q := x**3 + 5*x - y**4\bound{q }}
\indentrel{3}\begin{verbatim}
           4    3
   (4)  - y  + x  + 5x
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialSubstitutionPageEmpty4}
\begin{paste}{PolynomialSubstitutionPageEmpty4}{PolynomialSubstitutionPagePatch4}
\pastebutton{PolynomialSubstitutionPageEmpty4}{\showpaste}
\tab{5}\spadcommand{q := x**3 + 5*x - y**4\bound{q }}
\end{paste}\end{patch}

\begin{patch}{PolynomialSubstitutionPagePatch5}
\begin{paste}{PolynomialSubstitutionPageFull5}{PolynomialSubstitutionPageEmpty5}
\pastebutton{PolynomialSubstitutionPageFull5}{\hidepaste}
```

```
\tab{5}\spadcommand{eval(q,[x=y,y=x])\free{q }}
\indentrel{3}\begin{verbatim}
        3        4
   (5)  y  + 5y - x
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialSubstitutionPageEmpty5}
\begin{paste}{PolynomialSubstitutionPageEmpty5}{PolynomialSubstitutionPagePatch5}
\pastebutton{PolynomialSubstitutionPageEmpty5}{\showpaste}
\tab{5}\spadcommand{eval(q,[x=y,y=x])\free{q }}
\end{paste}\end{patch}

\begin{patch}{PolynomialSubstitutionPagePatch6}
\begin{paste}{PolynomialSubstitutionPageFull6}{PolynomialSubstitutionPageEmpty6}
\pastebutton{PolynomialSubstitutionPageFull6}{\hidepaste}
\tab{5}\spadcommand{px := eval(p, y = sin(2.0))\bound{px }\free{p }}
\indentrel{3}\begin{verbatim}
        2
   (6)  x  + 0.8268218104 3180595732
                                 Type: Polynomial Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialSubstitutionPageEmpty6}
\begin{paste}{PolynomialSubstitutionPageEmpty6}{PolynomialSubstitutionPagePatch6}
\pastebutton{PolynomialSubstitutionPageEmpty6}{\showpaste}
\tab{5}\spadcommand{px := eval(p, y = sin(2.0))\bound{px }\free{p }}
\end{paste}\end{patch}

\begin{patch}{PolynomialSubstitutionPagePatch7}
\begin{paste}{PolynomialSubstitutionPageFull7}{PolynomialSubstitutionPageEmpty7}
\pastebutton{PolynomialSubstitutionPageFull7}{\hidepaste}
\tab{5}\spadcommand{eval(px, x = cos(2.0))\free{px }}
\indentrel{3}\begin{verbatim}
   (7)  1.0
                                 Type: Polynomial Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialSubstitutionPageEmpty7}
\begin{paste}{PolynomialSubstitutionPageEmpty7}{PolynomialSubstitutionPagePatch7}
\pastebutton{PolynomialSubstitutionPageEmpty7}{\showpaste}
\tab{5}\spadcommand{eval(px, x = cos(2.0))\free{px }}
\end{paste}\end{patch}
```

Greatest Common Divisors, Resultants, and Discriminants

— poly.ht —

```
\begin{page}{PolynomialGCDPage}
{Greatest Common Divisors, Resultants, and Discriminants}
\beginscroll
You can compute the greatest common divisor of two polynomials using
the function \spadfun{gcd}.
Here's an example:
\spadpaste{p := 3*x**8 + 2*x**7 + 6*x**2 + 7*x + 2 \bound{p}}
\spadpaste{q := 2*x**13 + 9*x**7 + 2*x**6 + 10*x + 5 \bound{q}}
\spadpaste{gcd(p,q) \free{p q}}
You could also see that \spad{p} and \spad{q} have a factor in
common by using the function \spadfun{resultant}:
\spadpaste{resultant(p,q,x) \free{p q}}
The resultant of two polynomials vanishes precisely when they have a
factor in common.
(In the example above
we specified the variable with
which we wanted to compute the resultant because the
polynomials could have involved variables other than x.)
\endscroll
\autobuttons
\end{page}

\begin{patch}{PolynomialGCDPagePatch1}
\begin{paste}{PolynomialGCDPageFull1}{PolynomialGCDPageEmpty1}
\pastebutton{PolynomialGCDPageFull1}{\hidepaste}
\tab{5}\spadcommand{p := 3*x**8 + 2*x**7 + 6*x**2 + 7*x + 2\bound{p }}
\indentrel{3}\begin{verbatim}
          8     7     2
   (1)  3x  + 2x  + 6x  + 7x + 2
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialGCDPageEmpty1}
\begin{paste}{PolynomialGCDPageEmpty1}{PolynomialGCDPagePatch1}
\pastebutton{PolynomialGCDPageEmpty1}{\showpaste}
\tab{5}\spadcommand{p := 3*x**8 + 2*x**7 + 6*x**2 + 7*x + 2\bound{p }}
\end{paste}\end{patch}

\begin{patch}{PolynomialGCDPagePatch2}
\begin{paste}{PolynomialGCDPageFull2}{PolynomialGCDPageEmpty2}
\pastebutton{PolynomialGCDPageFull2}{\hidepaste}
\tab{5}\spadcommand{q := 2*x**13 + 9*x**7 + 2*x**6 + 10*x + 5\bound{q }}
\indentrel{3}\begin{verbatim}
          13     7     6
```

```
   (2)  2x   + 9x  + 2x  + 10x + 5
                              Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialGCDPageEmpty2}
\begin{paste}{PolynomialGCDPageEmpty2}{PolynomialGCDPagePatch2}
\pastebutton{PolynomialGCDPageEmpty2}{\showpaste}
\tab{5}\spadcommand{q := 2*x**13 + 9*x**7 + 2*x**6 + 10*x + 5\bound{q }}
\end{paste}\end{patch}

\begin{patch}{PolynomialGCDPagePatch3}
\begin{paste}{PolynomialGCDPageFull3}{PolynomialGCDPageEmpty3}
\pastebutton{PolynomialGCDPageFull3}{\hidepaste}
\tab{5}\spadcommand{gcd(p,q)\free{p q }}
\indentrel{3}\begin{verbatim}
         7
   (3)  x  + 2x + 1
                              Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialGCDPageEmpty3}
\begin{paste}{PolynomialGCDPageEmpty3}{PolynomialGCDPagePatch3}
\pastebutton{PolynomialGCDPageEmpty3}{\showpaste}
\tab{5}\spadcommand{gcd(p,q)\free{p q }}
\end{paste}\end{patch}

\begin{patch}{PolynomialGCDPagePatch4}
\begin{paste}{PolynomialGCDPageFull4}{PolynomialGCDPageEmpty4}
\pastebutton{PolynomialGCDPageFull4}{\hidepaste}
\tab{5}\spadcommand{resultant(p,q,x)\free{p q }}
\indentrel{3}\begin{verbatim}
   (4)  0
                              Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialGCDPageEmpty4}
\begin{paste}{PolynomialGCDPageEmpty4}{PolynomialGCDPagePatch4}
\pastebutton{PolynomialGCDPageEmpty4}{\showpaste}
\tab{5}\spadcommand{resultant(p,q,x)\free{p q }}
\end{paste}\end{patch}
```

Roots of Polynomials

— **poly.ht** —

```
\begin{page}{PolynomialRootPage}{Roots of Polynomials}
\beginscroll
\beginmenu
\menulink{Using a Single Root of a Polynomial}{ugxProblemSymRootOnePage}
\newline
Working with a single root of a polynomial.
\menulink{Using All Roots of a Polynomial}{ugxProblemSymRootAllPage}
\newline
Working with all the roots of a polynomial.
\menulink{Solution of a Single Polynomial Equation}
{ugxProblemOnePolPage}
\newline
Finding the roots of one polynomial.
\menulink{Solution of Systems of Polynomial Equations}
{ugxProblemPolSysPage}
\newline
Finding the roots of a system of polynomials.
\endmenu
\endscroll
\autobuttons
\end{page}
```

———-

3.88 poly1.ht

Polynomial

⇒ “notitle” (DistributedMultivariatePolyXmpPage) 3.24 on page 375
⇒ “notitle” (MultivariatePolyXmpPage) 3.77 on page 1011
⇒ “notitle” (UnivariatePolyXmpPage) 3.112 on page 1327
⇒ “notitle” (FactoredXmpPage) 3.43 on page 516
⇒ “notitle” (ugProblemFactorPage) 12 on page 2101

— **poly1.ht** —

```
\begin{page}{PolynomialXmpPage}{Polynomial}
\beginscroll

The domain constructor \spadtype{Polynomial}
(abbreviation: \spadtype{POLY})
```

```
provides polynomials with an arbitrary number of unspecified
variables.

\xtc{
It is used to create the default polynomial domains
in Axiom.
Here the coefficients are integers.
}{
\spadpaste{x + 1}
}
\xtc{
Here the coefficients have type \spadtype{Float}.
}{
\spadpaste{z - 2.3}
}
\xtc{
And here we have a polynomial in two variables with coefficients
which have type \spadtype{Fraction Integer}.
}{
\spadpaste{y**2 - z + 3/4}
}

The representation of objects of domains created by \spadtype{Polynomial}
is that of recursive univariate polynomials.\footnote{The term
\spad{univariate} means ``one variable.'' \spad{multivariate} means
``possibly more than one variable.''}
\xtc{
This recursive structure is sometimes obvious from the display of
a polynomial.
}{
\spadpaste{y **2 + x*y + y \bound{prev}}
}
In this example, you see that the polynomial is stored as a polynomial in
\spad{y} with coefficients that are polynomials in \spad{x} with integer
coefficients.
In fact, you really don't need to worry about the representation unless
you are working on an advanced application where it is critical.
The polynomial types created from
\spadtype{DistributedMultivariatePoly} and
\spadtype{NewDistributedMultivariatePoly} (discussed in
\downlink{`DistributedMultivariatePoly'}
{DistributedMultivariatePolyXmpPage}
\ignore{DistributedMultivariatePoly}) are stored and displayed in a
non-recursive manner.
\xtc{
You see a ``flat'' display of the above
polynomial by converting to one of those types.
}{
\spadpaste{\% :: DMP([y,x],INT) \free{prev}}
}
```

```
We will demonstrate many of the polynomial facilities by using two
polynomials with integer coefficients.
\xtc{
By default, the interpreter expands polynomial expressions, even if they
are written in a factored format.
}{
\spadpaste{p := (y-1)**2 * x * z \bound{p}}
}
\xtc{
See \downlink{`Factored'}{FactoredXmpPage}\ignore{Factored} to see
how to create objects in factored form directly.
}{
\spadpaste{q := (y-1) * x * (z+5) \bound{q}}
}
\xtc{
The fully factored form can be recovered by using
\spadfunFrom{factor}{Polynomial}.
}{
\spadpaste{factor(q) \free{q}}
}
This is the same name used for the operation to factor integers.  Such
reuse of names is called \spadglos{overloading} and makes it much
easier to think of solving problems in general ways.  Axiom facilities
for factoring polynomials created with \spadtype{Polynomial} are
currently restricted to the integer and rational number coefficient
cases.  There are more complete facilities for factoring univariate
polynomials: see
\downlink{``Polynomial Factorization''}{ugProblemFactorPage} in
Section 8.2\ignore{ugProblemFactor}.

\xtc{
The standard arithmetic operations are available for polynomials.
}{
\spadpaste{p - q**2\free{p q}}
}
\xtc{
The operation \spadfunFrom{gcd}{Polynomial} is used to compute the
greatest common divisor of two polynomials.
}{
\spadpaste{gcd(p,q) \free{p q}\bound{prev4}}
}
\xtc{
In the case of \spad{p} and \spad{q}, the gcd is obvious from their
definitions.
We factor the gcd to show this relationship better.
}{
\spadpaste{factor \% \free{prev4}}
}
\xtc{
```

```
The least common multiple is computed by using
\spadfunFrom{lcm}{Polynomial}.
}{
\spadpaste{lcm(p,q) \free{p q}}
}
\xtc{
Use \spadfunFrom{content}{Polynomial} to compute the
greatest common divisor of the
coefficients of the polynomial.
}{
\spadpaste{content p \free{p}}
}

Many of the operations on polynomials require you to specify a variable.
For example, \spadfunFrom{resultant}{Polynomial} requires you to give the
variable in which the polynomials should be expressed.
\xtc{
This computes the resultant of the values of \spad{p} and
\spad{q}, considering them as polynomials in the variable \spad{z}.
They do not share a root when thought of as polynomials in \spad{z}.
}{
\spadpaste{resultant(p,q,z) \free{p q}}
}
\xtc{
This value is \spad{0} because as polynomials in \spad{x} the polynomials
have a common root.
}{
\spadpaste{resultant(p,q,x) \free{p}\free{q}}
}
The data type used for the variables created by \spadtype{Polynomial} is
\spadtype{Symbol}.
As mentioned above, the representation used by \spadtype{Polynomial} is
recursive and so there is a main variable for nonconstant polynomials.
\xtc{
The operation \spadfunFrom{mainVariable}{Polynomial} returns this
variable.
The return type is actually a union of \spadtype{Symbol} and
\spad{"failed"}.
}{
\spadpaste{mainVariable p \free{p}}
}
\xtc{
The latter branch of the union is be used if the polynomial has no
variables, that is, is a constant.
}{
\spadpaste{mainVariable(1 :: POLY INT)}
}
\xtc{
You can also use the predicate \spadfunFrom{ground?}{Polynomial} to test
whether a polynomial is in fact a member of its ground ring.
```

```
}{
\spadpaste{ground? p \free{p}}
}
\xtc{
}{
\spadpaste{ground?(1 :: POLY INT)}
}
\xtc{
The complete list of variables actually used in a particular polynomial is
returned by \spadfunFrom{variables}{Polynomial}.
For constant polynomials, this list is empty.
}{
\spadpaste{variables p \free{p}}
}

\xtc{
The \spadfunFrom{degree}{Polynomial} operation returns the
degree of a polynomial in a specific variable.
}{
\spadpaste{degree(p,x) \free{p}}
}
\xtc{
}{
\spadpaste{degree(p,y) \free{p}}
}
\xtc{
}{
\spadpaste{degree(p,z) \free{p}}
}
\xtc{
If you give a list of variables for the second argument, a list
of the degrees in those variables is returned.
}{
\spadpaste{degree(p,[x,y,z]) \free{p}}
}
\xtc{
The minimum degree of a variable in a polynomial is computed using
\spadfunFrom{minimumDegree}{Polynomial}.
}{
\spadpaste{minimumDegree(p,z) \free{p}}
}
\xtc{
The total degree of a polynomial is returned by
\spadfunFrom{totalDegree}{Polynomial}.
}{
\spadpaste{totalDegree p \free{p}}
}

\xtc{
It is often convenient to think of a polynomial as a leading monomial
```

```
plus the remaining terms.
}{
\spadpaste{leadingMonomial p \free{p}}
}
\xtc{
The \spadfunFrom{reductum}{Polynomial} operation returns a polynomial
consisting of the sum of the monomials after the first.
}{
\spadpaste{reductum p \free{p}}
}
\xtc{
These have the obvious relationship that the original polynomial
is equal to the leading monomial plus the reductum.
}{
\spadpaste{p - leadingMonomial p - reductum p \free{p}}
}
\xtc{
The value returned by \spadfunFrom{leadingMonomial}{Polynomial} includes
the coefficient of that term.
This is extracted by using
\spadfunFrom{leadingCoefficient}{Polynomial} on the original polynomial.
}{
\spadpaste{leadingCoefficient p \free{p}}
}
\xtc{
The operation \spadfunFrom{eval}{Polynomial} is used to substitute a
value for a variable in a polynomial.
}{
\spadpaste{p  \free{p}}
}
\xtc{
This value may be another variable, a constant or a polynomial.
}{
\spadpaste{eval(p,x,w) \free{p}}
}
\xtc{
}{
\spadpaste{eval(p,x,1) \free{p}}
}
\xtc{
Actually, all the things being substituted are just polynomials,
some more trivial than others.
}{
\spadpaste{eval(p,x,y**2 - 1) \free{p}}
}

\xtc{
Derivatives are computed using the \spadfunFrom{D}{Polynomial}
operation.
}{
```

```
\spadpaste{D(p,x) \free{p}}
}
\xtc{
The first argument is the polynomial and the second is the variable.
}{
\spadpaste{D(p,y) \free{p}}
}
\xtc{
Even if the polynomial has only one variable, you must specify it.
}{
\spadpaste{D(p,z) \free{p}}
}

Integration of polynomials is similar and the
\spadfunFrom{integrate}{Polynomial} operation is used.

\xtc{
Integration requires that the coefficients support division.
Consequently,
Axiom converts polynomials over the integers to polynomials over
the rational numbers before integrating them.
}{
\spadpaste{integrate(p,y) \free{p}}
}

It is not possible, in general, to divide two polynomials.
In our example using polynomials over the integers, the operation
\spadfunFrom{monicDivide}{Polynomial} divides a polynomial by a monic
polynomial (that is, a polynomial with leading coefficient equal to 1).
The result is a record of the quotient and remainder of the
division.
\xtc{
You must specify the variable in which to express the polynomial.
}{
\spadpaste{qr := monicDivide(p,x+1,x) \free{p}\bound{qr}}
}
\xtc{
The selectors of the components of the record are
\spad{quotient} and \spad{remainder}.
Issue this to extract the remainder.
}{
\spadpaste{qr.remainder \free{qr}}
}
\xtc{
Now that we can extract the components, we can demonstrate the
relationship among them and the arguments to our original expression
\spad{qr := monicDivide(p,x+1,x)}.
}{
\spadpaste{p - ((x+1) * qr.quotient + qr.remainder) \free{p}\free{qr}}
}
```

```
\xtc{
If the \spadopFrom{/}{Fraction} operator is used with polynomials, a
fraction object is created.
In this example, the result is an object of type \spadtype{Fraction
Polynomial Integer}.
}{
\spadpaste{p/q \free{p}\free{q}}
}
\xtc{
If you use rational numbers as polynomial coefficients, the
resulting object is of type \spadtype{Polynomial Fraction Integer}.
}{
\spadpaste{(2/3) * x**2 - y + 4/5 \bound{prev1}}
}
\xtc{
This can be converted to a fraction of polynomials and back again, if
required.
}{
\spadpaste{\% :: FRAC POLY INT \free{prev1}\bound{prev2}}
}
\xtc{
}{
\spadpaste{\% :: POLY FRAC INT \free{prev2}\bound{prev3}}
}
\xtc{
To convert the coefficients to floating point,
map the \spadfun{numeric} operation on the coefficients of the polynomial.
}{
\spadpaste{map(numeric,\%) \free{prev3}}
}

For more information on related topics, see
\downlink{`UnivariatePolynomial'}{UnivariatePolyXmpPage}
\ignore{UnivariatePolynomial},
\downlink{`MultivariatePolynomial'}{MultivariatePolyXmpPage}
\ignore{MultivariatePolynomial}, and
\downlink{`DistributedMultivariatePoly'}
{DistributedMultivariatePolyXmpPage}
\ignore{DistributedMultivariatePoly}.
You can also issue the system command
\spadcmd{)show Polynomial}
to display the full list of operations defined by
\spadtype{Polynomial}.
\endscroll
\autobuttons
\end{page}

\begin{patch}{PolynomialXmpPagePatch1}
\begin{paste}{PolynomialXmpPageFull1}{PolynomialXmpPageEmpty1}
```

```
\pastebutton{PolynomialXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{x + 1}
\indentrel{3}\begin{verbatim}
   (1)  x + 1
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty1}
\begin{paste}{PolynomialXmpPageEmpty1}{PolynomialXmpPagePatch1}
\pastebutton{PolynomialXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{x + 1}
\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPagePatch2}
\begin{paste}{PolynomialXmpPageFull2}{PolynomialXmpPageEmpty2}
\pastebutton{PolynomialXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{z - 2.3}
\indentrel{3}\begin{verbatim}
   (2)  z - 2.3
                                 Type: Polynomial Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty2}
\begin{paste}{PolynomialXmpPageEmpty2}{PolynomialXmpPagePatch2}
\pastebutton{PolynomialXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{z - 2.3}
\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPagePatch3}
\begin{paste}{PolynomialXmpPageFull3}{PolynomialXmpPageEmpty3}
\pastebutton{PolynomialXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{y**2 - z + 3/4}
\indentrel{3}\begin{verbatim}
               2   3
   (3)  - z + y  +
                   4
                      Type: Polynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty3}
\begin{paste}{PolynomialXmpPageEmpty3}{PolynomialXmpPagePatch3}
\pastebutton{PolynomialXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{y**2 - z + 3/4}
\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPagePatch4}
\begin{paste}{PolynomialXmpPageFull4}{PolynomialXmpPageEmpty4}
```

```
\pastebutton{PolynomialXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{y **2 + x*y + y\bound{prev }}
\indentrel{3}\begin{verbatim}
        2
   (4)  y  + (x + 1)y
                              Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty4}
\begin{paste}{PolynomialXmpPageEmpty4}{PolynomialXmpPagePatch4}
\pastebutton{PolynomialXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{y **2 + x*y + y\bound{prev }}
\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPagePatch5}
\begin{paste}{PolynomialXmpPageFull5}{PolynomialXmpPageEmpty5}
\pastebutton{PolynomialXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{\% :: DMP([y,x],INT)\free{prev }}
\indentrel{3}\begin{verbatim}
        2
   (5)  y  + y x + y
 Type: DistributedMultivariatePolynomial([y,x],Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty5}
\begin{paste}{PolynomialXmpPageEmpty5}{PolynomialXmpPagePatch5}
\pastebutton{PolynomialXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{\% :: DMP([y,x],INT)\free{prev }}
\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPagePatch6}
\begin{paste}{PolynomialXmpPageFull6}{PolynomialXmpPageEmpty6}
\pastebutton{PolynomialXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{p := (y-1)**2 * x * z\bound{p }}
\indentrel{3}\begin{verbatim}
           2
   (6)  (x y  - 2x y + x)z
                              Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty6}
\begin{paste}{PolynomialXmpPageEmpty6}{PolynomialXmpPagePatch6}
\pastebutton{PolynomialXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{p := (y-1)**2 * x * z\bound{p }}
\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPagePatch7}
```

```
\begin{paste}{PolynomialXmpPageFull7}{PolynomialXmpPageEmpty7}
\pastebutton{PolynomialXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{q := (y-1) * x * (z+5)\bound{q }}
\indentrel{3}\begin{verbatim}
   (7)  (x y - x)z + 5x y - 5x
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty7}
\begin{paste}{PolynomialXmpPageEmpty7}{PolynomialXmpPagePatch7}
\pastebutton{PolynomialXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{q := (y-1) * x * (z+5)\bound{q }}
\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPagePatch8}
\begin{paste}{PolynomialXmpPageFull8}{PolynomialXmpPageEmpty8}
\pastebutton{PolynomialXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{factor(q)\free{q }}
\indentrel{3}\begin{verbatim}
   (8)  x(y - 1)(z + 5)
                      Type: Factored Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty8}
\begin{paste}{PolynomialXmpPageEmpty8}{PolynomialXmpPagePatch8}
\pastebutton{PolynomialXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{factor(q)\free{q }}
\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPagePatch9}
\begin{paste}{PolynomialXmpPageFull9}{PolynomialXmpPageEmpty9}
\pastebutton{PolynomialXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{p - q**2\free{p q }}
\indentrel{3}\begin{verbatim}
   (9)
        2 2     2     2  2
     (- x y  + 2x y - x )z
   +
           2      2       2             2           2 2
     ((- 10x  + x)y  + (20x  - 2x)y - 10x  + x)z - 25x y
   +
        2       2
     50x y - 25x
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty9}
```

```
\begin{paste}{PolynomialXmpPageEmpty9}{PolynomialXmpPagePatch9}
\pastebutton{PolynomialXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{p - q**2\free{p q }}
\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPagePatch10}
\begin{paste}{PolynomialXmpPageFull10}{PolynomialXmpPageEmpty10}
\pastebutton{PolynomialXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{gcd(p,q)\free{p q }\bound{prev4 }}
\indentrel{3}\begin{verbatim}
   (10)  x y - x
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty10}
\begin{paste}{PolynomialXmpPageEmpty10}{PolynomialXmpPagePatch10}
\pastebutton{PolynomialXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{gcd(p,q)\free{p q }\bound{prev4 }}
\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPagePatch11}
\begin{paste}{PolynomialXmpPageFull11}{PolynomialXmpPageEmpty11}
\pastebutton{PolynomialXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{factor \%\free{prev4 }}
\indentrel{3}\begin{verbatim}
   (11)  x(y - 1)
                      Type: Factored Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty11}
\begin{paste}{PolynomialXmpPageEmpty11}{PolynomialXmpPagePatch11}
\pastebutton{PolynomialXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{factor \%\free{prev4 }}
\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPagePatch12}
\begin{paste}{PolynomialXmpPageFull12}{PolynomialXmpPageEmpty12}
\pastebutton{PolynomialXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{lcm(p,q)\free{p q }}
\indentrel{3}\begin{verbatim}
             2             2        2
   (12)  (x y  - 2x y + x)z  + (5x y  - 10x y + 5x)z
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty12}
\begin{paste}{PolynomialXmpPageEmpty12}{PolynomialXmpPagePatch12}
```

```
\pastebutton{PolynomialXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{lcm(p,q)\free{p q }}
\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPagePatch13}
\begin{paste}{PolynomialXmpPageFull13}{PolynomialXmpPageEmpty13}
\pastebutton{PolynomialXmpPageFull13}{\hidepaste}
\tab{5}\spadcommand{content p\free{p }}
\indentrel{3}\begin{verbatim}
   (13)  1
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty13}
\begin{paste}{PolynomialXmpPageEmpty13}{PolynomialXmpPagePatch13}
\pastebutton{PolynomialXmpPageEmpty13}{\showpaste}
\tab{5}\spadcommand{content p\free{p }}
\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPagePatch14}
\begin{paste}{PolynomialXmpPageFull14}{PolynomialXmpPageEmpty14}
\pastebutton{PolynomialXmpPageFull14}{\hidepaste}
\tab{5}\spadcommand{resultant(p,q,z)\free{p q }}
\indentrel{3}\begin{verbatim}
           2 3      2 2      2      2
   (14)  5x y  - 15x y  + 15x y - 5x
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty14}
\begin{paste}{PolynomialXmpPageEmpty14}{PolynomialXmpPagePatch14}
\pastebutton{PolynomialXmpPageEmpty14}{\showpaste}
\tab{5}\spadcommand{resultant(p,q,z)\free{p q }}
\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPagePatch15}
\begin{paste}{PolynomialXmpPageFull15}{PolynomialXmpPageEmpty15}
\pastebutton{PolynomialXmpPageFull15}{\hidepaste}
\tab{5}\spadcommand{resultant(p,q,x)\free{p }\free{q }}
\indentrel{3}\begin{verbatim}
   (15)  0
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty15}
\begin{paste}{PolynomialXmpPageEmpty15}{PolynomialXmpPagePatch15}
\pastebutton{PolynomialXmpPageEmpty15}{\showpaste}
```

```
\tab{5}\spadcommand{resultant(p,q,x)\free{p }\free{q }}
\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPagePatch16}
\begin{paste}{PolynomialXmpPageFull16}{PolynomialXmpPageEmpty16}
\pastebutton{PolynomialXmpPageFull16}{\hidepaste}
\tab{5}\spadcommand{mainVariable p\free{p }}
\indentrel{3}\begin{verbatim}
   (16)  z
                              Type: Union(Symbol,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty16}
\begin{paste}{PolynomialXmpPageEmpty16}{PolynomialXmpPagePatch16}
\pastebutton{PolynomialXmpPageEmpty16}{\showpaste}
\tab{5}\spadcommand{mainVariable p\free{p }}
\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPagePatch17}
\begin{paste}{PolynomialXmpPageFull17}{PolynomialXmpPageEmpty17}
\pastebutton{PolynomialXmpPageFull17}{\hidepaste}
\tab{5}\spadcommand{mainVariable(1 :: POLY INT)}
\indentrel{3}\begin{verbatim}
   (17)  "failed"
                            Type: Union("failed",...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty17}
\begin{paste}{PolynomialXmpPageEmpty17}{PolynomialXmpPagePatch17}
\pastebutton{PolynomialXmpPageEmpty17}{\showpaste}
\tab{5}\spadcommand{mainVariable(1 :: POLY INT)}
\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPagePatch18}
\begin{paste}{PolynomialXmpPageFull18}{PolynomialXmpPageEmpty18}
\pastebutton{PolynomialXmpPageFull18}{\hidepaste}
\tab{5}\spadcommand{ground? p\free{p }}
\indentrel{3}\begin{verbatim}
   (18)  false
                                        Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty18}
\begin{paste}{PolynomialXmpPageEmpty18}{PolynomialXmpPagePatch18}
\pastebutton{PolynomialXmpPageEmpty18}{\showpaste}
\tab{5}\spadcommand{ground? p\free{p }}
\end{paste}\end{patch}
```

```
\begin{patch}{PolynomialXmpPagePatch19}
\begin{paste}{PolynomialXmpPageFull19}{PolynomialXmpPageEmpty19}
\pastebutton{PolynomialXmpPageFull19}{\hidepaste}
\tab{5}\spadcommand{ground?(1 :: POLY INT)}
\indentrel{3}\begin{verbatim}
   (19)  true
                                            Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty19}
\begin{paste}{PolynomialXmpPageEmpty19}{PolynomialXmpPagePatch19}
\pastebutton{PolynomialXmpPageEmpty19}{\showpaste}
\tab{5}\spadcommand{ground?(1 :: POLY INT)}
\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPagePatch20}
\begin{paste}{PolynomialXmpPageFull20}{PolynomialXmpPageEmpty20}
\pastebutton{PolynomialXmpPageFull20}{\hidepaste}
\tab{5}\spadcommand{variables p\free{p }}
\indentrel{3}\begin{verbatim}
   (20)  [z,y,x]
                                        Type: List Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty20}
\begin{paste}{PolynomialXmpPageEmpty20}{PolynomialXmpPagePatch20}
\pastebutton{PolynomialXmpPageEmpty20}{\showpaste}
\tab{5}\spadcommand{variables p\free{p }}
\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPagePatch21}
\begin{paste}{PolynomialXmpPageFull21}{PolynomialXmpPageEmpty21}
\pastebutton{PolynomialXmpPageFull21}{\hidepaste}
\tab{5}\spadcommand{degree(p,x)\free{p }}
\indentrel{3}\begin{verbatim}
   (21)  1
                                    Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty21}
\begin{paste}{PolynomialXmpPageEmpty21}{PolynomialXmpPagePatch21}
\pastebutton{PolynomialXmpPageEmpty21}{\showpaste}
\tab{5}\spadcommand{degree(p,x)\free{p }}
\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPagePatch22}
```

```
\begin{paste}{PolynomialXmpPageFull22}{PolynomialXmpPageEmpty22}
\pastebutton{PolynomialXmpPageFull22}{\hidepaste}
\tab{5}\spadcommand{degree(p,y)\free{p }}
\indentrel{3}\begin{verbatim}
   (22)  2
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty22}
\begin{paste}{PolynomialXmpPageEmpty22}{PolynomialXmpPagePatch22}
\pastebutton{PolynomialXmpPageEmpty22}{\showpaste}
\tab{5}\spadcommand{degree(p,y)\free{p }}
\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPagePatch23}
\begin{paste}{PolynomialXmpPageFull23}{PolynomialXmpPageEmpty23}
\pastebutton{PolynomialXmpPageFull23}{\hidepaste}
\tab{5}\spadcommand{degree(p,z)\free{p }}
\indentrel{3}\begin{verbatim}
   (23)  1
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty23}
\begin{paste}{PolynomialXmpPageEmpty23}{PolynomialXmpPagePatch23}
\pastebutton{PolynomialXmpPageEmpty23}{\showpaste}
\tab{5}\spadcommand{degree(p,z)\free{p }}
\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPagePatch24}
\begin{paste}{PolynomialXmpPageFull24}{PolynomialXmpPageEmpty24}
\pastebutton{PolynomialXmpPageFull24}{\hidepaste}
\tab{5}\spadcommand{degree(p,[x,y,z])\free{p }}
\indentrel{3}\begin{verbatim}
   (24)  [1,2,1]
                          Type: List NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty24}
\begin{paste}{PolynomialXmpPageEmpty24}{PolynomialXmpPagePatch24}
\pastebutton{PolynomialXmpPageEmpty24}{\showpaste}
\tab{5}\spadcommand{degree(p,[x,y,z])\free{p }}
\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPagePatch25}
\begin{paste}{PolynomialXmpPageFull25}{PolynomialXmpPageEmpty25}
\pastebutton{PolynomialXmpPageFull25}{\hidepaste}
```

```
\tab{5}\spadcommand{minimumDegree(p,z)\free{p }}
\indentrel{3}\begin{verbatim}
   (25)  1
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty25}
\begin{paste}{PolynomialXmpPageEmpty25}{PolynomialXmpPagePatch25}
\pastebutton{PolynomialXmpPageEmpty25}{\showpaste}
\tab{5}\spadcommand{minimumDegree(p,z)\free{p }}
\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPagePatch26}
\begin{paste}{PolynomialXmpPageFull26}{PolynomialXmpPageEmpty26}
\pastebutton{PolynomialXmpPageFull26}{\hidepaste}
\tab{5}\spadcommand{totalDegree p\free{p }}
\indentrel{3}\begin{verbatim}
   (26)  4
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty26}
\begin{paste}{PolynomialXmpPageEmpty26}{PolynomialXmpPagePatch26}
\pastebutton{PolynomialXmpPageEmpty26}{\showpaste}
\tab{5}\spadcommand{totalDegree p\free{p }}
\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPagePatch27}
\begin{paste}{PolynomialXmpPageFull27}{PolynomialXmpPageEmpty27}
\pastebutton{PolynomialXmpPageFull27}{\hidepaste}
\tab{5}\spadcommand{leadingMonomial p\free{p }}
\indentrel{3}\begin{verbatim}
            2
   (27)  x y z
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty27}
\begin{paste}{PolynomialXmpPageEmpty27}{PolynomialXmpPagePatch27}
\pastebutton{PolynomialXmpPageEmpty27}{\showpaste}
\tab{5}\spadcommand{leadingMonomial p\free{p }}
\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPagePatch28}
\begin{paste}{PolynomialXmpPageFull28}{PolynomialXmpPageEmpty28}
\pastebutton{PolynomialXmpPageFull28}{\hidepaste}
\tab{5}\spadcommand{reductum p\free{p }}
```

```
\indentrel{3}\begin{verbatim}
   (28)  (- 2x y + x)z
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty28}
\begin{paste}{PolynomialXmpPageEmpty28}{PolynomialXmpPagePatch28}
\pastebutton{PolynomialXmpPageEmpty28}{\showpaste}
\tab{5}\spadcommand{reductum p\free{p }}
\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPagePatch29}
\begin{paste}{PolynomialXmpPageFull29}{PolynomialXmpPageEmpty29}
\pastebutton{PolynomialXmpPageFull29}{\hidepaste}
\tab{5}\spadcommand{p - leadingMonomial p - reductum p\free{p }}
\indentrel{3}\begin{verbatim}
   (29)  0
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty29}
\begin{paste}{PolynomialXmpPageEmpty29}{PolynomialXmpPagePatch29}
\pastebutton{PolynomialXmpPageEmpty29}{\showpaste}
\tab{5}\spadcommand{p - leadingMonomial p - reductum p\free{p }}
\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPagePatch30}
\begin{paste}{PolynomialXmpPageFull30}{PolynomialXmpPageEmpty30}
\pastebutton{PolynomialXmpPageFull30}{\hidepaste}
\tab{5}\spadcommand{leadingCoefficient p\free{p }}
\indentrel{3}\begin{verbatim}
   (30)  1
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty30}
\begin{paste}{PolynomialXmpPageEmpty30}{PolynomialXmpPagePatch30}
\pastebutton{PolynomialXmpPageEmpty30}{\showpaste}
\tab{5}\spadcommand{leadingCoefficient p\free{p }}
\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPagePatch31}
\begin{paste}{PolynomialXmpPageFull31}{PolynomialXmpPageEmpty31}
\pastebutton{PolynomialXmpPageFull31}{\hidepaste}
\tab{5}\spadcommand{p\free{p }}
\indentrel{3}\begin{verbatim}
             2
```

```
   (31)  (x y  - 2x y + x)z
                              Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty31}
\begin{paste}{PolynomialXmpPageEmpty31}{PolynomialXmpPagePatch31}
\pastebutton{PolynomialXmpPageEmpty31}{\showpaste}
\tab{5}\spadcommand{p\free{p }}
\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPagePatch32}
\begin{paste}{PolynomialXmpPageFull32}{PolynomialXmpPageEmpty32}
\pastebutton{PolynomialXmpPageFull32}{\hidepaste}
\tab{5}\spadcommand{eval(p,x,w)\free{p }}
\indentrel{3}\begin{verbatim}
             2
   (32)  (w y  - 2w y + w)z
                              Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty32}
\begin{paste}{PolynomialXmpPageEmpty32}{PolynomialXmpPagePatch32}
\pastebutton{PolynomialXmpPageEmpty32}{\showpaste}
\tab{5}\spadcommand{eval(p,x,w)\free{p }}
\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPagePatch33}
\begin{paste}{PolynomialXmpPageFull33}{PolynomialXmpPageEmpty33}
\pastebutton{PolynomialXmpPageFull33}{\hidepaste}
\tab{5}\spadcommand{eval(p,x,1)\free{p }}
\indentrel{3}\begin{verbatim}
           2
   (33)  (y  - 2y + 1)z
                              Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty33}
\begin{paste}{PolynomialXmpPageEmpty33}{PolynomialXmpPagePatch33}
\pastebutton{PolynomialXmpPageEmpty33}{\showpaste}
\tab{5}\spadcommand{eval(p,x,1)\free{p }}
\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPagePatch34}
\begin{paste}{PolynomialXmpPageFull34}{PolynomialXmpPageEmpty34}
\pastebutton{PolynomialXmpPageFull34}{\hidepaste}
\tab{5}\spadcommand{eval(p,x,y**2 - 1)\free{p }}
\indentrel{3}\begin{verbatim}
```

```
          4     3
   (34)  (y  - 2y  + 2y - 1)z
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty34}
\begin{paste}{PolynomialXmpPageEmpty34}{PolynomialXmpPagePatch34}
\pastebutton{PolynomialXmpPageEmpty34}{\showpaste}
\tab{5}\spadcommand{eval(p,x,y**2 - 1)\free{p }}
\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPagePatch35}
\begin{paste}{PolynomialXmpPageFull35}{PolynomialXmpPageEmpty35}
\pastebutton{PolynomialXmpPageFull35}{\hidepaste}
\tab{5}\spadcommand{D(p,x)\free{p }}
\indentrel{3}\begin{verbatim}
          2
   (35)  (y  - 2y + 1)z
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty35}
\begin{paste}{PolynomialXmpPageEmpty35}{PolynomialXmpPagePatch35}
\pastebutton{PolynomialXmpPageEmpty35}{\showpaste}
\tab{5}\spadcommand{D(p,x)\free{p }}
\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPagePatch36}
\begin{paste}{PolynomialXmpPageFull36}{PolynomialXmpPageEmpty36}
\pastebutton{PolynomialXmpPageFull36}{\hidepaste}
\tab{5}\spadcommand{D(p,y)\free{p }}
\indentrel{3}\begin{verbatim}
   (36)  (2x y - 2x)z
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty36}
\begin{paste}{PolynomialXmpPageEmpty36}{PolynomialXmpPagePatch36}
\pastebutton{PolynomialXmpPageEmpty36}{\showpaste}
\tab{5}\spadcommand{D(p,y)\free{p }}
\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPagePatch37}
\begin{paste}{PolynomialXmpPageFull37}{PolynomialXmpPageEmpty37}
\pastebutton{PolynomialXmpPageFull37}{\hidepaste}
\tab{5}\spadcommand{D(p,z)\free{p }}
\indentrel{3}\begin{verbatim}
```

```
           2
   (37)  x y  - 2x y + x
                              Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty37}
\begin{paste}{PolynomialXmpPageEmpty37}{PolynomialXmpPagePatch37}
\pastebutton{PolynomialXmpPageEmpty37}{\showpaste}
\tab{5}\spadcommand{D(p,z)\free{p }}
\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPagePatch38}
\begin{paste}{PolynomialXmpPageFull38}{PolynomialXmpPageEmpty38}
\pastebutton{PolynomialXmpPageFull38}{\hidepaste}
\tab{5}\spadcommand{integrate(p,y)\free{p }}
\indentrel{3}\begin{verbatim}
         1    3      2
   (38)  ( x y  - x y  + x y)z
         3
                     Type: Polynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty38}
\begin{paste}{PolynomialXmpPageEmpty38}{PolynomialXmpPagePatch38}
\pastebutton{PolynomialXmpPageEmpty38}{\showpaste}
\tab{5}\spadcommand{integrate(p,y)\free{p }}
\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPagePatch39}
\begin{paste}{PolynomialXmpPageFull39}{PolynomialXmpPageEmpty39}
\pastebutton{PolynomialXmpPageFull39}{\hidepaste}
\tab{5}\spadcommand{qr := monicDivide(p,x+1,x)\free{p }\bound{qr }}
\indentrel{3}\begin{verbatim}
   (39)
             2                            2
   [quotient= (y  - 2y + 1)z,remainder= (- y  + 2y - 1)z]
Type: Record(quotient: Polynomial Integer,remainder: Polynomial Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty39}
\begin{paste}{PolynomialXmpPageEmpty39}{PolynomialXmpPagePatch39}
\pastebutton{PolynomialXmpPageEmpty39}{\showpaste}
\tab{5}\spadcommand{qr := monicDivide(p,x+1,x)\free{p }\bound{qr }}
\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPagePatch40}
\begin{paste}{PolynomialXmpPageFull40}{PolynomialXmpPageEmpty40}
```

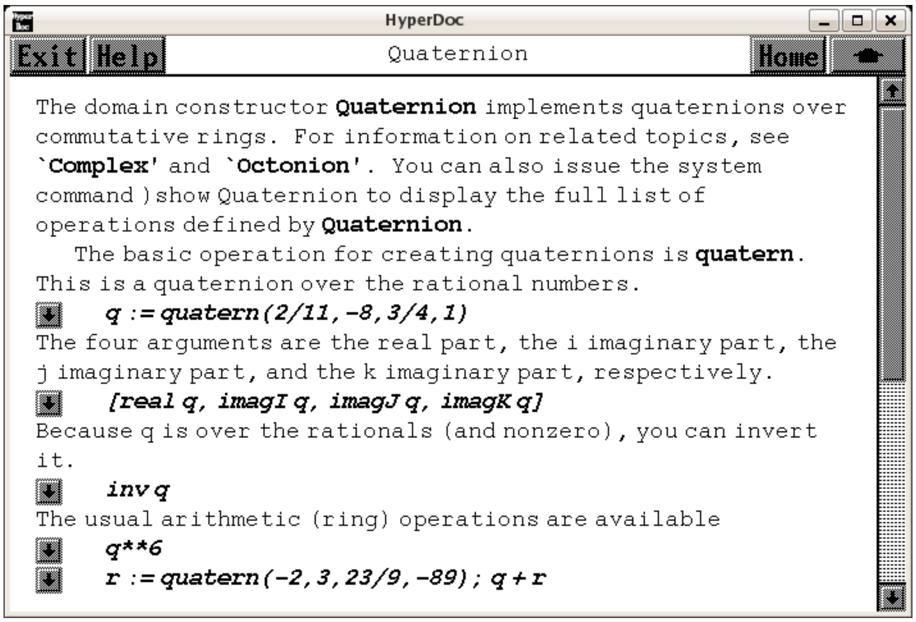
```
\pastebutton{PolynomialXmpPageFull40}{\hidepaste}
\tab{5}\spadcommand{qr.remainder\free{qr }}
\indentrel{3}\begin{verbatim}
             2
   (40)  (- y  + 2y - 1)z
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty40}
\begin{paste}{PolynomialXmpPageEmpty40}{PolynomialXmpPagePatch40}
\pastebutton{PolynomialXmpPageEmpty40}{\showpaste}
\tab{5}\spadcommand{qr.remainder\free{qr }}
\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPagePatch41}
\begin{paste}{PolynomialXmpPageFull41}{PolynomialXmpPageEmpty41}
\pastebutton{PolynomialXmpPageFull41}{\hidepaste}
\tab{5}\spadcommand{p - ((x+1) * qr.quotient + qr.remainder)\free{p }\free{qr }}
\indentrel{3}\begin{verbatim}
   (41)  0
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty41}
\begin{paste}{PolynomialXmpPageEmpty41}{PolynomialXmpPagePatch41}
\pastebutton{PolynomialXmpPageEmpty41}{\showpaste}
\tab{5}\spadcommand{p - ((x+1) * qr.quotient + qr.remainder)\free{p }\free{qr }}
\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPagePatch42}
\begin{paste}{PolynomialXmpPageFull42}{PolynomialXmpPageEmpty42}
\pastebutton{PolynomialXmpPageFull42}{\hidepaste}
\tab{5}\spadcommand{p/q\free{p }\free{q }}
\indentrel{3}\begin{verbatim}
         (y - 1)z
   (42)  
           z + 5
                      Type: Fraction Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty42}
\begin{paste}{PolynomialXmpPageEmpty42}{PolynomialXmpPagePatch42}
\pastebutton{PolynomialXmpPageEmpty42}{\showpaste}
\tab{5}\spadcommand{p/q\free{p }\free{q }}
\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPagePatch43}
```

```
\begin{paste}{PolynomialXmpPageFull43}{PolynomialXmpPageEmpty43}
\pastebutton{PolynomialXmpPageFull43}{\hidepaste}
\tab{5}\spadcommand{(2/3) * x**2 - y + 4/5\bound{prev1 }}
\indentrel{3}\begin{verbatim}
               2  2   4
   (43)  - y +  x  +
               3      5
                      Type: Polynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty43}
\begin{paste}{PolynomialXmpPageEmpty43}{PolynomialXmpPagePatch43}
\pastebutton{PolynomialXmpPageEmpty43}{\showpaste}
\tab{5}\spadcommand{(2/3) * x**2 - y + 4/5\bound{prev1 }}
\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPagePatch44}
\begin{paste}{PolynomialXmpPageFull44}{PolynomialXmpPageEmpty44}
\pastebutton{PolynomialXmpPageFull44}{\hidepaste}
\tab{5}\spadcommand{\% :: FRAC POLY INT\free{prev1 }\bound{prev2 }}
\indentrel{3}\begin{verbatim}
                    2
         - 15y + 10x  + 12
   (44)
                 15
                      Type: Fraction Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty44}
\begin{paste}{PolynomialXmpPageEmpty44}{PolynomialXmpPagePatch44}
\pastebutton{PolynomialXmpPageEmpty44}{\showpaste}
\tab{5}\spadcommand{\% :: FRAC POLY INT\free{prev1 }\bound{prev2 }}
\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPagePatch45}
\begin{paste}{PolynomialXmpPageFull45}{PolynomialXmpPageEmpty45}
\pastebutton{PolynomialXmpPageFull45}{\hidepaste}
\tab{5}\spadcommand{\% :: POLY FRAC INT\free{prev2 }\bound{prev3 }}
\indentrel{3}\begin{verbatim}
               2  2   4
   (45)  - y +  x  +
               3      5
                      Type: Polynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty45}
\begin{paste}{PolynomialXmpPageEmpty45}{PolynomialXmpPagePatch45}
```

```
\pastebutton{PolynomialXmpPageEmpty45}{\showpaste}
\tab{5}\spadcommand{\% :: POLY FRAC INT\free{prev2 }\bound{prev3 }}
\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPagePatch46}
\begin{paste}{PolynomialXmpPageFull46}{PolynomialXmpPageEmpty46}
\pastebutton{PolynomialXmpPageFull46}{\hidepaste}
\tab{5}\spadcommand{map(numeric,\%)\free{prev3 }}
\indentrel{3}\begin{verbatim}
                                            2
   (46)  - 1.0 y + 0.6666666666 6666666667 x  + 0.8
                                 Type: Polynomial Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{PolynomialXmpPageEmpty46}
\begin{paste}{PolynomialXmpPageEmpty46}{PolynomialXmpPagePatch46}
\pastebutton{PolynomialXmpPageEmpty46}{\showpaste}
\tab{5}\spadcommand{map(numeric,\%)\free{prev3 }}
\end{paste}\end{patch}
```

3.89 quat.ht

Quaternion



⇐ "Axiom Number Types" (NumberPage) 3.80 on page 1021
⇒ "Complex" (ComplexXmpPage) 3.16 on page 254
⇒ "Octonion" (OctonionXmpPage) 3.81 on page 1044

— **quat.ht** —

```
\begin{page}{QuaternionXmpPage}{Quaternion}
\beginscroll
The domain constructor \spadtype{Quaternion} implements quaternions over
commutative rings.
For information on related topics, see
\downlink{‘Complex’}{ComplexXmpPage}\ignore{Complex} and
\downlink{‘Octonion’}{OctonionXmpPage}\ignore{Octonion}.
You can also issue the system command
\spadcmd{)show Quaternion}
to display the full list of operations defined by
\spadtype{Quaternion}.

\xtc{
The basic operation for creating quaternions is
\spadfunFrom{quatern}{Quaternion}.
This is a quaternion over the rational numbers.
}{
\spadpaste{q := quatern(2/11,-8,3/4,1) \bound{q}}
```

```
}
\xtc{
The four arguments are the real part, the \spad{i} imaginary part, the
\spad{j} imaginary part, and the \spad{k} imaginary part, respectively.
}{
\spadpaste{[real q, imagI q, imagJ q, imagK q] \free{q}}
}
\xtc{
Because \spad{q} is over the rationals (and nonzero), you can invert it.
}{
\spadpaste{inv q \free{q}}
}
\xtc{
The usual arithmetic (ring) operations are available
}{
\spadpaste{q**6 \free{q}}
}
\xtc{
}{
\spadpaste{r := quatern(-2,3,23/9,-89); q + r \bound{r}\free{q}}
}
%
\xtc{
In general, multiplication is not commutative.
}{
\spadpaste{q * r - r * q\free{q r}}
}
\xtc{
There are no predefined constants for the imaginary \spad{i, j},
and \spad{k} parts, but you can easily define them.
}{
\spadpaste{i:=quatern(0,1,0,0); j:=quatern(0,0,1,0);
k:=quatern(0,0,0,1) \bound{i j k}}
}
\xtc{
These satisfy the normal identities.
}{
\spadpaste{[i*i, j*j, k*k, i*j, j*k, k*i, q*i] \free{i j k q}}
}
\xtc{
The norm is the quaternion times its conjugate.
}{
\spadpaste{norm q \free{q}}
}
\xtc{
}{
\spadpaste{conjugate q \free{q} \bound{prev}}
}
\xtc{
}{
```

```
\spadpaste{q * \% \free{q prev}}
}
\endscroll
\autobuttons
\end{page}

\begin{patch}{QuaternionXmpPagePatch1}
\begin{paste}{QuaternionXmpPageFull1}{QuaternionXmpPageEmpty1}
\pastebutton{QuaternionXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{q := quatern(2/11,-8,3/4,1)\bound{q }}
\indentrel{3}\begin{verbatim}
         2       3
   (1)   - 8i +  j + k
        11       4
                      Type: Quaternion Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{QuaternionXmpPageEmpty1}
\begin{paste}{QuaternionXmpPageEmpty1}{QuaternionXmpPagePatch1}
\pastebutton{QuaternionXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{q := quatern(2/11,-8,3/4,1)\bound{q }}
\end{paste}\end{patch}

\begin{patch}{QuaternionXmpPagePatch2}
\begin{paste}{QuaternionXmpPageFull2}{QuaternionXmpPageEmpty2}
\pastebutton{QuaternionXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{[real q, imagI q, imagJ q, imagK q]\free{q }}
\indentrel{3}\begin{verbatim}
          2     3
   (2)  [,- 8,,1]
         11     4
                            Type: List Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{QuaternionXmpPageEmpty2}
\begin{paste}{QuaternionXmpPageEmpty2}{QuaternionXmpPagePatch2}
\pastebutton{QuaternionXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{[real q, imagI q, imagJ q, imagK q]\free{q }}
\end{paste}\end{patch}

\begin{patch}{QuaternionXmpPagePatch3}
\begin{paste}{QuaternionXmpPageFull3}{QuaternionXmpPageEmpty3}
\pastebutton{QuaternionXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{inv q\free{q }}
\indentrel{3}\begin{verbatim}
          352     15488      484       1936
   (3)   +  i -  j -  k
        126993   126993     42331     126993
```

```
                      Type: Quaternion Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{QuaternionXmpPageEmpty3}
\begin{paste}{QuaternionXmpPageEmpty3}{QuaternionXmpPagePatch3}
\pastebutton{QuaternionXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{inv q\free{q }}
\end{paste}\end{patch}

\begin{patch}{QuaternionXmpPagePatch4}
\begin{paste}{QuaternionXmpPageFull4}{QuaternionXmpPageEmpty4}
\pastebutton{QuaternionXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{q**6\free{q }}
\indentrel{3}\begin{verbatim}
   (4)
       2029490709319345   48251690851     144755072553
     -  -  i +  j
          7256313856        1288408         41229056
   +
     48251690851
      k
       10307264
                      Type: Quaternion Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{QuaternionXmpPageEmpty4}
\begin{paste}{QuaternionXmpPageEmpty4}{QuaternionXmpPagePatch4}
\pastebutton{QuaternionXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{q**6\free{q }}
\end{paste}\end{patch}

\begin{patch}{QuaternionXmpPagePatch5}
\begin{paste}{QuaternionXmpPageFull5}{QuaternionXmpPageEmpty5}
\pastebutton{QuaternionXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{r := quatern(-2,3,23/9,-89); q + r\bound{r }\free{q }}
\indentrel{3}\begin{verbatim}
          20        119
   (5)  -  - 5i +  j - 88k
          11         36
                      Type: Quaternion Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{QuaternionXmpPageEmpty5}
\begin{paste}{QuaternionXmpPageEmpty5}{QuaternionXmpPagePatch5}
\pastebutton{QuaternionXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{r := quatern(-2,3,23/9,-89); q + r\bound{r }\free{q }}
\end{paste}\end{patch}
```

```
\begin{patch}{QuaternionXmpPagePatch6}
\begin{paste}{QuaternionXmpPageFull6}{QuaternionXmpPageEmpty6}
\pastebutton{QuaternionXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{q * r - r * q\free{q r }}
\indentrel{3}\begin{verbatim}
         2495             817
   (6)  -  i - 1418j -  k
          18               18
                      Type: Quaternion Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{QuaternionXmpPageEmpty6}
\begin{paste}{QuaternionXmpPageEmpty6}{QuaternionXmpPagePatch6}
\pastebutton{QuaternionXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{q * r - r * q\free{q r }}
\end{paste}\end{patch}

\begin{patch}{QuaternionXmpPagePatch7}
\begin{paste}{QuaternionXmpPageFull7}{QuaternionXmpPageEmpty7}
\pastebutton{QuaternionXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{i:=quatern(0,1,0,0); j:=quatern(0,0,1,0); k:=quatern(0,0,0,1)\bound{i j k }}
\indentrel{3}\begin{verbatim}
   (7)  k
                               Type: Quaternion Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{QuaternionXmpPageEmpty7}
\begin{paste}{QuaternionXmpPageEmpty7}{QuaternionXmpPagePatch7}
\pastebutton{QuaternionXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{i:=quatern(0,1,0,0); j:=quatern(0,0,1,0); k:=quatern(0,0,0,1)\bound{i j k }}
\end{paste}\end{patch}

\begin{patch}{QuaternionXmpPagePatch8}
\begin{paste}{QuaternionXmpPageFull8}{QuaternionXmpPageEmpty8}
\pastebutton{QuaternionXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{[i*i, j*j, k*k, i*j, j*k, k*i, q*i]\free{i j k q }}
\indentrel{3}\begin{verbatim}
                               2         3
   (8)  [- 1,- 1,- 1,k,i,j,8 +  i + j -  k]
                              11         4
                 Type: List Quaternion Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{QuaternionXmpPageEmpty8}
\begin{paste}{QuaternionXmpPageEmpty8}{QuaternionXmpPagePatch8}
\pastebutton{QuaternionXmpPageEmpty8}{\showpaste}
```

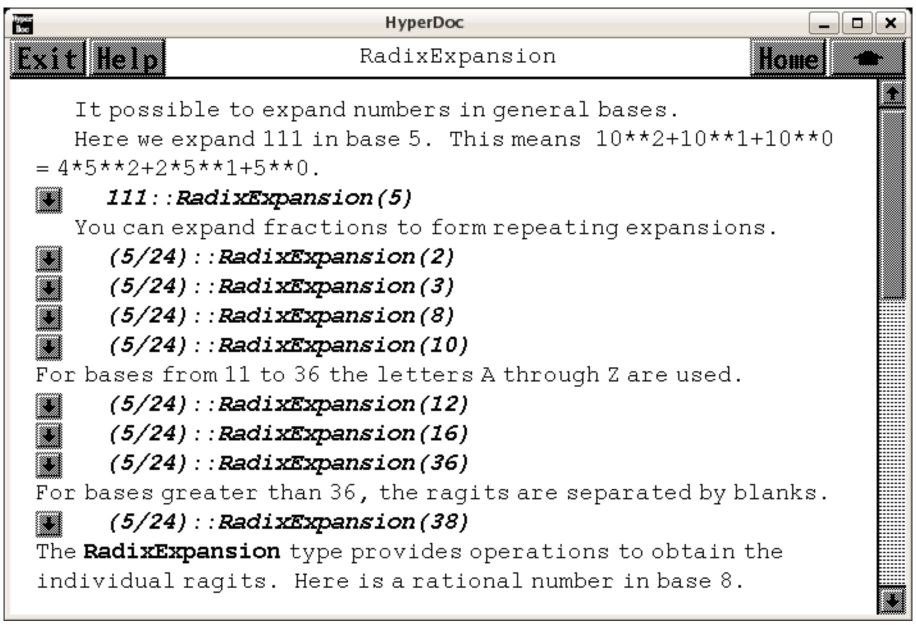
```
\tab{5}\spadcommand{[i*i, j*j, k*k, i*j, j*k, k*i, q*i]\free{i j k q }}
\end{paste}\end{patch}

\begin{patch}{QuaternionXmpPagePatch9}
\begin{paste}{QuaternionXmpPageFull9}{QuaternionXmpPageEmpty9}
\pastebutton{QuaternionXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{norm q\free{q }}
\indentrel{3}\begin{verbatim}
        126993
   (9)
         1936
                                 Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{QuaternionXmpPageEmpty9}
\begin{paste}{QuaternionXmpPageEmpty9}{QuaternionXmpPagePatch9}
\pastebutton{QuaternionXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{norm q\free{q }}
\end{paste}\end{patch}

\begin{patch}{QuaternionXmpPagePatch10}
\begin{paste}{QuaternionXmpPageFull10}{QuaternionXmpPageEmpty10}
\pastebutton{QuaternionXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{conjugate q\free{q }\bound{prev }}
\indentrel{3}\begin{verbatim}
          2        3
   (10)   + 8i -  j - k
         11        4
                      Type: Quaternion Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{QuaternionXmpPageEmpty10}
\begin{paste}{QuaternionXmpPageEmpty10}{QuaternionXmpPagePatch10}
\pastebutton{QuaternionXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{conjugate q\free{q }\bound{prev }}
\end{paste}\end{patch}

\begin{patch}{QuaternionXmpPagePatch11}
\begin{paste}{QuaternionXmpPageFull11}{QuaternionXmpPageEmpty11}
\pastebutton{QuaternionXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{q * \%\free{q prev }}
\indentrel{3}\begin{verbatim}
         126993
   (11)
          1936
                      Type: Quaternion Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{QuaternionXmpPageEmpty11}
\begin{paste}{QuaternionXmpPageEmpty11}{QuaternionXmpPagePatch11}
\pastebutton{QuaternionXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{q * \%\free{q prev }}
\end{paste}\end{patch}
```

———-

3.90 radix.ht

RadixExpansion



⇐ “Axiom Number Types” (NumberPage) 3.80 on page 1021
⇐ “Integer” (IntegerXmpPage) 3.55 on page 696
⇒ “HexadecimalExpansion” (HexExpansionXmpPage) 3.54 on page 692
⇒ “DecimalExpansion” (DecimalExpansionXmpPage) 3.21 on page 348
⇒ “BinaryExpansion” (BinaryExpansionXmpPage) 3.8 on page 163

— **radix.ht** —

```
\begin{page}{RadixExpansionXmpPage}{RadixExpansion}
\beginscroll
It is possible to expand numbers in general bases.
\labelSpace{2pc}
```

```
\xtc{
Here we expand \spad{111} in base \spad{5}.
This means
\texht{$10^2+10^1+10^0 = 4 \cdot 5^2+2 \cdot 5^1 + 5^0.$}{%
\spad{10**2+10**1+10**0 = 4*5**2+2*5**1+5**0.}}
}{
\spadpaste{111::RadixExpansion(5)}
}

\xtc{
You can expand fractions to form repeating expansions.
}{
\spadpaste{(5/24)::RadixExpansion(2)}
}
\xtc{
}{
\spadpaste{(5/24)::RadixExpansion(3)}
}
\xtc{
}{
\spadpaste{(5/24)::RadixExpansion(8)}
}
\xtc{
}{
\spadpaste{(5/24)::RadixExpansion(10)}
}
\xtc{
For bases from 11 to 36 the letters A through Z are used.
}{
\spadpaste{(5/24)::RadixExpansion(12)}
}
\xtc{
}{
\spadpaste{(5/24)::RadixExpansion(16)}
}
\xtc{
}{
\spadpaste{(5/24)::RadixExpansion(36)}
}
\xtc{
For bases greater than 36, the ragits are separated by blanks.
}{
\spadpaste{(5/24)::RadixExpansion(38)}
}
\xtc{
The \spadtype{RadixExpansion} type provides operations to obtain the
individual ragits.
Here is a rational number in base \spad{8}.
}{
\spadpaste{a := (76543/210)::RadixExpansion(8) \bound{a}}
```

```
}
\xtc{
The operation \spadfunFrom{wholeRagits}{RadixExpansion} returns a
list of the ragits for the integral part of the number.
}{
\spadpaste{w := wholeRagits a \free{a}\bound{w}}
}
\xtc{
The operations \spadfunFrom{prefixRagits}{RadixExpansion} and
\spadfunFrom{cycleRagits}{RadixExpansion}
return lists of the initial and repeating ragits in the
fractional part of the number.
}{
\spadpaste{f0 := prefixRagits a \free{a}\bound{f0}}
}
\xtc{
}{
\spadpaste{f1 := cycleRagits a \free{a}\bound{f1}}
}
\xtc{
You can construct any radix expansion by giving the
whole, prefix and cycle parts.
The declaration is necessary to let Axiom
know the base of the ragits.
}{
\spadpaste{u:RadixExpansion(8):=wholeRadix(w)+fractRadix(f0,f1)
\free{w f0 f1}\bound{u}}
}
\xtc{
If there is no repeating part, then the list \spad{[0]} should be used.
}{
\spadpaste{v: RadixExpansion(12) := fractRadix([1,2,3,11], [0]) \bound{v}}
}
\xtc{
If you are not interested in the repeating nature of the expansion,
an infinite stream of ragits can be obtained using
\spadfunFrom{fractRagits}{RadixExpansion}.
}{
\spadpaste{fractRagits(u) \free{u}}
}
\xtc{
Of course, it's possible to recover the fraction representation:
}{
\spadpaste{a :: Fraction(Integer) \free{a}}
}

\showBlurb{RadixExpansion}
More examples of expansions are available in
\downlink{`DecimalExpansion'}{DecimalExpansionXmpPage}
\ignore{DecimalExpansion},
```

```
\downlink{‘BinaryExpansion’}{BinaryExpansionXmpPage}
\ignore{BinaryExpansion}, and
\downlink{‘HexadecimalExpansion’}{HexExpansionXmpPage}
\ignore{HexadecimalExpansion}.
\endscroll
\autobuttons
\end{page}

\begin{patch}{RadixExpansionXmpPagePatch1}
\begin{paste}{RadixExpansionXmpPageFull1}{RadixExpansionXmpPageEmpty1}
\pastebutton{RadixExpansionXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{111::RadixExpansion(5)}
\indentrel{3}\begin{verbatim}
   (1)  421
                                 Type: RadixExpansion 5
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RadixExpansionXmpPageEmpty1}
\begin{paste}{RadixExpansionXmpPageEmpty1}{RadixExpansionXmpPagePatch1}
\pastebutton{RadixExpansionXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{111::RadixExpansion(5)}
\end{paste}\end{patch}

\begin{patch}{RadixExpansionXmpPagePatch2}
\begin{paste}{RadixExpansionXmpPageFull2}{RadixExpansionXmpPageEmpty2}
\pastebutton{RadixExpansionXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{(5/24)::RadixExpansion(2)}
\indentrel{3}\begin{verbatim}
              --
   (2)  0.00110
                                 Type: RadixExpansion 2
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RadixExpansionXmpPageEmpty2}
\begin{paste}{RadixExpansionXmpPageEmpty2}{RadixExpansionXmpPagePatch2}
\pastebutton{RadixExpansionXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{(5/24)::RadixExpansion(2)}
\end{paste}\end{patch}

\begin{patch}{RadixExpansionXmpPagePatch3}
\begin{paste}{RadixExpansionXmpPageFull3}{RadixExpansionXmpPageEmpty3}
\pastebutton{RadixExpansionXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{(5/24)::RadixExpansion(3)}
\indentrel{3}\begin{verbatim}
            --
   (3)  0.012
                                 Type: RadixExpansion 3
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RadixExpansionXmpPageEmpty3}
\begin{paste}{RadixExpansionXmpPageEmpty3}{RadixExpansionXmpPagePatch3}
\pastebutton{RadixExpansionXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{(5/24)::RadixExpansion(3)}
\end{paste}\end{patch}

\begin{patch}{RadixExpansionXmpPagePatch4}
\begin{paste}{RadixExpansionXmpPageFull4}{RadixExpansionXmpPageEmpty4}
\pastebutton{RadixExpansionXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{(5/24)::RadixExpansion(8)}
\indentrel{3}\begin{verbatim}
          --
   (4)  0.152
                                 Type: RadixExpansion 8
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RadixExpansionXmpPageEmpty4}
\begin{paste}{RadixExpansionXmpPageEmpty4}{RadixExpansionXmpPagePatch4}
\pastebutton{RadixExpansionXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{(5/24)::RadixExpansion(8)}
\end{paste}\end{patch}

\begin{patch}{RadixExpansionXmpPagePatch5}
\begin{paste}{RadixExpansionXmpPageFull5}{RadixExpansionXmpPageEmpty5}
\pastebutton{RadixExpansionXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{(5/24)::RadixExpansion(10)}
\indentrel{3}\begin{verbatim}
             -
   (5)  0.2083
                                Type: RadixExpansion 10
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RadixExpansionXmpPageEmpty5}
\begin{paste}{RadixExpansionXmpPageEmpty5}{RadixExpansionXmpPagePatch5}
\pastebutton{RadixExpansionXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{(5/24)::RadixExpansion(10)}
\end{paste}\end{patch}

\begin{patch}{RadixExpansionXmpPagePatch6}
\begin{paste}{RadixExpansionXmpPageFull6}{RadixExpansionXmpPageEmpty6}
\pastebutton{RadixExpansionXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{(5/24)::RadixExpansion(12)}
\indentrel{3}\begin{verbatim}
   (6)  0.26
                                Type: RadixExpansion 12
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RadixExpansionXmpPageEmpty6}
\begin{paste}{RadixExpansionXmpPageEmpty6}{RadixExpansionXmpPagePatch6}
\pastebutton{RadixExpansionXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{(5/24)::RadixExpansion(12)}
\end{paste}\end{patch}

\begin{patch}{RadixExpansionXmpPagePatch7}
\begin{paste}{RadixExpansionXmpPageFull7}{RadixExpansionXmpPageEmpty7}
\pastebutton{RadixExpansionXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{(5/24)::RadixExpansion(16)}
\indentrel{3}\begin{verbatim}
           -
   (7)  0.35
                                Type: RadixExpansion 16
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RadixExpansionXmpPageEmpty7}
\begin{paste}{RadixExpansionXmpPageEmpty7}{RadixExpansionXmpPagePatch7}
\pastebutton{RadixExpansionXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{(5/24)::RadixExpansion(16)}
\end{paste}\end{patch}

\begin{patch}{RadixExpansionXmpPagePatch8}
\begin{paste}{RadixExpansionXmpPageFull8}{RadixExpansionXmpPageEmpty8}
\pastebutton{RadixExpansionXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{(5/24)::RadixExpansion(36)}
\indentrel{3}\begin{verbatim}
   (8)  0.7I
                                Type: RadixExpansion 36
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RadixExpansionXmpPageEmpty8}
\begin{paste}{RadixExpansionXmpPageEmpty8}{RadixExpansionXmpPagePatch8}
\pastebutton{RadixExpansionXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{(5/24)::RadixExpansion(36)}
\end{paste}\end{patch}

\begin{patch}{RadixExpansionXmpPagePatch9}
\begin{paste}{RadixExpansionXmpPageFull9}{RadixExpansionXmpPageEmpty9}
\pastebutton{RadixExpansionXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{(5/24)::RadixExpansion(38)}
\indentrel{3}\begin{verbatim}
                    -----
   (9)  0 . 7 34 31 25 12
                                Type: RadixExpansion 38
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RadixExpansionXmpPageEmpty9}
\begin{paste}{RadixExpansionXmpPageEmpty9}{RadixExpansionXmpPagePatch9}
\pastebutton{RadixExpansionXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{(5/24)::RadixExpansion(38)}
\end{paste}\end{patch}

\begin{patch}{RadixExpansionXmpPagePatch10}
\begin{paste}{RadixExpansionXmpPageFull10}{RadixExpansionXmpPageEmpty10}
\pastebutton{RadixExpansionXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{a := (76543/210)::RadixExpansion(8)\bound{a }}
\indentrel{3}\begin{verbatim}
             ----
   (10)  554.37307
                                Type: RadixExpansion 8
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RadixExpansionXmpPageEmpty10}
\begin{paste}{RadixExpansionXmpPageEmpty10}{RadixExpansionXmpPagePatch10}
\pastebutton{RadixExpansionXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{a := (76543/210)::RadixExpansion(8)\bound{a }}
\end{paste}\end{patch}

\begin{patch}{RadixExpansionXmpPagePatch11}
\begin{paste}{RadixExpansionXmpPageFull11}{RadixExpansionXmpPageEmpty11}
\pastebutton{RadixExpansionXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{w := wholeRagits a\free{a }\bound{w }}
\indentrel{3}\begin{verbatim}
   (11)  [5,5,4]
                                    Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RadixExpansionXmpPageEmpty11}
\begin{paste}{RadixExpansionXmpPageEmpty11}{RadixExpansionXmpPagePatch11}
\pastebutton{RadixExpansionXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{w := wholeRagits a\free{a }\bound{w }}
\end{paste}\end{patch}

\begin{patch}{RadixExpansionXmpPagePatch12}
\begin{paste}{RadixExpansionXmpPageFull12}{RadixExpansionXmpPageEmpty12}
\pastebutton{RadixExpansionXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{f0 := prefixRagits a\free{a }\bound{f0 }}
\indentrel{3}\begin{verbatim}
   (12)  [3]
                                    Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{RadixExpansionXmpPageEmpty12}
\begin{paste}{RadixExpansionXmpPageEmpty12}{RadixExpansionXmpPagePatch12}
\pastebutton{RadixExpansionXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{f0 := prefixRagits a\free{a }\bound{f0 }}
\end{paste}\end{patch}

\begin{patch}{RadixExpansionXmpPagePatch13}
\begin{paste}{RadixExpansionXmpPageFull13}{RadixExpansionXmpPageEmpty13}
\pastebutton{RadixExpansionXmpPageFull13}{\hidepaste}
\tab{5}\spadcommand{f1 := cycleRagits a\free{a }\bound{f1 }}
\indentrel{3}\begin{verbatim}
   (13)  [7,3,0,7]
                                     Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RadixExpansionXmpPageEmpty13}
\begin{paste}{RadixExpansionXmpPageEmpty13}{RadixExpansionXmpPagePatch13}
\pastebutton{RadixExpansionXmpPageEmpty13}{\showpaste}
\tab{5}\spadcommand{f1 := cycleRagits a\free{a }\bound{f1 }}
\end{paste}\end{patch}

\begin{patch}{RadixExpansionXmpPagePatch14}
\begin{paste}{RadixExpansionXmpPageFull14}{RadixExpansionXmpPageEmpty14}
\pastebutton{RadixExpansionXmpPageFull14}{\hidepaste}
\tab{5}\spadcommand{u:RadixExpansion(8):=wholeRadix(w)+fractRadix(f0,f1)\free{w f0 f1 }\bound{u }}
\indentrel{3}\begin{verbatim}
                ----
   (14)  554.37307
                                 Type: RadixExpansion 8
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RadixExpansionXmpPageEmpty14}
\begin{paste}{RadixExpansionXmpPageEmpty14}{RadixExpansionXmpPagePatch14}
\pastebutton{RadixExpansionXmpPageEmpty14}{\showpaste}
\tab{5}\spadcommand{u:RadixExpansion(8):=wholeRadix(w)+fractRadix(f0,f1)\free{w f0 f1 }\bound{u }}
\end{paste}\end{patch}

\begin{patch}{RadixExpansionXmpPagePatch15}
\begin{paste}{RadixExpansionXmpPageFull15}{RadixExpansionXmpPageEmpty15}
\pastebutton{RadixExpansionXmpPageFull15}{\hidepaste}
\tab{5}\spadcommand{v: RadixExpansion(12) := fractRadix([1,2,3,11], [0])\bound{v }}
\indentrel{3}\begin{verbatim}
                 -
   (15)  0.123B0
                                Type: RadixExpansion 12
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{RadixExpansionXmpPageEmpty15}
\begin{paste}{RadixExpansionXmpPageEmpty15}{RadixExpansionXmpPagePatch15}
\pastebutton{RadixExpansionXmpPageEmpty15}{\showpaste}
\tab{5}\spadcommand{v: RadixExpansion(12) := fractRadix([1,2,3,11], [0])\bound{v }}
\end{paste}\end{patch}

\begin{patch}{RadixExpansionXmpPagePatch16}
\begin{paste}{RadixExpansionXmpPageFull16}{RadixExpansionXmpPageEmpty16}
\pastebutton{RadixExpansionXmpPageFull16}{\hidepaste}
\tab{5}\spadcommand{fractRagits(u)\free{u }}
\indentrel{3}\begin{verbatim}
              -------
   (16)  [3,7,3,0,7,7]
                                    Type: Stream Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RadixExpansionXmpPageEmpty16}
\begin{paste}{RadixExpansionXmpPageEmpty16}{RadixExpansionXmpPagePatch16}
\pastebutton{RadixExpansionXmpPageEmpty16}{\showpaste}
\tab{5}\spadcommand{fractRagits(u)\free{u }}
\end{paste}\end{patch}

\begin{patch}{RadixExpansionXmpPagePatch17}
\begin{paste}{RadixExpansionXmpPageFull17}{RadixExpansionXmpPageEmpty17}
\pastebutton{RadixExpansionXmpPageFull17}{\hidepaste}
\tab{5}\spadcommand{a :: Fraction(Integer)\free{a }}
\indentrel{3}\begin{verbatim}
         76543
   (17)
          210
                                  Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RadixExpansionXmpPageEmpty17}
\begin{paste}{RadixExpansionXmpPageEmpty17}{RadixExpansionXmpPagePatch17}
\pastebutton{RadixExpansionXmpPageEmpty17}{\showpaste}
\tab{5}\spadcommand{a :: Fraction(Integer)\free{a }}
\end{paste}\end{patch}
```

3.91 reclos.ht

RealClosure

— reclos.ht —

```
\begin{page}{RealClosureXmpPage}{RealClosure}
\beginscroll

The Real Closure 1.0 package provided by
Renaud Rioboo (Renaud.Rioboo@lip6.fr) consists of different
packages, categories and domains :

\begin{items}
\item The package \axiomType{RealPolynomialUtilitiesPackage} which
needs a \axiomType{Field} {\em F} and a
\axiomType{UnivariatePolynomialCategory} domain with coefficients in
{\em F}.  It computes some simple functions such as Sturm and
Sylvester sequences
(\axiomOpFrom{sturmSequence}{RealPolynomialUtilitiesPackage},
\axiomOpFrom{sylvesterSequence}{RealPolynomialUtilitiesPackage}).

\item The category \axiomType{RealRootCharacterizationCategory}
provides abstract functions to work with "real roots" of univariate
polynomials. These resemble variables with some functionality needed
to compute important operations.

\item The category \axiomType{RealClosedField} provides common
operations available over real closed fiels. These include finding all
the roots of a univariate polynomial, taking square (and higher)
roots, ...

\item The domain \axiomType{RightOpenIntervalRootCharacterization} is
the main code that provides the functionality of
\axiomType{RealRootCharacterizationCategory} for the case of
archimedean fields. Abstract roots are encoded with a left closed
right open interval containing the root together with a defining
polynomial for the root.

\item The \axiomType{RealClosure} domain is the end-user code. It
provides usual arithmetic with real algebraic numbers, along with the
functionality of a real closed field. It also provides functions to
approximate a real algebraic number by an element of the base
field. This approximation may either be absolute
(\axiomOpFrom{approximate}{RealClosure}) or relative
(\axiomOpFrom{relativeApprox}{RealClosure}).
```

```
\end{items}

\centerline{CAVEATS}

Since real algebraic expressions are stored as depending on "real
roots" which are managed like variables, there is an ordering on
these. This ordering is dynamical in the sense that any new algebraic
takes precedence over older ones. In particular every creation
function raises a new "real root". This has the effect that when you
type something like \spad{sqrt(2) + sqrt(2)} you have two new
variables which happen to be equal. To avoid this name the expression
such as in \spad{s2 := sqrt(2) ; s2 + s2}

Also note that computing times depend strongly on the ordering you
implicitly provide. Please provide algebraics in the order which seems
most natural to you.

\centerline{LIMITATIONS}

This packages uses algorithms which are published in [1] and [2] which
are based on field arithmetics, in particular for polynomial gcd
related algorithms. This can be quite slow for high degree polynomials
and subresultants methods usually work best. Beta versions of the
package try to use these techniques in a better way and work
significantly faster. These are mostly based on unpublished algorithms
and cannot be distributed. Please contact the author if you have a
particular problem to solve or want to use these versions.

Be aware that approximations behave as post-processing and that all
computations are done exactly. They can thus be quite time consuming
when depending on several "real roots".

\centerline{REFERENCES}


[1]  R. Rioboo : Real Algebraic Closure of an ordered Field : Implementation
     in Axiom.
     In proceedings of the ISSAC'92 Conference, Berkeley 1992 pp. 206-215.

[2]  Z. Ligatsikas, R. Rioboo, M. F. Roy : Generic computation of the real
     closure of an ordered field.
     In Mathematics and Computers in Simulation Volume 42, Issue 4-6,
     November 1996.

\centerline{EXAMPLES}
\xtc{
We shall work with the real closure of the ordered field of
rational numbers.
}{
\spadpaste{Ran := RECLOS(FRAC INT) \bound{Ran}}
```

```
}
\xtc{
Some simple signs for square roots, these correspond to an extension
of degree 16 of the rational numbers. Examples provided by J. Abbot.
}{
\spadpaste{fourSquares(a:Ran,b:Ran,c:Ran,d:Ran):Ran == sqrt(a)+sqrt(b) -
sqrt(c)-sqrt(d) \free{Ran} \bound{fs}}
}
\xtc{
These produce values very close to zero.
}{
\spadpaste{squareDiff1 := fourSquares(73,548,60,586) \free{fs}
\bound{sd1}}
}
\xtc{
}{
\spadpaste{recip(squareDiff1)\free{sd1}}
}
\xtc{
}{
\spadpaste{sign(squareDiff1)\free{sd1}}
}
\xtc{
}{
\spadpaste{squareDiff2 := fourSquares(165,778,86,990) \free{fs}
\bound{sd2}}
}
\xtc{
}{
\spadpaste{recip(squareDiff2)\free{sd2}}
}
\xtc{
}{
\spadpaste{sign(squareDiff2)\free{sd2}}
}
\xtc{
}{
\spadpaste{squareDiff3 := fourSquares(217,708,226,692) \free{fs}
\bound{sd3}}
}
\xtc{
}{
\spadpaste{recip(squareDiff3)\free{sd3}}
}
\xtc{
}{
\spadpaste{sign(squareDiff3)\free{sd3}}
}
\xtc{
}{
```

```
\spadpaste{squareDiff4 := fourSquares(155,836,162,820)
\free{fs}\bound{sd4}}
}
\xtc{
}{
\spadpaste{recip(squareDiff4)\free{sd4}}
}
\xtc{
}{
\spadpaste{sign(squareDiff4)\free{sd4}}
}
\xtc{
}{
\spadpaste{squareDiff5 := fourSquares(591,772,552,818)
\free{fs}\bound{sd5}}
}
\xtc{
}{
\spadpaste{recip(squareDiff5)\free{sd5}}
}
\xtc{
}{
\spadpaste{sign(squareDiff5)\free{sd5}}
}
\xtc{
}{
\spadpaste{squareDiff6 := fourSquares(434,1053,412,1088)
\free{fs}\bound{sd6}}
}
\xtc{
}{
\spadpaste{recip(squareDiff6)\free{sd6}}
}
\xtc{
}{
\spadpaste{sign(squareDiff6)\free{sd6}}
}
\xtc{
}{
\spadpaste{squareDiff7 := fourSquares(514,1049,446,1152)
\free{fs}\bound{sd7}}
}
\xtc{
}{
\spadpaste{recip(squareDiff7)\free{sd7}}
}
\xtc{
}{
\spadpaste{sign(squareDiff7)\free{sd7}}
}
```

```
\xtc{
}{
\spadpaste{squareDiff8 := fourSquares(190,1751,208,1698)
\free{fs}\bound{sd8}}
}
\xtc{
}{
\spadpaste{recip(squareDiff8)\free{sd8}}
}
\xtc{
}{
\spadpaste{sign(squareDiff8)\free{sd8}}
}
\xtc{
This should give three digits of precision
}{
\spadpaste{relativeApprox(squareDiff8,10**(-3))::Float \free{sd8}}
}
\xtc{
The sum of these 4 roots is 0
}{
\spadpaste{l := allRootsOf((x**2-2)**2-2)$Ran \free{Ran} \bound{l}}
}
\xtc{
Check that they are all roots of the same polynomial
}{
\spadpaste{removeDuplicates map(mainDefiningPolynomial,l) \free{l}}
}
\xtc{
We can see at a glance that they are separate roots
}{
\spadpaste{map(mainCharacterization,l) \free{l}}
}
\xtc{
Check the sum and product
}{
\spadpaste{[reduce(+,l),reduce(*,l)-2] \free{l}}
}
\xtc{
A more complicated test that involve an extension of degree 256.
This is a way of checking nested radical identities.
}{
\spadpaste{(s2, s5, s10) := (sqrt(2)$Ran, sqrt(5)$Ran, sqrt(10)$Ran)
\free{Ran}\bound{s2}\bound{s5}\bound{s10}}
}
\xtc{
}{
\spadpaste{eq1:=sqrt(s10+3)*sqrt(s5+2) - sqrt(s10-3)*sqrt(s5-2) =
sqrt(10*s2+10) \free{s2}\free{s5}\free{s10}\bound{eq1}}
}
```

```
\xtc{
}{
\spadpaste{eq1::Boolean \free{eq1}}
}
\xtc{
}{
\spadpaste{eq2:=sqrt(s5+2)*sqrt(s2+1) - sqrt(s5-2)*sqrt(s2-1) =
sqrt(2*s10+2)\free{s2}\free{s5}\free{s10}\bound{eq2}}
}
\xtc{
}{
\spadpaste{eq2::Boolean \free{eq2}}
}
\xtc{
Some more examples from J. M. Arnaudies
}{
\spadpaste{s3 := sqrt(3)$Ran \free{Ran}\bound{s3}}
}
\xtc{
}{
\spadpaste{s7:= sqrt(7)$Ran \free{Ran}\bound{s7}}
}
\xtc{
}{
\spadpaste{e1 := sqrt(2*s7-3*s3,3) \free{s7} \free{s3} \bound{e1}}
}
\xtc{
}{
\spadpaste{e2 := sqrt(2*s7+3*s3,3) \free{s7} \free{s3} \bound{e2}}
}
\xtc{
This should be null
}{
\spadpaste{e2-e1-s3 \free{e2} \free{e1} \free{s3}}
}
\xtc{
A quartic polynomial
}{
\spadpaste{pol : UP(x,Ran) := x**4+(7/3)*x**2+30*x-(100/3)
\free{Ran} \bound{pol}}
}
\xtc{
Add some cubic roots
}{
\spadpaste{r1 := sqrt(7633)$Ran \free{Ran}\bound{r1}}
}
\xtc{
}{
\spadpaste{alpha := sqrt(5*r1-436,3)/3 \free{r1} \bound{alpha}}
}
```

```
\xtc{
}{
\spadpaste{beta := -sqrt(5*r1+436,3)/3 \free{r1} \bound{beta}}
}
\xtc{
this should be null
}{
\spadpaste{pol.(alpha+beta-1/3) \free{pol} \free{alpha} \free{beta}}
}
\xtc{
A quintic polynomial
}{
\spadpaste{qol : UP(x,Ran) := x**5+10*x**3+20*x+22 \free{Ran}\bound{qol}}
}
\xtc{
Add some cubic roots
}{
\spadpaste{r2 := sqrt(153)$Ran \free{Ran}\bound{r2}}
}
\xtc{
}{
\spadpaste{alpha2 := sqrt(r2-11,5) \free{r2}\bound{alpha2}}
}
\xtc{
}{
\spadpaste{beta2 := -sqrt(r2+11,5) \free{r2}\bound{beta2}}
}
\xtc{
this should be null
}{
\spadpaste{qol(alpha2+beta2) \free{qol}\free{alpha2}\free{beta2}}
}
\xtc{
Finally, some examples from the book Computer Algebra by
Davenport, Siret and Tournier (page 77).
The last one is due to Ramanujan.
}{
\spadpaste{dst1:=sqrt(9+4*s2)=1+2*s2 \free{s2}\bound{dst1}}
}
\xtc{
}{
\spadpaste{dst1::Boolean \free{dst1}}
}
\xtc{
}{
\spadpaste{s6:Ran:=sqrt 6 \free{Ran}\bound{s6}}
}
\xtc{
}{
\spadpaste{dst2:=sqrt(5+2*s6)+sqrt(5-2*s6) = 2*s3
```

```
\free{s6}\free{s3}\bound{dst2}}
}
\xtc{
}{
\spadpaste{dst2::Boolean \free{dst2}}
}
\xtc{
}{
\spadpaste{s29:Ran:=sqrt 29 \free{Ran}\bound{s29}}
}
\xtc{
}{
\spadpaste{dst4:=sqrt(16-2*s29+2*sqrt(55-10*s29)) =
sqrt(22+2*s5)-sqrt(11+2*s29)+s5 \free{s29}\free{s5}\bound{dst4}}
}
\xtc{
}{
\spadpaste{dst4::Boolean \free{dst4}}
}
\xtc{
}{
\spadpaste{dst6:=sqrt((112+70*s2)+(46+34*s2)*s5) =
(5+4*s2)+(3+s2)*s5 \free{s2}\free{s5}\bound{dst6}}
}
\xtc{
}{
\spadpaste{dst6::Boolean \free{dst6}}
}
\xtc{
}{
\spadpaste{f3:Ran:=sqrt(3,5) \free{Ran}\bound{f3}}
}
\xtc{
}{
\spadpaste{f25:Ran:=sqrt(1/25,5) \free{Ran}\bound{f25}}
}
\xtc{
}{
\spadpaste{f32:Ran:=sqrt(32/5,5) \free{Ran}\bound{f32}}
}
\xtc{
}{
\spadpaste{f27:Ran:=sqrt(27/5,5) \free{Ran}\bound{f27}}
}
\xtc{
}{
\spadpaste{dst5:=sqrt((f32-f27,3)) = f25*(1+f3-f3**2)
\free{f32}\free{f27}\free{f25}\free{f3}\bound{dst5}}
}
\xtc{
```

```
}{
\spadpaste{dst5::Boolean \free{dst5}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{RealClosureXmpPagePatch1}
\begin{paste}{RealClosureXmpPageFull1}{RealClosureXmpPageEmpty1}
\pastebutton{RealClosureXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{Ran := RECLOS(FRAC INT)\bound{Ran }}
\indentrel{3}\begin{verbatim}
   (1)  RealClosure Fraction Integer
                                          Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty1}
\begin{paste}{RealClosureXmpPageEmpty1}{RealClosureXmpPagePatch1}
\pastebutton{RealClosureXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{Ran := RECLOS(FRAC INT)\bound{Ran }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch2}
\begin{paste}{RealClosureXmpPageFull2}{RealClosureXmpPageEmpty2}
\pastebutton{RealClosureXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{fourSquares(a:Ran,b:Ran,c:Ran,d:Ran):Ran == sqrt(a)+sqrt(b) - sqrt(c)-sqrt(d)\free{Ran }
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty2}
\begin{paste}{RealClosureXmpPageEmpty2}{RealClosureXmpPagePatch2}
\pastebutton{RealClosureXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{fourSquares(a:Ran,b:Ran,c:Ran,d:Ran):Ran == sqrt(a)+sqrt(b) - sqrt(c)-sqrt(d)\free{Ran }
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch3}
\begin{paste}{RealClosureXmpPageFull3}{RealClosureXmpPageEmpty3}
\pastebutton{RealClosureXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{squareDiff1 := fourSquares(73,548,60,586)\free{fs }\bound{sd1 }}
\indentrel{3}\begin{verbatim}

   (3)  - \586  - \60  + \548  + \73
                      Type: RealClosure Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{RealClosureXmpPageEmpty3}
\begin{paste}{RealClosureXmpPageEmpty3}{RealClosureXmpPagePatch3}
\pastebutton{RealClosureXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{squareDiff1 := fourSquares(73,548,60,586)\free{fs }\bound{sd1 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch4}
\begin{paste}{RealClosureXmpPageFull4}{RealClosureXmpPageEmpty4}
\pastebutton{RealClosureXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{recip(squareDiff1)\free{sd1 }}
\indentrel{3}\begin{verbatim}
   (4)

         (54602\548  + 149602\73 )\60
       +

         49502\73 \548  + 9900895
    *

       \586
   +

     (154702\73 \548  + 30941947)\60  + 10238421\548
   +

     28051871\73
          Type: Union(RealClosure Fraction Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty4}
\begin{paste}{RealClosureXmpPageEmpty4}{RealClosureXmpPagePatch4}
\pastebutton{RealClosureXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{recip(squareDiff1)\free{sd1 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch5}
\begin{paste}{RealClosureXmpPageFull5}{RealClosureXmpPageEmpty5}
\pastebutton{RealClosureXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{sign(squareDiff1)\free{sd1 }}
\indentrel{3}\begin{verbatim}
   (5)  1
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty5}
\begin{paste}{RealClosureXmpPageEmpty5}{RealClosureXmpPagePatch5}
\pastebutton{RealClosureXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{sign(squareDiff1)\free{sd1 }}
```

```
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch6}
\begin{paste}{RealClosureXmpPageFull6}{RealClosureXmpPageEmpty6}
\pastebutton{RealClosureXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{squareDiff2 := fourSquares(165,778,86,990)\free{fs }\bound{sd2 }}
\indentrel{3}\begin{verbatim}

   (6)  - \990  - \86  + \778  + \165
                     Type: RealClosure Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty6}
\begin{paste}{RealClosureXmpPageEmpty6}{RealClosureXmpPagePatch6}
\pastebutton{RealClosureXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{squareDiff2 := fourSquares(165,778,86,990)\free{fs }\bound{sd2 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch7}
\begin{paste}{RealClosureXmpPageFull7}{RealClosureXmpPageEmpty7}
\pastebutton{RealClosureXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{recip(squareDiff2)\free{sd2 }}
\indentrel{3}\begin{verbatim}
   (7)

         (556778\778  + 1209010\165 )\86
       +

         401966\165 \778  + 144019431
    *

       \990
   +

     (1363822\165 \778  + 488640503)\86
   +

     162460913\778  + 352774119\165
          Type: Union(RealClosure Fraction Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty7}
\begin{paste}{RealClosureXmpPageEmpty7}{RealClosureXmpPagePatch7}
\pastebutton{RealClosureXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{recip(squareDiff2)\free{sd2 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch8}
```

```
\begin{paste}{RealClosureXmpPageFull8}{RealClosureXmpPageEmpty8}
\pastebutton{RealClosureXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{sign(squareDiff2)\free{sd2 }}
\indentrel{3}\begin{verbatim}
   (8)  1
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty8}
\begin{paste}{RealClosureXmpPageEmpty8}{RealClosureXmpPagePatch8}
\pastebutton{RealClosureXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{sign(squareDiff2)\free{sd2 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch9}
\begin{paste}{RealClosureXmpPageFull9}{RealClosureXmpPageEmpty9}
\pastebutton{RealClosureXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{squareDiff3 := fourSquares(217,708,226,692)\free{fs }\bound{sd3 }}
\indentrel{3}\begin{verbatim}

   (9)  - \692  - \226  + \708  + \217
                     Type: RealClosure Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty9}
\begin{paste}{RealClosureXmpPageEmpty9}{RealClosureXmpPagePatch9}
\pastebutton{RealClosureXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{squareDiff3 := fourSquares(217,708,226,692)\free{fs }\bound{sd3 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch10}
\begin{paste}{RealClosureXmpPageFull10}{RealClosureXmpPageEmpty10}
\pastebutton{RealClosureXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{recip(squareDiff3)\free{sd3 }}
\indentrel{3}\begin{verbatim}
   (10)

         (- 34102\708  - 61598\217 )\226
       +

         - 34802\217 \708  - 13641141
    *

       \692
   +

     (- 60898\217 \708  - 23869841)\226
   +
```

```
     - 13486123\708  - 24359809\217
          Type: Union(RealClosure Fraction Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty10}
\begin{paste}{RealClosureXmpPageEmpty10}{RealClosureXmpPagePatch10}
\pastebutton{RealClosureXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{recip(squareDiff3)\free{sd3 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch11}
\begin{paste}{RealClosureXmpPageFull11}{RealClosureXmpPageEmpty11}
\pastebutton{RealClosureXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{sign(squareDiff3)\free{sd3 }}
\indentrel{3}\begin{verbatim}
   (11)  - 1
                                          Type: Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty11}
\begin{paste}{RealClosureXmpPageEmpty11}{RealClosureXmpPagePatch11}
\pastebutton{RealClosureXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{sign(squareDiff3)\free{sd3 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch12}
\begin{paste}{RealClosureXmpPageFull12}{RealClosureXmpPageEmpty12}
\pastebutton{RealClosureXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{squareDiff4 := fourSquares(155,836,162,820)\free{fs }\bound{sd4 }}
\indentrel{3}\begin{verbatim}

   (12)  - \820  - \162  + \836  + \155
                     Type: RealClosure Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty12}
\begin{paste}{RealClosureXmpPageEmpty12}{RealClosureXmpPagePatch12}
\pastebutton{RealClosureXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{squareDiff4 := fourSquares(155,836,162,820)\free{fs }\bound{sd4 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch13}
\begin{paste}{RealClosureXmpPageFull13}{RealClosureXmpPageEmpty13}
\pastebutton{RealClosureXmpPageFull13}{\hidepaste}
\tab{5}\spadcommand{recip(squareDiff4)\free{sd4 }}
\indentrel{3}\begin{verbatim}
```

```
   (13)

         (- 37078\836  - 86110\155 )\162
       +

         - 37906\155 \836  - 13645107
    *

       \820
   +

     (- 85282\155 \836  - 30699151)\162
   +

     - 13513901\836  - 31384703\155
          Type: Union(RealClosure Fraction Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty13}
\begin{paste}{RealClosureXmpPageEmpty13}{RealClosureXmpPagePatch13}
\pastebutton{RealClosureXmpPageEmpty13}{\showpaste}
\tab{5}\spadcommand{recip(squareDiff4)\free{sd4 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch14}
\begin{paste}{RealClosureXmpPageFull14}{RealClosureXmpPageEmpty14}
\pastebutton{RealClosureXmpPageFull14}{\hidepaste}
\tab{5}\spadcommand{sign(squareDiff4)\free{sd4 }}
\indentrel{3}\begin{verbatim}
   (14)  - 1
                                          Type: Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty14}
\begin{paste}{RealClosureXmpPageEmpty14}{RealClosureXmpPagePatch14}
\pastebutton{RealClosureXmpPageEmpty14}{\showpaste}
\tab{5}\spadcommand{sign(squareDiff4)\free{sd4 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch15}
\begin{paste}{RealClosureXmpPageFull15}{RealClosureXmpPageEmpty15}
\pastebutton{RealClosureXmpPageFull15}{\hidepaste}
\tab{5}\spadcommand{squareDiff5 := fourSquares(591,772,552,818)\free{fs }\bound{sd5 }}
\indentrel{3}\begin{verbatim}

   (15)  - \818  - \552  + \772  + \591
                     Type: RealClosure Fraction Integer
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty15}
\begin{paste}{RealClosureXmpPageEmpty15}{RealClosureXmpPagePatch15}
\pastebutton{RealClosureXmpPageEmpty15}{\showpaste}
\tab{5}\spadcommand{squareDiff5 := fourSquares(591,772,552,818)\free{fs }\bound{sd5 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch16}
\begin{paste}{RealClosureXmpPageFull16}{RealClosureXmpPageEmpty16}
\pastebutton{RealClosureXmpPageFull16}{\hidepaste}
\tab{5}\spadcommand{recip(squareDiff5)\free{sd5 }}
\indentrel{3}\begin{verbatim}
   (16)

         (70922\772  + 81058\591 )\552
       +

         68542\591 \772  + 46297673
    *

       \818
   +

     (83438\591 \772  + 56359389)\552  + 47657051\772
   +

     54468081\591
          Type: Union(RealClosure Fraction Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty16}
\begin{paste}{RealClosureXmpPageEmpty16}{RealClosureXmpPagePatch16}
\pastebutton{RealClosureXmpPageEmpty16}{\showpaste}
\tab{5}\spadcommand{recip(squareDiff5)\free{sd5 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch17}
\begin{paste}{RealClosureXmpPageFull17}{RealClosureXmpPageEmpty17}
\pastebutton{RealClosureXmpPageFull17}{\hidepaste}
\tab{5}\spadcommand{sign(squareDiff5)\free{sd5 }}
\indentrel{3}\begin{verbatim}
   (17)  1
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty17}
\begin{paste}{RealClosureXmpPageEmpty17}{RealClosureXmpPagePatch17}
```

```
\pastebutton{RealClosureXmpPageEmpty17}{\showpaste}
\tab{5}\spadcommand{sign(squareDiff5)\free{sd5 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch18}
\begin{paste}{RealClosureXmpPageFull18}{RealClosureXmpPageEmpty18}
\pastebutton{RealClosureXmpPageFull18}{\hidepaste}
\tab{5}\spadcommand{squareDiff6 := fourSquares(434,1053,412,1088)\free{fs }\bound{sd6 }}
\indentrel{3}\begin{verbatim}

   (18)  - \1088  - \412  + \1053  + \434
                     Type: RealClosure Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty18}
\begin{paste}{RealClosureXmpPageEmpty18}{RealClosureXmpPagePatch18}
\pastebutton{RealClosureXmpPageEmpty18}{\showpaste}
\tab{5}\spadcommand{squareDiff6 := fourSquares(434,1053,412,1088)\free{fs }\bound{sd6 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch19}
\begin{paste}{RealClosureXmpPageFull19}{RealClosureXmpPageEmpty19}
\pastebutton{RealClosureXmpPageFull19}{\hidepaste}
\tab{5}\spadcommand{recip(squareDiff6)\free{sd6 }}
\indentrel{3}\begin{verbatim}
   (19)

         (115442\1053  + 179818\434 )\412
       +

         112478\434 \1053  + 76037291
    *

       \1088
   +

     (182782\434 \1053  + 123564147)\412
   +

     77290639\1053  + 120391609\434
          Type: Union(RealClosure Fraction Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty19}
\begin{paste}{RealClosureXmpPageEmpty19}{RealClosureXmpPagePatch19}
\pastebutton{RealClosureXmpPageEmpty19}{\showpaste}
\tab{5}\spadcommand{recip(squareDiff6)\free{sd6 }}
\end{paste}\end{patch}
```

```
\begin{patch}{RealClosureXmpPagePatch20}
\begin{paste}{RealClosureXmpPageFull20}{RealClosureXmpPageEmpty20}
\pastebutton{RealClosureXmpPageFull20}{\hidepaste}
\tab{5}\spadcommand{sign(squareDiff6)\free{sd6 }}
\indentrel{3}\begin{verbatim}
   (20)  1
                                 Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty20}
\begin{paste}{RealClosureXmpPageEmpty20}{RealClosureXmpPagePatch20}
\pastebutton{RealClosureXmpPageEmpty20}{\showpaste}
\tab{5}\spadcommand{sign(squareDiff6)\free{sd6 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch21}
\begin{paste}{RealClosureXmpPageFull21}{RealClosureXmpPageEmpty21}
\pastebutton{RealClosureXmpPageFull21}{\hidepaste}
\tab{5}\spadcommand{squareDiff7 := fourSquares(514,1049,446,1152)\free{fs }\bound{sd7 }}
\indentrel{3}\begin{verbatim}

   (21)  - \1152  - \446  + \1049  + \514
                     Type: RealClosure Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty21}
\begin{paste}{RealClosureXmpPageEmpty21}{RealClosureXmpPagePatch21}
\pastebutton{RealClosureXmpPageEmpty21}{\showpaste}
\tab{5}\spadcommand{squareDiff7 := fourSquares(514,1049,446,1152)\free{fs }\bound{sd7 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch22}
\begin{paste}{RealClosureXmpPageFull22}{RealClosureXmpPageEmpty22}
\pastebutton{RealClosureXmpPageFull22}{\hidepaste}
\tab{5}\spadcommand{recip(squareDiff7)\free{sd7 }}
\indentrel{3}\begin{verbatim}
   (22)

         (349522\1049  + 499322\514 )\446
       +

         325582\514 \1049  + 239072537
    *

       \1152
   +
```

```
     (523262\514 \1049  + 384227549)\446
   +

     250534873\1049  + 357910443\514
          Type: Union(RealClosure Fraction Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty22}
\begin{paste}{RealClosureXmpPageEmpty22}{RealClosureXmpPagePatch22}
\pastebutton{RealClosureXmpPageEmpty22}{\showpaste}
\tab{5}\spadcommand{recip(squareDiff7)\free{sd7 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch23}
\begin{paste}{RealClosureXmpPageFull23}{RealClosureXmpPageEmpty23}
\pastebutton{RealClosureXmpPageFull23}{\hidepaste}
\tab{5}\spadcommand{sign(squareDiff7)\free{sd7 }}
\indentrel{3}\begin{verbatim}
   (23)  1
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty23}
\begin{paste}{RealClosureXmpPageEmpty23}{RealClosureXmpPagePatch23}
\pastebutton{RealClosureXmpPageEmpty23}{\showpaste}
\tab{5}\spadcommand{sign(squareDiff7)\free{sd7 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch24}
\begin{paste}{RealClosureXmpPageFull24}{RealClosureXmpPageEmpty24}
\pastebutton{RealClosureXmpPageFull24}{\hidepaste}
\tab{5}\spadcommand{squareDiff8 := fourSquares(190,1751,208,1698)\free{fs }\bound{sd8 }}
\indentrel{3}\begin{verbatim}

   (24)  - \1698  - \208  + \1751  + \190
                     Type: RealClosure Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty24}
\begin{paste}{RealClosureXmpPageEmpty24}{RealClosureXmpPagePatch24}
\pastebutton{RealClosureXmpPageEmpty24}{\showpaste}
\tab{5}\spadcommand{squareDiff8 := fourSquares(190,1751,208,1698)\free{fs }\bound{sd8 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch25}
\begin{paste}{RealClosureXmpPageFull25}{RealClosureXmpPageEmpty25}
\pastebutton{RealClosureXmpPageFull25}{\hidepaste}
```

```
\tab{5}\spadcommand{recip(squareDiff8)\free{sd8 }}
\indentrel{3}\begin{verbatim}
   (25)

         (- 214702\1751  - 651782\190 )\208
       +

         - 224642\190 \1751  - 129571901
    *

       \1698
   +

     (- 641842\190 \1751  - 370209881)\208
   +

     - 127595865\1751  - 387349387\190
          Type: Union(RealClosure Fraction Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty25}
\begin{paste}{RealClosureXmpPageEmpty25}{RealClosureXmpPagePatch25}
\pastebutton{RealClosureXmpPageEmpty25}{\showpaste}
\tab{5}\spadcommand{recip(squareDiff8)\free{sd8 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch26}
\begin{paste}{RealClosureXmpPageFull26}{RealClosureXmpPageEmpty26}
\pastebutton{RealClosureXmpPageFull26}{\hidepaste}
\tab{5}\spadcommand{sign(squareDiff8)\free{sd8 }}
\indentrel{3}\begin{verbatim}
   (26)  - 1
                                          Type: Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty26}
\begin{paste}{RealClosureXmpPageEmpty26}{RealClosureXmpPagePatch26}
\pastebutton{RealClosureXmpPageEmpty26}{\showpaste}
\tab{5}\spadcommand{sign(squareDiff8)\free{sd8 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch27}
\begin{paste}{RealClosureXmpPageFull27}{RealClosureXmpPageEmpty27}
\pastebutton{RealClosureXmpPageFull27}{\hidepaste}
\tab{5}\spadcommand{relativeApprox(squareDiff8,10**(-3))::Float\free{sd8 }}
\indentrel{3}\begin{verbatim}
   (27)  - 0.2340527771 5937700123 E -10
                                            Type: Float
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty27}
\begin{paste}{RealClosureXmpPageEmpty27}{RealClosureXmpPagePatch27}
\pastebutton{RealClosureXmpPageEmpty27}{\showpaste}
\tab{5}\spadcommand{relativeApprox(squareDiff8,10**(-3))::Float\free{sd8 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch28}
\begin{paste}{RealClosureXmpPageFull28}{RealClosureXmpPageEmpty28}
\pastebutton{RealClosureXmpPageFull28}{\hidepaste}
\tab{5}\spadcommand{l := allRootsOf((x**2-2)**2-2)$Ran\free{Ran }\bound{l }}
\indentrel{3}\begin{verbatim}
   (28)  [%R33,%R34,%R35,%R36]
                Type: List RealClosure Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty28}
\begin{paste}{RealClosureXmpPageEmpty28}{RealClosureXmpPagePatch28}
\pastebutton{RealClosureXmpPageEmpty28}{\showpaste}
\tab{5}\spadcommand{l := allRootsOf((x**2-2)**2-2)$Ran\free{Ran }\bound{l }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch29}
\begin{paste}{RealClosureXmpPageFull29}{RealClosureXmpPageEmpty29}
\pastebutton{RealClosureXmpPageFull29}{\hidepaste}
\tab{5}\spadcommand{removeDuplicates map(mainDefiningPolynomial,l)\free{l }}
\indentrel{3}\begin{verbatim}
          4     2
   (29)  [?  - 4?  + 2]
Type: List Union(SparseUnivariatePolynomial RealClosure Fraction Integer,"failed")
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty29}
\begin{paste}{RealClosureXmpPageEmpty29}{RealClosureXmpPagePatch29}
\pastebutton{RealClosureXmpPageEmpty29}{\showpaste}
\tab{5}\spadcommand{removeDuplicates map(mainDefiningPolynomial,l)\free{l }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch30}
\begin{paste}{RealClosureXmpPageFull30}{RealClosureXmpPageEmpty30}
\pastebutton{RealClosureXmpPageFull30}{\hidepaste}
\tab{5}\spadcommand{map(mainCharacterization,l)\free{l }}
\indentrel{3}\begin{verbatim}
   (30)  [[- 2,- 1[,[- 1,0[,[0,1[,[1,2[]
Type: List Union(RightOpenIntervalRootCharacterization(RealClosure Fraction Integer,SparseUnivar
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty30}
\begin{paste}{RealClosureXmpPageEmpty30}{RealClosureXmpPagePatch30}
\pastebutton{RealClosureXmpPageEmpty30}{\showpaste}
\tab{5}\spadcommand{map(mainCharacterization,l)\free{l }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch31}
\begin{paste}{RealClosureXmpPageFull31}{RealClosureXmpPageEmpty31}
\pastebutton{RealClosureXmpPageFull31}{\hidepaste}
\tab{5}\spadcommand{[reduce(+,l),reduce(*,l)-2]\free{l }}
\indentrel{3}\begin{verbatim}
   (31)  [0,0]
                Type: List RealClosure Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty31}
\begin{paste}{RealClosureXmpPageEmpty31}{RealClosureXmpPagePatch31}
\pastebutton{RealClosureXmpPageEmpty31}{\showpaste}
\tab{5}\spadcommand{[reduce(+,l),reduce(*,l)-2]\free{l }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch32}
\begin{paste}{RealClosureXmpPageFull32}{RealClosureXmpPageEmpty32}
\pastebutton{RealClosureXmpPageFull32}{\hidepaste}
\tab{5}\spadcommand{(s2, s5, s10) := (sqrt(2)$Ran, sqrt(5)$Ran, sqrt(10)$Ran)\free{Ran }\bound{s2 }\bound{s5
\indentrel{3}\begin{verbatim}

   (32)  \10
                     Type: RealClosure Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty32}
\begin{paste}{RealClosureXmpPageEmpty32}{RealClosureXmpPagePatch32}
\pastebutton{RealClosureXmpPageEmpty32}{\showpaste}
\tab{5}\spadcommand{(s2, s5, s10) := (sqrt(2)$Ran, sqrt(5)$Ran, sqrt(10)$Ran)\free{Ran }\bound{s2 }\bound{s5
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch33}
\begin{paste}{RealClosureXmpPageFull33}{RealClosureXmpPageEmpty33}
\pastebutton{RealClosureXmpPageFull33}{\hidepaste}
\tab{5}\spadcommand{eq1:=sqrt(s10+3)*sqrt(s5+2) - sqrt(s10-3)*sqrt(s5-2) = sqrt(10*s2+10)\free{s2 }\free{s5
\indentrel{3}\begin{verbatim}
   (33)


   - \\10  - 3 \\5  - 2  + \\10  + 3 \\5  + 2  =
```

```
     \10\2  + 10
            Type: Equation RealClosure Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty33}
\begin{paste}{RealClosureXmpPageEmpty33}{RealClosureXmpPagePatch33}
\pastebutton{RealClosureXmpPageEmpty33}{\showpaste}
\tab{5}\spadcommand{eq1:=sqrt(s10+3)*sqrt(s5+2) - sqrt(s10-3)*sqrt(s5-2) = sqrt(10*s2+10)\free{s
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch34}
\begin{paste}{RealClosureXmpPageFull34}{RealClosureXmpPageEmpty34}
\pastebutton{RealClosureXmpPageFull34}{\hidepaste}
\tab{5}\spadcommand{eq1::Boolean\free{eq1 }}
\indentrel{3}\begin{verbatim}
   (34)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty34}
\begin{paste}{RealClosureXmpPageEmpty34}{RealClosureXmpPagePatch34}
\pastebutton{RealClosureXmpPageEmpty34}{\showpaste}
\tab{5}\spadcommand{eq1::Boolean\free{eq1 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch35}
\begin{paste}{RealClosureXmpPageFull35}{RealClosureXmpPageEmpty35}
\pastebutton{RealClosureXmpPageFull35}{\hidepaste}
\tab{5}\spadcommand{eq2:=sqrt(s5+2)*sqrt(s2+1) - sqrt(s5-2)*sqrt(s2-1) = sqrt(2*s10+2)\free{s2 }
\indentrel{3}\begin{verbatim}
   (35)


   - \\5  - 2 \\2  - 1  + \\5  + 2 \\2  + 1  =


     \2\10  + 2
            Type: Equation RealClosure Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty35}
\begin{paste}{RealClosureXmpPageEmpty35}{RealClosureXmpPagePatch35}
\pastebutton{RealClosureXmpPageEmpty35}{\showpaste}
\tab{5}\spadcommand{eq2:=sqrt(s5+2)*sqrt(s2+1) - sqrt(s5-2)*sqrt(s2-1) = sqrt(2*s10+2)\free{s2 }
\end{paste}\end{patch}
```

```
\begin{patch}{RealClosureXmpPagePatch36}
\begin{paste}{RealClosureXmpPageFull36}{RealClosureXmpPageEmpty36}
\pastebutton{RealClosureXmpPageFull36}{\hidepaste}
\tab{5}\spadcommand{eq2::Boolean\free{eq2 }}
\indentrel{3}\begin{verbatim}
   (36)  true
                                            Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty36}
\begin{paste}{RealClosureXmpPageEmpty36}{RealClosureXmpPagePatch36}
\pastebutton{RealClosureXmpPageEmpty36}{\showpaste}
\tab{5}\spadcommand{eq2::Boolean\free{eq2 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch37}
\begin{paste}{RealClosureXmpPageFull37}{RealClosureXmpPageEmpty37}
\pastebutton{RealClosureXmpPageFull37}{\hidepaste}
\tab{5}\spadcommand{s3 := sqrt(3)$Ran\free{Ran }\bound{s3 }}
\indentrel{3}\begin{verbatim}

   (37)  \3
                      Type: RealClosure Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty37}
\begin{paste}{RealClosureXmpPageEmpty37}{RealClosureXmpPagePatch37}
\pastebutton{RealClosureXmpPageEmpty37}{\showpaste}
\tab{5}\spadcommand{s3 := sqrt(3)$Ran\free{Ran }\bound{s3 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch38}
\begin{paste}{RealClosureXmpPageFull38}{RealClosureXmpPageEmpty38}
\pastebutton{RealClosureXmpPageFull38}{\hidepaste}
\tab{5}\spadcommand{s7:= sqrt(7)$Ran\free{Ran }\bound{s7 }}
\indentrel{3}\begin{verbatim}

   (38)  \7
                      Type: RealClosure Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty38}
\begin{paste}{RealClosureXmpPageEmpty38}{RealClosureXmpPagePatch38}
\pastebutton{RealClosureXmpPageEmpty38}{\showpaste}
\tab{5}\spadcommand{s7:= sqrt(7)$Ran\free{Ran }\bound{s7 }}
\end{paste}\end{patch}
```

```
\begin{patch}{RealClosureXmpPagePatch39}
\begin{paste}{RealClosureXmpPageFull39}{RealClosureXmpPageEmpty39}
\pastebutton{RealClosureXmpPageFull39}{\hidepaste}
\tab{5}\spadcommand{e1 := sqrt(2*s7-3*s3,3)\free{s7 }\free{s3 }\bound{e1 }}
\indentrel{3}\begin{verbatim}

         3
   (39)  \2\7  - 3\3
                     Type: RealClosure Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty39}
\begin{paste}{RealClosureXmpPageEmpty39}{RealClosureXmpPagePatch39}
\pastebutton{RealClosureXmpPageEmpty39}{\showpaste}
\tab{5}\spadcommand{e1 := sqrt(2*s7-3*s3,3)\free{s7 }\free{s3 }\bound{e1 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch40}
\begin{paste}{RealClosureXmpPageFull40}{RealClosureXmpPageEmpty40}
\pastebutton{RealClosureXmpPageFull40}{\hidepaste}
\tab{5}\spadcommand{e2 := sqrt(2*s7+3*s3,3)\free{s7 }\free{s3 }\bound{e2 }}
\indentrel{3}\begin{verbatim}

         3
   (40)  \2\7  + 3\3
                     Type: RealClosure Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty40}
\begin{paste}{RealClosureXmpPageEmpty40}{RealClosureXmpPagePatch40}
\pastebutton{RealClosureXmpPageEmpty40}{\showpaste}
\tab{5}\spadcommand{e2 := sqrt(2*s7+3*s3,3)\free{s7 }\free{s3 }\bound{e2 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch41}
\begin{paste}{RealClosureXmpPageFull41}{RealClosureXmpPageEmpty41}
\pastebutton{RealClosureXmpPageFull41}{\hidepaste}
\tab{5}\spadcommand{e2-e1-s3\free{e2 }\free{e1 }\free{s3 }}
\indentrel{3}\begin{verbatim}
   (41)  0
                     Type: RealClosure Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty41}
\begin{paste}{RealClosureXmpPageEmpty41}{RealClosureXmpPagePatch41}
\pastebutton{RealClosureXmpPageEmpty41}{\showpaste}
```

```
\tab{5}\spadcommand{e2-e1-s3\free{e2 }\free{e1 }\free{s3 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch42}
\begin{paste}{RealClosureXmpPageFull42}{RealClosureXmpPageEmpty42}
\pastebutton{RealClosureXmpPageFull42}{\hidepaste}
\tab{5}\spadcommand{pol : UP(x,Ran) := x**4+(7/3)*x**2+30*x-(100/3)\free{Ran }\bound{pol }}
\indentrel{3}\begin{verbatim}
          4   7  2         100
   (42)  x  +  x  + 30x -
              3             3
Type: UnivariatePolynomial(x,RealClosure Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty42}
\begin{paste}{RealClosureXmpPageEmpty42}{RealClosureXmpPagePatch42}
\pastebutton{RealClosureXmpPageEmpty42}{\showpaste}
\tab{5}\spadcommand{pol : UP(x,Ran) := x**4+(7/3)*x**2+30*x-(100/3)\free{Ran }\bound{pol }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch43}
\begin{paste}{RealClosureXmpPageFull43}{RealClosureXmpPageEmpty43}
\pastebutton{RealClosureXmpPageFull43}{\hidepaste}
\tab{5}\spadcommand{r1 := sqrt(7633)$Ran\free{Ran }\bound{r1 }}
\indentrel{3}\begin{verbatim}

   (43)  \7633
                    Type: RealClosure Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty43}
\begin{paste}{RealClosureXmpPageEmpty43}{RealClosureXmpPagePatch43}
\pastebutton{RealClosureXmpPageEmpty43}{\showpaste}
\tab{5}\spadcommand{r1 := sqrt(7633)$Ran\free{Ran }\bound{r1 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch44}
\begin{paste}{RealClosureXmpPageFull44}{RealClosureXmpPageEmpty44}
\pastebutton{RealClosureXmpPageFull44}{\hidepaste}
\tab{5}\spadcommand{alpha := sqrt(5*r1-436,3)/3\free{r1 }\bound{alpha }}
\indentrel{3}\begin{verbatim}

         1 3
   (44)   \5\7633  - 436
         3
                    Type: RealClosure Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{RealClosureXmpPageEmpty44}
\begin{paste}{RealClosureXmpPageEmpty44}{RealClosureXmpPagePatch44}
\pastebutton{RealClosureXmpPageEmpty44}{\showpaste}
\tab{5}\spadcommand{alpha := sqrt(5*r1-436,3)/3\free{r1 }\bound{alpha }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch45}
\begin{paste}{RealClosureXmpPageFull45}{RealClosureXmpPageEmpty45}
\pastebutton{RealClosureXmpPageFull45}{\hidepaste}
\tab{5}\spadcommand{beta := -sqrt(5*r1+436,3)/3\free{r1 }\bound{beta }}
\indentrel{3}\begin{verbatim}

          1 3
   (45)  -  \5\7633  + 436
          3
                    Type: RealClosure Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty45}
\begin{paste}{RealClosureXmpPageEmpty45}{RealClosureXmpPagePatch45}
\pastebutton{RealClosureXmpPageEmpty45}{\showpaste}
\tab{5}\spadcommand{beta := -sqrt(5*r1+436,3)/3\free{r1 }\bound{beta }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch46}
\begin{paste}{RealClosureXmpPageFull46}{RealClosureXmpPageEmpty46}
\pastebutton{RealClosureXmpPageFull46}{\hidepaste}
\tab{5}\spadcommand{pol.(alpha+beta-1/3)\free{pol }\free{alpha }\free{beta }}
\indentrel{3}\begin{verbatim}
   (46)  0
                    Type: RealClosure Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty46}
\begin{paste}{RealClosureXmpPageEmpty46}{RealClosureXmpPagePatch46}
\pastebutton{RealClosureXmpPageEmpty46}{\showpaste}
\tab{5}\spadcommand{pol.(alpha+beta-1/3)\free{pol }\free{alpha }\free{beta }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch47}
\begin{paste}{RealClosureXmpPageFull47}{RealClosureXmpPageEmpty47}
\pastebutton{RealClosureXmpPageFull47}{\hidepaste}
\tab{5}\spadcommand{qol : UP(x,Ran) := x**5+10*x**3+20*x+22\free{Ran }\bound{qol }}
\indentrel{3}\begin{verbatim}
         5      3
   (47)  x  + 10x  + 20x + 22
Type: UnivariatePolynomial(x,RealClosure Fraction Integer)
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty47}
\begin{paste}{RealClosureXmpPageEmpty47}{RealClosureXmpPagePatch47}
\pastebutton{RealClosureXmpPageEmpty47}{\showpaste}
\tab{5}\spadcommand{qol : UP(x,Ran) := x**5+10*x**3+20*x+22\free{Ran }\bound{qol }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch48}
\begin{paste}{RealClosureXmpPageFull48}{RealClosureXmpPageEmpty48}
\pastebutton{RealClosureXmpPageFull48}{\hidepaste}
\tab{5}\spadcommand{r2 := sqrt(153)$Ran\free{Ran }\bound{r2 }}
\indentrel{3}\begin{verbatim}

   (48)  \153
                    Type: RealClosure Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty48}
\begin{paste}{RealClosureXmpPageEmpty48}{RealClosureXmpPagePatch48}
\pastebutton{RealClosureXmpPageEmpty48}{\showpaste}
\tab{5}\spadcommand{r2 := sqrt(153)$Ran\free{Ran }\bound{r2 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch49}
\begin{paste}{RealClosureXmpPageFull49}{RealClosureXmpPageEmpty49}
\pastebutton{RealClosureXmpPageFull49}{\hidepaste}
\tab{5}\spadcommand{alpha2 := sqrt(r2-11,5)\free{r2 }\bound{alpha2 }}
\indentrel{3}\begin{verbatim}

         5
   (49)  \\153  - 11
                    Type: RealClosure Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty49}
\begin{paste}{RealClosureXmpPageEmpty49}{RealClosureXmpPagePatch49}
\pastebutton{RealClosureXmpPageEmpty49}{\showpaste}
\tab{5}\spadcommand{alpha2 := sqrt(r2-11,5)\free{r2 }\bound{alpha2 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch50}
\begin{paste}{RealClosureXmpPageFull50}{RealClosureXmpPageEmpty50}
\pastebutton{RealClosureXmpPageFull50}{\hidepaste}
\tab{5}\spadcommand{beta2 := -sqrt(r2+11,5)\free{r2 }\bound{beta2 }}
\indentrel{3}\begin{verbatim}
```

```
          5
   (50)  - \\153  + 11
                      Type: RealClosure Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty50}
\begin{paste}{RealClosureXmpPageEmpty50}{RealClosureXmpPagePatch50}
\pastebutton{RealClosureXmpPageEmpty50}{\showpaste}
\tab{5}\spadcommand{beta2 := -sqrt(r2+11,5)\free{r2 }\bound{beta2 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch51}
\begin{paste}{RealClosureXmpPageFull51}{RealClosureXmpPageEmpty51}
\pastebutton{RealClosureXmpPageFull51}{\hidepaste}
\tab{5}\spadcommand{qol(alpha2+beta2)\free{qol }\free{alpha2 }\free{beta2 }}
\indentrel{3}\begin{verbatim}
   (51)  0
                      Type: RealClosure Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty51}
\begin{paste}{RealClosureXmpPageEmpty51}{RealClosureXmpPagePatch51}
\pastebutton{RealClosureXmpPageEmpty51}{\showpaste}
\tab{5}\spadcommand{qol(alpha2+beta2)\free{qol }\free{alpha2 }\free{beta2 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch52}
\begin{paste}{RealClosureXmpPageFull52}{RealClosureXmpPageEmpty52}
\pastebutton{RealClosureXmpPageFull52}{\hidepaste}
\tab{5}\spadcommand{dst1:=sqrt(9+4*s2)=1+2*s2\free{s2 }\bound{dst1 }}
\indentrel{3}\begin{verbatim}


   (52)  \4\2  + 9 = 2\2  + 1
             Type: Equation RealClosure Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty52}
\begin{paste}{RealClosureXmpPageEmpty52}{RealClosureXmpPagePatch52}
\pastebutton{RealClosureXmpPageEmpty52}{\showpaste}
\tab{5}\spadcommand{dst1:=sqrt(9+4*s2)=1+2*s2\free{s2 }\bound{dst1 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch53}
\begin{paste}{RealClosureXmpPageFull53}{RealClosureXmpPageEmpty53}
\pastebutton{RealClosureXmpPageFull53}{\hidepaste}
\tab{5}\spadcommand{dst1::Boolean\free{dst1 }}
```

```
\indentrel{3}\begin{verbatim}
   (53)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty53}
\begin{paste}{RealClosureXmpPageEmpty53}{RealClosureXmpPagePatch53}
\pastebutton{RealClosureXmpPageEmpty53}{\showpaste}
\tab{5}\spadcommand{dst1::Boolean\free{dst1 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch54}
\begin{paste}{RealClosureXmpPageFull54}{RealClosureXmpPageEmpty54}
\pastebutton{RealClosureXmpPageFull54}{\hidepaste}
\tab{5}\spadcommand{s6:Ran:=sqrt 6\free{Ran }\bound{s6 }}
\indentrel{3}\begin{verbatim}

   (54)  \6
                     Type: RealClosure Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty54}
\begin{paste}{RealClosureXmpPageEmpty54}{RealClosureXmpPagePatch54}
\pastebutton{RealClosureXmpPageEmpty54}{\showpaste}
\tab{5}\spadcommand{s6:Ran:=sqrt 6\free{Ran }\bound{s6 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch55}
\begin{paste}{RealClosureXmpPageFull55}{RealClosureXmpPageEmpty55}
\pastebutton{RealClosureXmpPageFull55}{\hidepaste}
\tab{5}\spadcommand{dst2:=sqrt(5+2*s6)+sqrt(5-2*s6) = 2*s3\free{s6 }\free{s3 }\bound{dst2 }}
\indentrel{3}\begin{verbatim}


   (55)  \- 2\6  + 5  + \2\6  + 5 = 2\3
            Type: Equation RealClosure Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty55}
\begin{paste}{RealClosureXmpPageEmpty55}{RealClosureXmpPagePatch55}
\pastebutton{RealClosureXmpPageEmpty55}{\showpaste}
\tab{5}\spadcommand{dst2:=sqrt(5+2*s6)+sqrt(5-2*s6) = 2*s3\free{s6 }\free{s3 }\bound{dst2 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch56}
\begin{paste}{RealClosureXmpPageFull56}{RealClosureXmpPageEmpty56}
\pastebutton{RealClosureXmpPageFull56}{\hidepaste}
```

```
\tab{5}\spadcommand{dst2::Boolean\free{dst2 }}
\indentrel{3}\begin{verbatim}
   (56)  true
                                            Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty56}
\begin{paste}{RealClosureXmpPageEmpty56}{RealClosureXmpPagePatch56}
\pastebutton{RealClosureXmpPageEmpty56}{\showpaste}
\tab{5}\spadcommand{dst2::Boolean\free{dst2 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch57}
\begin{paste}{RealClosureXmpPageFull57}{RealClosureXmpPageEmpty57}
\pastebutton{RealClosureXmpPageFull57}{\hidepaste}
\tab{5}\spadcommand{s29:Ran:=sqrt 29\free{Ran }\bound{s29 }}
\indentrel{3}\begin{verbatim}

   (57)  \29
                      Type: RealClosure Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty57}
\begin{paste}{RealClosureXmpPageEmpty57}{RealClosureXmpPagePatch57}
\pastebutton{RealClosureXmpPageEmpty57}{\showpaste}
\tab{5}\spadcommand{s29:Ran:=sqrt 29\free{Ran }\bound{s29 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch58}
\begin{paste}{RealClosureXmpPageFull58}{RealClosureXmpPageEmpty58}
\pastebutton{RealClosureXmpPageFull58}{\hidepaste}
\tab{5}\spadcommand{dst4:=sqrt(16-2*s29+2*sqrt(55-10*s29)) = sqrt(22+2*s5)-sqrt(11+2*s29)+s5\fre
\indentrel{3}\begin{verbatim}
   (58)



   \2\- 10\29  + 55  - 2\29  + 16  =


     - \2\29  + 11  + \2\5  + 22  + \5
           Type: Equation RealClosure Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty58}
\begin{paste}{RealClosureXmpPageEmpty58}{RealClosureXmpPagePatch58}
\pastebutton{RealClosureXmpPageEmpty58}{\showpaste}
```

```
\tab{5}\spadcommand{dst4:=sqrt(16-2*s29+2*sqrt(55-10*s29)) = sqrt(22+2*s5)-sqrt(11+2*s29)+s5\free{s29 }\free
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch59}
\begin{paste}{RealClosureXmpPageFull59}{RealClosureXmpPageEmpty59}
\pastebutton{RealClosureXmpPageFull59}{\hidepaste}
\tab{5}\spadcommand{dst4::Boolean\free{dst4 }}
\indentrel{3}\begin{verbatim}
   (59)  true
                                            Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty59}
\begin{paste}{RealClosureXmpPageEmpty59}{RealClosureXmpPagePatch59}
\pastebutton{RealClosureXmpPageEmpty59}{\showpaste}
\tab{5}\spadcommand{dst4::Boolean\free{dst4 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch60}
\begin{paste}{RealClosureXmpPageFull60}{RealClosureXmpPageEmpty60}
\pastebutton{RealClosureXmpPageFull60}{\hidepaste}
\tab{5}\spadcommand{dst6:=sqrt((112+70*s2)+(46+34*s2)*s5) = (5+4*s2)+(3+s2)*s5\free{s2 }\free{s5 }\bound{dst
\indentrel{3}\begin{verbatim}
   (60)


   \(34\2  + 46)\5  + 70\2  + 112  =

     (\2  + 3)\5  + 4\2  + 5
            Type: Equation RealClosure Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty60}
\begin{paste}{RealClosureXmpPageEmpty60}{RealClosureXmpPagePatch60}
\pastebutton{RealClosureXmpPageEmpty60}{\showpaste}
\tab{5}\spadcommand{dst6:=sqrt((112+70*s2)+(46+34*s2)*s5) = (5+4*s2)+(3+s2)*s5\free{s2 }\free{s5 }\bound{dst
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch61}
\begin{paste}{RealClosureXmpPageFull61}{RealClosureXmpPageEmpty61}
\pastebutton{RealClosureXmpPageFull61}{\hidepaste}
\tab{5}\spadcommand{dst6::Boolean\free{dst6 }}
\indentrel{3}\begin{verbatim}
   (61)  true
                                            Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{RealClosureXmpPageEmpty61}
\begin{paste}{RealClosureXmpPageEmpty61}{RealClosureXmpPagePatch61}
\pastebutton{RealClosureXmpPageEmpty61}{\showpaste}
\tab{5}\spadcommand{dst6::Boolean\free{dst6 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch62}
\begin{paste}{RealClosureXmpPageFull62}{RealClosureXmpPageEmpty62}
\pastebutton{RealClosureXmpPageFull62}{\hidepaste}
\tab{5}\spadcommand{f3:Ran:=sqrt(3,5)\free{Ran }\bound{f3 }}
\indentrel{3}\begin{verbatim}
         5
   (62)  \3
                    Type: RealClosure Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty62}
\begin{paste}{RealClosureXmpPageEmpty62}{RealClosureXmpPagePatch62}
\pastebutton{RealClosureXmpPageEmpty62}{\showpaste}
\tab{5}\spadcommand{f3:Ran:=sqrt(3,5)\free{Ran }\bound{f3 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch63}
\begin{paste}{RealClosureXmpPageFull63}{RealClosureXmpPageEmpty63}
\pastebutton{RealClosureXmpPageFull63}{\hidepaste}
\tab{5}\spadcommand{f25:Ran:=sqrt(1/25,5)\free{Ran }\bound{f25 }}
\indentrel{3}\begin{verbatim}

           1
   (63)  5
         \25
                    Type: RealClosure Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty63}
\begin{paste}{RealClosureXmpPageEmpty63}{RealClosureXmpPagePatch63}
\pastebutton{RealClosureXmpPageEmpty63}{\showpaste}
\tab{5}\spadcommand{f25:Ran:=sqrt(1/25,5)\free{Ran }\bound{f25 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch64}
\begin{paste}{RealClosureXmpPageFull64}{RealClosureXmpPageEmpty64}
\pastebutton{RealClosureXmpPageFull64}{\hidepaste}
\tab{5}\spadcommand{f32:Ran:=sqrt(32/5,5)\free{Ran }\bound{f32 }}
\indentrel{3}\begin{verbatim}

          32
   (64)  5
```

```
        \ 5
                    Type: RealClosure Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty64}
\begin{paste}{RealClosureXmpPageEmpty64}{RealClosureXmpPagePatch64}
\pastebutton{RealClosureXmpPageEmpty64}{\showpaste}
\tab{5}\spadcommand{f32:Ran:=sqrt(32/5,5)\free{Ran }\bound{f32 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch65}
\begin{paste}{RealClosureXmpPageFull65}{RealClosureXmpPageEmpty65}
\pastebutton{RealClosureXmpPageFull65}{\hidepaste}
\tab{5}\spadcommand{f27:Ran:=sqrt(27/5,5)\free{Ran }\bound{f27 }}
\indentrel{3}\begin{verbatim}

         27
   (65)  5
        \ 5
                    Type: RealClosure Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty65}
\begin{paste}{RealClosureXmpPageEmpty65}{RealClosureXmpPagePatch65}
\pastebutton{RealClosureXmpPageEmpty65}{\showpaste}
\tab{5}\spadcommand{f27:Ran:=sqrt(27/5,5)\free{Ran }\bound{f27 }}
\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPagePatch66}
\begin{paste}{RealClosureXmpPageFull66}{RealClosureXmpPageEmpty66}
\pastebutton{RealClosureXmpPageFull66}{\hidepaste}
\tab{5}\spadcommand{dst5:=sqrt((f32-f27,3)) = f25*(1+f3-f3**2)\free{f32 }\free{f27 }\free{f25 }\free{f3 }\bo
\indentrel{3}\begin{verbatim}


            27     32       52   5        1
   (66)  3- 5  + 5  = (- \3   + \3  + 1) 5
        \  \ 5    \ 5                        \25
            Type: Equation RealClosure Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty66}
\begin{paste}{RealClosureXmpPageEmpty66}{RealClosureXmpPagePatch66}
\pastebutton{RealClosureXmpPageEmpty66}{\showpaste}
\tab{5}\spadcommand{dst5:=sqrt((f32-f27,3)) = f25*(1+f3-f3**2)\free{f32 }\free{f27 }\free{f25 }\free{f3 }\bo
\end{paste}\end{patch}
```

```
\begin{patch}{RealClosureXmpPagePatch67}
\begin{paste}{RealClosureXmpPageFull67}{RealClosureXmpPageEmpty67}
\pastebutton{RealClosureXmpPageFull67}{\hidepaste}
\tab{5}\spadcommand{dst5::Boolean\free{dst5 }}
\indentrel{3}\begin{verbatim}
   (67)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RealClosureXmpPageEmpty67}
\begin{paste}{RealClosureXmpPageEmpty67}{RealClosureXmpPagePatch67}
\pastebutton{RealClosureXmpPageEmpty67}{\showpaste}
\tab{5}\spadcommand{dst5::Boolean\free{dst5 }}
\end{paste}\end{patch}
```

——–

3.92 record.ht

Domain Record(a:A,...,b:B)

⇒ “Description” (LispFunctions) 3.71 on page 952
⇒ “Operations” (LispFunctions) 3.71 on page 952

— **record.ht** —

```
\begin{page}{DomainRecord}{Domain {\em Record(a:A,...,b:B)}}
\beginscroll
{\em Record} takes any number of selector-domain pairs as arguments:
\indentrel{2}
\newline \spad{a}, a selector, an element of domain \spadtype{Symbol}
\newline \spad{A}, a domain of category \spadtype{SetCategory}
\newline\tab{10}...
\newline \spad{b}, a selector, an element of domain \spadtype{Symbol}
\newline \spad{B}, a domain of category \spadtype{SetCategory}
\indentrel{-2}\newline
This constructor is a primitive in Axiom.
\newline
\beginmenu
\item\menulispdownlink{Description}
{(|dbSpecialDescription| '|Record|)}\tab{15}General description
\item\menulispdownlink{Operations}
{(|dbSpecialOperations| '|Record|)}
\tab{15}All exported operations of {\em Record(a:A,b:B)}
%\item\menudownlink{Examples}   {RecordExamples}
```

```
\tab{15}Examples illustrating use
%\item\menudownlink{Exports}    {RecordExports}
\tab{15}Explicit categories and operations
\endmenu
\vspace{1}\newline
The selectors {\em a,...,b} of a \spad{Record} type must be distinct.
\endscroll\end{page}
```

———-

Domain Constructor Record

— record.ht —

```
\begin{page}{RecordDescription}{Domain Constructor {\em Record}}
\beginscroll
\newline\menuitemstyle{}\tab{2}Record({\em a:A},{\em b:B})
\newline\tab{2}{\em Arguments:}\indent{17}\tab{-2}
{\em a}, a selector, an element of domain \spadtype{Symbol}
\newline\tab{-2}
{\em A}, a domain of category \spadtype{SetCategory}
\newline\tab{10}...
\newline\tab{-2}
{\em b}, a selector, an element of domain \spadtype{Symbol}
\newline\tab{-2}
{\em B}, a domain of category \spadtype{SetCategory}
\indent{0}\newline\tab{2}{\em Returns:}\indent{15}\tab{0}
a record object with component objects of type {\em A},...,{\em B} with
correponding selectors {\em a},...,{\em b}
as described below.
\indent{0}\newline\tab{2}{\em Description:}\indent{15}\tab{0}
{\em Record(a:A,b:B)} is used to create the class of pairs of objects made
up of a value of type {\em A} selected by the symbol {\em a} and
a value of type {\em B} selected by the symbol {\em b}.
In general, the {\em Record} constructor can take any
number of arguments and thus can
be used to create aggregates of
heterogeneous components of arbitrary size selectable by name.
{\em Record} is a primitive domain of Axiom which cannot be
defined in the Axiom language.
\endscroll
\end{page}
```

———-

3.93 regset.ht

RegularTriangularSet

— regset.ht —

```
\begin{page}{RegularTriangularSetXmpPage}{RegularTriangularSet}
\beginscroll
The \spadtype{RegularTriangularSet} domain constructor implements
regular triangular sets.  These particular triangular sets were
introduced by M. Kalkbrener (1991) in his PhD Thesis under the name
regular chains.  Regular chains and their related concepts are
presented in the paper "On the Theories of Triangular sets" By
P. Aubry, D. Lazard and M. Moreno Maza (to appear in the Journal of
Symbolic Computation).  The \spadtype{RegularTriangularSet}
constructor also provides a new method (by the third author) for
solving polynomial system by means of regular chains.  This method has
two ways of solving.  One has the same specifications as Kalkbrener's
algorithm (1991) and the other is closer to Lazard's method
(Discr. App. Math, 1991).  Moreover, this new method removes redundant
component from the decompositions when this is not {\em too
expensive}.  This is always the case with square-free regular chains.
So if you want to obtain decompositions without redundant components
just use the \spadtype{SquareFreeRegularTriangularSet} domain
constructor or the \spadtype{LazardSetSolvingPackage} package
constructor.  See also the \spadtype{LexTriangularPackage} and
\spadtype{ZeroDimensionalSolvePackage} for the case of algebraic
systems with a finite number of (complex) solutions.

One of the main features of regular triangular sets is that they
naturally define towers of simple extensions of a field.
This allows to perform with multivariate polynomials the
same kind of operations as one can do in an \spadtype{EuclideanDomain}.

The \spadtype{RegularTriangularSet} constructor takes four arguments.
The first one, {\bf R}, is the coefficient ring of the polynomials; it
must belong to the category \spadtype{GcdDomain}.  The second one,
{\bf E}, is the exponent monoid of the polynomials; it must belong to
the category \spadtype{OrderedAbelianMonoidSup}.  the third one, {\bf
V}, is the ordered set of variables; it must belong to the category
\spadtype{OrderedSet}.  The last one is the polynomial ring; it must
belong to the category \spadtype{RecursivePolynomialCategory(R,E,V)}.
The abbreviation for \spadtype{RegularTriangularSet} is
\spadtype{REGSET}.  See also the constructor \spadtype{RegularChain}
which only takes two arguments, the coefficient ring and the ordered
set of variables; in that case, polynomials are necessarily built with
the \spadtype{NewSparseMultivariatePolynomial} domain constructor.
```

```
We shall explain now how to use the constructor \spadtype{REGSET}
and how to read the decomposition of a polynomial system by means
of regular sets.

Let us give some examples.  We start with an easy one
(Donati-Traverso) in order to understand the two ways of solving
polynomial systems provided by the \spadtype{REGSET} constructor.
\xtc{
Define the coefficient ring.
}{
\spadpaste{R := Integer \bound{R}}
}
\xtc{
Define the list of variables,
}{
\spadpaste{ls : List Symbol := [x,y,z,t] \bound{ls}}
}
\xtc{
and make it an ordered set;
}{
\spadpaste{V := OVAR(ls) \free{ls} \bound{V}}
}
\xtc{
then define the exponent monoid.
}{
\spadpaste{E := IndexedExponents V \free{V} \bound{E}}
}
\xtc{
Define the polynomial ring.
}{
\spadpaste{P := NSMP(R, V) \free{R} \free{V} \bound{P}}
}
\xtc{
Let the variables be polynomial.
}{
\spadpaste{x: P := 'x \free{P} \bound{x}}
}
\xtc{
}{
\spadpaste{y: P := 'y \free{P} \bound{y}}
}
\xtc{
}{
\spadpaste{z: P := 'z \free{P} \bound{z}}
}
\xtc{
}{
\spadpaste{t: P := 't \free{P} \bound{t}}
}
\xtc{
```

```
Now call the \spadtype{RegularTriangularSet} domain constructor.
}{
\spadpaste{T := REGSET(R,E,V,P) \free{R} \free{E}
\free{V} \free{P} \bound{T} }
}
\xtc{
Define a polynomial system.
}{
\spadpaste{p1 := x ** 31 - x ** 6 - x - y \free{x} \free{y} \bound{p1}}
}
\xtc{
}{
\spadpaste{p2 := x ** 8  - z \free{x} \free{z} \bound{p2}}
}
\xtc{
}{
\spadpaste{p3 := x ** 10 - t \free{x} \free{t} \bound{p3}}
}
\xtc{
}{
\spadpaste{lp := [p1, p2, p3] \free{p1} \free{p2} \free{p3} \bound{lp}}
}
\xtc{
First of all, let us solve this system in the sense of Kalkbrener.
}{
\spadpaste{zeroSetSplit(lp)$T \free{lp} \free{T}}
}
\xtc{
And now in the sense of Lazard (or Wu and other authors).
}{
\spadpaste{lts := zeroSetSplit(lp,false)$T \free{lp} \free{T} \bound{lts}}
}

We can see that the first decomposition is a subset of the second.
So how can both be correct ?

Recall first that polynomials from a domain of the category
\spadtype{RecursivePolynomialCategory} are regarded as univariate
polynomials in their main variable.  For instance the second
polynomial in the first set of each decomposition has main variable
{\bf y} and its initial (i.e. its leading coefficient w.r.t. its main
variable) is {\bf t z}.

Now let us explain how to read the second decomposition.  Note that
the non-constant initials of the first set are \texht{$t^4-t$}{{\bf
t^4 - t}} and \texht{$t z$}{{\bf t z}}.  Then the solutions described
by this first set are the common zeros of its polynomials that do not
cancel the polynomials \texht{$t^4-t$}{{\bf t^4 - t}} and
\texht{$ty z$}{{\bf t z}}.
Now the solutions of the input system {\bf lp}
```

```
satisfying these equations are described by the second and the third
sets of the decomposition.  Thus, in some sense, they can be
considered as degenerated solutions.  The solutions given by the first
set are called the generic points of the system; they give the general
form of the solutions.  The first decomposition only provides these
generic points.  This latter decomposition is useful when they are
many degenerated solutions (which is sometimes hard to compute) and
when one is only interested in general informations, like the
dimension of the input system.

\xtc{
We can get the dimensions of each component
of a decomposition as follows.
}{
\spadpaste{[coHeight(ts) for ts in lts] \free{lts}}
}

Thus the first set has dimension one.
Indeed {\bf t} can take any value, except {\bf 0}
or any third root of {\bf 1}, whereas {\bf z}
is completely determined from {\bf t},
{\bf y} is given by {\bf z} and {\bf t},
and finally {\bf x} is given by the other three variables.
In the second and the third sets of the second decomposition
the four variables are completely determined and thus
these sets have dimension zero.

We give now the precise specifications of each decomposition.
This assume some mathematical knowledge.
However, for the non-expert user, the above explanations will
be sufficient to understand the other features of the
\spadtype{RSEGSET} constructor.

The input system {\bf lp} is decomposed in the sense
of Kalkbrener as finitely many regular sets {\bf T1,...,Ts}
such that the radical ideal generated by {\bf lp}
is the intersection of the radicals of the
saturated ideals of {\bf T1,...,Ts}.
In other words, the affine variety associated with {\bf lp}
is the union of the closures (w.r.t. Zarisky topology)
of the regular-zeros sets of {\bf T1,...,Ts}.

{\bf N. B.} The prime ideals associated with the
radical of the saturated ideal of
a regular triangular set have all the same dimension;
moreover these prime ideals can be given by characteristic
sets with the same main variables.
Thus a decomposition in the sense of Kalkbrener
is unmixed dimensional.
Then it can be viewed as a {\em lazy}
```

```
decomposition into prime ideals (some of these
prime ideals being merged into unmixed dimensional ideals).

Now we explain the other way of solving by means of regular
triangular sets.
The input system {\bf lp} is decomposed in the sense
of Lazard as finitely many regular triangular sets {\bf T1,...,Ts}
such that the affine variety associated with {\bf lp}
is the union of the regular-zeros sets of {\bf T1,...,Ts}.
Thus a decomposition in the sense of Lazard is also
a decomposition in the sense of Kalkbrener; the converse
is false as we have seen before.

When the input system has a finite number of solutions,
both ways of solving provide similar decompositions as
we shall see with this second example (Caprasse).


\xtc{
Define a polynomial system.
}{
\spadpaste{f1 := y**2*z+2*x*y*t-2*x-z \free{z} \free{x}
\free{y} \free{t} \bound{f1}}
}
\xtc{
}{
\spadpaste{f2 :=   -x**3*z+ 4*x*y**2*z+ 4*x**2*y*t+ 2*y**3*t+
4*x**2- 10*y**2+ 4*x*z- 10*y*t+ 2 \free{z} \free{x} \free{y}
\free{t} \bound{f2}}
}
\xtc{
}{
\spadpaste{f3 :=  2*y*z*t+x*t**2-x-2*z \free{z} \free{x} \free{y}
\free{t} \bound{f3}}
}
\xtc{
}{
\spadpaste{f4 :=   -x*z**3+ 4*y*z**2*t+ 4*x*z*t**2+ 2*y*t**3+
4*x*z+ 4*z**2-10*y*t- 10*t**2+2 \free{z} \free{x} \free{y}
\free{t} \bound{f4}}
}
\xtc{
}{
\spadpaste{lf := [f1, f2, f3, f4] \free{f1} \free{f2}
\free{f3}  \free{f4} \bound{lf}}
}

\xtc{
First of all, let us solve this system in the sense of Kalkbrener.
}{
```

```
\spadpaste{zeroSetSplit(lf)$T \free{lf} \free{T}}
}
\xtc{
And now in the sense of Lazard (or Wu and other authors).
}{
\spadpaste{lts2 := zeroSetSplit(lf,false)$T \free{lf}
\free{T} \bound{lts2}}
}

Up to the ordering of the components, both decompositions are identical.

\xtc{
Let us check that each component has a finite number of solutions.
}{
\spadpaste{[coHeight(ts) for ts in lts2] \free{lts2}}
}

\xtc{
Let us count the degrees of each component,
}{
\spadpaste{degrees := [degree(ts) for ts in lts2]
\free{lts2} \bound{degrees}}
}
\xtc{
and compute their sum.
}{
\spadpaste{reduce(+,degrees) \free{degrees}}
}

We study now the options of the \spadfun{zeroSetSplit} operation.
As we have seen yet, there is an optional second argument
which is a boolean value. If this value is true (this
is the default) then the decomposition is computed
in the sense of Kalkbrener, otherwise it is computed
in the sense of Lazard.

There is a second boolean optional argument that
can be used (in that case the first optional
argument must be present).
This second option allows you to get some information
during the computations.

Therefore, we need to understand a little what is going on during the
computations.  An important feature of the algorithm is that the
intermediate computations are managed in some sense like the processes
of a Unix system.  Indeed, each intermediate computation may generate
other intermediate computations and the management of all these
computations is a crucial task for the efficiency.  Thus any
intermediate computation may be suspended, killed or resumed,
depending on algebraic considerations that determine priorities for
```

```
these processes.  The goal is of course to go as fast as possible
towards the final decomposition which means to avoid as much as
possible unnecessary computations.

To follow the computations, one needs to set to \spad{true} the second
argument.  Then a lot of numbers and letters are displayed.  Between a
\spad{[} and a \spad{]} one has the state of the processes at a given
time.  Just after \spad{[} one can see the number of processes.  Then
each process is represented by two numbers between \spad{<} and
\spad{>}.  A process consists of a list of polynomial {\bf ps} and a
triangular set {\bf ts}; its goal is to compute the common zeros of
{\bf ps} that belong to the regular-zeros set of {\bf ts}.  After the
processes, the number between pipes gives the total number of
polynomials in all the sets \spad{ps}.  Finally, the number between
braces gives the number of components of a decomposition that are
already computed. This number may decrease.

Let us take a third example (Czapor-Geddes-Wang) to see how these
informations are displayed.

\xtc{
Define a polynomial system.
}{
\spadpaste{u : R := 2  \free{R} \bound{u}}
}
\xtc{
}{
\spadpaste{q1 := 2*(u-1)**2+ 2*(x-z*x+z**2)+ y**2*(x-1)**2-
2*u*x+ 2*y*t*(1-x)*(x-z)+ 2*u*z*t*(t-y)+ u**2*t**2*(1-2*z)+
2*u*t**2*(z-x)+ 2*u*t*y*(z-1)+ 2*u*z*x*(y+1)+
(u**2-2*u)*z**2*t**2+ 2*u**2*z**2+ 4*u*(1-u)*z+ t**2*(z-x)**2
\free{z} \free{x} \free{y} \free{t} \free{u} \bound{q1}}
}
\xtc{
}{
\spadpaste{q2 := t*(2*z+1)*(x-z)+ y*(z+2)*(1-x)+ u*(u-2)*t+
u*(1-2*u)*z*t+ u*y*(x+u-z*x-1)+ u*(u+1)*z**2*t \free{z}
\free{x} \free{y} \free{t} \free{u} \bound{q2}}
}
\xtc{
}{
\spadpaste{q3 := -u**2*(z-1)**2+ 2*z*(z-x)-2*(x-1) \free{z}
\free{x} \free{y} \free{t} \free{u}  \bound{q3}}
}
\xtc{
}{
\spadpaste{q4 :=   u**2+4*(z-x**2)+3*y**2*(x-1)**2-
3*t**2*(z-x)**2 +3*u**2*t**2*(z-1)**2+u**2*z*(z-2)+
6*u*t*y*(z+x+z*x-1) \free{z} \free{x} \free{y} \free{t}
\free{u} \bound{q4}}
```

```
}
\xtc{
}{
\spadpaste{lq := [q1, q2, q3, q4] \free{q1} \free{q2}
\free{q3}  \free{q4} \bound{lq}}
}

\xtc{
Let us try the information option.
N.B. The timing should be between 1 and 10 minutes, depending on your machine.
}{
\spadpaste{zeroSetSplit(lq,true,true)$T \free{lq} \free{T}}
}

Between a sequence of processes, thus between a \spad{]} and a
\spad{[} you can see capital letters \spad{W, G, I} and lower case
letters \spad{i, w}. Each time a capital letter appears a non-trivial
computation has be performed and its result is put in a hash-table.
Each time a lower case letter appears a needed result has been found
in an hash-table.  The use of these hash-tables generally speed up the
computations.  However, on very large systems, it may happen that
these hash-tables become too big to be handle by your Axiom
configuration.  Then in these exceptional cases, you may prefer
getting a result (even if it takes a long time) than getting nothing.
Hence you need to know how to prevent the \spadtype{RSEGSET}
constructor from using these hash-tables.  In that case you will be
using the \spadfun{zeroSetSplit} with five arguments.  The first one
is the input system {\bf lp} as above.  The second one is a boolean
value \spad{hash?} which is \spad{true} iff you want to use
hash-tables.  The third one is boolean value \spad{clos?} which is
\spad{true} iff you want to solve your system in the sense of
Kalkbrener, the other way remaining that of Lazard.  The fourth
argument is boolean value \spad{info?} which is \spad{true} iff you
want to display information during the computations.  The last one is
boolean value \spad{prep?} which is \spad{true} iff you want to use
some heuristics that are performed on the input system before starting
the real algorithm.  The value of this flag is \spad{true} when you
are using \spadfun{zeroSetSplit} with less than five arguments.  Note
that there is no available signature for \spadfun{zeroSetSplit} with
four arguments.

We finish this section by some remarks about both ways of solving, in
the sense of Kalkbrener or in the sense of Lazard.  For problems with
a finite number of solutions, there are theoretically equivalent and
the resulting decompositions are identical, up to the ordering of the
components.  However, when solving in the sense of Lazard, the
algorithm behaves differently.  In that case, it becomes more
incremental than in the sense of Kalkbrener. That means the
polynomials of the input system are considered one after another
whereas in the sense of Kalkbrener the input system is treated more
```

```
globally.

This makes an important difference in positive dimension.  Indeed when
solving in the sense of Kalkbrener, the {\em Primeidealkettensatz} of
Krull is used.  That means any regular triangular containing more
polynomials than the input system can be deleted.  This is not
possible when solving in the sense of Lazard.  This explains why
Kalkbrener's decompositions usually contain less components than those
of Lazard.  However, it may happen with some examples that the
incremental process (that cannot be used when solving in the sense of
Kalkbrener) provide a more efficient way of solving than the global
one even if the {\em Primeidealkettensatz} is used.  Thus just try
both, with the various options, before concluding that you cannot
solve your favorite system with \spadfun{zeroSetSplit}.  There exist
more options at the development level that are not currently available
in this public version.  So you are welcome to contact {\em
marc@nag.co.uk} for more information and help.

\endscroll
\autobuttons
\end{page}

\begin{patch}{RegularTriangularSetXmpPagePatch1}
\begin{paste}{RegularTriangularSetXmpPageFull1}{RegularTriangularSetXmpPageEmpty1}
\pastebutton{RegularTriangularSetXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{R := Integer\bound{R }}
\indentrel{3}\begin{verbatim}
   (1)  Integer
                                           Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPageEmpty1}
\begin{paste}{RegularTriangularSetXmpPageEmpty1}{RegularTriangularSetXmpPagePatch1}
\pastebutton{RegularTriangularSetXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{R := Integer\bound{R }}
\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPagePatch2}
\begin{paste}{RegularTriangularSetXmpPageFull2}{RegularTriangularSetXmpPageEmpty2}
\pastebutton{RegularTriangularSetXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{ls : List Symbol := [x,y,z,t]\bound{ls }}
\indentrel{3}\begin{verbatim}
   (2)  [x,y,z,t]
                                      Type: List Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPageEmpty2}
\begin{paste}{RegularTriangularSetXmpPageEmpty2}{RegularTriangularSetXmpPagePatch2}
```

```
\pastebutton{RegularTriangularSetXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{ls : List Symbol := [x,y,z,t]\bound{ls }}
\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPagePatch3}
\begin{paste}{RegularTriangularSetXmpPageFull3}{RegularTriangularSetXmpPageEmpty3}
\pastebutton{RegularTriangularSetXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{V := OVAR(ls)\free{ls }\bound{V }}
\indentrel{3}\begin{verbatim}
   (3)  OrderedVariableList [x,y,z,t]
                                          Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPageEmpty3}
\begin{paste}{RegularTriangularSetXmpPageEmpty3}{RegularTriangularSetXmpPagePatch3}
\pastebutton{RegularTriangularSetXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{V := OVAR(ls)\free{ls }\bound{V }}
\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPagePatch4}
\begin{paste}{RegularTriangularSetXmpPageFull4}{RegularTriangularSetXmpPageEmpty4}
\pastebutton{RegularTriangularSetXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{E := IndexedExponents V\free{V }\bound{E }}
\indentrel{3}\begin{verbatim}
   (4)  IndexedExponents OrderedVariableList [x,y,z,t]
                                          Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPageEmpty4}
\begin{paste}{RegularTriangularSetXmpPageEmpty4}{RegularTriangularSetXmpPagePatch4}
\pastebutton{RegularTriangularSetXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{E := IndexedExponents V\free{V }\bound{E }}
\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPagePatch5}
\begin{paste}{RegularTriangularSetXmpPageFull5}{RegularTriangularSetXmpPageEmpty5}
\pastebutton{RegularTriangularSetXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{P := NSMP(R, V)\free{R }\free{V }\bound{P }}
\indentrel{3}\begin{verbatim}
   (5)
  NewSparseMultivariatePolynomial(Integer,OrderedVariable
  List [x,y,z,t])
                                          Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPageEmpty5}
\begin{paste}{RegularTriangularSetXmpPageEmpty5}{RegularTriangularSetXmpPagePatch5}
```

```
\pastebutton{RegularTriangularSetXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{P := NSMP(R, V)\free{R }\free{V }\bound{P }}
\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPagePatch6}
\begin{paste}{RegularTriangularSetXmpPageFull6}{RegularTriangularSetXmpPageEmpty6}
\pastebutton{RegularTriangularSetXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{x: P := 'x\free{P }\bound{x }}
\indentrel{3}\begin{verbatim}
   (6)  x
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [x,y,z,t])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPageEmpty6}
\begin{paste}{RegularTriangularSetXmpPageEmpty6}{RegularTriangularSetXmpPagePatch6}
\pastebutton{RegularTriangularSetXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{x: P := 'x\free{P }\bound{x }}
\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPagePatch7}
\begin{paste}{RegularTriangularSetXmpPageFull7}{RegularTriangularSetXmpPageEmpty7}
\pastebutton{RegularTriangularSetXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{y: P := 'y\free{P }\bound{y }}
\indentrel{3}\begin{verbatim}
   (7)  y
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [x,y,z,t])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPageEmpty7}
\begin{paste}{RegularTriangularSetXmpPageEmpty7}{RegularTriangularSetXmpPagePatch7}
\pastebutton{RegularTriangularSetXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{y: P := 'y\free{P }\bound{y }}
\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPagePatch8}
\begin{paste}{RegularTriangularSetXmpPageFull8}{RegularTriangularSetXmpPageEmpty8}
\pastebutton{RegularTriangularSetXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{z: P := 'z\free{P }\bound{z }}
\indentrel{3}\begin{verbatim}
   (8)  z
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [x,y,z,t])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPageEmpty8}
\begin{paste}{RegularTriangularSetXmpPageEmpty8}{RegularTriangularSetXmpPagePatch8}
\pastebutton{RegularTriangularSetXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{z: P := 'z\free{P }\bound{z }}
```

```
\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPagePatch9}
\begin{paste}{RegularTriangularSetXmpPageFull9}{RegularTriangularSetXmpPageEmpty9}
\pastebutton{RegularTriangularSetXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{t: P := 't\free{P }\bound{t }}
\indentrel{3}\begin{verbatim}
   (9)  t
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [x,y,z,t])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPageEmpty9}
\begin{paste}{RegularTriangularSetXmpPageEmpty9}{RegularTriangularSetXmpPagePatch9}
\pastebutton{RegularTriangularSetXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{t: P := 't\free{P }\bound{t }}
\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPagePatch10}
\begin{paste}{RegularTriangularSetXmpPageFull10}{RegularTriangularSetXmpPageEmpty10}
\pastebutton{RegularTriangularSetXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{T := REGSET(R,E,V,P)\free{R }\free{E }\free{V }\free{P }\bound{T }}
\indentrel{3}\begin{verbatim}
   (10)
  RegularTriangularSet(Integer,IndexedExponents OrderedVa
  riableList [x,y,z,t],OrderedVariableList [x,y,z,t],NewS
  parseMultivariatePolynomial(Integer,OrderedVariableList
   [x,y,z,t]))
                                             Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPageEmpty10}
\begin{paste}{RegularTriangularSetXmpPageEmpty10}{RegularTriangularSetXmpPagePatch10}
\pastebutton{RegularTriangularSetXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{T := REGSET(R,E,V,P)\free{R }\free{E }\free{V }\free{P }\bound{T }}
\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPagePatch11}
\begin{paste}{RegularTriangularSetXmpPageFull11}{RegularTriangularSetXmpPageEmpty11}
\pastebutton{RegularTriangularSetXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{p1 := x ** 31 - x ** 6 - x - y\free{x }\free{y }\bound{p1 }}
\indentrel{3}\begin{verbatim}
          31    6
   (11)  x   - x  - x - y
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [x,y,z,t])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPageEmpty11}
```

```
\begin{paste}{RegularTriangularSetXmpPageEmpty11}{RegularTriangularSetXmpPagePatch11}
\pastebutton{RegularTriangularSetXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{p1 := x ** 31 - x ** 6 - x - y\free{x }\free{y }\bound{p1 }}
\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPagePatch12}
\begin{paste}{RegularTriangularSetXmpPageFull12}{RegularTriangularSetXmpPageEmpty12}
\pastebutton{RegularTriangularSetXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{p2 := x ** 8 - z\free{x }\free{z }\bound{p2 }}
\indentrel{3}\begin{verbatim}
          8
   (12)  x  - z
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [x,y,z,t])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPageEmpty12}
\begin{paste}{RegularTriangularSetXmpPageEmpty12}{RegularTriangularSetXmpPagePatch12}
\pastebutton{RegularTriangularSetXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{p2 := x ** 8 - z\free{x }\free{z }\bound{p2 }}
\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPagePatch13}
\begin{paste}{RegularTriangularSetXmpPageFull13}{RegularTriangularSetXmpPageEmpty13}
\pastebutton{RegularTriangularSetXmpPageFull13}{\hidepaste}
\tab{5}\spadcommand{p3 := x ** 10 - t\free{x }\free{t }\bound{p3 }}
\indentrel{3}\begin{verbatim}
          10
   (13)  x   - t
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [x,y,z,t])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPageEmpty13}
\begin{paste}{RegularTriangularSetXmpPageEmpty13}{RegularTriangularSetXmpPagePatch13}
\pastebutton{RegularTriangularSetXmpPageEmpty13}{\showpaste}
\tab{5}\spadcommand{p3 := x ** 10 - t\free{x }\free{t }\bound{p3 }}
\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPagePatch14}
\begin{paste}{RegularTriangularSetXmpPageFull14}{RegularTriangularSetXmpPageEmpty14}
\pastebutton{RegularTriangularSetXmpPageFull14}{\hidepaste}
\tab{5}\spadcommand{lp := [p1, p2, p3]\free{p1 }\free{p2 }\free{p3 }\bound{lp }}
\indentrel{3}\begin{verbatim}
           31    6           8      10
   (14)  [x   - x  - x - y,x  - z,x   - t]
Type: List NewSparseMultivariatePolynomial(Integer,OrderedVariableList [x,y,z,t])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{RegularTriangularSetXmpPageEmpty14}
\begin{paste}{RegularTriangularSetXmpPageEmpty14}{RegularTriangularSetXmpPagePatch14}
\pastebutton{RegularTriangularSetXmpPageEmpty14}{\showpaste}
\tab{5}\spadcommand{lp := [p1, p2, p3]\free{p1 }\free{p2 }\free{p3 }\bound{lp }}
\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPagePatch15}
\begin{paste}{RegularTriangularSetXmpPageFull15}{RegularTriangularSetXmpPageEmpty15}
\pastebutton{RegularTriangularSetXmpPageFull15}{\hidepaste}
\tab{5}\spadcommand{zeroSetSplit(lp)$T\free{lp }\free{T }}
\indentrel{3}\begin{verbatim}
   (15)
   [
      5    4       2     3     8     5    3    2
     {z  - t , t z y  + 2z y - t  + 2t  + t  - t ,
        4              2
      (t  - t)x - t y - z }
     ]
Type: List RegularTriangularSet(Integer,IndexedExponents OrderedVariableList [x,y,z,t],OrderedVariableList
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPageEmpty15}
\begin{paste}{RegularTriangularSetXmpPageEmpty15}{RegularTriangularSetXmpPagePatch15}
\pastebutton{RegularTriangularSetXmpPageEmpty15}{\showpaste}
\tab{5}\spadcommand{zeroSetSplit(lp)$T\free{lp }\free{T }}
\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPagePatch16}
\begin{paste}{RegularTriangularSetXmpPageFull16}{RegularTriangularSetXmpPageEmpty16}
\pastebutton{RegularTriangularSetXmpPageFull16}{\hidepaste}
\tab{5}\spadcommand{lts := zeroSetSplit(lp,false)$T\free{lp }\free{T }\bound{lts }}
\indentrel{3}\begin{verbatim}
   (16)
   [
      5    4       2     3     8     5    3    2
     {z  - t , t z y  + 2z y - t  + 2t  + t  - t ,
        4              2
      (t  - t)x - t y - z }
     ,
      3      5          2     3         2
    {t  - 1,z  - t,t z y  + 2z y + 1,z x  - t},
    {t,z,y,x}]
Type: List RegularTriangularSet(Integer,IndexedExponents OrderedVariableList [x,y,z,t],OrderedVariableList
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPageEmpty16}
\begin{paste}{RegularTriangularSetXmpPageEmpty16}{RegularTriangularSetXmpPagePatch16}
\pastebutton{RegularTriangularSetXmpPageEmpty16}{\showpaste}
```

```
\tab{5}\spadcommand{lts := zeroSetSplit(lp,false)$T\free{lp }\free{T }\bound{lts }}
\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPagePatch17}
\begin{paste}{RegularTriangularSetXmpPageFull17}{RegularTriangularSetXmpPageEmpty17}
\pastebutton{RegularTriangularSetXmpPageFull17}{\hidepaste}
\tab{5}\spadcommand{[coHeight(ts) for ts in lts]\free{lts }}
\indentrel{3}\begin{verbatim}
   (17)  [1,0,0]
                          Type: List NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPageEmpty17}
\begin{paste}{RegularTriangularSetXmpPageEmpty17}{RegularTriangularSetXmpPagePatch17}
\pastebutton{RegularTriangularSetXmpPageEmpty17}{\showpaste}
\tab{5}\spadcommand{[coHeight(ts) for ts in lts]\free{lts }}
\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPagePatch18}
\begin{paste}{RegularTriangularSetXmpPageFull18}{RegularTriangularSetXmpPageEmpty18}
\pastebutton{RegularTriangularSetXmpPageFull18}{\hidepaste}
\tab{5}\spadcommand{f1 := y**2*z+2*x*y*t-2*x-z\free{z }\free{x }\free{y }\free{t }\bound{f1 }}
\indentrel{3}\begin{verbatim}
                          2
   (18)  (2t y - 2)x + z y  - z
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [x,y,z,t])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPageEmpty18}
\begin{paste}{RegularTriangularSetXmpPageEmpty18}{RegularTriangularSetXmpPagePatch18}
\pastebutton{RegularTriangularSetXmpPageEmpty18}{\showpaste}
\tab{5}\spadcommand{f1 := y**2*z+2*x*y*t-2*x-z\free{z }\free{x }\free{y }\free{t }\bound{f1 }}
\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPagePatch19}
\begin{paste}{RegularTriangularSetXmpPageFull19}{RegularTriangularSetXmpPageEmpty19}
\pastebutton{RegularTriangularSetXmpPageFull19}{\hidepaste}
\tab{5}\spadcommand{f2 := -x**3*z+ 4*x*y**2*z+ 4*x**2*y*t+ 2*y**3*t+ 4*x**2- 10*y**2+ 4*x*z- 10*
\indentrel{3}\begin{verbatim}
   (19)
          3               2        2               3      2
     - z x  + (4t y + 4)x  + (4z y  + 4z)x + 2t y  - 10y
   +
     - 10t y + 2
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [x,y,z,t])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{RegularTriangularSetXmpPageEmpty19}
\begin{paste}{RegularTriangularSetXmpPageEmpty19}{RegularTriangularSetXmpPagePatch19}
\pastebutton{RegularTriangularSetXmpPageEmpty19}{\showpaste}
\tab{5}\spadcommand{f2 := -x**3*z+ 4*x*y**2*z+ 4*x**2*y*t+ 2*y**3*t+ 4*x**2- 10*y**2+ 4*x*z- 10*y*t+ 2\free
\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPagePatch20}
\begin{paste}{RegularTriangularSetXmpPageFull20}{RegularTriangularSetXmpPageEmpty20}
\pastebutton{RegularTriangularSetXmpPageFull20}{\hidepaste}
\tab{5}\spadcommand{f3 := 2*y*z*t+x*t**2-x-2*z\free{z }\free{x }\free{y }\free{t }\bound{f3 }}
\indentrel{3}\begin{verbatim}
           2
   (20)  (t  - 1)x + 2t z y - 2z
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [x,y,z,t])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPageEmpty20}
\begin{paste}{RegularTriangularSetXmpPageEmpty20}{RegularTriangularSetXmpPagePatch20}
\pastebutton{RegularTriangularSetXmpPageEmpty20}{\showpaste}
\tab{5}\spadcommand{f3 := 2*y*z*t+x*t**2-x-2*z\free{z }\free{x }\free{y }\free{t }\bound{f3 }}
\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPagePatch21}
\begin{paste}{RegularTriangularSetXmpPageFull21}{RegularTriangularSetXmpPageEmpty21}
\pastebutton{RegularTriangularSetXmpPageFull21}{\hidepaste}
\tab{5}\spadcommand{f4 := -x*z**3+ 4*y*z**2*t+ 4*x*z*t**2+ 2*y*t**3+ 4*x*z+ 4*z**2-10*y*t- 10*t**2+2\free{z
\indentrel{3}\begin{verbatim}
   (21)
        3      2                2     3             2
     (- z  + (4t  + 4)z)x + (4t z  + 2t  - 10t)y + 4z
   +
          2
     - 10t  + 2
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [x,y,z,t])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPageEmpty21}
\begin{paste}{RegularTriangularSetXmpPageEmpty21}{RegularTriangularSetXmpPagePatch21}
\pastebutton{RegularTriangularSetXmpPageEmpty21}{\showpaste}
\tab{5}\spadcommand{f4 := -x*z**3+ 4*y*z**2*t+ 4*x*z*t**2+ 2*y*t**3+ 4*x*z+ 4*z**2-10*y*t- 10*t**2+2\free{z
\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPagePatch22}
\begin{paste}{RegularTriangularSetXmpPageFull22}{RegularTriangularSetXmpPageEmpty22}
\pastebutton{RegularTriangularSetXmpPageFull22}{\hidepaste}
\tab{5}\spadcommand{lf := [f1, f2, f3, f4]\free{f1 }\free{f2 }\free{f3 }\free{f4 }\bound{lf }}
\indentrel{3}\begin{verbatim}
   (22)
```

```
                  2
   [(2t y - 2)x + z y  - z,

          3             2       2             3
      - z x  + (4t y + 4)x  + (4z y  + 4z)x + 2t y
     +
          2
      - 10y  - 10t y + 2
     ,
      2
    (t  - 1)x + 2t z y - 2z,

         3      2               2     3            2
      (- z  + (4t  + 4)z)x + (4t z  + 2t  - 10t)y + 4z
     +
          2
      - 10t  + 2
     ]
Type: List NewSparseMultivariatePolynomial(Integer,OrderedVariableList [x,y,z,t])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPageEmpty22}
\begin{paste}{RegularTriangularSetXmpPageEmpty22}{RegularTriangularSetXmpPagePatch22}
\pastebutton{RegularTriangularSetXmpPageEmpty22}{\showpaste}
\tab{5}\spadcommand{lf := [f1, f2, f3, f4]\free{f1 }\free{f2 }\free{f3 }\free{f4 }\bound{lf }}
\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPagePatch23}
\begin{paste}{RegularTriangularSetXmpPageFull23}{RegularTriangularSetXmpPageEmpty23}
\pastebutton{RegularTriangularSetXmpPageFull23}{\hidepaste}
\tab{5}\spadcommand{zeroSetSplit(lf)$T\free{lf }\free{T }}
\indentrel{3}\begin{verbatim}
   (23)
   [
       2       8      6       2
     {t  - 1, z  - 16z  + 256z  - 256, t y - 1,
        3           2
      (z  - 8z)x - 8z  + 16}
     ,
       2      2     2
    {3t  + 1,z  - 7t  - 1,y + t,x + z},
      8      6      2        3           2
    {t  - 10t  + 10t  - 1,z,(t  - 5t)y - 5t  + 1,x},
      2      2
    {t  + 3,z  - 4,y + t,x - z}]
Type: List RegularTriangularSet(Integer,IndexedExponents OrderedVariableList [x,y,z,t],OrderedVa
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{RegularTriangularSetXmpPageEmpty23}
\begin{paste}{RegularTriangularSetXmpPageEmpty23}{RegularTriangularSetXmpPagePatch23}
\pastebutton{RegularTriangularSetXmpPageEmpty23}{\showpaste}
\tab{5}\spadcommand{zeroSetSplit(lf)$T\free{lf }\free{T }}
\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPagePatch24}
\begin{paste}{RegularTriangularSetXmpPageFull24}{RegularTriangularSetXmpPageEmpty24}
\pastebutton{RegularTriangularSetXmpPageFull24}{\hidepaste}
\tab{5}\spadcommand{lts2 := zeroSetSplit(lf,false)$T\free{lf }\free{T }\bound{lts2 }}
\indentrel{3}\begin{verbatim}
   (24)
     8      6      2         3            2
   [{t  - 10t  + 10t  - 1,z,(t  - 5t)y - 5t  + 1,x},

      2       8      6       2
     {t  - 1, z  - 16z  + 256z  - 256, t y - 1,
       3           2
      (z  - 8z)x - 8z  + 16}
     ,
      2      2     2
    {3t  + 1,z  - 7t  - 1,y + t,x + z},
     2      2
    {t  + 3,z  - 4,y + t,x - z}]
Type: List RegularTriangularSet(Integer,IndexedExponents OrderedVariableList [x,y,z,t],OrderedVariableList
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPageEmpty24}
\begin{paste}{RegularTriangularSetXmpPageEmpty24}{RegularTriangularSetXmpPagePatch24}
\pastebutton{RegularTriangularSetXmpPageEmpty24}{\showpaste}
\tab{5}\spadcommand{lts2 := zeroSetSplit(lf,false)$T\free{lf }\free{T }\bound{lts2 }}
\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPagePatch25}
\begin{paste}{RegularTriangularSetXmpPageFull25}{RegularTriangularSetXmpPageEmpty25}
\pastebutton{RegularTriangularSetXmpPageFull25}{\hidepaste}
\tab{5}\spadcommand{[coHeight(ts) for ts in lts2]\free{lts2 }}
\indentrel{3}\begin{verbatim}
   (25)  [0,0,0,0]
                          Type: List NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPageEmpty25}
\begin{paste}{RegularTriangularSetXmpPageEmpty25}{RegularTriangularSetXmpPagePatch25}
\pastebutton{RegularTriangularSetXmpPageEmpty25}{\showpaste}
\tab{5}\spadcommand{[coHeight(ts) for ts in lts2]\free{lts2 }}
\end{paste}\end{patch}
```

```
\begin{patch}{RegularTriangularSetXmpPagePatch26}
\begin{paste}{RegularTriangularSetXmpPageFull26}{RegularTriangularSetXmpPageEmpty26}
\pastebutton{RegularTriangularSetXmpPageFull26}{\hidepaste}
\tab{5}\spadcommand{degrees := [degree(ts) for ts in lts2]\free{lts2 }\bound{degrees }}
\indentrel{3}\begin{verbatim}
   (26)  [8,16,4,4]
                         Type: List NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPageEmpty26}
\begin{paste}{RegularTriangularSetXmpPageEmpty26}{RegularTriangularSetXmpPagePatch26}
\pastebutton{RegularTriangularSetXmpPageEmpty26}{\showpaste}
\tab{5}\spadcommand{degrees := [degree(ts) for ts in lts2]\free{lts2 }\bound{degrees }}
\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPagePatch27}
\begin{paste}{RegularTriangularSetXmpPageFull27}{RegularTriangularSetXmpPageEmpty27}
\pastebutton{RegularTriangularSetXmpPageFull27}{\hidepaste}
\tab{5}\spadcommand{reduce(+,degrees)\free{degrees }}
\indentrel{3}\begin{verbatim}
   (27)  32
                                 Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPageEmpty27}
\begin{paste}{RegularTriangularSetXmpPageEmpty27}{RegularTriangularSetXmpPagePatch27}
\pastebutton{RegularTriangularSetXmpPageEmpty27}{\showpaste}
\tab{5}\spadcommand{reduce(+,degrees)\free{degrees }}
\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPagePatch28}
\begin{paste}{RegularTriangularSetXmpPageFull28}{RegularTriangularSetXmpPageEmpty28}
\pastebutton{RegularTriangularSetXmpPageFull28}{\hidepaste}
\tab{5}\spadcommand{u : R := 2\free{R }\bound{u }}
\indentrel{3}\begin{verbatim}
   (28)  2
                                         Type: Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPageEmpty28}
\begin{paste}{RegularTriangularSetXmpPageEmpty28}{RegularTriangularSetXmpPagePatch28}
\pastebutton{RegularTriangularSetXmpPageEmpty28}{\showpaste}
\tab{5}\spadcommand{u : R := 2\free{R }\bound{u }}
\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPagePatch29}
\begin{paste}{RegularTriangularSetXmpPageFull29}{RegularTriangularSetXmpPageEmpty29}
```

```
\pastebutton{RegularTriangularSetXmpPageFull29}{\hidepaste}
\tab{5}\spadcommand{q1 := 2*(u-1)**2+ 2*(x-z*x+z**2)+ y**2*(x-1)**2- 2*u*x+ 2*y*t*(1-x)*(x-z)+ 2*u*z*t*(t-y)
\indentrel{3}\begin{verbatim}
   (29)
       2           2  2
     (y  - 2t y + t )x
   +
          2                            2           2
     (- 2y  + ((2t + 4)z + 2t)y + (- 2t  + 2)z - 4t  - 2)x
   +
      2                      2       2          2
     y  + (- 2t z - 4t)y + (t  + 10)z  - 8z + 4t  + 2
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [x,y,z,t])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPageEmpty29}
\begin{paste}{RegularTriangularSetXmpPageEmpty29}{RegularTriangularSetXmpPagePatch29}
\pastebutton{RegularTriangularSetXmpPageEmpty29}{\showpaste}
\tab{5}\spadcommand{q1 := 2*(u-1)**2+ 2*(x-z*x+z**2)+ y**2*(x-1)**2- 2*u*x+ 2*y*t*(1-x)*(x-z)+ 2*u*z*t*(t-y)
\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPagePatch30}
\begin{paste}{RegularTriangularSetXmpPageFull30}{RegularTriangularSetXmpPageEmpty30}
\pastebutton{RegularTriangularSetXmpPageFull30}{\hidepaste}
\tab{5}\spadcommand{q2 := t*(2*z+1)*(x-z)+ y*(z+2)*(1-x)+ u*(u-2)*t+ u*(1-2*u)*z*t+ u*y*(x+u-z*x-1)+ u*(u+1)
\indentrel{3}\begin{verbatim}
                                              2
   (30)  (- 3z y + 2t z + t)x + (z + 4)y + 4t z  - 7t z
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [x,y,z,t])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPageEmpty30}
\begin{paste}{RegularTriangularSetXmpPageEmpty30}{RegularTriangularSetXmpPagePatch30}
\pastebutton{RegularTriangularSetXmpPageEmpty30}{\showpaste}
\tab{5}\spadcommand{q2 := t*(2*z+1)*(x-z)+ y*(z+2)*(1-x)+ u*(u-2)*t+ u*(1-2*u)*z*t+ u*y*(x+u-z*x-1)+ u*(u+1)
\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPagePatch31}
\begin{paste}{RegularTriangularSetXmpPageFull31}{RegularTriangularSetXmpPageEmpty31}
\pastebutton{RegularTriangularSetXmpPageFull31}{\hidepaste}
\tab{5}\spadcommand{q3 := -u**2*(z-1)**2+ 2*z*(z-x)-2*(x-1)\free{z }\free{x }\free{y }\free{t }\free{u }\bou
\indentrel{3}\begin{verbatim}
                          2
   (31)  (- 2z - 2)x - 2z  + 8z - 2
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [x,y,z,t])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{RegularTriangularSetXmpPageEmpty31}
\begin{paste}{RegularTriangularSetXmpPageEmpty31}{RegularTriangularSetXmpPagePatch31}
\pastebutton{RegularTriangularSetXmpPageEmpty31}{\showpaste}
\tab{5}\spadcommand{q3 := -u**2*(z-1)**2+ 2*z*(z-x)-2*(x-1)\free{z }\free{x }\free{y }\free{t }\
\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPagePatch32}
\begin{paste}{RegularTriangularSetXmpPageFull32}{RegularTriangularSetXmpPageEmpty32}
\pastebutton{RegularTriangularSetXmpPageFull32}{\hidepaste}
\tab{5}\spadcommand{q4 := u**2+4*(z-x**2)+3*y**2*(x-1)**2- 3*t**2*(z-x)**2 +3*u**2*t**2*(z-1)**2
\indentrel{3}\begin{verbatim}
   (32)
        2     2      2        2                      2
     (3y  - 3t  - 4)x  + (- 6y  + (12t z + 12t)y + 6t z)x
   +
       2                        2      2         2
     3y  + (12t z - 12t)y + (9t  + 4)z  + (- 24t  - 4)z
   +
        2
     12t  + 4
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [x,y,z,t])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPageEmpty32}
\begin{paste}{RegularTriangularSetXmpPageEmpty32}{RegularTriangularSetXmpPagePatch32}
\pastebutton{RegularTriangularSetXmpPageEmpty32}{\showpaste}
\tab{5}\spadcommand{q4 := u**2+4*(z-x**2)+3*y**2*(x-1)**2- 3*t**2*(z-x)**2 +3*u**2*t**2*(z-1)**2
\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPagePatch33}
\begin{paste}{RegularTriangularSetXmpPageFull33}{RegularTriangularSetXmpPageEmpty33}
\pastebutton{RegularTriangularSetXmpPageFull33}{\hidepaste}
\tab{5}\spadcommand{lq := [q1, q2, q3, q4]\free{q1 }\free{q2 }\free{q3 }\free{q4 }\bound{lq }}
\indentrel{3}\begin{verbatim}
   (33)
   [
         2           2  2
       (y  - 2t y + t )x
     +
             2                            2           2
         - 2y  + ((2t + 4)z + 2t)y + (- 2t  + 2)z - 4t
       +
         - 2
      *
        x
     +
        2                      2      2          2
       y  + (- 2t z - 4t)y + (t  + 10)z  - 8z + 4t  + 2
     ,
```

```
                                      2
    (- 3z y + 2t z + t)x + (z + 4)y + 4t z  - 7t z,
                   2
    (- 2z - 2)x - 2z  + 8z - 2,

          2     2      2
       (3y  - 3t  - 4)x
     +
            2                      2        2
       (- 6y  + (12t z + 12t)y + 6t z)x + 3y
     +
                             2      2         2            2
       (12t z - 12t)y + (9t  + 4)z  + (- 24t  - 4)z + 12t
     +
       4
     ]
Type: List NewSparseMultivariatePolynomial(Integer,OrderedVariableList [x,y,z,t])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPageEmpty33}
\begin{paste}{RegularTriangularSetXmpPageEmpty33}{RegularTriangularSetXmpPagePatch33}
\pastebutton{RegularTriangularSetXmpPageEmpty33}{\showpaste}
\tab{5}\spadcommand{lq := [q1, q2, q3, q4]\free{q1 }\free{q2 }\free{q3 }\free{q4 }\bound{lq }}
\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPagePatch34}
\begin{paste}{RegularTriangularSetXmpPageFull34}{RegularTriangularSetXmpPageEmpty34}
\pastebutton{RegularTriangularSetXmpPageFull34}{\hidepaste}
\tab{5}\spadcommand{zeroSetSplit(lq,true,true)$T\free{lq }\free{T }}
\indentrel{3}\begin{verbatim}

   *** QCMPACK Statistics ***
      Table     size:  36
      Entries reused:  255

   *** REGSETGCD: Gcd Statistics ***
      Table     size:  125
      Entries reused:  0

   *** REGSETGCD: Inv Set Statistics ***
      Table     size:  30
      Entries reused:  0
   (34)
   [
     {
                         24                     23
        960725655771966t   + 386820897948702t
      +
                          22                     21
```

```
  8906817198608181t   + 2704966893949428t
+
                    20                     19
  37304033340228264t   + 7924782817170207t
+
                    18                     17
  93126799040354990t   + 131012736531309l0t
+
                     16                     15
  156146250424711858t   + 166264909572591l9t
+
                     14                     13
  190699288479805763t   + 243391733676252 75t
+
                     12                     11
  180532313014960135t   + 352880890309753 78t
+
                     10                     9
  135054975747656285t   + 347337369524885 40t
+
                    8                     7
  75947600354493972t  + 197725556924570 88t
+
                    6                     5
  28871558573755428t  + 5576152439081664t
+
                   4                    3
  6321711820352976t  + 438314209312320t
+
                2
581105748367008t  - 60254467992576t + 1449115951104
,

       266042108694913023855152657370520823616684_
        74181372891857784
     *
         23
        t
    +
       443104378424686086067294899528296664238693_
        556855017735265295
     *
         22
        t
    +
       279078393286701234679141342358988327155321_
        305829547090310242
     *
         21
        t
```

```
+
   33902763614132324651076171766155430546206 2_
    6391823613392185226
 *
     20
    t
+
   94147817950354057555419864522035280371979 3_
    196473813837434129
 *
     19
    t
+
   11547855194679475242211696749673949352585 7_
    47674184320988144390
 *
     18
    t
+
   13436095667655977898817016566994132164672 1_
    5660333356417241432
 *
     17
    t
+
   23233813868147873503933551617175640859899 1_
    02987800663566699334
 *
     16
    t
+
   86957402053767233695084544050879074085093 1_
    336484983573386433
 *
     15
    t
+
   31561554305876934875419461486969926554241 7_
    50065103460820476969
 *
     14
    t
+
   12714009902877174874420659525477318795548 2_
    3889855386072264931
 *
     13
    t
+
   31945089913863736044802526964079540198337 0_
```

```
     495505032958251605 23
   *
      12
     t
  +
    3738735704288144509871371560232845884439 10_
     2270778010470931960
   *
      11
     t
  +
    252939975123914120261446014357711315875619_
     05532992045692885927
   *
      10
     t
  +
    521023900984606712346926279987005277341047_
     1135950175008046524
   *
      9
     t
  +
    150838879869302971662598705686082704274031_
     87606238713491129188
   *
      8
     t
  +
    352208723469293012638368627077577955348176_
     9125670839075109000
   *
      7
     t
  +
    607994520039568101308653379256888649110124_
     4247440034969288588
   *
      6
     t
  +
    109063485243390088819991375624798602319698_
     7723469934933603680
   *
      5
     t
  +
    140581943087190710229443253753833540210283_
     8994019667487458352
   *
```

```
          4
         t
      +
        88071527950320450072536671265507748878347 8_
         28884933605202432
       *
          3
         t
      +
        13588248943364093322978117715597776801606 5_
         765482378657129440
       *
          2
         t
      +
        -
           13957283442882262230559894607400314082 5_
            16690749975646520320
         *
            t
      +
       33463769297318929927725832570930847259211711_
        2855749713920
   *
      z
  +
     85671754840439528797567259645068339321496371 01_
      090521164936
   *
       23
      t
  +
     14979239286420179184570837403272894249879751 92_
      51667250945721
   *
       22
      t
  +
     77258371783645822157410861582159764138123003 07_
      4190374021550
   *
       21
      t
  +
     11088622541268542144989189407086122111845605 56_
      764334742191654
   *
       20
      t
  +
```

```
     213250494460678865219774480106826053783815789б_
      21501732672327
   *
       19
      t
  +
     366892907516066619572917789417834351450198789_
      410131431699882
   *
       18
      t
  +
     171388906471001872879490124368748236314765459_
      39567820048872
   *
       17
      t
  +
     719243074691460216666023347733102248314492177_
      645523139658986
   *
       16
      t
  +
    -
        128798674689690072812879965633090291959663_
         43108437362453385
      *
          15
         t
  +
     955301085834142590930642313292113404085602879_
      803526430270671
   *
       14
      t
  +
    -
        132960962456754928745386876463004378246584_
         8709144441096603
      *
          13
         t
  +
     947580680581414532638308551832533310688169056_
      644274964864413
   *
       12
      t
  +
```

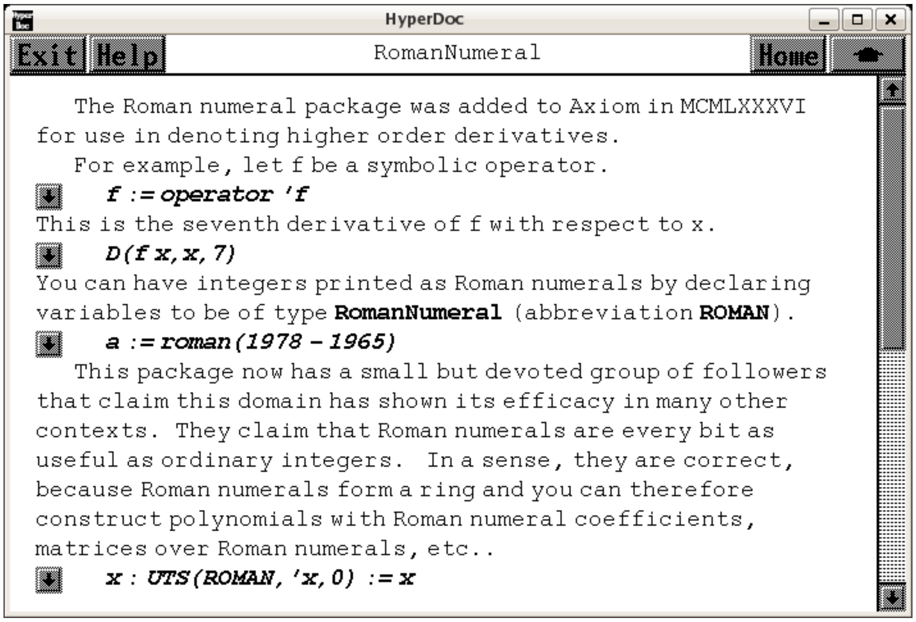
```
   8032346879251334588616598556640849276062987947_
    99856265539336
  *
     11
    t
 +
   7338202759292865165994622349207516400662174302_
    614595173333825
  *
     10
    t
 +
   1308004628480367351164369613111971668880538855_
    640917200187108
  *
     9
    t
 +
   4268059455741255498880229598973705747098216067_
    697754352634748
  *
     8
    t
 +
   8928935268585140957913187759040933001030456015_
    14470613580600
  *
     7
    t
 +
   1679152575460683956631925852181341501981598137_
    465328797013652
  *
     6
    t
 +
   2697574157679229803789671541433578355441131582_
    80591408043936
  *
     5
    t
 +
   3809515278646575290335808298012827240813453726_
    80202920198224
  *
     4
    t
 +
   1978554529422849503299882693760134113272503533_
    9452913286656
```

```
       *
           3
          t
      +
        3647741205738478294236663530339663776330392817_
          4935079178528
       *
           2
          t
      +
        -
             3722212879279038648713080422224976273210890_
              229485838670848
          *
             t
      +
       890797248531143483612306344840138620247285999068_
        74105856
      ,
        3      2                   3       2
    (3z  - 11z  + 8z + 4)y + 2t z  + 4t z  - 5t z - t,
                2
    (z + 1)x + z  - 4z + 1}
   ]
Type: List RegularTriangularSet(Integer,IndexedExponents OrderedVariableList [x,y,z,t],OrderedVa
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RegularTriangularSetXmpPageEmpty34}
\begin{paste}{RegularTriangularSetXmpPageEmpty34}{RegularTriangularSetXmpPagePatch34}
\pastebutton{RegularTriangularSetXmpPageEmpty34}{\showpaste}
\tab{5}\spadcommand{zeroSetSplit(lq,true,true)$T\free{lq }\free{T }}
\end{paste}\end{patch}
```

————

3.94 roman.ht

RomanNumeral



⇐ "Axiom Number Types" (NumberPage) 3.80 on page 1021

— roman.ht —

```
\begin{page}{RomanNumeralXmpPage}{RomanNumeral}
\beginscroll
The Roman numeral package was added to Axiom in MCMLXXXVI
for use in denoting higher order derivatives.

\xtc{
For example, let \spad{f} be a symbolic operator.
}{
\spadpaste{f := operator 'f \bound{f}}
}
\xtc{
This is the seventh derivative of \spad{f} with respect to \spad{x}.
}{
\spadpaste{D(f x,x,7) \free{f}}
}
\xtc{
You can have integers printed as Roman numerals by declaring variables to
be of type \spadtype{RomanNumeral} (abbreviation \spadtype{ROMAN}).
}{
\spadpaste{a := roman(1978 - 1965) \bound{a}}
```

```
}

This package now has a small but devoted group of followers that claim
this domain has shown its efficacy in many other contexts.
They claim that Roman numerals are every bit as useful as ordinary
integers.
\xtc{
In a sense, they are correct, because Roman numerals form a ring and you
can therefore construct polynomials with Roman numeral coefficients,
matrices over Roman numerals, etc..
}{
\spadpaste{x : UTS(ROMAN,'x,0) := x \bound{x}}
}
\xtc{
Was Fibonacci Italian or ROMAN?
}{
\spadpaste{recip(1 - x - x**2) \free{x}}
}
\xtc{
You can also construct fractions with Roman numeral numerators and
denominators, as this matrix Hilberticus illustrates.
}{
\spadpaste{m : MATRIX FRAC ROMAN \bound{m}}
}
\xtc{
}{
\spadpaste{m := matrix [[1/(i + j) for i in 1..3] for j in 1..3]
\free{m} \bound{m1}}
}
\xtc{
Note that the inverse of the matrix has integral \spadtype{ROMAN}
entries.
}{
\spadpaste{inverse m \free{m1}}
}
\xtc{
Unfortunately, the spoil-sports say that the fun stops when
the numbers get big---mostly
because the Romans didn't establish conventions about representing
very large numbers.
}{
\spadpaste{y := factorial 10 \bound{y}}
}
\xtc{
You work it out!
}{
\spadpaste{roman y \free{y}}
}
Issue the system command
\spadcmd{)show RomanNumeral}
```

```
to display the full list of operations defined by
\spadtype{RomanNumeral}.
\endscroll
\autobuttons
\end{page}

\begin{patch}{RomanNumeralXmpPagePatch1}
\begin{paste}{RomanNumeralXmpPageFull1}{RomanNumeralXmpPageEmpty1}
\pastebutton{RomanNumeralXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{f := operator 'f\bound{f }}
\indentrel{3}\begin{verbatim}
   (1)  f
                                    Type: BasicOperator
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RomanNumeralXmpPageEmpty1}
\begin{paste}{RomanNumeralXmpPageEmpty1}{RomanNumeralXmpPagePatch1}
\pastebutton{RomanNumeralXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{f := operator 'f\bound{f }}
\end{paste}\end{patch}

\begin{patch}{RomanNumeralXmpPagePatch2}
\begin{paste}{RomanNumeralXmpPageFull2}{RomanNumeralXmpPageEmpty2}
\pastebutton{RomanNumeralXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{D(f x,x,7)\free{f }}
\indentrel{3}\begin{verbatim}
         (vii)
   (2)  f     (x)

                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RomanNumeralXmpPageEmpty2}
\begin{paste}{RomanNumeralXmpPageEmpty2}{RomanNumeralXmpPagePatch2}
\pastebutton{RomanNumeralXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{D(f x,x,7)\free{f }}
\end{paste}\end{patch}

\begin{patch}{RomanNumeralXmpPagePatch3}
\begin{paste}{RomanNumeralXmpPageFull3}{RomanNumeralXmpPageEmpty3}
\pastebutton{RomanNumeralXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{a := roman(1978 - 1965)\bound{a }}
\indentrel{3}\begin{verbatim}
   (3)  XIII
                                     Type: RomanNumeral
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{RomanNumeralXmpPageEmpty3}
\begin{paste}{RomanNumeralXmpPageEmpty3}{RomanNumeralXmpPagePatch3}
\pastebutton{RomanNumeralXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{a := roman(1978 - 1965)\bound{a }}
\end{paste}\end{patch}

\begin{patch}{RomanNumeralXmpPagePatch4}
\begin{paste}{RomanNumeralXmpPageFull4}{RomanNumeralXmpPageEmpty4}
\pastebutton{RomanNumeralXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{x : UTS(ROMAN,'x,0) := x\bound{x }}
\indentrel{3}\begin{verbatim}
   (4)  x
         Type: UnivariateTaylorSeries(RomanNumeral,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RomanNumeralXmpPageEmpty4}
\begin{paste}{RomanNumeralXmpPageEmpty4}{RomanNumeralXmpPagePatch4}
\pastebutton{RomanNumeralXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{x : UTS(ROMAN,'x,0) := x\bound{x }}
\end{paste}\end{patch}

\begin{patch}{RomanNumeralXmpPagePatch5}
\begin{paste}{RomanNumeralXmpPageFull5}{RomanNumeralXmpPageEmpty5}
\pastebutton{RomanNumeralXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{recip(1 - x - x**2)\free{x }}
\indentrel{3}\begin{verbatim}
   (5)
                 2        3      4         5         6
     I + x + II x  + III x  + V x  + VIII x  + XIII x
   +
          7          8       9           10      11
     XXI x  + XXXIV x  + LV x  + LXXXIX x   + O(x  )
Type: Union(UnivariateTaylorSeries(RomanNumeral,x,0),...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RomanNumeralXmpPageEmpty5}
\begin{paste}{RomanNumeralXmpPageEmpty5}{RomanNumeralXmpPagePatch5}
\pastebutton{RomanNumeralXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{recip(1 - x - x**2)\free{x }}
\end{paste}\end{patch}

\begin{patch}{RomanNumeralXmpPagePatch6}
\begin{paste}{RomanNumeralXmpPageFull6}{RomanNumeralXmpPageEmpty6}
\pastebutton{RomanNumeralXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{m : MATRIX FRAC ROMAN\bound{m }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RomanNumeralXmpPageEmpty6}
\begin{paste}{RomanNumeralXmpPageEmpty6}{RomanNumeralXmpPagePatch6}
\pastebutton{RomanNumeralXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{m : MATRIX FRAC ROMAN\bound{m }}
\end{paste}\end{patch}

\begin{patch}{RomanNumeralXmpPagePatch7}
\begin{paste}{RomanNumeralXmpPageFull7}{RomanNumeralXmpPageEmpty7}
\pastebutton{RomanNumeralXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{m := matrix [[1/(i + j) for i in 1..3] for j in 1..3]\free{m }\bound{m1 }}
\indentrel{3}\begin{verbatim}
         I    I    I

        II   III  IV

         I    I   I
   (7)
        III  IV   V

         I    I    I

        IV    V   VI
                    Type: Matrix Fraction RomanNumeral
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RomanNumeralXmpPageEmpty7}
\begin{paste}{RomanNumeralXmpPageEmpty7}{RomanNumeralXmpPagePatch7}
\pastebutton{RomanNumeralXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{m := matrix [[1/(i + j) for i in 1..3] for j in 1..3]\free{m }\bound{m1 }}
\end{paste}\end{patch}

\begin{patch}{RomanNumeralXmpPagePatch8}
\begin{paste}{RomanNumeralXmpPageFull8}{RomanNumeralXmpPageEmpty8}
\pastebutton{RomanNumeralXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{inverse m\free{m1 }}
\indentrel{3}\begin{verbatim}
        LXXII   - CCXL    CLXXX

   (8)  - CCXL    CM     - DCCXX

        CLXXX   - DCCXX    DC
          Type: Union(Matrix Fraction RomanNumeral,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RomanNumeralXmpPageEmpty8}
\begin{paste}{RomanNumeralXmpPageEmpty8}{RomanNumeralXmpPagePatch8}
```

```
\pastebutton{RomanNumeralXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{inverse m\free{m1 }}
\end{paste}\end{patch}

\begin{patch}{RomanNumeralXmpPagePatch9}
\begin{paste}{RomanNumeralXmpPageFull9}{RomanNumeralXmpPageEmpty9}
\pastebutton{RomanNumeralXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{y := factorial 10\bound{y }}
\indentrel{3}\begin{verbatim}
   (9)  3628800
                                 Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RomanNumeralXmpPageEmpty9}
\begin{paste}{RomanNumeralXmpPageEmpty9}{RomanNumeralXmpPagePatch9}
\pastebutton{RomanNumeralXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{y := factorial 10\bound{y }}
\end{paste}\end{patch}

\begin{patch}{RomanNumeralXmpPagePatch10}
\begin{paste}{RomanNumeralXmpPageFull10}{RomanNumeralXmpPageEmpty10}
\pastebutton{RomanNumeralXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{roman y\free{y }}
\indentrel{3}\begin{verbatim}
   (10)
  ((((I))))((((I))))((((I)))) (((I)))(((I)))(((I)))(((I))
  )(((I)))(((I))) ((I))((I)) MMMMMMMMDCCC
                                    Type: RomanNumeral
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{RomanNumeralXmpPageEmpty10}
\begin{paste}{RomanNumeralXmpPageEmpty10}{RomanNumeralXmpPagePatch10}
\pastebutton{RomanNumeralXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{roman y\free{y }}
\end{paste}\end{patch}
```

3.95 seg.ht

Segment

⇒ "notitle" (SegmentBindingXmpPage) 3.96 on page 1224

⇒ “notitle” (UniversalSegmentXmpPage) 3.111 on page 1322

— **seg.ht** —

```
\begin{page}{SegmentXmpPage}{Segment}
\beginscroll

The \spadtype{Segment} domain provides a generalized interval type.

\labelSpace{2pc}
\xtc{
Segments are created using the \spadSyntax{..} construct
by indicating the (included) end points.
}{
\spadpaste{s := 3..10 \bound{s}}
}
\xtc{
The first end point is called the \spadfunFrom{lo}{Segment}
and the second is called \spadfunFrom{hi}{Segment}.
}{
\spadpaste{lo s \free{s}}
}
\xtc{
These names are used even though the end points might belong to an
unordered set.
}{
\spadpaste{hi s \free{s}}
}

\xtc{
In addition to the end points, each segment has an integer ‘‘increment.’’
An increment can be specified using the ‘‘\spad{by}’’ construct.
\spadkey{by}
}{
\spadpaste{t := 10..3 by -2 \bound{t}}
}
\xtc{
This part can be obtained using the \spadfunFrom{incr}{Segment}
function.
}{
\spadpaste{incr s \free{s}}
}
\xtc{
Unless otherwise specified, the increment is \spad{1}.
}{
\spadpaste{incr t \free{t}}
}

\xtc{
A single value can be converted to a segment with equal end points.
This happens if segments and single values are mixed in a list.
```

```
}{
\spadpaste{l := [1..3, 5, 9, 15..11 by -1] \bound{l}}
}

\xtc{
If the underlying type is an ordered ring, it is possible to perform
additional operations.
The \spadfunFrom{expand}{Segment} operation creates a list of
points in a segment.
}{
\spadpaste{expand s \free{s}}
}
\xtc{
If \spad{k > 0}, then \spad{expand(l..h by k)} creates the list
\spad{[l, l+k, ..., lN]} where \spad{lN <= h < lN+k}.
If \spad{k < 0}, then \spad{lN >= h > lN+k}.
}{
\spadpaste{expand t \free{t}}
}

\xtc{
It is also possible to expand a list of segments.  This is equivalent
to appending lists obtained by expanding each segment individually.
}{
\spadpaste{expand l \free{l}}
}

For more information on related topics, see
\downlink{`SegmentBinding'}{SegmentBindingXmpPage}\ignore{SegmentBinding}
and \downlink{`UniversalSegment'}{UniversalSegmentXmpPage}
\ignore{UniversalSegment}.
%
\showBlurb{Segment}
\endscroll
\autobuttons
\end{page}

\begin{patch}{SegmentXmpPagePatch1}
\begin{paste}{SegmentXmpPageFull1}{SegmentXmpPageEmpty1}
\pastebutton{SegmentXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{s := 3..10\bound{s }}
\indentrel{3}\begin{verbatim}
   (1)  3..10
                          Type: Segment PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SegmentXmpPageEmpty1}
\begin{paste}{SegmentXmpPageEmpty1}{SegmentXmpPagePatch1}
\pastebutton{SegmentXmpPageEmpty1}{\showpaste}
```

```
\tab{5}\spadcommand{s := 3..10\bound{s }}
\end{paste}\end{patch}

\begin{patch}{SegmentXmpPagePatch2}
\begin{paste}{SegmentXmpPageFull2}{SegmentXmpPageEmpty2}
\pastebutton{SegmentXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{lo s\free{s }}
\indentrel{3}\begin{verbatim}
   (2)  3
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SegmentXmpPageEmpty2}
\begin{paste}{SegmentXmpPageEmpty2}{SegmentXmpPagePatch2}
\pastebutton{SegmentXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{lo s\free{s }}
\end{paste}\end{patch}

\begin{patch}{SegmentXmpPagePatch3}
\begin{paste}{SegmentXmpPageFull3}{SegmentXmpPageEmpty3}
\pastebutton{SegmentXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{hi s\free{s }}
\indentrel{3}\begin{verbatim}
   (3)  10
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SegmentXmpPageEmpty3}
\begin{paste}{SegmentXmpPageEmpty3}{SegmentXmpPagePatch3}
\pastebutton{SegmentXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{hi s\free{s }}
\end{paste}\end{patch}

\begin{patch}{SegmentXmpPagePatch4}
\begin{paste}{SegmentXmpPageFull4}{SegmentXmpPageEmpty4}
\pastebutton{SegmentXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{t := 10..3 by -2\bound{t }}
\indentrel{3}\begin{verbatim}
   (4)  10..3 by - 2
                          Type: Segment PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SegmentXmpPageEmpty4}
\begin{paste}{SegmentXmpPageEmpty4}{SegmentXmpPagePatch4}
\pastebutton{SegmentXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{t := 10..3 by -2\bound{t }}
\end{paste}\end{patch}
```

```
\begin{patch}{SegmentXmpPagePatch5}
\begin{paste}{SegmentXmpPageFull5}{SegmentXmpPageEmpty5}
\pastebutton{SegmentXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{incr s\free{s }}
\indentrel{3}\begin{verbatim}
   (5)  1
                                 Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SegmentXmpPageEmpty5}
\begin{paste}{SegmentXmpPageEmpty5}{SegmentXmpPagePatch5}
\pastebutton{SegmentXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{incr s\free{s }}
\end{paste}\end{patch}

\begin{patch}{SegmentXmpPagePatch6}
\begin{paste}{SegmentXmpPageFull6}{SegmentXmpPageEmpty6}
\pastebutton{SegmentXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{incr t\free{t }}
\indentrel{3}\begin{verbatim}
   (6)  - 2
                                         Type: Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SegmentXmpPageEmpty6}
\begin{paste}{SegmentXmpPageEmpty6}{SegmentXmpPagePatch6}
\pastebutton{SegmentXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{incr t\free{t }}
\end{paste}\end{patch}

\begin{patch}{SegmentXmpPagePatch7}
\begin{paste}{SegmentXmpPageFull7}{SegmentXmpPageEmpty7}
\pastebutton{SegmentXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{l := [1..3, 5, 9, 15..11 by -1]\bound{l }}
\indentrel{3}\begin{verbatim}
   (7)  [1..3,5..5,9..9,15..11 by - 1]
                     Type: List Segment PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SegmentXmpPageEmpty7}
\begin{paste}{SegmentXmpPageEmpty7}{SegmentXmpPagePatch7}
\pastebutton{SegmentXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{l := [1..3, 5, 9, 15..11 by -1]\bound{l }}
\end{paste}\end{patch}

\begin{patch}{SegmentXmpPagePatch8}
```

```
\begin{paste}{SegmentXmpPageFull8}{SegmentXmpPageEmpty8}
\pastebutton{SegmentXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{expand s\free{s }}
\indentrel{3}\begin{verbatim}
   (8)  [3,4,5,6,7,8,9,10]
                                    Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SegmentXmpPageEmpty8}
\begin{paste}{SegmentXmpPageEmpty8}{SegmentXmpPagePatch8}
\pastebutton{SegmentXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{expand s\free{s }}
\end{paste}\end{patch}

\begin{patch}{SegmentXmpPagePatch9}
\begin{paste}{SegmentXmpPageFull9}{SegmentXmpPageEmpty9}
\pastebutton{SegmentXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{expand t\free{t }}
\indentrel{3}\begin{verbatim}
   (9)  [10,8,6,4]
                                    Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SegmentXmpPageEmpty9}
\begin{paste}{SegmentXmpPageEmpty9}{SegmentXmpPagePatch9}
\pastebutton{SegmentXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{expand t\free{t }}
\end{paste}\end{patch}

\begin{patch}{SegmentXmpPagePatch10}
\begin{paste}{SegmentXmpPageFull10}{SegmentXmpPageEmpty10}
\pastebutton{SegmentXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{expand l\free{l }}
\indentrel{3}\begin{verbatim}
   (10)  [1,2,3,5,9,15,14,13,12,11]
                                    Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SegmentXmpPageEmpty10}
\begin{paste}{SegmentXmpPageEmpty10}{SegmentXmpPagePatch10}
\pastebutton{SegmentXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{expand l\free{l }}
\end{paste}\end{patch}
```

3.96 segbind.ht

SegmentBinding

⇒ "notitle" (SegmentXmpPage) 3.95 on page 1218
⇒ "notitle" (UniversalSegmentXmpPage) 3.111 on page 1322

— **segbind.ht** —

```
\begin{page}{SegmentBindingXmpPage}{SegmentBinding}
\beginscroll

The \spadtype{SegmentBinding} type is used
to indicate a range for a named symbol.

\labelSpace{2pc}
\xtc{
First give the symbol, then an \spadSyntax{=} and finally a
segment of values.
}{
\spadpaste{x = a..b}
}
\xtc{
This is used to provide a convenient
syntax for arguments to certain operations.
}{
\spadpaste{sum(i**2, i = 0..n)}
}
\graphpaste{draw(x**2, x = -2..2)}

\xtc{
The left-hand side must be of type \spadtype{Symbol} but the
right-hand side can be a segment over any type.
}{
\spadpaste{sb := y = 1/2..3/2 \bound{sb}}
}
\xtc{
The left- and right-hand sides can be obtained using the
\spadfunFrom{variable}{SegmentBinding} and
\spadfunFrom{segment}{SegmentBinding} operations.
}{
\spadpaste{variable(sb) \free{sb}}
}
\xtc{
}{
\spadpaste{segment(sb)  \free{sb}}
}

For more information on related topics, see
```

```
\downlink{`Segment'}{SegmentXmpPage}\ignore{Segment} and
\downlink{`UniversalSegment'}{UniversalSegmentXmpPage}
\ignore{UniversalSegment}.
%
\showBlurb{SegmentBinding}
\endscroll
\autobuttons
\end{page}

\begin{patch}{SegmentBindingXmpPagePatch1}
\begin{paste}{SegmentBindingXmpPageFull1}{SegmentBindingXmpPageEmpty1}
\pastebutton{SegmentBindingXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{x = a..b}
\indentrel{3}\begin{verbatim}
   (1)  x= a..b
                            Type: SegmentBinding Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SegmentBindingXmpPageEmpty1}
\begin{paste}{SegmentBindingXmpPageEmpty1}{SegmentBindingXmpPagePatch1}
\pastebutton{SegmentBindingXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{x = a..b}
\end{paste}\end{patch}

\begin{patch}{SegmentBindingXmpPagePatch2}
\begin{paste}{SegmentBindingXmpPageFull2}{SegmentBindingXmpPageEmpty2}
\pastebutton{SegmentBindingXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{sum(i**2, i = 0..n)}
\indentrel{3}\begin{verbatim}
          3     2
        2n  + 3n  + n
   (2)
              6
                      Type: Fraction Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SegmentBindingXmpPageEmpty2}
\begin{paste}{SegmentBindingXmpPageEmpty2}{SegmentBindingXmpPagePatch2}
\pastebutton{SegmentBindingXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{sum(i**2, i = 0..n)}
\end{paste}\end{patch}

\begin{patch}{SegmentBindingXmpPagePatch3}
\begin{paste}{SegmentBindingXmpPageFull3}{SegmentBindingXmpPageEmpty3}
\pastebutton{SegmentBindingXmpPageFull3}{\hidepaste}
\tab{5}\spadgraph{draw(x**2, x = -2..2)}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/segmentbindingxmppage3.view/image}}{viewalone\spa
\end{paste}\end{patch}
```

```
\begin{patch}{SegmentBindingXmpPageEmpty3}
\begin{paste}{SegmentBindingXmpPageEmpty3}{SegmentBindingXmpPagePatch3}
\pastebutton{SegmentBindingXmpPageEmpty3}{\showpaste}
\tab{5}\spadgraph{draw(x**2, x = -2..2)}
\end{paste}\end{patch}

\begin{patch}{SegmentBindingXmpPagePatch4}
\begin{paste}{SegmentBindingXmpPageFull4}{SegmentBindingXmpPageEmpty4}
\pastebutton{SegmentBindingXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{sb := y = 1/2..3/2\bound{sb }}
\indentrel{3}\begin{verbatim}
           1    3
   (4)  y= ()..()
           2    2
                 Type: SegmentBinding Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SegmentBindingXmpPageEmpty4}
\begin{paste}{SegmentBindingXmpPageEmpty4}{SegmentBindingXmpPagePatch4}
\pastebutton{SegmentBindingXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{sb := y = 1/2..3/2\bound{sb }}
\end{paste}\end{patch}

\begin{patch}{SegmentBindingXmpPagePatch5}
\begin{paste}{SegmentBindingXmpPageFull5}{SegmentBindingXmpPageEmpty5}
\pastebutton{SegmentBindingXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{variable(sb)\free{sb }}
\indentrel{3}\begin{verbatim}
   (5)  y
                                         Type: Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SegmentBindingXmpPageEmpty5}
\begin{paste}{SegmentBindingXmpPageEmpty5}{SegmentBindingXmpPagePatch5}
\pastebutton{SegmentBindingXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{variable(sb)\free{sb }}
\end{paste}\end{patch}

\begin{patch}{SegmentBindingXmpPagePatch6}
\begin{paste}{SegmentBindingXmpPageFull6}{SegmentBindingXmpPageEmpty6}
\pastebutton{SegmentBindingXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{segment(sb)\free{sb }}
\indentrel{3}\begin{verbatim}
        1    3
   (6)  ()..()
        2    2
                        Type: Segment Fraction Integer
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SegmentBindingXmpPageEmpty6}
\begin{paste}{SegmentBindingXmpPageEmpty6}{SegmentBindingXmpPagePatch6}
\pastebutton{SegmentBindingXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{segment(sb)\free{sb }}
\end{paste}\end{patch}
```

———-

3.97 set.ht

Set

⇒ "notitle" (ListXmpPage) 3.64 on page 866

— **set.ht** —

```
\begin{page}{SetXmpPage}{Set}
\beginscroll

The \spadtype{Set} domain allows one to represent explicit finite
sets of values.
These are similar to lists, but duplicate elements are not allowed.
\xtc{
Sets can be created by giving a fixed set of values \ldots
}{
\spadpaste{s := set [x**2-1, y**2-1, z**2-1] \bound{s}}
}
\xtc{
or by using a collect form, just as for lists.
In either case, the set is formed from a finite collection of values.
}{
\spadpaste{t := set [x**i - i+1 for i in 2..10 | prime? i] \bound{t}}
}

\xtc{
The basic operations on sets are
\spadfunFrom{intersect}{Set}, \spadfunFrom{union}{Set},
\spadfunFrom{difference}{Set},
and \spadfunFrom{symmetricDifference}{Set}.
}{
\spadpaste{i := intersect(s,t)      \free{s t}\bound{i}}
}
\xtc{
}{
```

```
\spadpaste{u := union(s,t)          \free{s t}\bound{u}}
}
\xtc{
The set \spad{difference(s,t)} contains those members of \spad{s} which
are not in \spad{t}.
}{
\spadpaste{difference(s,t)          \free{s t}}
}
\xtc{
The set \spad{symmetricDifference(s,t)} contains those elements which are
in \spad{s} or \spad{t} but not in both.
}{
\spadpaste{symmetricDifference(s,t)          \free{s t}}
}

\xtc{
Set membership is tested using the \spadfunFrom{member?}{Set} operation.
}{
\spadpaste{member?(y, s)               \free{s}}
}
\xtc{
}{
\spadpaste{member?((y+1)*(y-1), s)     \free{s}}
}
\xtc{
The \spadfunFrom{subset?}{Set} function determines whether one set
is a subset of another.
}{
\spadpaste{subset?(i, s)               \free{i s}}
}
\xtc{
}{
\spadpaste{subset?(u, s)               \free{u s}}
}

\xtc{
When the base type is finite, the absolute complement of a set is
defined.
This finds the set of all multiplicative generators of
\spadtype{PrimeField 11}---the integers mod \spad{11.}
}{
\spadpaste{gs := set [g for i in 1..11 | primitive?(g := i::PF 11)]
\bound{gs}}
}
\xtc{
The following values are not generators.
}{
\spadpaste{complement gs \free{gs}}
}
```

```
Often the members of a set are computed individually; in addition,
values can be inserted or removed from a set over the course of a
computation.
\xtc{
There are two ways to do this:
}{
\spadpaste{a := set [i**2 for i in 1..5] \bound{a}}
}
\xtc{
One is to view a set as a data structure and to apply updating
operations.
}{
\spadpaste{insert!(32, a) \free{a}\bound{ainsert}}
}
\xtc{
}{
\spadpaste{remove!(25, a) \free{a}\bound{aremove}}
}
\xtc{
}{
\spadpaste{a \free{aremove ainsert}}
}
\xtc{
The other way is to view a set as a mathematical entity and to
create new sets from old.
}{
\spadpaste{b := b0 := set [i**2 for i in 1..5] \bound{b}}
}
\xtc{
}{
\spadpaste{b := union(b, {32})         \free{b}\bound{binsert}}
}
\xtc{
}{
\spadpaste{b := difference(b, {25})    \free{binsert}\bound{bremove}}
}
\xtc{
}{
\spadpaste{b0 \free{bremove}}
}

For more information about lists, see
\downlink{`List'}{ListXmpPage}\ignore{List}.
\showBlurb{Set}
\endscroll
\autobuttons
\end{page}

\begin{patch}{SetXmpPagePatch1}
\begin{paste}{SetXmpPageFull1}{SetXmpPageEmpty1}
```

```
\pastebutton{SetXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{s := set [x**2-1, y**2-1, z**2-1]\bound{s }}
\indentrel{3}\begin{verbatim}
          2      2      2
   (1)  {x  - 1,y  - 1,z  - 1}
                           Type: Set Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SetXmpPageEmpty1}
\begin{paste}{SetXmpPageEmpty1}{SetXmpPagePatch1}
\pastebutton{SetXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{s := set [x**2-1, y**2-1, z**2-1]\bound{s }}
\end{paste}\end{patch}

\begin{patch}{SetXmpPagePatch2}
\begin{paste}{SetXmpPageFull2}{SetXmpPageEmpty2}
\pastebutton{SetXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{t := set [x**i - i+1 for i in 2..10 | prime? i]\bound{t }}
\indentrel{3}\begin{verbatim}
          2      3      5      7
   (2)  {x  - 1,x  - 2,x  - 4,x  - 6}
                           Type: Set Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SetXmpPageEmpty2}
\begin{paste}{SetXmpPageEmpty2}{SetXmpPagePatch2}
\pastebutton{SetXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{t := set [x**i - i+1 for i in 2..10 | prime? i]\bound{t }}
\end{paste}\end{patch}

\begin{patch}{SetXmpPagePatch3}
\begin{paste}{SetXmpPageFull3}{SetXmpPageEmpty3}
\pastebutton{SetXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{i := intersect(s,t)\free{s t }\bound{i }}
\indentrel{3}\begin{verbatim}
          2
   (3)  {x  - 1}
                           Type: Set Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SetXmpPageEmpty3}
\begin{paste}{SetXmpPageEmpty3}{SetXmpPagePatch3}
\pastebutton{SetXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{i := intersect(s,t)\free{s t }\bound{i }}
\end{paste}\end{patch}

\begin{patch}{SetXmpPagePatch4}
```

```
\begin{paste}{SetXmpPageFull4}{SetXmpPageEmpty4}
\pastebutton{SetXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{u := union(s,t)\free{s t }\bound{u }}
\indentrel{3}\begin{verbatim}
          2      3      5      7      2      2
   (4)  {x  - 1,x  - 2,x  - 4,x  - 6,y  - 1,z  - 1}
                           Type: Set Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SetXmpPageEmpty4}
\begin{paste}{SetXmpPageEmpty4}{SetXmpPagePatch4}
\pastebutton{SetXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{u := union(s,t)\free{s t }\bound{u }}
\end{paste}\end{patch}

\begin{patch}{SetXmpPagePatch5}
\begin{paste}{SetXmpPageFull5}{SetXmpPageEmpty5}
\pastebutton{SetXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{difference(s,t)\free{s t }}
\indentrel{3}\begin{verbatim}
          2      2
   (5)  {y  - 1,z  - 1}
                           Type: Set Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SetXmpPageEmpty5}
\begin{paste}{SetXmpPageEmpty5}{SetXmpPagePatch5}
\pastebutton{SetXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{difference(s,t)\free{s t }}
\end{paste}\end{patch}

\begin{patch}{SetXmpPagePatch6}
\begin{paste}{SetXmpPageFull6}{SetXmpPageEmpty6}
\pastebutton{SetXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{symmetricDifference(s,t)\free{s t }}
\indentrel{3}\begin{verbatim}
          3      5      7      2      2
   (6)  {x  - 2,x  - 4,x  - 6,y  - 1,z  - 1}
                           Type: Set Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SetXmpPageEmpty6}
\begin{paste}{SetXmpPageEmpty6}{SetXmpPagePatch6}
\pastebutton{SetXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{symmetricDifference(s,t)\free{s t }}
\end{paste}\end{patch}
```

```
\begin{patch}{SetXmpPagePatch7}
\begin{paste}{SetXmpPageFull7}{SetXmpPageEmpty7}
\pastebutton{SetXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{member?(y, s)\free{s }}
\indentrel{3}\begin{verbatim}
   (7)  false
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SetXmpPageEmpty7}
\begin{paste}{SetXmpPageEmpty7}{SetXmpPagePatch7}
\pastebutton{SetXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{member?(y, s)\free{s }}
\end{paste}\end{patch}

\begin{patch}{SetXmpPagePatch8}
\begin{paste}{SetXmpPageFull8}{SetXmpPageEmpty8}
\pastebutton{SetXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{member?((y+1)*(y-1), s)\free{s }}
\indentrel{3}\begin{verbatim}
   (8)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SetXmpPageEmpty8}
\begin{paste}{SetXmpPageEmpty8}{SetXmpPagePatch8}
\pastebutton{SetXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{member?((y+1)*(y-1), s)\free{s }}
\end{paste}\end{patch}

\begin{patch}{SetXmpPagePatch9}
\begin{paste}{SetXmpPageFull9}{SetXmpPageEmpty9}
\pastebutton{SetXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{subset?(i, s)\free{i s }}
\indentrel{3}\begin{verbatim}
   (9)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SetXmpPageEmpty9}
\begin{paste}{SetXmpPageEmpty9}{SetXmpPagePatch9}
\pastebutton{SetXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{subset?(i, s)\free{i s }}
\end{paste}\end{patch}

\begin{patch}{SetXmpPagePatch10}
\begin{paste}{SetXmpPageFull10}{SetXmpPageEmpty10}
```

```
\pastebutton{SetXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{subset?(u, s)\free{u s }}
\indentrel{3}\begin{verbatim}
   (10)  false
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SetXmpPageEmpty10}
\begin{paste}{SetXmpPageEmpty10}{SetXmpPagePatch10}
\pastebutton{SetXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{subset?(u, s)\free{u s }}
\end{paste}\end{patch}

\begin{patch}{SetXmpPagePatch11}
\begin{paste}{SetXmpPageFull11}{SetXmpPageEmpty11}
\pastebutton{SetXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{gs := set [g for i in 1..11 | primitive?(g := i::PF 11)]\bound{gs }}
\indentrel{3}\begin{verbatim}
   (11)  {2,6,7,8}
                                Type: Set PrimeField 11
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SetXmpPageEmpty11}
\begin{paste}{SetXmpPageEmpty11}{SetXmpPagePatch11}
\pastebutton{SetXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{gs := set [g for i in 1..11 | primitive?(g := i::PF 11)]\bound{gs }}
\end{paste}\end{patch}

\begin{patch}{SetXmpPagePatch12}
\begin{paste}{SetXmpPageFull12}{SetXmpPageEmpty12}
\pastebutton{SetXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{complement gs\free{gs }}
\indentrel{3}\begin{verbatim}
   (12)  {1,3,4,5,9,10,0}
                                Type: Set PrimeField 11
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SetXmpPageEmpty12}
\begin{paste}{SetXmpPageEmpty12}{SetXmpPagePatch12}
\pastebutton{SetXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{complement gs\free{gs }}
\end{paste}\end{patch}

\begin{patch}{SetXmpPagePatch13}
\begin{paste}{SetXmpPageFull13}{SetXmpPageEmpty13}
\pastebutton{SetXmpPageFull13}{\hidepaste}
\tab{5}\spadcommand{a := set [i**2 for i in 1..5]\bound{a }}
```

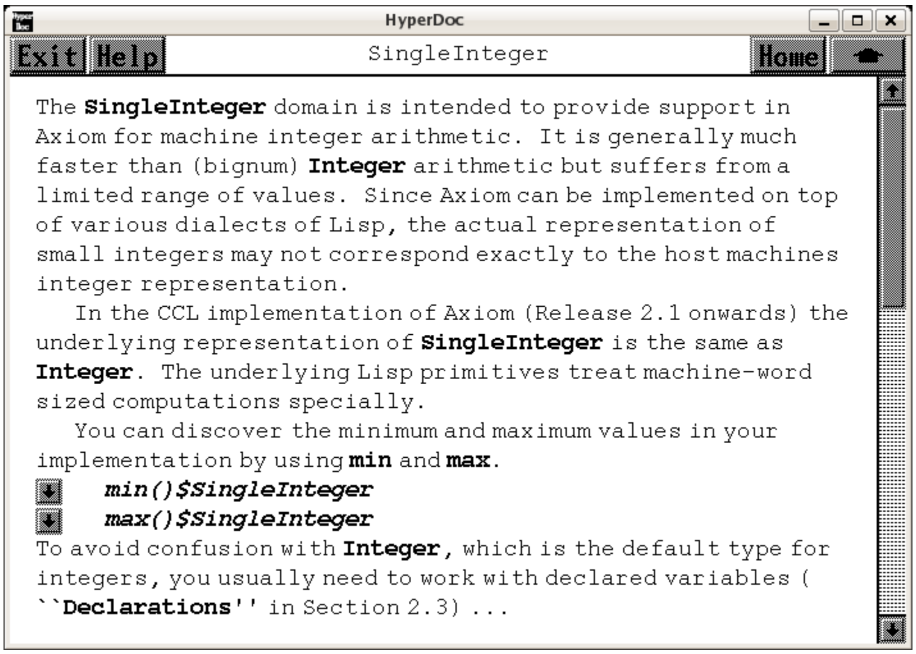
```
\indentrel{3}\begin{verbatim}
   (13)  {1,4,9,16,25}
                              Type: Set PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SetXmpPageEmpty13}
\begin{paste}{SetXmpPageEmpty13}{SetXmpPagePatch13}
\pastebutton{SetXmpPageEmpty13}{\showpaste}
\tab{5}\spadcommand{a := set [i**2 for i in 1..5]\bound{a }}
\end{paste}\end{patch}

\begin{patch}{SetXmpPagePatch14}
\begin{paste}{SetXmpPageFull14}{SetXmpPageEmpty14}
\pastebutton{SetXmpPageFull14}{\hidepaste}
\tab{5}\spadcommand{insert!(32, a)\free{a }\bound{ainsert }}
\indentrel{3}\begin{verbatim}
   (14)  {1,4,9,16,25,32}
                              Type: Set PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SetXmpPageEmpty14}
\begin{paste}{SetXmpPageEmpty14}{SetXmpPagePatch14}
\pastebutton{SetXmpPageEmpty14}{\showpaste}
\tab{5}\spadcommand{insert!(32, a)\free{a }\bound{ainsert }}
\end{paste}\end{patch}

\begin{patch}{SetXmpPagePatch15}
\begin{paste}{SetXmpPageFull15}{SetXmpPageEmpty15}
\pastebutton{SetXmpPageFull15}{\hidepaste}
\tab{5}\spadcommand{remove!(25, a)\free{a }\bound{aremove }}
\indentrel{3}\begin{verbatim}
   (15)  {1,4,9,16,32}
                              Type: Set PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SetXmpPageEmpty15}
\begin{paste}{SetXmpPageEmpty15}{SetXmpPagePatch15}
\pastebutton{SetXmpPageEmpty15}{\showpaste}
\tab{5}\spadcommand{remove!(25, a)\free{a }\bound{aremove }}
\end{paste}\end{patch}

\begin{patch}{SetXmpPagePatch16}
\begin{paste}{SetXmpPageFull16}{SetXmpPageEmpty16}
\pastebutton{SetXmpPageFull16}{\hidepaste}
\tab{5}\spadcommand{a\free{aremove ainsert }}
\indentrel{3}\begin{verbatim}
   (16)  {1,4,9,16,32}
```

```
                              Type: Set PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SetXmpPageEmpty16}
\begin{paste}{SetXmpPageEmpty16}{SetXmpPagePatch16}
\pastebutton{SetXmpPageEmpty16}{\showpaste}
\tab{5}\spadcommand{a\free{aremove ainsert }}
\end{paste}\end{patch}

\begin{patch}{SetXmpPagePatch17}
\begin{paste}{SetXmpPageFull17}{SetXmpPageEmpty17}
\pastebutton{SetXmpPageFull17}{\hidepaste}
\tab{5}\spadcommand{b := b0 := set [i**2 for i in 1..5]\bound{b }}
\indentrel{3}\begin{verbatim}
   (17)  {1,4,9,16,25}
                              Type: Set PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SetXmpPageEmpty17}
\begin{paste}{SetXmpPageEmpty17}{SetXmpPagePatch17}
\pastebutton{SetXmpPageEmpty17}{\showpaste}
\tab{5}\spadcommand{b := b0 := set [i**2 for i in 1..5]\bound{b }}
\end{paste}\end{patch}

\begin{patch}{SetXmpPagePatch18}
\begin{paste}{SetXmpPageFull18}{SetXmpPageEmpty18}
\pastebutton{SetXmpPageFull18}{\hidepaste}
\tab{5}\spadcommand{b := union(b,32)\free{b }\bound{binsert }}
\indentrel{3}\begin{verbatim}
   (18)  {1,4,9,16,25,32}
                              Type: Set PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SetXmpPageEmpty18}
\begin{paste}{SetXmpPageEmpty18}{SetXmpPagePatch18}
\pastebutton{SetXmpPageEmpty18}{\showpaste}
\tab{5}\spadcommand{b := union(b,32)\free{b }\bound{binsert }}
\end{paste}\end{patch}

\begin{patch}{SetXmpPagePatch19}
\begin{paste}{SetXmpPageFull19}{SetXmpPageEmpty19}
\pastebutton{SetXmpPageFull19}{\hidepaste}
\tab{5}\spadcommand{b := difference(b,25)\free{binsert }\bound{bremove }}
\indentrel{3}\begin{verbatim}
   (19)  {1,4,9,16,32}
                              Type: Set PositiveInteger
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SetXmpPageEmpty19}
\begin{paste}{SetXmpPageEmpty19}{SetXmpPagePatch19}
\pastebutton{SetXmpPageEmpty19}{\showpaste}
\tab{5}\spadcommand{b := difference(b,25)\free{binsert }\bound{bremove }}
\end{paste}\end{patch}

\begin{patch}{SetXmpPagePatch20}
\begin{paste}{SetXmpPageFull20}{SetXmpPageEmpty20}
\pastebutton{SetXmpPageFull20}{\hidepaste}
\tab{5}\spadcommand{b0\free{bremove }}
\indentrel{3}\begin{verbatim}
   (20)  {1,4,9,16,25}
                              Type: Set PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SetXmpPageEmpty20}
\begin{paste}{SetXmpPageEmpty20}{SetXmpPagePatch20}
\pastebutton{SetXmpPageEmpty20}{\showpaste}
\tab{5}\spadcommand{b0\free{bremove }}
\end{paste}\end{patch}
```

3.98 sint.ht

SingleInteger



⇐ “Axiom Number Types” (NumberPage) 3.80 on page 1021
⇒ “Declarations” (ugTypesDeclarePage) 7 on page 1641
⇒ “Package Calling and Target Types” (ugTypesPkgCallPage) 7 on page 1686
⇒ “Browse” (ugBrowsePage) 17 on page 2471

— **sint.ht** —

```
\begin{page}{SingleIntegerXmpPage}{SingleInteger}
\beginscroll
The \axiomType{SingleInteger} domain is intended to provide support in
Axiom for machine integer arithmetic.  It is generally much faster
than (bignum) \axiomType{Integer} arithmetic but suffers from a
limited range of values.  Since Axiom can be implemented on top of
various dialects of Lisp, the actual representation of small integers
may not correspond exactly to the host machines integer
representation.

In the CCL implementation of Axiom (Release 2.1 onwards) the
underlying representation of \axiomType{SingleInteger} is the same as
\axiomType{Integer}.  The underlying Lisp primitives treat
machine-word sized computations specially.
```

```
\xtc{
You can discover the minimum and maximum values in your implementation
by using \spadfunFrom{min}{SingleInteger} and \spadfunFrom{max}{SingleInteger}.
}{
\spadpaste{min()\$SingleInteger}
}
\xtc{
}{
\spadpaste{max()\$SingleInteger}
}
\xtc{
To avoid confusion with \axiomType{Integer}, which is the default
type for integers, you usually need to work with declared variables
(\downlink{``Declarations''}{ugTypesDeclarePage}
in Section 2.3\ignore{ugTypesDeclare}) \ldots
}{
\spadpaste{a := 1234 :: SingleInteger \bound{a}}
}
\xtc{
or use package calling
(\downlink{``Package Calling and Target Types''}{ugTypesPkgCallPage}
in Section 2.9\ignore{ugTypesPkgCall}).
}{
\spadpaste{b := 124\$SingleInteger \bound{b}}
}
\xtc{
You can add, multiply and subtract \axiomType{SingleInteger} objects,
and ask for the greatest common divisor (\spadfun{gcd}).
}{
\spadpaste{gcd(a,b) \free{a}\free{b}}
}
\xtc{
The least common multiple (\spadfun{lcm}) is also available.
}{
\spadpaste{lcm(a,b) \free{a}\free{b}}
}

\xtc{
Operations \spadfunFrom{mulmod}{SingleInteger},
\spadfunFrom{addmod}{SingleInteger},
\spadfunFrom{submod}{SingleInteger}, and
\spadfunFrom{invmod}{SingleInteger} are similar---they provide
arithmetic modulo a given small integer.
Here is \spad{5 * 6 {\tt mod} 13}.
}{
\spadpaste{mulmod(5,6,13)\$SingleInteger}
}
\xtc{
To reduce a small integer modulo a prime, use
\spadfunFrom{positiveRemainder}{SingleInteger}.
```

```
}{
\spadpaste{positiveRemainder(37,13)\$SingleInteger}
}
\xtc{
Operations
\spadfunFrom{And}{SingleInteger},
\spadfunFrom{Or}{SingleInteger},
\spadfunFrom{xor}{SingleInteger},
and \spadfunFrom{Not}{SingleInteger}
provide bit level operations on small integers.
}{
\spadpaste{And(3,4)\$SingleInteger}
}
\xtc{
Use
\spad{shift(int,numToShift)} to shift bits, where
\spad{i} is shifted left if \spad{numToShift} is positive, right
if negative.
}{
\spadpaste{shift(1,4)\$SingleInteger}
}
\xtc{
}{
\spadpaste{shift(31,-1)\$SingleInteger}
}

Many other operations are available for small integers, including
many of those provided for \axiomType{Integer}.
To see the other operations, use the Browse Hyperdoc facility
(\downlink{``Browse''}{ugBrowsePage} in Section 14\ignore{ugBrowse}).
\showBlurb{SingleInteger}.
\endscroll
\autobuttons
\end{page}

\begin{patch}{SingleIntegerXmpPagePatch1}
\begin{paste}{SingleIntegerXmpPageFull1}{SingleIntegerXmpPageEmpty1}
\pastebutton{SingleIntegerXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{min()$SingleInteger}
\indentrel{3}\begin{verbatim}
   (1)  - 134217728
                                  Type: SingleInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SingleIntegerXmpPageEmpty1}
\begin{paste}{SingleIntegerXmpPageEmpty1}{SingleIntegerXmpPagePatch1}
\pastebutton{SingleIntegerXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{min()$SingleInteger}
\end{paste}\end{patch}
```

```
\begin{patch}{SingleIntegerXmpPagePatch2}
\begin{paste}{SingleIntegerXmpPageFull2}{SingleIntegerXmpPageEmpty2}
\pastebutton{SingleIntegerXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{max()$SingleInteger}
\indentrel{3}\begin{verbatim}
   (2)  134217727
                                    Type: SingleInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SingleIntegerXmpPageEmpty2}
\begin{paste}{SingleIntegerXmpPageEmpty2}{SingleIntegerXmpPagePatch2}
\pastebutton{SingleIntegerXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{max()$SingleInteger}
\end{paste}\end{patch}

\begin{patch}{SingleIntegerXmpPagePatch3}
\begin{paste}{SingleIntegerXmpPageFull3}{SingleIntegerXmpPageEmpty3}
\pastebutton{SingleIntegerXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{a := 1234 :: SingleInteger\bound{a }}
\indentrel{3}\begin{verbatim}
   (3)  1234
                                    Type: SingleInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SingleIntegerXmpPageEmpty3}
\begin{paste}{SingleIntegerXmpPageEmpty3}{SingleIntegerXmpPagePatch3}
\pastebutton{SingleIntegerXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{a := 1234 :: SingleInteger\bound{a }}
\end{paste}\end{patch}

\begin{patch}{SingleIntegerXmpPagePatch4}
\begin{paste}{SingleIntegerXmpPageFull4}{SingleIntegerXmpPageEmpty4}
\pastebutton{SingleIntegerXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{b := 124$SingleInteger\bound{b }}
\indentrel{3}\begin{verbatim}
   (4)  124
                                    Type: SingleInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SingleIntegerXmpPageEmpty4}
\begin{paste}{SingleIntegerXmpPageEmpty4}{SingleIntegerXmpPagePatch4}
\pastebutton{SingleIntegerXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{b := 124$SingleInteger\bound{b }}
\end{paste}\end{patch}

\begin{patch}{SingleIntegerXmpPagePatch5}
```

```
\begin{paste}{SingleIntegerXmpPageFull5}{SingleIntegerXmpPageEmpty5}
\pastebutton{SingleIntegerXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{gcd(a,b)\free{a }\free{b }}
\indentrel{3}\begin{verbatim}
   (5)  2
                                    Type: SingleInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SingleIntegerXmpPageEmpty5}
\begin{paste}{SingleIntegerXmpPageEmpty5}{SingleIntegerXmpPagePatch5}
\pastebutton{SingleIntegerXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{gcd(a,b)\free{a }\free{b }}
\end{paste}\end{patch}

\begin{patch}{SingleIntegerXmpPagePatch6}
\begin{paste}{SingleIntegerXmpPageFull6}{SingleIntegerXmpPageEmpty6}
\pastebutton{SingleIntegerXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{lcm(a,b)\free{a }\free{b }}
\indentrel{3}\begin{verbatim}
   (6)  76508
                                    Type: SingleInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SingleIntegerXmpPageEmpty6}
\begin{paste}{SingleIntegerXmpPageEmpty6}{SingleIntegerXmpPagePatch6}
\pastebutton{SingleIntegerXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{lcm(a,b)\free{a }\free{b }}
\end{paste}\end{patch}

\begin{patch}{SingleIntegerXmpPagePatch7}
\begin{paste}{SingleIntegerXmpPageFull7}{SingleIntegerXmpPageEmpty7}
\pastebutton{SingleIntegerXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{mulmod(5,6,13)$SingleInteger}
\indentrel{3}\begin{verbatim}
   (7)  4
                                    Type: SingleInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SingleIntegerXmpPageEmpty7}
\begin{paste}{SingleIntegerXmpPageEmpty7}{SingleIntegerXmpPagePatch7}
\pastebutton{SingleIntegerXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{mulmod(5,6,13)$SingleInteger}
\end{paste}\end{patch}

\begin{patch}{SingleIntegerXmpPagePatch8}
\begin{paste}{SingleIntegerXmpPageFull8}{SingleIntegerXmpPageEmpty8}
\pastebutton{SingleIntegerXmpPageFull8}{\hidepaste}
```

```
\tab{5}\spadcommand{positiveRemainder(37,13)$SingleInteger}
\indentrel{3}\begin{verbatim}
   (8)  11
                                    Type: SingleInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SingleIntegerXmpPageEmpty8}
\begin{paste}{SingleIntegerXmpPageEmpty8}{SingleIntegerXmpPagePatch8}
\pastebutton{SingleIntegerXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{positiveRemainder(37,13)$SingleInteger}
\end{paste}\end{patch}

\begin{patch}{SingleIntegerXmpPagePatch9}
\begin{paste}{SingleIntegerXmpPageFull9}{SingleIntegerXmpPageEmpty9}
\pastebutton{SingleIntegerXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{And(3,4)$SingleInteger}
\indentrel{3}\begin{verbatim}
   (9)  0
                                    Type: SingleInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SingleIntegerXmpPageEmpty9}
\begin{paste}{SingleIntegerXmpPageEmpty9}{SingleIntegerXmpPagePatch9}
\pastebutton{SingleIntegerXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{And(3,4)$SingleInteger}
\end{paste}\end{patch}

\begin{patch}{SingleIntegerXmpPagePatch10}
\begin{paste}{SingleIntegerXmpPageFull10}{SingleIntegerXmpPageEmpty10}
\pastebutton{SingleIntegerXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{shift(1,4)$SingleInteger}
\indentrel{3}\begin{verbatim}
   (10)  16
                                    Type: SingleInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SingleIntegerXmpPageEmpty10}
\begin{paste}{SingleIntegerXmpPageEmpty10}{SingleIntegerXmpPagePatch10}
\pastebutton{SingleIntegerXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{shift(1,4)$SingleInteger}
\end{paste}\end{patch}

\begin{patch}{SingleIntegerXmpPagePatch11}
\begin{paste}{SingleIntegerXmpPageFull11}{SingleIntegerXmpPageEmpty11}
\pastebutton{SingleIntegerXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{shift(31,-1)$SingleInteger}
\indentrel{3}\begin{verbatim}
```

```
   (11)  15
                                       Type: SingleInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SingleIntegerXmpPageEmpty11}
\begin{paste}{SingleIntegerXmpPageEmpty11}{SingleIntegerXmpPagePatch11}
\pastebutton{SingleIntegerXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{shift(31,-1)$SingleInteger}
\end{paste}\end{patch}
```

———-

3.99 sqmatrix.ht

SquareMatrix

⇒ "notitle" (MatrixXmpPage) 3.75 on page 984
⇒ "notitle" (ugTypesWritingModesPage) 7 on page 1637
⇒ "notitle" (ugTypesExposePage) 7 on page 1699

— **sqmatrix.ht** —

```
\begin{page}{SqMatrixXmpPage}{SquareMatrix}
\beginscroll

The top level matrix type in Axiom is \spadtype{Matrix} (see
\downlink{`Matrix'}{MatrixXmpPage}\ignore{Matrix}), which provides
basic arithmetic and linear algebra functions.  However, since the
matrices can be of any size it is not true that any pair can be added
or multiplied.  Thus \spadtype{Matrix} has little algebraic structure.

Sometimes you want to use matrices as coefficients for polynomials or
in other algebraic contexts.  In this case, \spadtype{SquareMatrix}
should be used.  The domain \spadtype{SquareMatrix(n,R)} gives the
ring of \spad{n} by \spad{n} square matrices over \spad{R}.

\xtc{
Since \spadtype{SquareMatrix} is not normally exposed at the top level,
you must expose it before it can be used.
}{
\spadpaste{)set expose add constructor SquareMatrix \bound{SQ}}
}
\xtc{
Once \spad{SQMATRIX} has been exposed,
values can be created using the \spadfunFrom{squareMatrix}{SquareMatrix}
```

```
function.
}{
\spadpaste{m := squareMatrix [[1,-\%i],[\%i,4]] \bound{m}\free{SQ}}
}
\xtc{
The usual arithmetic operations are available.
}{
\spadpaste{m*m - m \free{m}}
}
\xtc{
Square matrices can be used where ring elements are required.
For example, here is a matrix with matrix entries.
}{
\spadpaste{mm := squareMatrix [[m, 1], [1-m, m**2]] \free{m}\bound{mm}}
}
\xtc{
Or you can construct a polynomial with  square matrix coefficients.
}{
\spadpaste{p := (x + m)**2 \free{m}\bound{p}}
}
\xtc{
This value can be converted to a square matrix with polynomial
coefficients.
}{
\spadpaste{p::SquareMatrix(2, ?) \free{p}}
}

For more information on related topics, see
\downlink{``Modes''}{ugTypesWritingModesPage}
in Section 2.2.4\ignore{ugTypesWritingModes},
\downlink{``Exposing Domains and Packages''}{ugTypesExposePage}
in Section 2.11\ignore{ugTypesExpose}, and
\downlink{`Matrix'}{MatrixXmpPage}\ignore{Matrix}.
\showBlurb{SquareMatrix}
\endscroll
\autobuttons
\end{page}

\begin{patch}{SqMatrixXmpPagePatch1}
\begin{paste}{SqMatrixXmpPageFull1}{SqMatrixXmpPageEmpty1}
\pastebutton{SqMatrixXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{)set expose add constructor SquareMatrix\bound{SQ }}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SqMatrixXmpPageEmpty1}
\begin{paste}{SqMatrixXmpPageEmpty1}{SqMatrixXmpPagePatch1}
\pastebutton{SqMatrixXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{)set expose add constructor SquareMatrix\bound{SQ }}
```

```
\end{paste}\end{patch}

\begin{patch}{SqMatrixXmpPagePatch2}
\begin{paste}{SqMatrixXmpPageFull2}{SqMatrixXmpPageEmpty2}
\pastebutton{SqMatrixXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{m := squareMatrix [[1,-\%i],[\%i,4]]\bound{m }\free{SQ }}
\indentrel{3}\begin{verbatim}
        1   - %i
   (1)
        %i   4
                  Type: SquareMatrix(2,Complex Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SqMatrixXmpPageEmpty2}
\begin{paste}{SqMatrixXmpPageEmpty2}{SqMatrixXmpPagePatch2}
\pastebutton{SqMatrixXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{m := squareMatrix [[1,-\%i],[\%i,4]]\bound{m }\free{SQ }}
\end{paste}\end{patch}

\begin{patch}{SqMatrixXmpPagePatch3}
\begin{paste}{SqMatrixXmpPageFull3}{SqMatrixXmpPageEmpty3}
\pastebutton{SqMatrixXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{m*m - m\free{m }}
\indentrel{3}\begin{verbatim}
         1   - 4%i
   (2)
        4%i   13
                  Type: SquareMatrix(2,Complex Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SqMatrixXmpPageEmpty3}
\begin{paste}{SqMatrixXmpPageEmpty3}{SqMatrixXmpPagePatch3}
\pastebutton{SqMatrixXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{m*m - m\free{m }}
\end{paste}\end{patch}

\begin{patch}{SqMatrixXmpPagePatch4}
\begin{paste}{SqMatrixXmpPageFull4}{SqMatrixXmpPageEmpty4}
\pastebutton{SqMatrixXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{mm := squareMatrix [[m, 1], [1-m, m**2]]\free{m }\bound{mm }}
\indentrel{3}\begin{verbatim}
        1   - %i      1  0

        %i   4        0  1
   (3)
         0    %i    2   - 5%i

        - %i  - 3  5%i   17
```

```
  Type: SquareMatrix(2,SquareMatrix(2,Complex Integer))
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SqMatrixXmpPageEmpty4}
\begin{paste}{SqMatrixXmpPageEmpty4}{SqMatrixXmpPagePatch4}
\pastebutton{SqMatrixXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{mm := squareMatrix [[m, 1], [1-m, m**2]]\free{m }\bound{mm }}
\end{paste}\end{patch}

\begin{patch}{SqMatrixXmpPagePatch5}
\begin{paste}{SqMatrixXmpPageFull5}{SqMatrixXmpPageEmpty5}
\pastebutton{SqMatrixXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{p := (x + m)**2\free{m }\bound{p }}
\indentrel{3}\begin{verbatim}
         2    2   - 2%i     2   - 5%i
   (4)  x  +           x +
             2%i    8      5%i   17
       Type: Polynomial SquareMatrix(2,Complex Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SqMatrixXmpPageEmpty5}
\begin{paste}{SqMatrixXmpPageEmpty5}{SqMatrixXmpPagePatch5}
\pastebutton{SqMatrixXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{p := (x + m)**2\free{m }\bound{p }}
\end{paste}\end{patch}

\begin{patch}{SqMatrixXmpPagePatch6}
\begin{paste}{SqMatrixXmpPageFull6}{SqMatrixXmpPageEmpty6}
\pastebutton{SqMatrixXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{p::SquareMatrix(2, ?)\free{p }}
\indentrel{3}\begin{verbatim}
         2
        x  + 2x + 2  - 2%i x - 5%i
   (5)
                         2
        2%i x + 5%i   x  + 8x + 17
       Type: SquareMatrix(2,Polynomial Complex Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SqMatrixXmpPageEmpty6}
\begin{paste}{SqMatrixXmpPageEmpty6}{SqMatrixXmpPagePatch6}
\pastebutton{SqMatrixXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{p::SquareMatrix(2, ?)\free{p }}
\end{paste}\end{patch}
```

3.100 sregset.ht

SquareFreeRegularTriangularSet

— sregset.ht —

```
\begin{page}{SqFreeRegTriangSetXmpPage}
{SquareFreeRegularTriangularSet}
\beginscroll

The \spadtype{SquareFreeRegularTriangularSet} domain constructor
implements square-free regular triangular sets.  See the
\spadtype{RegularTriangularSet} domain constructor for general regular
triangular sets.  Let {\em T} be a regular triangular set consisting
of polynomials {\em t1, ..., tm} ordered by increasing main variables.
The regular triangular set {\em T} is square-free if {\em T} is empty
or if {\em t1, ..., tm-1} is square-free and if the polynomial {\em
tm} is square-free as a univariate polynomial with coefficients in the
tower of simple extensions associated with {\em t1, ..., tm-1}.

The main interest of square-free regular triangular sets is that their
associated towers of simple extensions are product of fields.
Consequently, the saturated ideal of a square-free regular triangular
set is radical.  This property simplifies some of the operations
related to regular triangular sets.  However, building square-free
regular triangular sets is generally more expensive than building
general regular triangular sets.

As the \spadtype{RegularTriangularSet} domain constructor, the
\spadtype{SquareFreeRegularTriangularSet} domain constructor also
implements a method for solving polynomial systems by means of regular
triangular sets.  This is in fact the same method with some
adaptations to take into account the fact that the computed regular
chains are square-free.  Note that it is also possible to pass from a
decomposition into general regular triangular sets to a decomposition
into square-free regular triangular sets.  This conversion is used
internally by the \spadtype{LazardSetSolvingPackage} package
constructor.

{\bf N.B.} When solving polynomial systems with the
\spadtype{SquareFreeRegularTriangularSet} domain constructor or the
\spadtype{LazardSetSolvingPackage} package constructor, decompositions
have no redundant components.  See also
\spadtype{LexTriangularPackage} and
\spadtype{ZeroDimensionalSolvePackage} for the case of algebraic
systems with a finite number of (complex) solutions.

We shall explain now how to use the constructor
```

```
\spadtype{SquareFreeRegularTriangularSet}.

This constructor takes four arguments.  The first one, {\bf R}, is the
coefficient ring of the polynomials; it must belong to the category
\spadtype{GcdDomain}.  The second one, {\bf E}, is the exponent monoid
of the polynomials; it must belong to the category
\spadtype{OrderedAbelianMonoidSup}.  the third one, {\bf V}, is the
ordered set of variables; it must belong to the category
\spadtype{OrderedSet}.  The last one is the polynomial ring; it must
belong to the category \spadtype{RecursivePolynomialCategory(R,E,V)}.
The abbreviation for \spadtype{SquareFreeRegularTriangularSet} is
\spadtype{SREGSET}.

Note that the way of understanding triangular decompositions
is detailed in the example of the \spadtype{RegularTriangularSet}
constructor.

\xtc{
Let us illustrate the use of this constructor
with one example (Donati-Traverso).
Define the coefficient ring.
}{
\spadpaste{R := Integer \bound{R}}
}
\xtc{
Define the list of variables,
}{
\spadpaste{ls : List Symbol := [x,y,z,t] \bound{ls}}
}
\xtc{
and make it an ordered set;
}{
\spadpaste{V := OVAR(ls) \free{ls} \bound{V}}
}
\xtc{
then define the exponent monoid.
}{
\spadpaste{E := IndexedExponents V \free{V} \bound{E}}
}
\xtc{
Define the polynomial ring.
}{
\spadpaste{P := NSMP(R, V) \free{R} \free{V} \bound{P}}
}
\xtc{
Let the variables be polynomial.
}{
\spadpaste{x: P := 'x \free{P} \bound{x}}
}
\xtc{
```

```
}{
\spadpaste{y: P := 'y \free{P} \bound{y}}
}
\xtc{
}{
\spadpaste{z: P := 'z \free{P} \bound{z}}
}
\xtc{
}{
\spadpaste{t: P := 't \free{P} \bound{t}}
}
\xtc{
Now call the \spadtype{SquareFreeRegularTriangularSet} domain
constructor.
}{
\spadpaste{ST := SREGSET(R,E,V,P) \free{R} \free{E} \free{V}
\free{P} \bound{ST} }
}
\xtc{
Define a polynomial system.
}{
\spadpaste{p1 := x ** 31 - x ** 6 - x - y \free{x} \free{y}
\bound{p1}}
}
\xtc{
}{
\spadpaste{p2 := x ** 8  - z \free{x} \free{z} \bound{p2}}
}
\xtc{
}{
\spadpaste{p3 := x ** 10 - t \free{x} \free{t} \bound{p3}}
}
\xtc{
}{
\spadpaste{lp := [p1, p2, p3] \free{p1} \free{p2} \free{p3}
\bound{lp}}
}

\xtc{
First of all, let us solve this system in the sense of Kalkbrener.
}{
\spadpaste{zeroSetSplit(lp)$ST \free{lp} \free{ST}}
}
\xtc{
And now in the sense of Lazard (or Wu and other authors).
}{
\spadpaste{zeroSetSplit(lp,false)$ST \free{lp} \free{ST}
\bound{lts}}
}
```

```
Now to see the difference with the \spadtype{RegularTriangularSet}
domain constructor,
\xtc{
we define:
}{
\spadpaste{T := REGSET(R,E,V,P) \free{R} \free{E} \free{V}
\free{P} \bound{T} }
}
\xtc{
and compute:
}{
\spadpaste{lts := zeroSetSplit(lp,false)$T \free{lp}
\free{T} \bound{lts}}
}

If you look at the second set in both decompositions in the sense of
Lazard, you will see that the polynomial with main variable {\bf y} is
not the same.

Let us understand what has happened.
\xtc{
We define:
}{
\spadpaste{ts := lts.2  \free{lts}  \bound{ts}}
}
\xtc{
}{
\spadpaste{pol := select(ts,'y)$T \free{ts} \free{y} \free{T} \bound{pol}}
}
\xtc{
}{
\spadpaste{tower := collectUnder(ts,'y)$T \free{ts} \free{y}
\free{T} \bound{tower}}
}
\xtc{
}{
\spadpaste{pack := RegularTriangularSetGcdPackage(R,E,V,P,T)
\free{R} \free{E} \free{V} \free{P} \free{T} \bound{pack}}
}
\xtc{
Then we compute:
}{
\spadpaste{toseSquareFreePart(pol,tower)$pack \free{pol}
\free{tower} \free{pack}}
}
\endscroll
\autobuttons
\end{page}

\begin{patch}{SqFreeRegTriangSetXmpPagePatch1}
```

```
\begin{paste}{SqFreeRegTriangSetXmpPageFull1}{SqFreeRegTriangSetXmpPageEmpty1}
\pastebutton{SqFreeRegTriangSetXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{R := Integer\bound{R }}
\indentrel{3}\begin{verbatim}
   (1)  Integer
                                          Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPageEmpty1}
\begin{paste}{SqFreeRegTriangSetXmpPageEmpty1}{SqFreeRegTriangSetXmpPagePatch1}
\pastebutton{SqFreeRegTriangSetXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{R := Integer\bound{R }}
\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPagePatch2}
\begin{paste}{SqFreeRegTriangSetXmpPageFull2}{SqFreeRegTriangSetXmpPageEmpty2}
\pastebutton{SqFreeRegTriangSetXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{ls : List Symbol := [x,y,z,t]\bound{ls }}
\indentrel{3}\begin{verbatim}
   (2)  [x,y,z,t]
                                     Type: List Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPageEmpty2}
\begin{paste}{SqFreeRegTriangSetXmpPageEmpty2}{SqFreeRegTriangSetXmpPagePatch2}
\pastebutton{SqFreeRegTriangSetXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{ls : List Symbol := [x,y,z,t]\bound{ls }}
\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPagePatch3}
\begin{paste}{SqFreeRegTriangSetXmpPageFull3}{SqFreeRegTriangSetXmpPageEmpty3}
\pastebutton{SqFreeRegTriangSetXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{V := OVAR(ls)\free{ls }\bound{V }}
\indentrel{3}\begin{verbatim}
   (3)  OrderedVariableList [x,y,z,t]
                                          Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPageEmpty3}
\begin{paste}{SqFreeRegTriangSetXmpPageEmpty3}{SqFreeRegTriangSetXmpPagePatch3}
\pastebutton{SqFreeRegTriangSetXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{V := OVAR(ls)\free{ls }\bound{V }}
\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPagePatch4}
\begin{paste}{SqFreeRegTriangSetXmpPageFull4}{SqFreeRegTriangSetXmpPageEmpty4}
\pastebutton{SqFreeRegTriangSetXmpPageFull4}{\hidepaste}
```

```
\tab{5}\spadcommand{E := IndexedExponents V\free{V }\bound{E }}
\indentrel{3}\begin{verbatim}
   (4)  IndexedExponents OrderedVariableList [x,y,z,t]
                                           Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPageEmpty4}
\begin{paste}{SqFreeRegTriangSetXmpPageEmpty4}{SqFreeRegTriangSetXmpPagePatch4}
\pastebutton{SqFreeRegTriangSetXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{E := IndexedExponents V\free{V }\bound{E }}
\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPagePatch5}
\begin{paste}{SqFreeRegTriangSetXmpPageFull5}{SqFreeRegTriangSetXmpPageEmpty5}
\pastebutton{SqFreeRegTriangSetXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{P := NSMP(R, V)\free{R }\free{V }\bound{P }}
\indentrel{3}\begin{verbatim}
   (5)
  NewSparseMultivariatePolynomial(Integer,OrderedVariable
  List [x,y,z,t])
                                           Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPageEmpty5}
\begin{paste}{SqFreeRegTriangSetXmpPageEmpty5}{SqFreeRegTriangSetXmpPagePatch5}
\pastebutton{SqFreeRegTriangSetXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{P := NSMP(R, V)\free{R }\free{V }\bound{P }}
\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPagePatch6}
\begin{paste}{SqFreeRegTriangSetXmpPageFull6}{SqFreeRegTriangSetXmpPageEmpty6}
\pastebutton{SqFreeRegTriangSetXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{x: P := 'x\free{P }\bound{x }}
\indentrel{3}\begin{verbatim}
   (6)  x
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [x,y,z,t])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPageEmpty6}
\begin{paste}{SqFreeRegTriangSetXmpPageEmpty6}{SqFreeRegTriangSetXmpPagePatch6}
\pastebutton{SqFreeRegTriangSetXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{x: P := 'x\free{P }\bound{x }}
\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPagePatch7}
\begin{paste}{SqFreeRegTriangSetXmpPageFull7}{SqFreeRegTriangSetXmpPageEmpty7}
\pastebutton{SqFreeRegTriangSetXmpPageFull7}{\hidepaste}
```

```
\tab{5}\spadcommand{y: P := 'y\free{P }\bound{y }}
\indentrel{3}\begin{verbatim}
   (7)  y
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [x,y,z,t])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPageEmpty7}
\begin{paste}{SqFreeRegTriangSetXmpPageEmpty7}{SqFreeRegTriangSetXmpPagePatch7}
\pastebutton{SqFreeRegTriangSetXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{y: P := 'y\free{P }\bound{y }}
\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPagePatch8}
\begin{paste}{SqFreeRegTriangSetXmpPageFull8}{SqFreeRegTriangSetXmpPageEmpty8}
\pastebutton{SqFreeRegTriangSetXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{z: P := 'z\free{P }\bound{z }}
\indentrel{3}\begin{verbatim}
   (8)  z
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [x,y,z,t])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPageEmpty8}
\begin{paste}{SqFreeRegTriangSetXmpPageEmpty8}{SqFreeRegTriangSetXmpPagePatch8}
\pastebutton{SqFreeRegTriangSetXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{z: P := 'z\free{P }\bound{z }}
\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPagePatch9}
\begin{paste}{SqFreeRegTriangSetXmpPageFull9}{SqFreeRegTriangSetXmpPageEmpty9}
\pastebutton{SqFreeRegTriangSetXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{t: P := 't\free{P }\bound{t }}
\indentrel{3}\begin{verbatim}
   (9)  t
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [x,y,z,t])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPageEmpty9}
\begin{paste}{SqFreeRegTriangSetXmpPageEmpty9}{SqFreeRegTriangSetXmpPagePatch9}
\pastebutton{SqFreeRegTriangSetXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{t: P := 't\free{P }\bound{t }}
\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPagePatch10}
\begin{paste}{SqFreeRegTriangSetXmpPageFull10}{SqFreeRegTriangSetXmpPageEmpty10}
\pastebutton{SqFreeRegTriangSetXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{ST := SREGSET(R,E,V,P)\free{R }\free{E }\free{V }\free{P }\bound{ST }}
\indentrel{3}\begin{verbatim}
```

```
   (10)
  SquareFreeRegularTriangularSet(Integer,IndexedExponents
   OrderedVariableList [x,y,z,t],OrderedVariableList [x,y
  ,z,t],NewSparseMultivariatePolynomial(Integer,OrderedVa
  riableList [x,y,z,t]))
                                            Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPageEmpty10}
\begin{paste}{SqFreeRegTriangSetXmpPageEmpty10}{SqFreeRegTriangSetXmpPagePatch10}
\pastebutton{SqFreeRegTriangSetXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{ST := SREGSET(R,E,V,P)\free{R }\free{E }\free{V }\free{P }\bound{ST }}
\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPagePatch11}
\begin{paste}{SqFreeRegTriangSetXmpPageFull11}{SqFreeRegTriangSetXmpPageEmpty11}
\pastebutton{SqFreeRegTriangSetXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{p1 := x ** 31 - x ** 6 - x - y\free{x }\free{y }\bound{p1 }}
\indentrel{3}\begin{verbatim}
          31    6
   (11)  x   - x  - x - y
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [x,y,z,t])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPageEmpty11}
\begin{paste}{SqFreeRegTriangSetXmpPageEmpty11}{SqFreeRegTriangSetXmpPagePatch11}
\pastebutton{SqFreeRegTriangSetXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{p1 := x ** 31 - x ** 6 - x - y\free{x }\free{y }\bound{p1 }}
\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPagePatch12}
\begin{paste}{SqFreeRegTriangSetXmpPageFull12}{SqFreeRegTriangSetXmpPageEmpty12}
\pastebutton{SqFreeRegTriangSetXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{p2 := x ** 8 - z\free{x }\free{z }\bound{p2 }}
\indentrel{3}\begin{verbatim}
          8
   (12)  x  - z
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [x,y,z,t])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPageEmpty12}
\begin{paste}{SqFreeRegTriangSetXmpPageEmpty12}{SqFreeRegTriangSetXmpPagePatch12}
\pastebutton{SqFreeRegTriangSetXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{p2 := x ** 8 - z\free{x }\free{z }\bound{p2 }}
\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPagePatch13}
```

```
\begin{paste}{SqFreeRegTriangSetXmpPageFull13}{SqFreeRegTriangSetXmpPageEmpty13}
\pastebutton{SqFreeRegTriangSetXmpPageFull13}{\hidepaste}
\tab{5}\spadcommand{p3 := x ** 10 - t\free{x }\free{t }\bound{p3 }}
\indentrel{3}\begin{verbatim}
          10
   (13)  x   - t
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [x,y,z,t])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPageEmpty13}
\begin{paste}{SqFreeRegTriangSetXmpPageEmpty13}{SqFreeRegTriangSetXmpPagePatch13}
\pastebutton{SqFreeRegTriangSetXmpPageEmpty13}{\showpaste}
\tab{5}\spadcommand{p3 := x ** 10 - t\free{x }\free{t }\bound{p3 }}
\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPagePatch14}
\begin{paste}{SqFreeRegTriangSetXmpPageFull14}{SqFreeRegTriangSetXmpPageEmpty14}
\pastebutton{SqFreeRegTriangSetXmpPageFull14}{\hidepaste}
\tab{5}\spadcommand{lp := [p1, p2, p3]\free{p1 }\free{p2 }\free{p3 }\bound{lp }}
\indentrel{3}\begin{verbatim}
           31    6          8      10
   (14)  [x   - x  - x - y,x  - z,x   - t]
Type: List NewSparseMultivariatePolynomial(Integer,OrderedVariableList [x,y,z,t])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPageEmpty14}
\begin{paste}{SqFreeRegTriangSetXmpPageEmpty14}{SqFreeRegTriangSetXmpPagePatch14}
\pastebutton{SqFreeRegTriangSetXmpPageEmpty14}{\showpaste}
\tab{5}\spadcommand{lp := [p1, p2, p3]\free{p1 }\free{p2 }\free{p3 }\bound{lp }}
\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPagePatch15}
\begin{paste}{SqFreeRegTriangSetXmpPageFull15}{SqFreeRegTriangSetXmpPageEmpty15}
\pastebutton{SqFreeRegTriangSetXmpPageFull15}{\hidepaste}
\tab{5}\spadcommand{zeroSetSplit(lp)$ST\free{lp }\free{ST }}
\indentrel{3}\begin{verbatim}
   (15)
   [
       5    4       2     3     8     5    3    2
     {z  - t , t z y  + 2z y - t  + 2t  + t  - t ,
        4               2
      (t  - t)x - t y - z }
     ]
Type: List SquareFreeRegularTriangularSet(Integer,IndexedExponents OrderedVariableList [x,y,z,t],OrderedVar
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPageEmpty15}
```

```
\begin{paste}{SqFreeRegTriangSetXmpPageEmpty15}{SqFreeRegTriangSetXmpPagePatch15}
\pastebutton{SqFreeRegTriangSetXmpPageEmpty15}{\showpaste}
\tab{5}\spadcommand{zeroSetSplit(lp)$ST\free{lp }\free{ST }}
\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPagePatch16}
\begin{paste}{SqFreeRegTriangSetXmpPageFull16}{SqFreeRegTriangSetXmpPageEmpty16}
\pastebutton{SqFreeRegTriangSetXmpPageFull16}{\hidepaste}
\tab{5}\spadcommand{zeroSetSplit(lp,false)$ST\free{lp }\free{ST }\bound{lts }}
\indentrel{3}\begin{verbatim}
   (16)
   [
      5    4       2     3     8     5    3    2
     {z  - t , t z y  + 2z y - t  + 2t  + t  - t ,
        4              2
      (t  - t)x - t y - z }
     ,
      3      5            2    2
    {t  - 1,z  - t,t y + z ,z x  - t}, {t,z,y,x}]
Type: List SquareFreeRegularTriangularSet(Integer,IndexedExponents OrderedVariableList [x,y,z,t]
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPageEmpty16}
\begin{paste}{SqFreeRegTriangSetXmpPageEmpty16}{SqFreeRegTriangSetXmpPagePatch16}
\pastebutton{SqFreeRegTriangSetXmpPageEmpty16}{\showpaste}
\tab{5}\spadcommand{zeroSetSplit(lp,false)$ST\free{lp }\free{ST }\bound{lts }}
\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPagePatch17}
\begin{paste}{SqFreeRegTriangSetXmpPageFull17}{SqFreeRegTriangSetXmpPageEmpty17}
\pastebutton{SqFreeRegTriangSetXmpPageFull17}{\hidepaste}
\tab{5}\spadcommand{T := REGSET(R,E,V,P)\free{R }\free{E }\free{V }\free{P }\bound{T }}
\indentrel{3}\begin{verbatim}
   (17)
  RegularTriangularSet(Integer,IndexedExponents OrderedVa
  riableList [x,y,z,t],OrderedVariableList [x,y,z,t],NewS
  parseMultivariatePolynomial(Integer,OrderedVariableList
   [x,y,z,t]))
                                           Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPageEmpty17}
\begin{paste}{SqFreeRegTriangSetXmpPageEmpty17}{SqFreeRegTriangSetXmpPagePatch17}
\pastebutton{SqFreeRegTriangSetXmpPageEmpty17}{\showpaste}
\tab{5}\spadcommand{T := REGSET(R,E,V,P)\free{R }\free{E }\free{V }\free{P }\bound{T }}
\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPagePatch18}
```

```
\begin{paste}{SqFreeRegTriangSetXmpPageFull18}{SqFreeRegTriangSetXmpPageEmpty18}
\pastebutton{SqFreeRegTriangSetXmpPageFull18}{\hidepaste}
\tab{5}\spadcommand{lts := zeroSetSplit(lp,false)$T\free{lp }\free{T }\bound{lts }}
\indentrel{3}\begin{verbatim}
   (18)
   [
      5    4       2     3     8     5    3    2
     {z  - t , t z y  + 2z y - t  + 2t  + t  - t ,
       4              2
      (t  - t)x - t y - z }
     ,
      3      5          2     3         2
    {t  - 1,z  - t,t z y  + 2z y + 1,z x  - t},
    {t,z,y,x}]
Type: List RegularTriangularSet(Integer,IndexedExponents OrderedVariableList [x,y,z,t],OrderedVariableList
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPageEmpty18}
\begin{paste}{SqFreeRegTriangSetXmpPageEmpty18}{SqFreeRegTriangSetXmpPagePatch18}
\pastebutton{SqFreeRegTriangSetXmpPageEmpty18}{\showpaste}
\tab{5}\spadcommand{lts := zeroSetSplit(lp,false)$T\free{lp }\free{T }\bound{lts }}
\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPagePatch19}
\begin{paste}{SqFreeRegTriangSetXmpPageFull19}{SqFreeRegTriangSetXmpPageEmpty19}
\pastebutton{SqFreeRegTriangSetXmpPageFull19}{\hidepaste}
\tab{5}\spadcommand{ts := lts.2\free{lts }\bound{ts }}
\indentrel{3}\begin{verbatim}
          3      5          2     3         2
   (19)  {t  - 1,z  - t,t z y  + 2z y + 1,z x  - t}
Type: RegularTriangularSet(Integer,IndexedExponents OrderedVariableList [x,y,z,t],OrderedVariableList [x,y,z
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPageEmpty19}
\begin{paste}{SqFreeRegTriangSetXmpPageEmpty19}{SqFreeRegTriangSetXmpPagePatch19}
\pastebutton{SqFreeRegTriangSetXmpPageEmpty19}{\showpaste}
\tab{5}\spadcommand{ts := lts.2\free{lts }\bound{ts }}
\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPagePatch20}
\begin{paste}{SqFreeRegTriangSetXmpPageFull20}{SqFreeRegTriangSetXmpPageEmpty20}
\pastebutton{SqFreeRegTriangSetXmpPageFull20}{\hidepaste}
\tab{5}\spadcommand{pol := select(ts,'y)$T\free{ts }\free{y }\free{T }\bound{pol }}
\indentrel{3}\begin{verbatim}
              2     3
   (20)  t z y  + 2z y + 1
Type: Union(NewSparseMultivariatePolynomial(Integer,OrderedVariableList [x,y,z,t]),...)
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPageEmpty20}
\begin{paste}{SqFreeRegTriangSetXmpPageEmpty20}{SqFreeRegTriangSetXmpPagePatch20}
\pastebutton{SqFreeRegTriangSetXmpPageEmpty20}{\showpaste}
\tab{5}\spadcommand{pol := select(ts,'y)$T\free{ts }\free{y }\free{T }\bound{pol }}
\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPagePatch21}
\begin{paste}{SqFreeRegTriangSetXmpPageFull21}{SqFreeRegTriangSetXmpPageEmpty21}
\pastebutton{SqFreeRegTriangSetXmpPageFull21}{\hidepaste}
\tab{5}\spadcommand{tower := collectUnder(ts,'y)$T\free{ts }\free{y }\free{T }\bound{tower }}
\indentrel{3}\begin{verbatim}
          3      5
   (21)  {t  - 1,z  - t}
Type: RegularTriangularSet(Integer,IndexedExponents OrderedVariableList [x,y,z,t],OrderedVariabl
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPageEmpty21}
\begin{paste}{SqFreeRegTriangSetXmpPageEmpty21}{SqFreeRegTriangSetXmpPagePatch21}
\pastebutton{SqFreeRegTriangSetXmpPageEmpty21}{\showpaste}
\tab{5}\spadcommand{tower := collectUnder(ts,'y)$T\free{ts }\free{y }\free{T }\bound{tower }}
\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPagePatch22}
\begin{paste}{SqFreeRegTriangSetXmpPageFull22}{SqFreeRegTriangSetXmpPageEmpty22}
\pastebutton{SqFreeRegTriangSetXmpPageFull22}{\hidepaste}
\tab{5}\spadcommand{pack := RegularTriangularSetGcdPackage(R,E,V,P,T)\free{R }\free{E }\free{V }
\indentrel{3}\begin{verbatim}
   (22)
  RegularTriangularSetGcdPackage(Integer,IndexedExponents
   OrderedVariableList [x,y,z,t],OrderedVariableList [x,y
  ,z,t],NewSparseMultivariatePolynomial(Integer,OrderedVa
  riableList [x,y,z,t]),RegularTriangularSet(Integer,Inde
  xedExponents OrderedVariableList [x,y,z,t],OrderedVaria
  bleList [x,y,z,t],NewSparseMultivariatePolynomial(Integ
  er,OrderedVariableList [x,y,z,t])))
                                            Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPageEmpty22}
\begin{paste}{SqFreeRegTriangSetXmpPageEmpty22}{SqFreeRegTriangSetXmpPagePatch22}
\pastebutton{SqFreeRegTriangSetXmpPageEmpty22}{\showpaste}
\tab{5}\spadcommand{pack := RegularTriangularSetGcdPackage(R,E,V,P,T)\free{R }\free{E }\free{V }
\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPagePatch23}
\begin{paste}{SqFreeRegTriangSetXmpPageFull23}{SqFreeRegTriangSetXmpPageEmpty23}
```

```
\pastebutton{SqFreeRegTriangSetXmpPageFull23}{\hidepaste}
\tab{5}\spadcommand{toseSquareFreePart(pol,tower)$pack\free{pol }\free{tower }\free{pack }}
\indentrel{3}\begin{verbatim}
                       2          3      5
   (23)  [[val= t y + z ,tower= {t  - 1,z  - t}]]
Type: List Record(val: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [x,y,z,t]),tower: Regular
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SqFreeRegTriangSetXmpPageEmpty23}
\begin{paste}{SqFreeRegTriangSetXmpPageEmpty23}{SqFreeRegTriangSetXmpPagePatch23}
\pastebutton{SqFreeRegTriangSetXmpPageEmpty23}{\showpaste}
\tab{5}\spadcommand{toseSquareFreePart(pol,tower)$pack\free{pol }\free{tower }\free{pack }}
\end{paste}\end{patch}
```

———-

3.101 stbl.ht

SparseTable

⇒ "notitle" (TableXmpPage) 3.106 on page 1298
⇒ "notitle" (GeneralSparseTableXmpPage) 3.52 on page 687

— **stbl.ht** —

```
\begin{page}{SparseTableXmpPage}{SparseTable}
\beginscroll
%
The \spadtype{SparseTable} domain provides a general purpose
table type with default entries.
\xtc{
Here we create a table to save strings under integer keys.
The value \spad{"Try again!"} is returned if no other value has been
stored for a key.
}{
\spadpaste{t: SparseTable(Integer, String, "Try again!") := table()
\bound{t}}
}
\xtc{
Entries can be stored in the table.
}{
\spadpaste{t.3 := "Number three" \free{t}\bound{t1}}
}
\xtc{
}{
```

```
\spadpaste{t.4 := "Number four" \free{t}\bound{t2}}
}
\xtc{
These values can be retrieved as usual, but if a look up fails
the default entry will be returned.
}{
\spadpaste{t.3 \free{t1}}
}
\xtc{
}{
\spadpaste{t.2 \free{t}}
}
\xtc{
To see which values are explicitly stored, the
\spadfunFrom{keys}{SparseTable} and \spadfunFrom{entries}{SparseTable}
functions can be used.
}{
\spadpaste{keys t \free{t1 t2}}
}
\xtc{
}{
\spadpaste{entries t \free{t1 t2}}
}
If a specific table representation
is required, the \spadtype{GeneralSparseTable} constructor should be
used. The domain \spadtype{SparseTable(K, E, dflt)} is equivalent to
\spadtype{GeneralSparseTable(K,E, Table(K,E), dflt)}.
For more information, see
\downlink{`Table'}{TableXmpPage}\ignore{Table} and
\downlink{`GeneralSparseTable'}{GeneralSparseTableXmpPage}
\ignore{GeneralSparseTable}.
\showBlurb{SparseTable}
\endscroll
\autobuttons
\end{page}

\begin{patch}{SparseTableXmpPagePatch1}
\begin{paste}{SparseTableXmpPageFull1}{SparseTableXmpPageEmpty1}
\pastebutton{SparseTableXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{t: SparseTable(Integer, String, "Try again!") := table()\bound{t }}
\indentrel{3}\begin{verbatim}
   (1)  table()
           Type: SparseTable(Integer,String,Try again!)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SparseTableXmpPageEmpty1}
\begin{paste}{SparseTableXmpPageEmpty1}{SparseTableXmpPagePatch1}
\pastebutton{SparseTableXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{t: SparseTable(Integer, String, "Try again!") := table()\bound{t }}
```

```
\end{paste}\end{patch}

\begin{patch}{SparseTableXmpPagePatch2}
\begin{paste}{SparseTableXmpPageFull2}{SparseTableXmpPageEmpty2}
\pastebutton{SparseTableXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{t.3 := "Number three"\free{t }\bound{t1 }}
\indentrel{3}\begin{verbatim}
   (2)  "Number three"
                                           Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SparseTableXmpPageEmpty2}
\begin{paste}{SparseTableXmpPageEmpty2}{SparseTableXmpPagePatch2}
\pastebutton{SparseTableXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{t.3 := "Number three"\free{t }\bound{t1 }}
\end{paste}\end{patch}

\begin{patch}{SparseTableXmpPagePatch3}
\begin{paste}{SparseTableXmpPageFull3}{SparseTableXmpPageEmpty3}
\pastebutton{SparseTableXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{t.4 := "Number four"\free{t }\bound{t2 }}
\indentrel{3}\begin{verbatim}
   (3)  "Number four"
                                           Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SparseTableXmpPageEmpty3}
\begin{paste}{SparseTableXmpPageEmpty3}{SparseTableXmpPagePatch3}
\pastebutton{SparseTableXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{t.4 := "Number four"\free{t }\bound{t2 }}
\end{paste}\end{patch}

\begin{patch}{SparseTableXmpPagePatch4}
\begin{paste}{SparseTableXmpPageFull4}{SparseTableXmpPageEmpty4}
\pastebutton{SparseTableXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{t.3\free{t1 }}
\indentrel{3}\begin{verbatim}
   (4)  "Number three"
                                           Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SparseTableXmpPageEmpty4}
\begin{paste}{SparseTableXmpPageEmpty4}{SparseTableXmpPagePatch4}
\pastebutton{SparseTableXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{t.3\free{t1 }}
\end{paste}\end{patch}
```

```
\begin{patch}{SparseTableXmpPagePatch5}
\begin{paste}{SparseTableXmpPageFull5}{SparseTableXmpPageEmpty5}
\pastebutton{SparseTableXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{t.2\free{t }}
\indentrel{3}\begin{verbatim}
   (5)  "Try again!"
                                         Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SparseTableXmpPageEmpty5}
\begin{paste}{SparseTableXmpPageEmpty5}{SparseTableXmpPagePatch5}
\pastebutton{SparseTableXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{t.2\free{t }}
\end{paste}\end{patch}

\begin{patch}{SparseTableXmpPagePatch6}
\begin{paste}{SparseTableXmpPageFull6}{SparseTableXmpPageEmpty6}
\pastebutton{SparseTableXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{keys t\free{t1 t2 }}
\indentrel{3}\begin{verbatim}
   (6)  [4,3]
                                   Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SparseTableXmpPageEmpty6}
\begin{paste}{SparseTableXmpPageEmpty6}{SparseTableXmpPagePatch6}
\pastebutton{SparseTableXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{keys t\free{t1 t2 }}
\end{paste}\end{patch}

\begin{patch}{SparseTableXmpPagePatch7}
\begin{paste}{SparseTableXmpPageFull7}{SparseTableXmpPageEmpty7}
\pastebutton{SparseTableXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{entries t\free{t1 t2 }}
\indentrel{3}\begin{verbatim}
   (7)  ["Number four","Number three"]
                                    Type: List String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SparseTableXmpPageEmpty7}
\begin{paste}{SparseTableXmpPageEmpty7}{SparseTableXmpPagePatch7}
\pastebutton{SparseTableXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{entries t\free{t1 t2 }}
\end{paste}\end{patch}
```

3.102 stream.ht

Stream

⇒ "notitle" (ugLangItsPage) 9 on page 1808
⇒ "notitle" (ugProblemSeriesPage) 12 on page 2164
⇒ "notitle" (ContinuedFractionXmpPage) 3.17 on page 262
⇒ "notitle" (ListXmpPage) 3.64 on page 866

— **stream.ht** —

```
\begin{page}{StreamXmpPage}{Stream}
\beginscroll

A \spadtype{Stream} object is represented as a list whose
last element contains the
wherewithal to create the next element, should it ever be required.
\xtc{
Let \spad{ints} be the infinite stream of non-negative integers.
}{
\spadpaste{ints := [i for i in 0..] \bound{ints}}
}
By default, ten stream elements are calculated.
This number may be changed to something else by the system command
\spadcmd{)set streams calculate}.
For the display purposes of this book, we have chosen a smaller value.
\xtc{
More generally, you can construct a stream by specifying its initial
value and a function which, when given an element, creates the next
element.
}{
\spadpaste{f : List INT -> List INT \bound{fdec}}
}
\xtc{
}{
\spadpaste{f x == [x.1 + x.2, x.1] \bound{f}\free{fdec}}
}
\xtc{
}{
\spadpaste{fibs := [i.2 for i in [generate(f,[1,1])]] \bound{fibs}\free{f}}
}
\xtc{
You can create the stream of odd non-negative integers by either
filtering them from the integers, or by evaluating an expression for
each integer.
}{
\spadpaste{[i for i in ints | odd? i] \free{ints}}
}
\xtc{
```

```
}{
\spadpaste{odds := [2*i+1 for i in ints]\bound{odds}\free{ints}}
}
\xtc{
You can accumulate the initial segments of a stream using the
\spadfunFrom{scan}{StreamFunctions2} operation.
}{
\spadpaste{scan(0,+,odds) \free{odds}}
}
\xtc{
The corresponding elements of
two or more streams can be combined in this way.
}{
\spadpaste{[i*j for i in ints for j in odds]\free{ints} \free{odds}}
}
\xtc{
}{
\spadpaste{map(*,ints,odds)\free{ints odds}}
}
\xtc{
Many operations similar to those applicable to lists are available for
streams.
}{
\spadpaste{first ints \free{ints}}
}
\xtc{
}{
\spadpaste{rest ints \free{ints}}
}
\xtc{
}{
\spadpaste{fibs 20 \free{fibs}}
}
The packages \spadtype{StreamFunctions1},
\spadtype{StreamFunctions2} and
\spadtype{StreamFunctions3} export some useful stream manipulation
operations.
For more information, see
\downlink{``Creating Lists and Streams with Iterators''}
{ugLangItsPage} in Section 5.5\ignore{ugLangIts},
\downlink{``Working with Power Series''}{ugProblemSeriesPage}
in Section 8.9\ignore{ugProblemSeries},
\downlink{`ContinuedFraction'}{ContinuedFractionXmpPage}
\ignore{ContinuedFraction}, and
\downlink{`List'}{ListXmpPage}\ignore{List}.
%
\showBlurb{Stream}
\endscroll
\autobuttons
\end{page}
```

```
\begin{patch}{StreamXmpPagePatch1}
\begin{paste}{StreamXmpPageFull1}{StreamXmpPageEmpty1}
\pastebutton{StreamXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{ints := [i for i in 0..]\bound{ints }}
\indentrel{3}\begin{verbatim}
   (1)  [0,1,2,3,4,5,6,7,8,9,...]
                        Type: Stream NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StreamXmpPageEmpty1}
\begin{paste}{StreamXmpPageEmpty1}{StreamXmpPagePatch1}
\pastebutton{StreamXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{ints := [i for i in 0..]\bound{ints }}
\end{paste}\end{patch}

\begin{patch}{StreamXmpPagePatch2}
\begin{paste}{StreamXmpPageFull2}{StreamXmpPageEmpty2}
\pastebutton{StreamXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{f : List INT -> List INT\bound{fdec }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StreamXmpPageEmpty2}
\begin{paste}{StreamXmpPageEmpty2}{StreamXmpPagePatch2}
\pastebutton{StreamXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{f : List INT -> List INT\bound{fdec }}
\end{paste}\end{patch}

\begin{patch}{StreamXmpPagePatch3}
\begin{paste}{StreamXmpPageFull3}{StreamXmpPageEmpty3}
\pastebutton{StreamXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{f x == [x.1 + x.2, x.1]\bound{f }\free{fdec }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StreamXmpPageEmpty3}
\begin{paste}{StreamXmpPageEmpty3}{StreamXmpPagePatch3}
\pastebutton{StreamXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{f x == [x.1 + x.2, x.1]\bound{f }\free{fdec }}
\end{paste}\end{patch}

\begin{patch}{StreamXmpPagePatch4}
\begin{paste}{StreamXmpPageFull4}{StreamXmpPageEmpty4}
\pastebutton{StreamXmpPageFull4}{\hidepaste}
```

```
\tab{5}\spadcommand{fibs := [i.2 for i in [generate(f,[1,1])]]\bound{fibs }\free{f }}
\indentrel{3}\begin{verbatim}
   (4)  [1,1,2,3,5,8,13,21,34,55,...]
                                   Type: Stream Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StreamXmpPageEmpty4}
\begin{paste}{StreamXmpPageEmpty4}{StreamXmpPagePatch4}
\pastebutton{StreamXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{fibs := [i.2 for i in [generate(f,[1,1])]]\bound{fibs }\free{f }}
\end{paste}\end{patch}

\begin{patch}{StreamXmpPagePatch5}
\begin{paste}{StreamXmpPageFull5}{StreamXmpPageEmpty5}
\pastebutton{StreamXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{[i for i in ints | odd? i]\free{ints }}
\indentrel{3}\begin{verbatim}
   (5)  [1,3,5,7,9,11,13,15,17,19,...]
                        Type: Stream NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StreamXmpPageEmpty5}
\begin{paste}{StreamXmpPageEmpty5}{StreamXmpPagePatch5}
\pastebutton{StreamXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{[i for i in ints | odd? i]\free{ints }}
\end{paste}\end{patch}

\begin{patch}{StreamXmpPagePatch6}
\begin{paste}{StreamXmpPageFull6}{StreamXmpPageEmpty6}
\pastebutton{StreamXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{odds := [2*i+1 for i in ints]\bound{odds }\free{ints }}
\indentrel{3}\begin{verbatim}
   (6)  [1,3,5,7,9,11,13,15,17,19,...]
                        Type: Stream NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StreamXmpPageEmpty6}
\begin{paste}{StreamXmpPageEmpty6}{StreamXmpPagePatch6}
\pastebutton{StreamXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{odds := [2*i+1 for i in ints]\bound{odds }\free{ints }}
\end{paste}\end{patch}

\begin{patch}{StreamXmpPagePatch7}
\begin{paste}{StreamXmpPageFull7}{StreamXmpPageEmpty7}
\pastebutton{StreamXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{scan(0,+,odds)\free{odds }}
\indentrel{3}\begin{verbatim}
```

```
   (7)  [1,4,9,16,25,36,49,64,81,100,...]
                          Type: Stream NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StreamXmpPageEmpty7}
\begin{paste}{StreamXmpPageEmpty7}{StreamXmpPagePatch7}
\pastebutton{StreamXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{scan(0,+,odds)\free{odds }}
\end{paste}\end{patch}

\begin{patch}{StreamXmpPagePatch8}
\begin{paste}{StreamXmpPageFull8}{StreamXmpPageEmpty8}
\pastebutton{StreamXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{[i*j for i in ints for j in odds]\free{ints }\free{odds }}
\indentrel{3}\begin{verbatim}
   (8)  [0,3,10,21,36,55,78,105,136,171,...]
                          Type: Stream NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StreamXmpPageEmpty8}
\begin{paste}{StreamXmpPageEmpty8}{StreamXmpPagePatch8}
\pastebutton{StreamXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{[i*j for i in ints for j in odds]\free{ints }\free{odds }}
\end{paste}\end{patch}

\begin{patch}{StreamXmpPagePatch9}
\begin{paste}{StreamXmpPageFull9}{StreamXmpPageEmpty9}
\pastebutton{StreamXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{map(*,ints,odds)\free{ints odds }}
\indentrel{3}\begin{verbatim}
   (9)  [0,3,10,21,36,55,78,105,136,171,...]
                          Type: Stream NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StreamXmpPageEmpty9}
\begin{paste}{StreamXmpPageEmpty9}{StreamXmpPagePatch9}
\pastebutton{StreamXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{map(*,ints,odds)\free{ints odds }}
\end{paste}\end{patch}

\begin{patch}{StreamXmpPagePatch10}
\begin{paste}{StreamXmpPageFull10}{StreamXmpPageEmpty10}
\pastebutton{StreamXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{first ints\free{ints }}
\indentrel{3}\begin{verbatim}
   (10)  0
                                 Type: NonNegativeInteger
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StreamXmpPageEmpty10}
\begin{paste}{StreamXmpPageEmpty10}{StreamXmpPagePatch10}
\pastebutton{StreamXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{first ints\free{ints }}
\end{paste}\end{patch}

\begin{patch}{StreamXmpPagePatch11}
\begin{paste}{StreamXmpPageFull11}{StreamXmpPageEmpty11}
\pastebutton{StreamXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{rest ints\free{ints }}
\indentrel{3}\begin{verbatim}
   (11)  [1,2,3,4,5,6,7,8,9,10,...]
                        Type: Stream NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StreamXmpPageEmpty11}
\begin{paste}{StreamXmpPageEmpty11}{StreamXmpPagePatch11}
\pastebutton{StreamXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{rest ints\free{ints }}
\end{paste}\end{patch}

\begin{patch}{StreamXmpPagePatch12}
\begin{paste}{StreamXmpPageFull12}{StreamXmpPageEmpty12}
\pastebutton{StreamXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{fibs 20\free{fibs }}
\indentrel{3}\begin{verbatim}
   (12)  6765
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StreamXmpPageEmpty12}
\begin{paste}{StreamXmpPageEmpty12}{StreamXmpPagePatch12}
\pastebutton{StreamXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{fibs 20\free{fibs }}
\end{paste}\end{patch}
```

3.103 string.ht

String

⇒ "notitle" (CharacterXmpPage) 3.15 on page 228
⇒ "notitle" (CharacterClassXmpPage) 3.14 on page 221

— **string.ht** —

```
\begin{page}{StringXmpPage}{String}
\beginscroll

The type \spadtype{String} provides character strings.  Character
strings provide all the operations for a one-dimensional array of
characters, plus additional operations for manipulating text.  For
more information on related topics, see
\downlink{`Character'}{CharacterXmpPage}\ignore{Character} and
\downlink{`CharacterClass'}{CharacterClassXmpPage}\ignore{CharacterClass}.
You can also issue the system command \spadcmd{)show String} to
display the full list of operations defined by \spadtype{String}.

\xtc{
String values can be created using double quotes.
}{
\spadpaste{hello := "Hello, I'm Axiom!" \bound{hello}}
}
\xtc{
Note, however, that double quotes and underscores must be preceded by
an extra underscore.
}{
\spadpaste{said  := "Jane said, _"Look!_"" \bound{said}}
}
\xtc{
}{
\spadpaste{saw   := "She saw exactly one underscore: __." \bound{saw}}
}
\xtc{
It is also possible to use \spadfunFrom{new}{String} to create a
string of any size filled with a given character.  Since there are
many \spadfun{new} functions it is necessary to indicate the desired
type.
}{
\spadpaste{gasp: String := new(32, char "x") \bound{gasp}}
}
\xtc{
The length of a string is given by \spadopFrom{\#}{List}.
}{
\spadpaste{\#gasp \free{gasp}}
}
```

```
\xtc{
Indexing operations allow characters to be extracted or replaced in
strings. For any string \spad{s},
indices lie in the range \spad{1..\#s}.
}{
\spadpaste{hello.2 \free{hello}}
}
\xtc{
Indexing is really just the application of a string to a subscript,
so any application syntax works.
}{
\spadpaste{hello 2  \free{hello}}
}
\xtc{
}{
\spadpaste{hello(2) \free{hello}}
}
\xtc{
If it is important not to modify a given string, it should be copied
before any updating operations are used.
}{
\spadpaste{hullo := copy hello \free{hello}\bound{hullo0}}
}
\xtc{
}{
\spadpaste{hullo.2 := char "u"; [hello, hullo] \free{hullo0 hello}
\bound{hullo}}
}

\xtc{
Operations are provided to split and join strings.
The \spadfunFrom{concat}{String} operation allows several strings
to be joined together.
}{
\spadpaste{saidsaw := concat ["alpha","---","omega"] \bound{saidsaw}}
}
\xtc{
There is a version of \spadfunFrom{concat}{String} that works with
two strings.
}{
\spadpaste{concat("hello ","goodbye")}
}
\xtc{
Juxtaposition can also be used to concatenate strings.
}{
\spadpaste{"This " "is " "several " "strings " "concatenated."}
}
\xtc{
Substrings are obtained by giving an index range.
}{
```

```
\spadpaste{hello(1..5) \free{hello}}
}
\xtc{
}{
\spadpaste{hello(8..) \free{hello}}
}
\xtc{
A string can be split into several substrings by giving a
separation character or character class.
}{
\spadpaste{split(hello, char " ")              \free{hello}}
}
\xtc{
}{
\spadpaste{other := complement alphanumeric(); \bound{other}}
}
\xtc{
}{
\spadpaste{split(saidsaw, other)               \free{saidsaw other}}
}
\xtc{
Unwanted characters can be trimmed from the beginning or end of a
string using the operations \spadfunFrom{trim}{String},
\spadfunFrom{leftTrim}{String} and \spadfunFrom{rightTrim}{String}.
}{
\spadpaste{trim     ("\#\# ++ relax ++ \#\#", char "\#")}
}
\xtc{
Each of these functions takes a string and a second argument to specify
the characters to be discarded.
}{
\spadpaste{trim     ("\#\# ++ relax ++ \#\#", other) \free{other}}
}
\xtc{
The second argument can be given
either as a single character or as a character class.
}{
\spadpaste{leftTrim ("\#\# ++ relax ++ \#\#", other) \free{other}}
}
\xtc{
}{
\spadpaste{rightTrim("\#\# ++ relax ++ \#\#", other) \free{other}}
}

\xtc{
Strings can be changed to upper case or lower case using the operations
\spadfunFrom{upperCase}{String}, \spadfunFromX{upperCase}{String},
\spadfunFrom{lowerCase}{String} and
\spadfunFromX{lowerCase}{String}.
}{
```

```
\spadpaste{upperCase hello \free{hello}}
}
\xtc{
The versions with the exclamation mark
change the original string, while the others produce a copy.
}{
\spadpaste{lowerCase hello \free{hello}}
}

\xtc{
Some basic string matching is provided.
The function \spadfunFrom{prefix?}{String}
tests whether one string is an initial prefix of another.
}{
\spadpaste{prefix?("He", "Hello")}
}
\xtc{
}{
\spadpaste{prefix?("Her", "Hello")}
}
\xtc{
A similar function, \spadfunFrom{suffix?}{String}, tests for suffixes.
}{
\spadpaste{suffix?("", "Hello")}
}
\xtc{
}{
\spadpaste{suffix?("LO", "Hello")}
}
\xtc{
The function \spadfunFrom{substring?}{String} tests for a substring
given a starting
position.
}{
\spadpaste{substring?("ll", "Hello", 3)}
}
\xtc{
}{
\spadpaste{substring?("ll", "Hello", 4)}
}

\xtc{
A number of \spadfunFrom{position}{String} functions locate things in
strings.  If the first argument to position is a string, then
\spad{position(s,t,i)} finds the location of \spad{s} as a substring
of \spad{t} starting the search at position \spad{i}.
}{
\spadpaste{n := position("nd", "underground",   1) \bound{n}}
}
\xtc{
```

```
}{
\spadpaste{n := position("nd", "underground", n+1) \free{n} \bound{n1}}
}
\xtc{
If \spad{s} is not found, then \spad{0} is returned (\spad{minIndex(s)-1}
in \spadtype{IndexedString}).
}{
\spadpaste{n := position("nd", "underground", n+1) \free{n1}\bound{n2}}
}
\xtc{
To search for a specific character or a member of a character class,
a different first argument is used.
}{
\spadpaste{position(char "d", "underground", 1)}
}
\xtc{
}{
\spadpaste{position(hexDigit(), "underground", 1)}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{StringXmpPagePatch1}
\begin{paste}{StringXmpPageFull1}{StringXmpPageEmpty1}
\pastebutton{StringXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{hello := "Hello, I'm AXIOM!"\bound{hello }}
\indentrel{3}\begin{verbatim}
   (1)  "Hello, I'm AXIOM!"
                                           Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StringXmpPageEmpty1}
\begin{paste}{StringXmpPageEmpty1}{StringXmpPagePatch1}
\pastebutton{StringXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{hello := "Hello, I'm AXIOM!"\bound{hello }}
\end{paste}\end{patch}

\begin{patch}{StringXmpPagePatch2}
\begin{paste}{StringXmpPageFull2}{StringXmpPageEmpty2}
\pastebutton{StringXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{said := "Jane said, _"Look!_""\bound{said }}
\indentrel{3}\begin{verbatim}
   (2)  "Jane said, "Look!""
                                           Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{StringXmpPageEmpty2}
\begin{paste}{StringXmpPageEmpty2}{StringXmpPagePatch2}
\pastebutton{StringXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{said := "Jane said, _"Look!_""\bound{said }}
\end{paste}\end{patch}

\begin{patch}{StringXmpPagePatch3}
\begin{paste}{StringXmpPageFull3}{StringXmpPageEmpty3}
\pastebutton{StringXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{saw := "She saw exactly one underscore: __."\bound{saw }}
\indentrel{3}\begin{verbatim}
   (3)  "She saw exactly one underscore: _."
                                            Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StringXmpPageEmpty3}
\begin{paste}{StringXmpPageEmpty3}{StringXmpPagePatch3}
\pastebutton{StringXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{saw := "She saw exactly one underscore: __."\bound{saw }}
\end{paste}\end{patch}

\begin{patch}{StringXmpPagePatch4}
\begin{paste}{StringXmpPageFull4}{StringXmpPageEmpty4}
\pastebutton{StringXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{gasp: String := new(32, char "x")\bound{gasp }}
\indentrel{3}\begin{verbatim}
   (4)  "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
                                            Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StringXmpPageEmpty4}
\begin{paste}{StringXmpPageEmpty4}{StringXmpPagePatch4}
\pastebutton{StringXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{gasp: String := new(32, char "x")\bound{gasp }}
\end{paste}\end{patch}

\begin{patch}{StringXmpPagePatch5}
\begin{paste}{StringXmpPageFull5}{StringXmpPageEmpty5}
\pastebutton{StringXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{\#gasp\free{gasp }}
\indentrel{3}\begin{verbatim}
   (5)  32
                                   Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StringXmpPageEmpty5}
\begin{paste}{StringXmpPageEmpty5}{StringXmpPagePatch5}
```

```
\pastebutton{StringXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{\#gasp\free{gasp }}
\end{paste}\end{patch}

\begin{patch}{StringXmpPagePatch6}
\begin{paste}{StringXmpPageFull6}{StringXmpPageEmpty6}
\pastebutton{StringXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{hello.2\free{hello }}
\indentrel{3}\begin{verbatim}
   (6)  e
                                        Type: Character
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StringXmpPageEmpty6}
\begin{paste}{StringXmpPageEmpty6}{StringXmpPagePatch6}
\pastebutton{StringXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{hello.2\free{hello }}
\end{paste}\end{patch}

\begin{patch}{StringXmpPagePatch7}
\begin{paste}{StringXmpPageFull7}{StringXmpPageEmpty7}
\pastebutton{StringXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{hello 2\free{hello }}
\indentrel{3}\begin{verbatim}
   (7)  e
                                        Type: Character
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StringXmpPageEmpty7}
\begin{paste}{StringXmpPageEmpty7}{StringXmpPagePatch7}
\pastebutton{StringXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{hello 2\free{hello }}
\end{paste}\end{patch}

\begin{patch}{StringXmpPagePatch8}
\begin{paste}{StringXmpPageFull8}{StringXmpPageEmpty8}
\pastebutton{StringXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{hello(2)\free{hello }}
\indentrel{3}\begin{verbatim}
   (8)  e
                                        Type: Character
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StringXmpPageEmpty8}
\begin{paste}{StringXmpPageEmpty8}{StringXmpPagePatch8}
\pastebutton{StringXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{hello(2)\free{hello }}
```

```
\end{paste}\end{patch}

\begin{patch}{StringXmpPagePatch9}
\begin{paste}{StringXmpPageFull9}{StringXmpPageEmpty9}
\pastebutton{StringXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{hullo := copy hello\free{hello }\bound{hullo0 }}
\indentrel{3}\begin{verbatim}
   (9)  "Hello, I'm AXIOM!"
                                            Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StringXmpPageEmpty9}
\begin{paste}{StringXmpPageEmpty9}{StringXmpPagePatch9}
\pastebutton{StringXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{hullo := copy hello\free{hello }\bound{hullo0 }}
\end{paste}\end{patch}

\begin{patch}{StringXmpPagePatch10}
\begin{paste}{StringXmpPageFull10}{StringXmpPageEmpty10}
\pastebutton{StringXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{hullo.2 := char "u"; [hello, hullo]\free{hullo0 hello }\bound{hullo }}
\indentrel{3}\begin{verbatim}
   (10)  ["Hello, I'm AXIOM!","Hullo, I'm AXIOM!"]
                                       Type: List String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StringXmpPageEmpty10}
\begin{paste}{StringXmpPageEmpty10}{StringXmpPagePatch10}
\pastebutton{StringXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{hullo.2 := char "u"; [hello, hullo]\free{hullo0 hello }\bound{hullo }}
\end{paste}\end{patch}

\begin{patch}{StringXmpPagePatch11}
\begin{paste}{StringXmpPageFull11}{StringXmpPageEmpty11}
\pastebutton{StringXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{saidsaw := concat ["alpha","---","omega"]\bound{saidsaw }}
\indentrel{3}\begin{verbatim}
   (11)  "alpha---omega"
                                            Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StringXmpPageEmpty11}
\begin{paste}{StringXmpPageEmpty11}{StringXmpPagePatch11}
\pastebutton{StringXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{saidsaw := concat ["alpha","---","omega"]\bound{saidsaw }}
\end{paste}\end{patch}
```

```
\begin{patch}{StringXmpPagePatch12}
\begin{paste}{StringXmpPageFull12}{StringXmpPageEmpty12}
\pastebutton{StringXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{concat("hello ","goodbye")}
\indentrel{3}\begin{verbatim}
   (12)  "hello goodbye"
                                          Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StringXmpPageEmpty12}
\begin{paste}{StringXmpPageEmpty12}{StringXmpPagePatch12}
\pastebutton{StringXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{concat("hello ","goodbye")}
\end{paste}\end{patch}

\begin{patch}{StringXmpPagePatch13}
\begin{paste}{StringXmpPageFull13}{StringXmpPageEmpty13}
\pastebutton{StringXmpPageFull13}{\hidepaste}
\tab{5}\spadcommand{"This " "is " "several " "strings " "concatenated."}
\indentrel{3}\begin{verbatim}
   (13)  "This is several strings concatenated."
                                          Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StringXmpPageEmpty13}
\begin{paste}{StringXmpPageEmpty13}{StringXmpPagePatch13}
\pastebutton{StringXmpPageEmpty13}{\showpaste}
\tab{5}\spadcommand{"This " "is " "several " "strings " "concatenated."}
\end{paste}\end{patch}

\begin{patch}{StringXmpPagePatch14}
\begin{paste}{StringXmpPageFull14}{StringXmpPageEmpty14}
\pastebutton{StringXmpPageFull14}{\hidepaste}
\tab{5}\spadcommand{hello(1..5)\free{hello }}
\indentrel{3}\begin{verbatim}
   (14)  "Hello"
                                          Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StringXmpPageEmpty14}
\begin{paste}{StringXmpPageEmpty14}{StringXmpPagePatch14}
\pastebutton{StringXmpPageEmpty14}{\showpaste}
\tab{5}\spadcommand{hello(1..5)\free{hello }}
\end{paste}\end{patch}

\begin{patch}{StringXmpPagePatch15}
\begin{paste}{StringXmpPageFull15}{StringXmpPageEmpty15}
```

```
\pastebutton{StringXmpPageFull15}{\hidepaste}
\tab{5}\spadcommand{hello(8..)\free{hello }}
\indentrel{3}\begin{verbatim}
   (15)  "I'm AXIOM!"
                                              Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StringXmpPageEmpty15}
\begin{paste}{StringXmpPageEmpty15}{StringXmpPagePatch15}
\pastebutton{StringXmpPageEmpty15}{\showpaste}
\tab{5}\spadcommand{hello(8..)\free{hello }}
\end{paste}\end{patch}

\begin{patch}{StringXmpPagePatch16}
\begin{paste}{StringXmpPageFull16}{StringXmpPageEmpty16}
\pastebutton{StringXmpPageFull16}{\hidepaste}
\tab{5}\spadcommand{split(hello, char " ")\free{hello }}
\indentrel{3}\begin{verbatim}
   (16)  ["Hello,","I'm","AXIOM!"]
                                         Type: List String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StringXmpPageEmpty16}
\begin{paste}{StringXmpPageEmpty16}{StringXmpPagePatch16}
\pastebutton{StringXmpPageEmpty16}{\showpaste}
\tab{5}\spadcommand{split(hello, char " ")\free{hello }}
\end{paste}\end{patch}

\begin{patch}{StringXmpPagePatch17}
\begin{paste}{StringXmpPageFull17}{StringXmpPageEmpty17}
\pastebutton{StringXmpPageFull17}{\hidepaste}
\tab{5}\spadcommand{other := complement alphanumeric();\bound{other }}
\indentrel{3}\begin{verbatim}
                                      Type: CharacterClass
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StringXmpPageEmpty17}
\begin{paste}{StringXmpPageEmpty17}{StringXmpPagePatch17}
\pastebutton{StringXmpPageEmpty17}{\showpaste}
\tab{5}\spadcommand{other := complement alphanumeric();\bound{other }}
\end{paste}\end{patch}

\begin{patch}{StringXmpPagePatch18}
\begin{paste}{StringXmpPageFull18}{StringXmpPageEmpty18}
\pastebutton{StringXmpPageFull18}{\hidepaste}
\tab{5}\spadcommand{split(saidsaw, other)\free{saidsaw other }}
\indentrel{3}\begin{verbatim}
```

```
   (18)  ["alpha","omega"]
                                        Type: List String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StringXmpPageEmpty18}
\begin{paste}{StringXmpPageEmpty18}{StringXmpPagePatch18}
\pastebutton{StringXmpPageEmpty18}{\showpaste}
\tab{5}\spadcommand{split(saidsaw, other)\free{saidsaw other }}
\end{paste}\end{patch}

\begin{patch}{StringXmpPagePatch19}
\begin{paste}{StringXmpPageFull19}{StringXmpPageEmpty19}
\pastebutton{StringXmpPageFull19}{\hidepaste}
\tab{5}\spadcommand{trim ("\#\# ++ relax ++ \#\#", char "\#")}
\indentrel{3}\begin{verbatim}
   (19)  " ++ relax ++ "
                                             Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StringXmpPageEmpty19}
\begin{paste}{StringXmpPageEmpty19}{StringXmpPagePatch19}
\pastebutton{StringXmpPageEmpty19}{\showpaste}
\tab{5}\spadcommand{trim ("\#\# ++ relax ++ \#\#", char "\#")}
\end{paste}\end{patch}

\begin{patch}{StringXmpPagePatch20}
\begin{paste}{StringXmpPageFull20}{StringXmpPageEmpty20}
\pastebutton{StringXmpPageFull20}{\hidepaste}
\tab{5}\spadcommand{trim ("\#\# ++ relax ++ \#\#", other)\free{other }}
\indentrel{3}\begin{verbatim}
   (20)  "relax"
                                             Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StringXmpPageEmpty20}
\begin{paste}{StringXmpPageEmpty20}{StringXmpPagePatch20}
\pastebutton{StringXmpPageEmpty20}{\showpaste}
\tab{5}\spadcommand{trim ("\#\# ++ relax ++ \#\#", other)\free{other }}
\end{paste}\end{patch}

\begin{patch}{StringXmpPagePatch21}
\begin{paste}{StringXmpPageFull21}{StringXmpPageEmpty21}
\pastebutton{StringXmpPageFull21}{\hidepaste}
\tab{5}\spadcommand{leftTrim ("\#\# ++ relax ++ \#\#", other)\free{other }}
\indentrel{3}\begin{verbatim}
   (21)  "relax ++ ##"
                                             Type: String
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StringXmpPageEmpty21}
\begin{paste}{StringXmpPageEmpty21}{StringXmpPagePatch21}
\pastebutton{StringXmpPageEmpty21}{\showpaste}
\tab{5}\spadcommand{leftTrim ("\#\# ++ relax ++ \#\#", other)\free{other }}
\end{paste}\end{patch}

\begin{patch}{StringXmpPagePatch22}
\begin{paste}{StringXmpPageFull22}{StringXmpPageEmpty22}
\pastebutton{StringXmpPageFull22}{\hidepaste}
\tab{5}\spadcommand{rightTrim("\#\# ++ relax ++ \#\#", other)\free{other }}
\indentrel{3}\begin{verbatim}
   (22)  "## ++ relax"
                                          Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StringXmpPageEmpty22}
\begin{paste}{StringXmpPageEmpty22}{StringXmpPagePatch22}
\pastebutton{StringXmpPageEmpty22}{\showpaste}
\tab{5}\spadcommand{rightTrim("\#\# ++ relax ++ \#\#", other)\free{other }}
\end{paste}\end{patch}

\begin{patch}{StringXmpPagePatch23}
\begin{paste}{StringXmpPageFull23}{StringXmpPageEmpty23}
\pastebutton{StringXmpPageFull23}{\hidepaste}
\tab{5}\spadcommand{upperCase hello\free{hello }}
\indentrel{3}\begin{verbatim}
   (23)  "HELLO, I'M AXIOM!"
                                          Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StringXmpPageEmpty23}
\begin{paste}{StringXmpPageEmpty23}{StringXmpPagePatch23}
\pastebutton{StringXmpPageEmpty23}{\showpaste}
\tab{5}\spadcommand{upperCase hello\free{hello }}
\end{paste}\end{patch}

\begin{patch}{StringXmpPagePatch24}
\begin{paste}{StringXmpPageFull24}{StringXmpPageEmpty24}
\pastebutton{StringXmpPageFull24}{\hidepaste}
\tab{5}\spadcommand{lowerCase hello\free{hello }}
\indentrel{3}\begin{verbatim}
   (24)  "hello, i'm axiom!"
                                          Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{StringXmpPageEmpty24}
\begin{paste}{StringXmpPageEmpty24}{StringXmpPagePatch24}
\pastebutton{StringXmpPageEmpty24}{\showpaste}
\tab{5}\spadcommand{lowerCase hello\free{hello }}
\end{paste}\end{patch}

\begin{patch}{StringXmpPagePatch25}
\begin{paste}{StringXmpPageFull25}{StringXmpPageEmpty25}
\pastebutton{StringXmpPageFull25}{\hidepaste}
\tab{5}\spadcommand{prefix?("He", "Hello")}
\indentrel{3}\begin{verbatim}
   (25)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StringXmpPageEmpty25}
\begin{paste}{StringXmpPageEmpty25}{StringXmpPagePatch25}
\pastebutton{StringXmpPageEmpty25}{\showpaste}
\tab{5}\spadcommand{prefix?("He", "Hello")}
\end{paste}\end{patch}

\begin{patch}{StringXmpPagePatch26}
\begin{paste}{StringXmpPageFull26}{StringXmpPageEmpty26}
\pastebutton{StringXmpPageFull26}{\hidepaste}
\tab{5}\spadcommand{prefix?("Her", "Hello")}
\indentrel{3}\begin{verbatim}
   (26)  false
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StringXmpPageEmpty26}
\begin{paste}{StringXmpPageEmpty26}{StringXmpPagePatch26}
\pastebutton{StringXmpPageEmpty26}{\showpaste}
\tab{5}\spadcommand{prefix?("Her", "Hello")}
\end{paste}\end{patch}

\begin{patch}{StringXmpPagePatch27}
\begin{paste}{StringXmpPageFull27}{StringXmpPageEmpty27}
\pastebutton{StringXmpPageFull27}{\hidepaste}
\tab{5}\spadcommand{suffix?("", "Hello")}
\indentrel{3}\begin{verbatim}
   (27)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StringXmpPageEmpty27}
```

```
\begin{paste}{StringXmpPageEmpty27}{StringXmpPagePatch27}
\pastebutton{StringXmpPageEmpty27}{\showpaste}
\tab{5}\spadcommand{suffix?("", "Hello")}
\end{paste}\end{patch}

\begin{patch}{StringXmpPagePatch28}
\begin{paste}{StringXmpPageFull28}{StringXmpPageEmpty28}
\pastebutton{StringXmpPageFull28}{\hidepaste}
\tab{5}\spadcommand{suffix?("LO", "Hello")}
\indentrel{3}\begin{verbatim}
   (28)  false
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StringXmpPageEmpty28}
\begin{paste}{StringXmpPageEmpty28}{StringXmpPagePatch28}
\pastebutton{StringXmpPageEmpty28}{\showpaste}
\tab{5}\spadcommand{suffix?("LO", "Hello")}
\end{paste}\end{patch}

\begin{patch}{StringXmpPagePatch29}
\begin{paste}{StringXmpPageFull29}{StringXmpPageEmpty29}
\pastebutton{StringXmpPageFull29}{\hidepaste}
\tab{5}\spadcommand{substring?("ll", "Hello", 3)}
\indentrel{3}\begin{verbatim}
   (29)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StringXmpPageEmpty29}
\begin{paste}{StringXmpPageEmpty29}{StringXmpPagePatch29}
\pastebutton{StringXmpPageEmpty29}{\showpaste}
\tab{5}\spadcommand{substring?("ll", "Hello", 3)}
\end{paste}\end{patch}

\begin{patch}{StringXmpPagePatch30}
\begin{paste}{StringXmpPageFull30}{StringXmpPageEmpty30}
\pastebutton{StringXmpPageFull30}{\hidepaste}
\tab{5}\spadcommand{substring?("ll", "Hello", 4)}
\indentrel{3}\begin{verbatim}
   (30)  false
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StringXmpPageEmpty30}
\begin{paste}{StringXmpPageEmpty30}{StringXmpPagePatch30}
\pastebutton{StringXmpPageEmpty30}{\showpaste}
```

```
\tab{5}\spadcommand{substring?("ll", "Hello", 4)}
\end{paste}\end{patch}

\begin{patch}{StringXmpPagePatch31}
\begin{paste}{StringXmpPageFull31}{StringXmpPageEmpty31}
\pastebutton{StringXmpPageFull31}{\hidepaste}
\tab{5}\spadcommand{n := position("nd", "underground", 1)\bound{n }}
\indentrel{3}\begin{verbatim}
   (31)  2
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StringXmpPageEmpty31}
\begin{paste}{StringXmpPageEmpty31}{StringXmpPagePatch31}
\pastebutton{StringXmpPageEmpty31}{\showpaste}
\tab{5}\spadcommand{n := position("nd", "underground", 1)\bound{n }}
\end{paste}\end{patch}

\begin{patch}{StringXmpPagePatch32}
\begin{paste}{StringXmpPageFull32}{StringXmpPageEmpty32}
\pastebutton{StringXmpPageFull32}{\hidepaste}
\tab{5}\spadcommand{n := position("nd", "underground", n+1)\free{n }\bound{n1 }}
\indentrel{3}\begin{verbatim}
   (32)  10
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StringXmpPageEmpty32}
\begin{paste}{StringXmpPageEmpty32}{StringXmpPagePatch32}
\pastebutton{StringXmpPageEmpty32}{\showpaste}
\tab{5}\spadcommand{n := position("nd", "underground", n+1)\free{n }\bound{n1 }}
\end{paste}\end{patch}

\begin{patch}{StringXmpPagePatch33}
\begin{paste}{StringXmpPageFull33}{StringXmpPageEmpty33}
\pastebutton{StringXmpPageFull33}{\hidepaste}
\tab{5}\spadcommand{n := position("nd", "underground", n+1)\free{n1 }\bound{n2 }}
\indentrel{3}\begin{verbatim}
   (33)  0
                               Type: NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StringXmpPageEmpty33}
\begin{paste}{StringXmpPageEmpty33}{StringXmpPagePatch33}
\pastebutton{StringXmpPageEmpty33}{\showpaste}
\tab{5}\spadcommand{n := position("nd", "underground", n+1)\free{n1 }\bound{n2 }}
\end{paste}\end{patch}
```

```
\begin{patch}{StringXmpPagePatch34}
\begin{paste}{StringXmpPageFull34}{StringXmpPageEmpty34}
\pastebutton{StringXmpPageFull34}{\hidepaste}
\tab{5}\spadcommand{position(char "d", "underground", 1)}
\indentrel{3}\begin{verbatim}
   (34)  3
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StringXmpPageEmpty34}
\begin{paste}{StringXmpPageEmpty34}{StringXmpPagePatch34}
\pastebutton{StringXmpPageEmpty34}{\showpaste}
\tab{5}\spadcommand{position(char "d", "underground", 1)}
\end{paste}\end{patch}

\begin{patch}{StringXmpPagePatch35}
\begin{paste}{StringXmpPageFull35}{StringXmpPageEmpty35}
\pastebutton{StringXmpPageFull35}{\hidepaste}
\tab{5}\spadcommand{position(hexDigit(), "underground", 1)}
\indentrel{3}\begin{verbatim}
   (35)  3
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StringXmpPageEmpty35}
\begin{paste}{StringXmpPageEmpty35}{StringXmpPagePatch35}
\pastebutton{StringXmpPageEmpty35}{\showpaste}
\tab{5}\spadcommand{position(hexDigit(), "underground", 1)}
\end{paste}\end{patch}
```

————-

3.104 strtbl.ht

StringTable

⇒ “notitle” (TableXmpPage) 3.106 on page 1298

— **strtbl.ht** —

```
\begin{page}{StringTableXmpPage}{StringTable}
\beginscroll
%
This domain provides a table type in which the keys are known to
```

```
be strings so special techniques can be used.
Other than performance, the type \spadtype{StringTable(S)} should
behave exactly the same way as \spadtype{Table(String,S)}.
See \downlink{`Table'}{TableXmpPage}\ignore{Table}
for general information about tables.
\showBlurb{StringTable}

\xtc{
This creates a new table whose keys are strings.
}{
\spadpaste{t: StringTable(Integer) := table()  \bound{t}}
}
\xtc{
The value associated with each string key is the number of
characters in the string.
}{
\begin{spadsrc}[\free{t}\bound{h}]
for s in split("My name is Ian Watt.",char " ")
  repeat
    t.s := #s
\end{spadsrc}
}
\xtc{
}{
\spadpaste{for key in keys t repeat output [key, t.key] \free{t h}}
}
\endscroll
\autobuttons
\end{page}

\begin{patch}{StringTableXmpPagePatch1}
\begin{paste}{StringTableXmpPageFull1}{StringTableXmpPageEmpty1}
\pastebutton{StringTableXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{t: StringTable(Integer) := table()\bound{t }}
\indentrel{3}\begin{verbatim}
   (1)  table()
                              Type: StringTable Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StringTableXmpPageEmpty1}
\begin{paste}{StringTableXmpPageEmpty1}{StringTableXmpPagePatch1}
\pastebutton{StringTableXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{t: StringTable(Integer) := table()\bound{t }}
\end{paste}\end{patch}

\begin{patch}{StringTableXmpPagePatch2}
\begin{paste}{StringTableXmpPageFull2}{StringTableXmpPageEmpty2}
\pastebutton{StringTableXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{for s in split("My name is Ian Watt.",char " ")
```

```
  repeat
    t.s := \#s
\free{t }\bound{h }}
\indentrel{3}\begin{verbatim}
                                          Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StringTableXmpPageEmpty2}
\begin{paste}{StringTableXmpPageEmpty2}{StringTableXmpPagePatch2}
\pastebutton{StringTableXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{for s in split("My name is Ian Watt.",char " ")
  repeat
    t.s := \#s
\free{t }\bound{h }}
\end{paste}\end{patch}

\begin{patch}{StringTableXmpPagePatch3}
\begin{paste}{StringTableXmpPageFull3}{StringTableXmpPageEmpty3}
\pastebutton{StringTableXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{for key in keys t repeat output [key, t.key]\free{t h }}
\indentrel{3}\begin{verbatim}
   ["Ian",3]
   ["My",2]
   ["Watt.",5]
   ["name",4]
   ["is",2]
                                          Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{StringTableXmpPageEmpty3}
\begin{paste}{StringTableXmpPageEmpty3}{StringTableXmpPagePatch3}
\pastebutton{StringTableXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{for key in keys t repeat output [key, t.key]\free{t h }}
\end{paste}\end{patch}
```

————-

3.105 symbol.ht

Symbol

— symbol.ht —

```
\begin{page}{SymbolXmpPage}{Symbol}
```

```
\beginscroll

Symbols are one of the basic types manipulated by Axiom.
The \spadtype{Symbol} domain provides ways to create
symbols of many varieties.
\showBlurb{Symbol}

\xtc{
The simplest way to create a symbol is to ``single quote'' an identifier.
}{
\spadpaste{X: Symbol := 'x \bound{X}}
}
\xtc{
This gives the symbol even if \spad{x} has been assigned a value.
If \spad{x} has not been assigned a value, then it is possible to omit
the quote.
}{
\spadpaste{XX: Symbol := x}
}
\xtc{
Declarations must be used when working
with symbols, because otherwise the interpreter tries to place
values in a more specialized type \spadtype{Variable}.
}{
\spadpaste{A := 'a}
}
\xtc{
}{
\spadpaste{B := b}
}
\xtc{
The normal way of entering polynomials uses this fact.
}{
\spadpaste{x**2 + 1}
}

\xtc{
Another convenient way to create symbols is to convert a string.
This is useful when the name is to be constructed by a program.
}{
\spadpaste{"Hello"::Symbol}
}
\xtc{
Sometimes it is necessary to generate new unique symbols, for example, to
name constants of integration.
The expression \spad{new()} generates a symbol starting with \spad{\%}.
}{
\spadpaste{new()\$Symbol}
}
\xtc{
```

```
Successive calls to \spadfunFrom{new}{Symbol} produce different symbols.
}{
\spadpaste{new()\$Symbol}
}
\xtc{
The expression \spad{new("s")} produces a symbol starting with \spad{\%s}.
}{
\spadpaste{new("xyz")\$Symbol}
}

\xtc{
A symbol can be adorned in various ways.
The most basic thing is applying a symbol to a list
of subscripts.
}{
\spadpaste{X[i,j] \free{X}}
}

\xtc{
Somewhat less pretty is to attach subscripts, superscripts or arguments.
}{
\spadpaste{U := subscript(u, [1,2,1,2]) \bound{U}}
}
\xtc{
}{
\spadpaste{V := superscript(v, [n]) \bound{V}}
}
\xtc{
}{
\spadpaste{P := argscript(p, [t]) \bound{P}}
}

\xtc{
It is possible to test whether a symbol has scripts using the
\spadfunFrom{scripted?}{Symbol} test.
}{
\spadpaste{scripted? U \free{U}}
}
\xtc{
}{
\spadpaste{scripted? X \free{X}}
}
\xtc{
If a symbol is not scripted, then it may be converted to a string.
}{
\spadpaste{string X \free{X}}
}
\xtc{
The basic parts can always be extracted using the
\spadfunFrom{name}{Symbol} and \spadfunFrom{scripts}{Symbol} operations.
```

```
}{
\spadpaste{name U \free{U}}
}
\xtc{
}{
\spadpaste{scripts U \free{U}}
}
\xtc{
}{
\spadpaste{name X \free{X}}
}
\xtc{
}{
\spadpaste{scripts X \free{X}}
}

\xtc{
The most general form is obtained using the \spadfunFrom{script}{Symbol}
operation.
This operation takes an argument which is a list containing, in this order,
lists of subscripts, superscripts, presuperscripts, presubscripts and
arguments to a symbol.
}{
\spadpaste{M := script(Mammoth, [[i,j],[k,l],[0,1],[2],[u,v,w]])
\bound{M}}
}
\xtc{
}{
\spadpaste{scripts M \free{M}}
}
\xtc{
If trailing lists of scripts are omitted, they are assumed to be empty.
}{
\spadpaste{N := script(Nut, [[i,j],[k,l],[0,1]]) \bound{N}}
}
\xtc{
}{
\spadpaste{scripts N \free{N}}
}
\endscroll
\autobuttons
\end{page}

\begin{patch}{SymbolXmpPagePatch1}
\begin{paste}{SymbolXmpPageFull1}{SymbolXmpPageEmpty1}
\pastebutton{SymbolXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{X: Symbol := 'x\bound{X }}
\indentrel{3}\begin{verbatim}
   (1)  x
                                          Type: Symbol
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SymbolXmpPageEmpty1}
\begin{paste}{SymbolXmpPageEmpty1}{SymbolXmpPagePatch1}
\pastebutton{SymbolXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{X: Symbol := 'x\bound{X }}
\end{paste}\end{patch}

\begin{patch}{SymbolXmpPagePatch2}
\begin{paste}{SymbolXmpPageFull2}{SymbolXmpPageEmpty2}
\pastebutton{SymbolXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{XX: Symbol := x}
\indentrel{3}\begin{verbatim}
   (2)  x
                                          Type: Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SymbolXmpPageEmpty2}
\begin{paste}{SymbolXmpPageEmpty2}{SymbolXmpPagePatch2}
\pastebutton{SymbolXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{XX: Symbol := x}
\end{paste}\end{patch}

\begin{patch}{SymbolXmpPagePatch3}
\begin{paste}{SymbolXmpPageFull3}{SymbolXmpPageEmpty3}
\pastebutton{SymbolXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{A := 'a}
\indentrel{3}\begin{verbatim}
   (3)  a
                                      Type: Variable a
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SymbolXmpPageEmpty3}
\begin{paste}{SymbolXmpPageEmpty3}{SymbolXmpPagePatch3}
\pastebutton{SymbolXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{A := 'a}
\end{paste}\end{patch}

\begin{patch}{SymbolXmpPagePatch4}
\begin{paste}{SymbolXmpPageFull4}{SymbolXmpPageEmpty4}
\pastebutton{SymbolXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{B := b}
\indentrel{3}\begin{verbatim}
   (4)  b
                                      Type: Variable b
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{SymbolXmpPageEmpty4}
\begin{paste}{SymbolXmpPageEmpty4}{SymbolXmpPagePatch4}
\pastebutton{SymbolXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{B := b}
\end{paste}\end{patch}

\begin{patch}{SymbolXmpPagePatch5}
\begin{paste}{SymbolXmpPageFull5}{SymbolXmpPageEmpty5}
\pastebutton{SymbolXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{x**2 + 1}
\indentrel{3}\begin{verbatim}
         2
   (5)  x  + 1
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SymbolXmpPageEmpty5}
\begin{paste}{SymbolXmpPageEmpty5}{SymbolXmpPagePatch5}
\pastebutton{SymbolXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{x**2 + 1}
\end{paste}\end{patch}

\begin{patch}{SymbolXmpPagePatch6}
\begin{paste}{SymbolXmpPageFull6}{SymbolXmpPageEmpty6}
\pastebutton{SymbolXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{"Hello"::Symbol}
\indentrel{3}\begin{verbatim}
   (6)  Hello
                                           Type: Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SymbolXmpPageEmpty6}
\begin{paste}{SymbolXmpPageEmpty6}{SymbolXmpPagePatch6}
\pastebutton{SymbolXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{"Hello"::Symbol}
\end{paste}\end{patch}

\begin{patch}{SymbolXmpPagePatch7}
\begin{paste}{SymbolXmpPageFull7}{SymbolXmpPageEmpty7}
\pastebutton{SymbolXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{new()$Symbol}
\indentrel{3}\begin{verbatim}
   (7)  %A
                                           Type: Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{SymbolXmpPageEmpty7}
\begin{paste}{SymbolXmpPageEmpty7}{SymbolXmpPagePatch7}
\pastebutton{SymbolXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{new()$Symbol}
\end{paste}\end{patch}

\begin{patch}{SymbolXmpPagePatch8}
\begin{paste}{SymbolXmpPageFull8}{SymbolXmpPageEmpty8}
\pastebutton{SymbolXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{new()$Symbol}
\indentrel{3}\begin{verbatim}
   (8)  %B
                                            Type: Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SymbolXmpPageEmpty8}
\begin{paste}{SymbolXmpPageEmpty8}{SymbolXmpPagePatch8}
\pastebutton{SymbolXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{new()$Symbol}
\end{paste}\end{patch}

\begin{patch}{SymbolXmpPagePatch9}
\begin{paste}{SymbolXmpPageFull9}{SymbolXmpPageEmpty9}
\pastebutton{SymbolXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{new("xyz")$Symbol}
\indentrel{3}\begin{verbatim}
   (9)  %xyz0
                                            Type: Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SymbolXmpPageEmpty9}
\begin{paste}{SymbolXmpPageEmpty9}{SymbolXmpPagePatch9}
\pastebutton{SymbolXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{new("xyz")$Symbol}
\end{paste}\end{patch}

\begin{patch}{SymbolXmpPagePatch10}
\begin{paste}{SymbolXmpPageFull10}{SymbolXmpPageEmpty10}
\pastebutton{SymbolXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{X[i,j]\free{X }}
\indentrel{3}\begin{verbatim}
   (10)  x
          i,j
                                            Type: Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SymbolXmpPageEmpty10}
```

```
\begin{paste}{SymbolXmpPageEmpty10}{SymbolXmpPagePatch10}
\pastebutton{SymbolXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{X[i,j]\free{X }}
\end{paste}\end{patch}

\begin{patch}{SymbolXmpPagePatch11}
\begin{paste}{SymbolXmpPageFull11}{SymbolXmpPageEmpty11}
\pastebutton{SymbolXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{U := subscript(u, [1,2,1,2])\bound{U }}
\indentrel{3}\begin{verbatim}
   (11)  u
          1,2,1,2
                                            Type: Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SymbolXmpPageEmpty11}
\begin{paste}{SymbolXmpPageEmpty11}{SymbolXmpPagePatch11}
\pastebutton{SymbolXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{U := subscript(u, [1,2,1,2])\bound{U }}
\end{paste}\end{patch}

\begin{patch}{SymbolXmpPagePatch12}
\begin{paste}{SymbolXmpPageFull12}{SymbolXmpPageEmpty12}
\pastebutton{SymbolXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{V := superscript(v, [n])\bound{V }}
\indentrel{3}\begin{verbatim}
          n
   (12)  v
                                            Type: Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SymbolXmpPageEmpty12}
\begin{paste}{SymbolXmpPageEmpty12}{SymbolXmpPagePatch12}
\pastebutton{SymbolXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{V := superscript(v, [n])\bound{V }}
\end{paste}\end{patch}

\begin{patch}{SymbolXmpPagePatch13}
\begin{paste}{SymbolXmpPageFull13}{SymbolXmpPageEmpty13}
\pastebutton{SymbolXmpPageFull13}{\hidepaste}
\tab{5}\spadcommand{P := argscript(p, [t])\bound{P }}
\indentrel{3}\begin{verbatim}
   (13)  p(t)
                                            Type: Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SymbolXmpPageEmpty13}
```

```
\begin{paste}{SymbolXmpPageEmpty13}{SymbolXmpPagePatch13}
\pastebutton{SymbolXmpPageEmpty13}{\showpaste}
\tab{5}\spadcommand{P := argscript(p, [t])\bound{P }}
\end{paste}\end{patch}

\begin{patch}{SymbolXmpPagePatch14}
\begin{paste}{SymbolXmpPageFull14}{SymbolXmpPageEmpty14}
\pastebutton{SymbolXmpPageFull14}{\hidepaste}
\tab{5}\spadcommand{scripted? U\free{U }}
\indentrel{3}\begin{verbatim}
   (14)  true
                                           Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SymbolXmpPageEmpty14}
\begin{paste}{SymbolXmpPageEmpty14}{SymbolXmpPagePatch14}
\pastebutton{SymbolXmpPageEmpty14}{\showpaste}
\tab{5}\spadcommand{scripted? U\free{U }}
\end{paste}\end{patch}

\begin{patch}{SymbolXmpPagePatch15}
\begin{paste}{SymbolXmpPageFull15}{SymbolXmpPageEmpty15}
\pastebutton{SymbolXmpPageFull15}{\hidepaste}
\tab{5}\spadcommand{scripted? X\free{X }}
\indentrel{3}\begin{verbatim}
   (15)  false
                                           Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SymbolXmpPageEmpty15}
\begin{paste}{SymbolXmpPageEmpty15}{SymbolXmpPagePatch15}
\pastebutton{SymbolXmpPageEmpty15}{\showpaste}
\tab{5}\spadcommand{scripted? X\free{X }}
\end{paste}\end{patch}

\begin{patch}{SymbolXmpPagePatch16}
\begin{paste}{SymbolXmpPageFull16}{SymbolXmpPageEmpty16}
\pastebutton{SymbolXmpPageFull16}{\hidepaste}
\tab{5}\spadcommand{string X\free{X }}
\indentrel{3}\begin{verbatim}
   (16)  "x"
                                            Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SymbolXmpPageEmpty16}
\begin{paste}{SymbolXmpPageEmpty16}{SymbolXmpPagePatch16}
\pastebutton{SymbolXmpPageEmpty16}{\showpaste}
```

```
\tab{5}\spadcommand{string X\free{X }}
\end{paste}\end{patch}

\begin{patch}{SymbolXmpPagePatch17}
\begin{paste}{SymbolXmpPageFull17}{SymbolXmpPageEmpty17}
\pastebutton{SymbolXmpPageFull17}{\hidepaste}
\tab{5}\spadcommand{name U\free{U }}
\indentrel{3}\begin{verbatim}
   (17)  u
                                            Type: Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SymbolXmpPageEmpty17}
\begin{paste}{SymbolXmpPageEmpty17}{SymbolXmpPagePatch17}
\pastebutton{SymbolXmpPageEmpty17}{\showpaste}
\tab{5}\spadcommand{name U\free{U }}
\end{paste}\end{patch}

\begin{patch}{SymbolXmpPagePatch18}
\begin{paste}{SymbolXmpPageFull18}{SymbolXmpPageEmpty18}
\pastebutton{SymbolXmpPageFull18}{\hidepaste}
\tab{5}\spadcommand{scripts U\free{U }}
\indentrel{3}\begin{verbatim}
   (18)
   [sub= [1,2,1,2],sup= [],presup= [],presub= [],args= []]
Type: Record(sub: List OutputForm,sup: List OutputForm,presup: List OutputForm,presub: List OutputForm,args
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SymbolXmpPageEmpty18}
\begin{paste}{SymbolXmpPageEmpty18}{SymbolXmpPagePatch18}
\pastebutton{SymbolXmpPageEmpty18}{\showpaste}
\tab{5}\spadcommand{scripts U\free{U }}
\end{paste}\end{patch}

\begin{patch}{SymbolXmpPagePatch19}
\begin{paste}{SymbolXmpPageFull19}{SymbolXmpPageEmpty19}
\pastebutton{SymbolXmpPageFull19}{\hidepaste}
\tab{5}\spadcommand{name X\free{X }}
\indentrel{3}\begin{verbatim}
   (19)  x
                                            Type: Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SymbolXmpPageEmpty19}
\begin{paste}{SymbolXmpPageEmpty19}{SymbolXmpPagePatch19}
\pastebutton{SymbolXmpPageEmpty19}{\showpaste}
\tab{5}\spadcommand{name X\free{X }}
```

```
\end{paste}\end{patch}

\begin{patch}{SymbolXmpPagePatch20}
\begin{paste}{SymbolXmpPageFull20}{SymbolXmpPageEmpty20}
\pastebutton{SymbolXmpPageFull20}{\hidepaste}
\tab{5}\spadcommand{scripts X\free{X }}
\indentrel{3}\begin{verbatim}
   (20)
   [sub= [],sup= [],presup= [],presub= [],args= []]
Type: Record(sub: List OutputForm,sup: List OutputForm,presup: List OutputForm,presub: List Outp
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SymbolXmpPageEmpty20}
\begin{paste}{SymbolXmpPageEmpty20}{SymbolXmpPagePatch20}
\pastebutton{SymbolXmpPageEmpty20}{\showpaste}
\tab{5}\spadcommand{scripts X\free{X }}
\end{paste}\end{patch}

\begin{patch}{SymbolXmpPagePatch21}
\begin{paste}{SymbolXmpPageFull21}{SymbolXmpPageEmpty21}
\pastebutton{SymbolXmpPageFull21}{\hidepaste}
\tab{5}\spadcommand{M := script(Mammoth, [[i,j],[k,l],[0,1],[2],[u,v,w]])\bound{M }}
\indentrel{3}\begin{verbatim}
         0,1       k,l
   (21)     Mammoth   (u,v,w)
           2       i,j
                                          Type: Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SymbolXmpPageEmpty21}
\begin{paste}{SymbolXmpPageEmpty21}{SymbolXmpPagePatch21}
\pastebutton{SymbolXmpPageEmpty21}{\showpaste}
\tab{5}\spadcommand{M := script(Mammoth, [[i,j],[k,l],[0,1],[2],[u,v,w]])\bound{M }}
\end{paste}\end{patch}

\begin{patch}{SymbolXmpPagePatch22}
\begin{paste}{SymbolXmpPageFull22}{SymbolXmpPageEmpty22}
\pastebutton{SymbolXmpPageFull22}{\hidepaste}
\tab{5}\spadcommand{scripts M\free{M }}
\indentrel{3}\begin{verbatim}
   (22)
   [sub= [i,j], sup= [k,l], presup= [0,1], presub= [2],
    args= [u,v,w]]
Type: Record(sub: List OutputForm,sup: List OutputForm,presup: List OutputForm,presub: List Outp
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SymbolXmpPageEmpty22}
```

```
\begin{paste}{SymbolXmpPageEmpty22}{SymbolXmpPagePatch22}
\pastebutton{SymbolXmpPageEmpty22}{\showpaste}
\tab{5}\spadcommand{scripts M\free{M }}
\end{paste}\end{patch}

\begin{patch}{SymbolXmpPagePatch23}
\begin{paste}{SymbolXmpPageFull23}{SymbolXmpPageEmpty23}
\pastebutton{SymbolXmpPageFull23}{\hidepaste}
\tab{5}\spadcommand{N := script(Nut, [[i,j],[k,l],[0,1]])\bound{N }}
\indentrel{3}\begin{verbatim}
         0,1   k,l
   (23)     Nut
               i,j
                                           Type: Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SymbolXmpPageEmpty23}
\begin{paste}{SymbolXmpPageEmpty23}{SymbolXmpPagePatch23}
\pastebutton{SymbolXmpPageEmpty23}{\showpaste}
\tab{5}\spadcommand{N := script(Nut, [[i,j],[k,l],[0,1]])\bound{N }}
\end{paste}\end{patch}

\begin{patch}{SymbolXmpPagePatch24}
\begin{paste}{SymbolXmpPageFull24}{SymbolXmpPageEmpty24}
\pastebutton{SymbolXmpPageFull24}{\hidepaste}
\tab{5}\spadcommand{scripts N\free{N }}
\indentrel{3}\begin{verbatim}
   (24)
   [sub= [i,j], sup= [k,l], presup= [0,1], presub= [],
    args= []]
Type: Record(sub: List OutputForm,sup: List OutputForm,presup: List OutputForm,presub: List OutputForm,args
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{SymbolXmpPageEmpty24}
\begin{paste}{SymbolXmpPageEmpty24}{SymbolXmpPagePatch24}
\pastebutton{SymbolXmpPageEmpty24}{\showpaste}
\tab{5}\spadcommand{scripts N\free{N }}
\end{paste}\end{patch}
```

3.106 table.ht

Table

⇒ “notitle” (AssociationListXmpPage) 3.3 on page 132
⇒ “notitle” (EqTableXmpPage) 3.26 on page 386
⇒ “notitle” (StringTableXmpPage) 3.104 on page 1284
⇒ “notitle” (SparseTableXmpPage) 3.101 on page 1259
⇒ “notitle” (KeyedAccessFileXmpPage) 3.57 on page 734

— **table.ht** —

```
\begin{page}{TableXmpPage}{Table}
\beginscroll

The \spadtype{Table} constructor provides a general structure for
associative storage.  This type provides hash tables in which data
objects can be saved according to keys of any type.  For a given
table, specific types must be chosen for the keys and entries.

\xtc{
In this example the keys to the table are polynomials with
integer coefficients.
The entries in the table are strings.
}{
\spadpaste{t: Table(Polynomial Integer, String) := table() \bound{t}}
}
\xtc{
To save an entry in the table, the \spadfunFrom{setelt}{Table}
operation is used.
This can be called directly, giving the table a key and an entry.
}{
\spadpaste{setelt(t, x**2 - 1, "Easy to factor") \bound{p1}\free{t}}
}
\xtc{
Alternatively, you can use assignment syntax.
}{
\spadpaste{t(x**3 + 1) := "Harder to factor" \bound{p2}\free{p1}}
}
\xtc{
}{
\spadpaste{t(x)        := "The easiest to factor" \bound{p3}\free{p2}}
}
\xtc{
Entries are retrieved from the table by calling the
\spadfunFrom{elt}{Table} operation.
}{
\spadpaste{elt(t, x) \free{p3}}
}
```

```
\xtc{
This operation is called when a table is ‘‘applied’’ to a key using
this or the following syntax.
}{
\spadpaste{t.x \free{p3}}
}
\xtc{
}{
\spadpaste{t x \free{p3}}
}
\xtc{
Parentheses are used only for grouping.  They are needed if the key is
an infixed expression.
}{
\spadpaste{t.(x**2 - 1) \free{p3}}
}
\xtc{
Note that the \spadfunFrom{elt}{Table} operation is used only when the
key is known to be in the table---otherwise an error is generated.
}{
\spadpaste{t (x**3 + 1) \free{p3}}
}

\xtc{
You can get a list of all the keys to a table using the
\spadfunFrom{keys}{Table} operation.
}{
\spadpaste{keys t \free{p3}}
}
\xtc{
If you wish to test whether a key is in a table, the
\spadfunFrom{search}{Table} operation is used.
This operation returns either an entry or \spad{"failed"}.
}{
\spadpaste{search(x, t) \free{p3}}
}
\xtc{
}{
\spadpaste{search(x**2, t) \free{p3}}
}
\xtc{
The return type is a union so the success of the search can be tested
using \spad{case}.
\spadkey{case}
}{
\spadpaste{search(x**2, t) case "failed" \free{p3}}
}
\xtc{
The \spadfunFromX{remove}{Table} operation is used to delete values from a
table.
```

```
}{
\spadpaste{remove!(x**2-1, t) \free{p3} \bound{p4}}
}
\xtc{
If an entry exists under the key, then it is returned.  Otherwise
\spadfunFromX{remove}{Table} returns \spad{"failed"}.
}{
\spadpaste{remove!(x-1, t) \free{p4}\bound{p5}}
}

\xtc{
The number of key-entry pairs can be found using the
\spadfunFrom{\#}{Table} operation.
}{
\spadpaste{\#t \free{p5}}
}
\xtc{
Just as \spadfunFrom{keys}{Table} returns a list of keys to the table, a
list of all the entries can be obtained using the
\spadfunFrom{members}{Table} operation.
}{
\spadpaste{members t \free{p5}}
}
\xtc{
A number of useful operations take functions and map them on to the
table to compute the result.  Here we count the entries which
have \spad{"Hard"} as a prefix.
}{
\spadpaste{count(s: String +-> prefix?("Hard", s), t) \free{p5}}
}

Other table types are provided to support various needs.
\indent{4}
\beginitems
\item[-] \spadtype{AssociationList} gives a list with a table view.
This allows new entries to be appended onto the front of the list to
cover up old entries.  This is useful when table entries need to be
stacked or when frequent list traversals are required.  See
\downlink{`AssociationList'}{AssociationListXmpPage}
\ignore{AssociationList} for more information.

\item[-] \spadtype{EqTable} gives tables in which keys are considered
equal only when they are in fact the same instance of a structure.
See \downlink{`EqTable'}{EqTableXmpPage}\ignore{EqTable} for more
information.

\item[-] \spadtype{StringTable} should be used when the keys are known
to be strings.
See \downlink{`StringTable'}{StringTableXmpPage}\ignore{StringTable}
for more information.
```

```
\item[-] \spadtype{SparseTable} provides tables with default entries,
so lookup never fails.  The \spadtype{GeneralSparseTable} constructor
can be used to make any table type behave this way.  See
\downlink{`SparseTable'}{SparseTableXmpPage}\ignore{SparseTable} for
more information.

\item[-] \spadtype{KeyedAccessFile} allows values to be saved in a
file, accessed as a table.  See
\downlink{`KeyedAccessFile'}{KeyedAccessFileXmpPage}
\ignore{KeyedAccessFile}
for more information.

\enditems
\indent{0}
%
\showBlurb{Table}
\endscroll
\autobuttons
\end{page}

\begin{patch}{TableXmpPagePatch1}
\begin{paste}{TableXmpPageFull1}{TableXmpPageEmpty1}
\pastebutton{TableXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{t: Table(Polynomial Integer, String) := table()\bound{t }}
\indentrel{3}\begin{verbatim}
   (1)  table()
                 Type: Table(Polynomial Integer,String)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TableXmpPageEmpty1}
\begin{paste}{TableXmpPageEmpty1}{TableXmpPagePatch1}
\pastebutton{TableXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{t: Table(Polynomial Integer, String) := table()\bound{t }}
\end{paste}\end{patch}

\begin{patch}{TableXmpPagePatch2}
\begin{paste}{TableXmpPageFull2}{TableXmpPageEmpty2}
\pastebutton{TableXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{setelt(t, x**2 - 1, "Easy to factor")\bound{p1 }\free{t }}
\indentrel{3}\begin{verbatim}
   (2)  "Easy to factor"
                                          Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TableXmpPageEmpty2}
\begin{paste}{TableXmpPageEmpty2}{TableXmpPagePatch2}
\pastebutton{TableXmpPageEmpty2}{\showpaste}
```

```
\tab{5}\spadcommand{setelt(t, x**2 - 1, "Easy to factor")\bound{p1 }\free{t }}
\end{paste}\end{patch}

\begin{patch}{TableXmpPagePatch3}
\begin{paste}{TableXmpPageFull3}{TableXmpPageEmpty3}
\pastebutton{TableXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{t(x**3 + 1) := "Harder to factor"\bound{p2 }\free{p1 }}
\indentrel{3}\begin{verbatim}
   (3)  "Harder to factor"
                                            Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TableXmpPageEmpty3}
\begin{paste}{TableXmpPageEmpty3}{TableXmpPagePatch3}
\pastebutton{TableXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{t(x**3 + 1) := "Harder to factor"\bound{p2 }\free{p1 }}
\end{paste}\end{patch}

\begin{patch}{TableXmpPagePatch4}
\begin{paste}{TableXmpPageFull4}{TableXmpPageEmpty4}
\pastebutton{TableXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{t(x) := "The easiest to factor"\bound{p3 }\free{p2 }}
\indentrel{3}\begin{verbatim}
   (4)  "The easiest to factor"
                                            Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TableXmpPageEmpty4}
\begin{paste}{TableXmpPageEmpty4}{TableXmpPagePatch4}
\pastebutton{TableXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{t(x) := "The easiest to factor"\bound{p3 }\free{p2 }}
\end{paste}\end{patch}

\begin{patch}{TableXmpPagePatch5}
\begin{paste}{TableXmpPageFull5}{TableXmpPageEmpty5}
\pastebutton{TableXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{elt(t, x)\free{p3 }}
\indentrel{3}\begin{verbatim}
   (5)  "The easiest to factor"
                                            Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TableXmpPageEmpty5}
\begin{paste}{TableXmpPageEmpty5}{TableXmpPagePatch5}
\pastebutton{TableXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{elt(t, x)\free{p3 }}
\end{paste}\end{patch}
```

```
\begin{patch}{TableXmpPagePatch6}
\begin{paste}{TableXmpPageFull6}{TableXmpPageEmpty6}
\pastebutton{TableXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{t.x\free{p3 }}
\indentrel{3}\begin{verbatim}
   (6)  "The easiest to factor"
                                           Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TableXmpPageEmpty6}
\begin{paste}{TableXmpPageEmpty6}{TableXmpPagePatch6}
\pastebutton{TableXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{t.x\free{p3 }}
\end{paste}\end{patch}

\begin{patch}{TableXmpPagePatch7}
\begin{paste}{TableXmpPageFull7}{TableXmpPageEmpty7}
\pastebutton{TableXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{t x\free{p3 }}
\indentrel{3}\begin{verbatim}
   (7)  "The easiest to factor"
                                           Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TableXmpPageEmpty7}
\begin{paste}{TableXmpPageEmpty7}{TableXmpPagePatch7}
\pastebutton{TableXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{t x\free{p3 }}
\end{paste}\end{patch}

\begin{patch}{TableXmpPagePatch8}
\begin{paste}{TableXmpPageFull8}{TableXmpPageEmpty8}
\pastebutton{TableXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{t.(x**2 - 1)\free{p3 }}
\indentrel{3}\begin{verbatim}
   (8)  "Easy to factor"
                                           Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TableXmpPageEmpty8}
\begin{paste}{TableXmpPageEmpty8}{TableXmpPagePatch8}
\pastebutton{TableXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{t.(x**2 - 1)\free{p3 }}
\end{paste}\end{patch}

\begin{patch}{TableXmpPagePatch9}
```

```
\begin{paste}{TableXmpPageFull9}{TableXmpPageEmpty9}
\pastebutton{TableXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{t (x**3 + 1)\free{p3 }}
\indentrel{3}\begin{verbatim}
   (9)  "Harder to factor"
                                          Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TableXmpPageEmpty9}
\begin{paste}{TableXmpPageEmpty9}{TableXmpPagePatch9}
\pastebutton{TableXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{t (x**3 + 1)\free{p3 }}
\end{paste}\end{patch}

\begin{patch}{TableXmpPagePatch10}
\begin{paste}{TableXmpPageFull10}{TableXmpPageEmpty10}
\pastebutton{TableXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{keys t\free{p3 }}
\indentrel{3}\begin{verbatim}
             3      2
   (10)  [x,x  + 1,x  - 1]
                          Type: List Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TableXmpPageEmpty10}
\begin{paste}{TableXmpPageEmpty10}{TableXmpPagePatch10}
\pastebutton{TableXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{keys t\free{p3 }}
\end{paste}\end{patch}

\begin{patch}{TableXmpPagePatch11}
\begin{paste}{TableXmpPageFull11}{TableXmpPageEmpty11}
\pastebutton{TableXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{search(x, t)\free{p3 }}
\indentrel{3}\begin{verbatim}
   (11)  "The easiest to factor"
                                Type: Union(String,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TableXmpPageEmpty11}
\begin{paste}{TableXmpPageEmpty11}{TableXmpPagePatch11}
\pastebutton{TableXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{search(x, t)\free{p3 }}
\end{paste}\end{patch}

\begin{patch}{TableXmpPagePatch12}
\begin{paste}{TableXmpPageFull12}{TableXmpPageEmpty12}
```

```
\pastebutton{TableXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{search(x**2, t)\free{p3 }}
\indentrel{3}\begin{verbatim}
   (12)  "failed"
                              Type: Union("failed",...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TableXmpPageEmpty12}
\begin{paste}{TableXmpPageEmpty12}{TableXmpPagePatch12}
\pastebutton{TableXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{search(x**2, t)\free{p3 }}
\end{paste}\end{patch}

\begin{patch}{TableXmpPagePatch13}
\begin{paste}{TableXmpPageFull13}{TableXmpPageEmpty13}
\pastebutton{TableXmpPageFull13}{\hidepaste}
\tab{5}\spadcommand{search(x**2, t) case "failed"\free{p3 }}
\indentrel{3}\begin{verbatim}
   (13)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TableXmpPageEmpty13}
\begin{paste}{TableXmpPageEmpty13}{TableXmpPagePatch13}
\pastebutton{TableXmpPageEmpty13}{\showpaste}
\tab{5}\spadcommand{search(x**2, t) case "failed"\free{p3 }}
\end{paste}\end{patch}

\begin{patch}{TableXmpPagePatch14}
\begin{paste}{TableXmpPageFull14}{TableXmpPageEmpty14}
\pastebutton{TableXmpPageFull14}{\hidepaste}
\tab{5}\spadcommand{remove!(x**2-1, t)\free{p3 }\bound{p4 }}
\indentrel{3}\begin{verbatim}
   (14)  "Easy to factor"
                                Type: Union(String,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TableXmpPageEmpty14}
\begin{paste}{TableXmpPageEmpty14}{TableXmpPagePatch14}
\pastebutton{TableXmpPageEmpty14}{\showpaste}
\tab{5}\spadcommand{remove!(x**2-1, t)\free{p3 }\bound{p4 }}
\end{paste}\end{patch}

\begin{patch}{TableXmpPagePatch15}
\begin{paste}{TableXmpPageFull15}{TableXmpPageEmpty15}
\pastebutton{TableXmpPageFull15}{\hidepaste}
\tab{5}\spadcommand{remove!(x-1, t)\free{p4 }\bound{p5 }}
```

```
\indentrel{3}\begin{verbatim}
   (15)  "failed"
                              Type: Union("failed",...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TableXmpPageEmpty15}
\begin{paste}{TableXmpPageEmpty15}{TableXmpPagePatch15}
\pastebutton{TableXmpPageEmpty15}{\showpaste}
\tab{5}\spadcommand{remove!(x-1, t)\free{p4 }\bound{p5 }}
\end{paste}\end{patch}

\begin{patch}{TableXmpPagePatch16}
\begin{paste}{TableXmpPageFull16}{TableXmpPageEmpty16}
\pastebutton{TableXmpPageFull16}{\hidepaste}
\tab{5}\spadcommand{\#t\free{p5 }}
\indentrel{3}\begin{verbatim}
   (16)  2
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TableXmpPageEmpty16}
\begin{paste}{TableXmpPageEmpty16}{TableXmpPagePatch16}
\pastebutton{TableXmpPageEmpty16}{\showpaste}
\tab{5}\spadcommand{\#t\free{p5 }}
\end{paste}\end{patch}

\begin{patch}{TableXmpPagePatch17}
\begin{paste}{TableXmpPageFull17}{TableXmpPageEmpty17}
\pastebutton{TableXmpPageFull17}{\hidepaste}
\tab{5}\spadcommand{members t\free{p5 }}
\indentrel{3}\begin{verbatim}
   (17)  ["The easiest to factor","Harder to factor"]
                                      Type: List String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TableXmpPageEmpty17}
\begin{paste}{TableXmpPageEmpty17}{TableXmpPagePatch17}
\pastebutton{TableXmpPageEmpty17}{\showpaste}
\tab{5}\spadcommand{members t\free{p5 }}
\end{paste}\end{patch}

\begin{patch}{TableXmpPagePatch18}
\begin{paste}{TableXmpPageFull18}{TableXmpPageEmpty18}
\pastebutton{TableXmpPageFull18}{\hidepaste}
\tab{5}\spadcommand{count(s: String +-> prefix?("Hard", s), t)\free{p5 }}
\indentrel{3}\begin{verbatim}
   (18)  1
```

```
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TableXmpPageEmpty18}
\begin{paste}{TableXmpPageEmpty18}{TableXmpPagePatch18}
\pastebutton{TableXmpPageEmpty18}{\showpaste}
\tab{5}\spadcommand{count(s: String +-> prefix?("Hard", s), t)\free{p5 }}
\end{paste}\end{patch}
```

————-

3.107 textfile.ht

TextFile

⇒ "notitle" (FileXmpPage) 3.40 on page 480
⇒ "notitle" (KeyedAccessFileXmpPage) 3.57 on page 734
⇒ "notitle" (LibraryXmpPage) 3.62 on page 840

— **textfile.ht** —

```
\begin{page}{TextFileXmpPage}{TextFile}
\beginscroll

The domain \spadtype{TextFile} allows Axiom to read and write
character data and exchange text with other programs.
This type behaves in Axiom much like a \spadtype{File} of strings,
with additional operations to cause new lines.
We give an example of how to produce an upper case copy of a file.
\xtc{
This is the file from which we read the text.
}{
\spadpaste{f1: TextFile := open("/etc/group", "input")   \bound{f1}}
}
\xtc{
This is the file to which we read the text.
}{
\spadpaste{f2: TextFile := open("/tmp/MOTD", "output")  \bound{f2}}
}
\xtc{
Entire lines are handled using the \spadfunFromX{readLine}{TextFile} and
\spadfunFromX{writeLine}{TextFile} operations.
}{
\spadpaste{l := readLine! f1 \free{f1}\bound{l}}
}
```

```
\xtc{
}{
\spadpaste{writeLine!(f2, upperCase l) \free{f2 l}}
}
\xtc{
Use the
\spadfunFrom{endOfFile?}{TextFile} operation to check if you have
reached the end of the file.
}{
\begin{spadsrc}[\free{f1 f2}\bound{Copied}]
while not endOfFile? f1 repeat
    s := readLine! f1
    writeLine!(f2, upperCase s)
\end{spadsrc}
}
\xtc{
The file \spad{f1} is exhausted and should be closed.
}{
\spadpaste{close! f1  \free{Copied}\bound{closed1}}
}

\xtc{
It is sometimes useful to write lines a bit at a time.
The \spadfunFromX{write}{TextFile} operation allows this.
}{
\spadpaste{write!(f2, "-The-")  \free{Copied}\bound{tthhee}}
}
\xtc{
}{
\spadpaste{write!(f2, "-End-")  \free{tthhee}\bound{eenndd}}
}
\xtc{
This ends the line.
This is done in a machine-dependent manner.
}{
\spadpaste{writeLine! f2        \free{eenndd}\bound{LastLine}}
}
\xtc{
}{
\spadpaste{close! f2              \free{LastLine}\bound{closed2}}
}
\noOutputXtc{
Finally, clean up.
}{
\spadpaste{)system rm /tmp/MOTD   \free{closed2}}
}

For more information on related topics,  see
\downlink{`File'}{FileXmpPage}\ignore{File},
\downlink{`KeyedAccessFile'}{KeyedAccessFileXmpPage}
```

```
\ignore{KeyedAccessFile}, and
\downlink{`Library'}{LibraryXmpPage}\ignore{Library}.
\showBlurb{TextFile}
\endscroll
\autobuttons
\end{page}

\begin{patch}{TextFileXmpPagePatch1}
\begin{paste}{TextFileXmpPageFull1}{TextFileXmpPageEmpty1}
\pastebutton{TextFileXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{f1: TextFile := open("/etc/group", "input")\bound{f1 }}
\indentrel{3}\begin{verbatim}
   (1)  "/etc/group"
                                        Type: TextFile
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TextFileXmpPageEmpty1}
\begin{paste}{TextFileXmpPageEmpty1}{TextFileXmpPagePatch1}
\pastebutton{TextFileXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{f1: TextFile := open("/etc/group", "input")\bound{f1 }}
\end{paste}\end{patch}

\begin{patch}{TextFileXmpPagePatch2}
\begin{paste}{TextFileXmpPageFull2}{TextFileXmpPageEmpty2}
\pastebutton{TextFileXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{f2: TextFile := open("/tmp/MOTD", "output")\bound{f2 }}
\indentrel{3}\begin{verbatim}
   (2)  "/tmp/MOTD"
                                        Type: TextFile
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TextFileXmpPageEmpty2}
\begin{paste}{TextFileXmpPageEmpty2}{TextFileXmpPagePatch2}
\pastebutton{TextFileXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{f2: TextFile := open("/tmp/MOTD", "output")\bound{f2 }}
\end{paste}\end{patch}

\begin{patch}{TextFileXmpPagePatch3}
\begin{paste}{TextFileXmpPageFull3}{TextFileXmpPageEmpty3}
\pastebutton{TextFileXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{l := readLine! f1\free{f1 }\bound{l }}
\indentrel{3}\begin{verbatim}
   (3)  "system:*:0:root"
                                          Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TextFileXmpPageEmpty3}
```

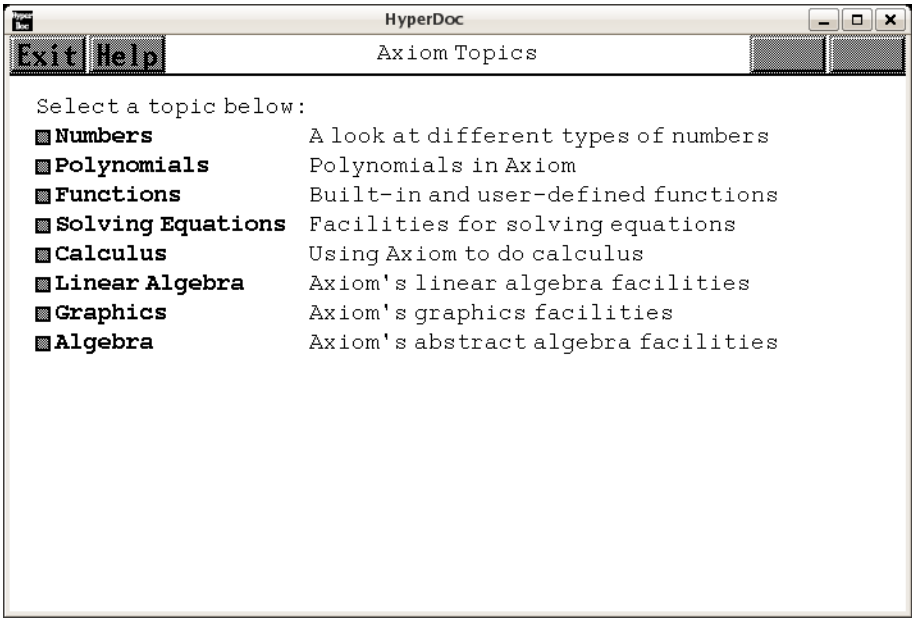
```
\begin{paste}{TextFileXmpPageEmpty3}{TextFileXmpPagePatch3}
\pastebutton{TextFileXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{l := readLine! f1\free{f1 }\bound{l }}
\end{paste}\end{patch}

\begin{patch}{TextFileXmpPagePatch4}
\begin{paste}{TextFileXmpPageFull4}{TextFileXmpPageEmpty4}
\pastebutton{TextFileXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{writeLine!(f2, upperCase l)\free{f2 l }}
\indentrel{3}\begin{verbatim}
   (4)  "SYSTEM:*:0:ROOT"
                                            Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TextFileXmpPageEmpty4}
\begin{paste}{TextFileXmpPageEmpty4}{TextFileXmpPagePatch4}
\pastebutton{TextFileXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{writeLine!(f2, upperCase l)\free{f2 l }}
\end{paste}\end{patch}

\begin{patch}{TextFileXmpPagePatch5}
\begin{paste}{TextFileXmpPageFull5}{TextFileXmpPageEmpty5}
\pastebutton{TextFileXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{while not endOfFile? f1 repeat
    s := readLine! f1
    writeLine!(f2, upperCase s)
\free{f1 f2 }\bound{Copied }}
\indentrel{3}\begin{verbatim}
                                              Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TextFileXmpPageEmpty5}
\begin{paste}{TextFileXmpPageEmpty5}{TextFileXmpPagePatch5}
\pastebutton{TextFileXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{while not endOfFile? f1 repeat
    s := readLine! f1
    writeLine!(f2, upperCase s)
\free{f1 f2 }\bound{Copied }}
\end{paste}\end{patch}

\begin{patch}{TextFileXmpPagePatch6}
\begin{paste}{TextFileXmpPageFull6}{TextFileXmpPageEmpty6}
\pastebutton{TextFileXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{close! f1\free{Copied }\bound{closed1 }}
\indentrel{3}\begin{verbatim}
   (6)  "/etc/group"
                                          Type: TextFile
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TextFileXmpPageEmpty6}
\begin{paste}{TextFileXmpPageEmpty6}{TextFileXmpPagePatch6}
\pastebutton{TextFileXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{close! f1\free{Copied }\bound{closed1 }}
\end{paste}\end{patch}

\begin{patch}{TextFileXmpPagePatch7}
\begin{paste}{TextFileXmpPageFull7}{TextFileXmpPageEmpty7}
\pastebutton{TextFileXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{write!(f2, "-The-")\free{Copied }\bound{tthhee }}
\indentrel{3}\begin{verbatim}
   (7)  "-The-"
                                          Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TextFileXmpPageEmpty7}
\begin{paste}{TextFileXmpPageEmpty7}{TextFileXmpPagePatch7}
\pastebutton{TextFileXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{write!(f2, "-The-")\free{Copied }\bound{tthhee }}
\end{paste}\end{patch}

\begin{patch}{TextFileXmpPagePatch8}
\begin{paste}{TextFileXmpPageFull8}{TextFileXmpPageEmpty8}
\pastebutton{TextFileXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{write!(f2, "-End-")\free{tthhee }\bound{eenndd }}
\indentrel{3}\begin{verbatim}
   (8)  "-End-"
                                          Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TextFileXmpPageEmpty8}
\begin{paste}{TextFileXmpPageEmpty8}{TextFileXmpPagePatch8}
\pastebutton{TextFileXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{write!(f2, "-End-")\free{tthhee }\bound{eenndd }}
\end{paste}\end{patch}

\begin{patch}{TextFileXmpPagePatch9}
\begin{paste}{TextFileXmpPageFull9}{TextFileXmpPageEmpty9}
\pastebutton{TextFileXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{writeLine! f2\free{eenndd }\bound{LastLine }}
\indentrel{3}\begin{verbatim}
   (9)  ""
                                          Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{TextFileXmpPageEmpty9}
\begin{paste}{TextFileXmpPageEmpty9}{TextFileXmpPagePatch9}
\pastebutton{TextFileXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{writeLine! f2\free{eenndd }\bound{LastLine }}
\end{paste}\end{patch}

\begin{patch}{TextFileXmpPagePatch10}
\begin{paste}{TextFileXmpPageFull10}{TextFileXmpPageEmpty10}
\pastebutton{TextFileXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{close! f2\free{LastLine }\bound{closed2 }}
\indentrel{3}\begin{verbatim}
   (10)  "/tmp/MOTD"
                                         Type: TextFile
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TextFileXmpPageEmpty10}
\begin{paste}{TextFileXmpPageEmpty10}{TextFileXmpPagePatch10}
\pastebutton{TextFileXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{close! f2\free{LastLine }\bound{closed2 }}
\end{paste}\end{patch}

\begin{patch}{TextFileXmpPagePatch11}
\begin{paste}{TextFileXmpPageFull11}{TextFileXmpPageEmpty11}
\pastebutton{TextFileXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{)system rm /tmp/MOTD\free{closed2 }}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{TextFileXmpPageEmpty11}
\begin{paste}{TextFileXmpPageEmpty11}{TextFileXmpPagePatch11}
\pastebutton{TextFileXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{)system rm /tmp/MOTD\free{closed2 }}
\end{paste}\end{patch}
```

3.108 topics.ht

Axiom Topics



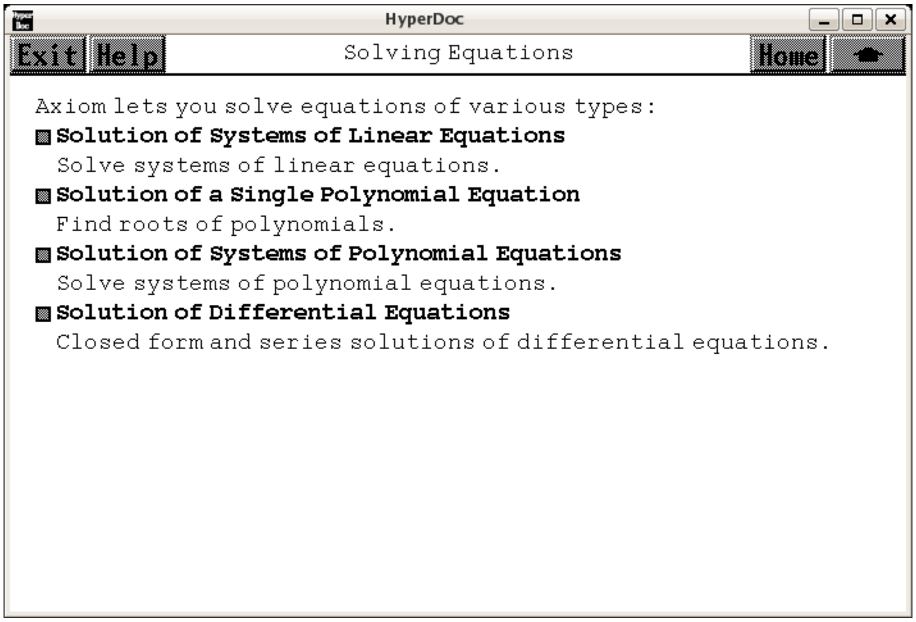
⇐ "Root Page" (RootPage) 3.1 on page 117
⇐ "Reference" (TopReferencePage) 3.1 on page 123
⇒ "Numbers" (NumberPage) 3.80 on page 1021
⇒ "Polynomials" (PolynomialPage) 3.87 on page 1095
⇒ "Functions" (FunctionPage) 3.47 on page 560
⇒ "Solving Equations" (EquationPage) 3.108 on page 1315
⇒ "Calculus" (CalculusPage) 3.108 on page 1318
⇒ "Linear Algebra" (LinAlgPage) 3.108 on page 1316
⇒ "Graphics" (GraphicsPage) 3.50 on page 601
⇒ "Algebra" (AlgebraPage) 3.2 on page 131

— **topics.ht** —

```
\begin{page}{TopicPage}{Axiom Topics}
\beginscroll
Select a topic below: % or
%\lispmemolink{search}{(|htTutorialSearch| '|\stringvalue{pattern}|))}
%for string (use {\em *} for wild card):
%\newline\inputstring{pattern}{58}{}
\beginmenu
\menumemolink{Numbers}{NumberPage}\tab{18}
A look at different types of numbers
```

```
\menumemolink{Polynomials}{PolynomialPage}\tab{18}
Polynomials in Axiom
%
\menumemolink{Functions}{FunctionPage}\tab{18}
Built-in and user-defined functions
%
\menumemolink{Solving Equations}{EquationPage}\tab{18}
Facilities for solving equations
%
\menumemolink{Calculus}{CalculusPage}\tab{18}
Using Axiom to do calculus
%
\menumemolink{Linear Algebra}{LinAlgPage}\tab{18}
Axiom's linear algebra facilities
%
\menumemolink{Graphics}{GraphicsPage}\tab{18}
Axiom's graphics facilities
%
\menumemolink{Algebra}{AlgebraPage}\tab{18}
Axiom's abstract algebra facilities
%
\endmenu
\endscroll
\end{page}
```

Solving Equations



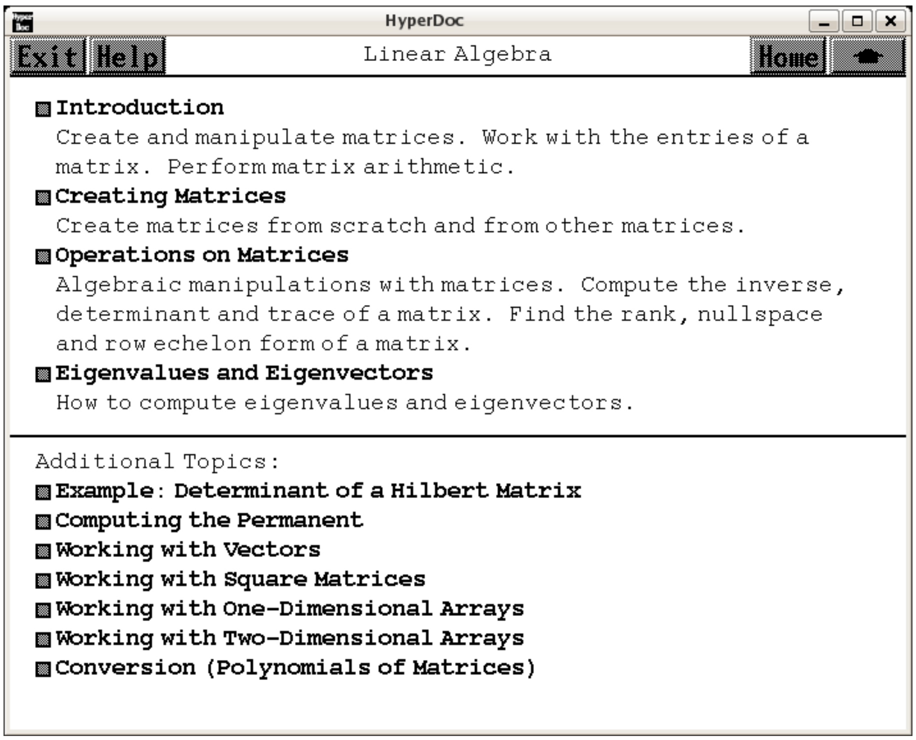
⇐ "Topics" (TopicPage) 3.108 on page 1313
⇒ "Sol...Systems of Linear Equations" (ugxProblemLinSysPage) 12 on page 2131
⇒ "Sol...Single Polynomial Equation" (ugxProblemOnePolPage) 12 on page 2135
⇒ "Sol...Systems of Polynomial Equations" (ugxProblemPolSysPage) 12 on page 2140
⇒ "Sol...Differential Equations" (ugProblemDEQPage) 12 on page 2211

— **topics.ht** —

```
\begin{page}{EquationPage}{Solving Equations}
\beginscroll
Axiom lets you solve equations of various types:
\beginmenu
  \menulink{Solution of Systems of Linear Equations}
{ugxProblemLinSysPage}
    \newline Solve systems of linear equations.
  \menulink{Solution of a Single Polynomial Equation}
{ugxProblemOnePolPage}
    \newline Find roots of polynomials.
  \menulink{Solution of Systems of Polynomial Equations}
{ugxProblemPolSysPage}
  \newline Solve systems of polynomial equations.
  \menulink{Solution of Differential Equations}{ugProblemDEQPage}
  \newline Closed form and series solutions of differential equations.
\endmenu
\endscroll
\autobuttons
```

```
\end{page}
```

Linear Algebra



⇐ "Topics" (TopicPage) 3.108 on page 1313
⇒ "Introduction" (ugIntroTwoDimPage) 6 on page 1548
⇒ "Creating Matrices" (ugxMatrixCreatePage) 3.75 on page 984
⇒ "Operations on Matrices" (ugxMatrixOpsPage) 3.75 on page 997
⇒ "Eigenvalues and Eigenvectors" (ugProblemEigenPage) 12 on page 2123
⇒ "Example: Determinant of a Hilbert Matrix" (ugxFloatHilbertPage) 3.41 on page 502
⇒ "Computing the Permanent" (PermanentXmpPage) 3.85 on page 1085
⇒ "Working with Vectors" (VectorXmpPage) 3.114 on page 1351
⇒ "Working with Square Matrices" (SqMatrixXmpPage) 3.99 on page 1243
⇒ "Working with One-dimensional Arrays" (OneDimensionalArrayXmpPage) 3.4 on page 138

⇒ "Working with Two-dimensional Arrays" (TwoDimensionalArrayXmpPage) 3.5 on page 143

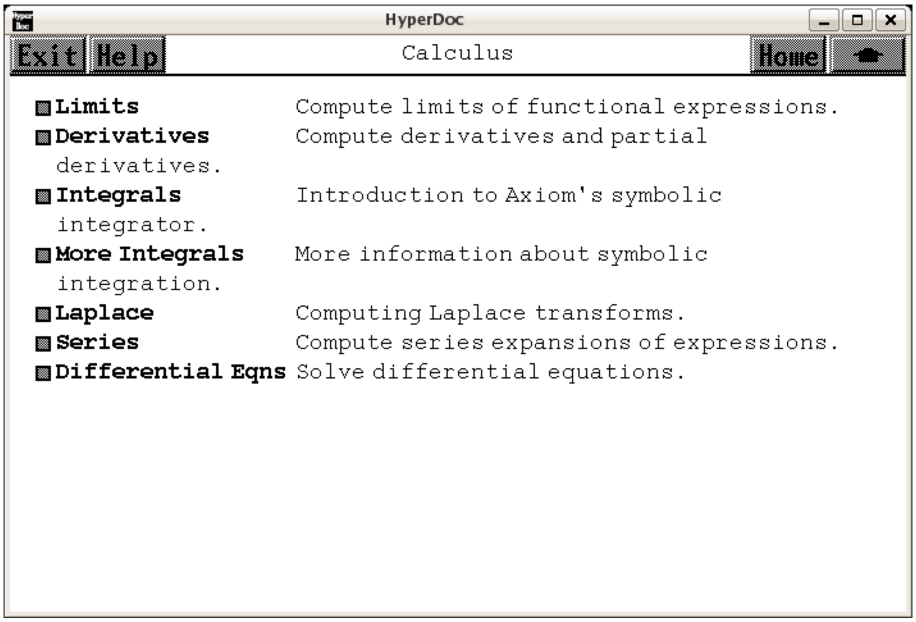
⇒ "Conversion (Polynomials of Matrices)" (ugTypesConvertPage) 7 on page 1671

— topics.ht —

```
\begin{page}{LinAlgPage}{Linear Algebra}
\beginscroll
\beginmenu

\menulink{Introduction}{ugIntroTwoDimPage}\newline

Create and manipulate matrices.
Work with the entries of a matrix.
Perform matrix arithmetic.

\menulink{Creating Matrices}{ugxMatrixCreatePage} \newline

Create matrices from scratch and from other matrices.

\menulink{Operations on Matrices}{ugxMatrixOpsPage} \newline

Algebraic manipulations with matrices.
Compute the inverse, determinant and trace of a matrix.
Find the rank, nullspace and row echelon form of a matrix.

\menulink{Eigenvalues and Eigenvectors}{ugProblemEigenPage} \newline

How to compute eigenvalues and eigenvectors.
\endmenu
\horizontalline\newline
Additional Topics:
\beginmenu
\menulink{Example: Determinant of a Hilbert Matrix}{ugxFloatHilbertPage}
\menulink{Computing the Permanent}{PermanentXmpPage}
\menulink{Working with Vectors}{VectorXmpPage}
\menulink{Working with Square Matrices}{SqMatrixXmpPage}
\menulink{Working with One-Dimensional Arrays}{OneDimensionalArrayXmpPage}
\menulink{Working with Two-Dimensional Arrays}{TwoDimensionalArrayXmpPage}
\menulink{Conversion (Polynomials of Matrices)}{ugTypesConvertPage}
\endmenu
\endscroll
\autobuttons
\end{page}
```

———-

Calculus



⇐ "Topics" (TopicPage) 3.108 on page 1313
⇒ "Limits" (ugProblemLimitsPage) 12 on page 2145
⇒ "Derivatives" (ugIntroCalcDerivPage) 6 on page 1580
⇒ "Integrals" (ugIntroIntegratePage) 6 on page 1587
⇒ "More Integrals" (ugProblemIntegrationPage) 12 on page 2157
⇒ "Laplace" (ugProblemLaplacePage) 12 on page 2152
⇒ "Series" (ugProblemSeriesPage) 12 on page 2164
⇒ "Differential Eqns" (ugProblemDEQPage) 12 on page 2211

— **topics.ht** —

```
\begin{page}{CalculusPage}{Calculus}
\beginscroll
\beginmenu
\menulink{Limits}{ugProblemLimitsPage} \tab{17}
Compute limits of functional expressions.
\menulink{Derivatives}{ugIntroCalcDerivPage}\tab{17}
Compute derivatives and partial derivatives.
\menulink{Integrals}{ugIntroIntegratePage}\tab{17}
Introduction to Axiom's symbolic integrator.
\menulink{More Integrals}{ugProblemIntegrationPage}\tab{17}
More information about symbolic integration.
\menulink{Laplace}{ugProblemLaplacePage}\tab{17}
Computing Laplace transforms.
\menulink{Series}{ugProblemSeriesPage}\tab{17}
Compute series expansions of expressions.
```

```
\menulink{Differential Eqns}{ugProblemDEQPage}\tab{17}
Solve differential equations.
\endmenu
\endscroll
\autobuttons \end{page}
```

——–

3.109 type.ht

Category Type

— type.ht —

```
\begin{page}{CategoryType}{Category {\em Type}}
\beginscroll
{\em Type} is a primitive category in Axiom,
one which is an ancestor of all Axiom categories.

{\em Type} is the root of Axiom's category hierarchy,
a category with no properties (exported operations
and attributes) of which all other categories are descendants.
Two important children of {\em Type} are
\spadtype{SetCategory}, the category of all algebraic domains,
and \spadtype{Aggregate}, the category of all data structures.
\endscroll
```

——–

3.110 union.ht

Domain Union(a:A,...,b:B)

⇒ "Description" (LispFunctions) 3.71 on page 952
⇒ "Operations" (LispFunctions) 3.71 on page 952
⇒ "Exports" (LispFunctions) 3.71 on page 952

— union.ht —

```
\begin{page}{DomainUnion}{Domain {\em Union(a:A,...,b:B)}}
\beginscroll
{\em Union} takes any number of "tag"-domain pairs of arguments:
```

```
\indentrel{2}
\newline \spad{a}, a tag, an element of domain \spadtype{Symbol}
\newline \spad{A}, a domain of category \spadtype{SetCategory}
\newline\tab{10}...
\newline \spad{b}, a tag, an element of domain \spadtype{Symbol}
\newline \spad{B}, a domain of category \spadtype{SetCategory}
\indentrel{-2}\newline
This constructor is a primitive in Axiom.
\newline
\beginmenu
\item\menulispdownlink{Description}
{(|dbSpecialDescription| '|Union|)) }\tab{15}General description
\item\menulispdownlink{Operations}
{(|dbSpecialOperations| '|Union|)}
\tab{15}All exported operations of \spad{Union(a:A,b:B)}
%\item\menudownlink{Examples}   {UnionExamples}
%\tab{15}Examples illustrating use
\item\menulispdownlink{Exports}{(|dbSpecialExports| '|Union|)}
\tab{15}Explicit categories and operations
\endmenu
\vspace{1}\newline
In this tagged \spad{Union}, tags \spad{a,...,b} must be distinct.
\newline
For an alternate "untagged" form of \spad{Union}, see
\downlink{Union(A,B)}{UntaggedUnion}.
\endscroll\end{page}
```

———-

Domain Constructor Union

— union.ht —

```
\begin{page}{UnionDescription}{Domain Constructor {\em Union}}
\beginscroll
\newline\menuitemstyle{}\tab{2}Union({\em a:A},{\em b:B})
\newline\tab{2}{\em Arguments:}\indent{17}\tab{-2}
{\em a}, a tag, an element of domain \spadtype{Symbol}
\newline\tab{-2}
{\em A}, a domain of category \spadtype{SetCategory}
\newline\tab{-2}
{\em b}, a tag, an element of domain \spadtype{Symbol}
\newline\tab{-2}
{\em B}, a domain of category \spadtype{SetCategory}
\indent{0}\newline\tab{2}{\em Returns:}\indent{17}\tab{-2}
the "union of {\em A} and {\em B}" as described below.
\indent{0}\newline\tab{2}{\em Description:}\indent{15}\tab{0}
```

```
{\em Union(a:A,b:B)} denotes the class of objects
which are either members of domain {\em A} or of domain {\em B}.
The symbols {\em a} and {\em b} are called "tags" and
are used to identify the two "branches"
of the union.
The {\em Union} constructor can take any number of arguments and
has an alternate form without {\em tags}.
This tagged {\em Union} type is necessary, for example, to disambiguate
two branches of a union where {\em A} and {\em B} denote the same type.
{\em Union} is a primitive domain of Axiom which cannot be
defined in the Axiom language.
\endscroll\end{page}
```

———-

Domain Union(A,...,B)

⇒ "Description" (LispFunctions) 3.71 on page 952
⇒ "Operations" (LispFunctions) 3.71 on page 952

— **union.ht** —

```
\begin{page}{UntaggedUnion}{Domain {\em Union(A,...,B)}}
\beginscroll
{\em Union} takes any number of domain arguments:
\indentrel{2}
\newline \spad{A}, a domain of category \spadtype{SetCategory}
\newline\tab{10}...
\newline \spad{B}, a domain of category \spadtype{SetCategory}
\indentrel{-2}\newline
\spad{Union} is a primitive constructor in Axiom.
\newline
\beginmenu
\item\menulispdownlink{Description}
{(|dbSpecialDescription| '|UntaggedUnion|)) }\tab{15}General description
\item\menulispdownlink{Operations}
{(|dbSpecialOperations| '|UntaggedUnion|)}
\tab{15}All exported operations of \spad{Union(A,B)}
%\item\menudownlink{Examples}   {UTUnionExamples}
%\tab{15}Examples illustrating use
%\item\menudownlink{Exports}    {UTUnionExports}
%\tab{15}Explicit categories and operations
\endmenu
\vspace{1}\newline
In this untagged form of \spad{Union}, domains \spad{A,...,B}
must be distinct.
\endscroll\end{page}
```

———-

Domain Constructor Union

— **union.ht** —

```
\begin{page}{UTUnionDescription}{Domain Constructor {\em Union}}
\beginscroll
\newline\menuitemstyle{}\tab{2}Union({\em A},{\em B})
\newline\tab{2}{\em Arguments:}\indent{17}\tab{-2}
{\em A}, a domain of category \spadtype{SetCategory}
\newline\tab{-2}
{\em B}, a domain of category \spadtype{SetCategory}
\indent{0}\newline\tab{2}{\em Returns:}\indent{17}\tab{-2}
the "union of {\em A} and {\em B}" as described below.
\indent{0}\newline\tab{2}{\em Description:}\indent{15}\tab{0}
{\em Union(A,B)} denotes the class of objects which are
which are either members of domain {\em A} or of domain {\em B}.
The {\em Union} constructor can take any number of arguments and
has an alternate form using {\em tags}.
{\em Union} is a primitive domain of Axiom which cannot be
defined in the Axiom language.
\endscroll\end{page}
```

———-

3.111 uniseg.ht

UniversalSegment

⇒ “notitle” (SegmentXmpPage) 3.95 on page 1218
⇒ “notitle” (SegmentBindingXmpPage) 3.96 on page 1224
⇒ “notitle” (ListXmpPage) 3.64 on page 866
⇒ “notitle” (StreamXmpPage) 3.102 on page 1263

— **uniseg.ht** —

```
\begin{page}{UniversalSegmentXmpPage}{UniversalSegment}
\beginscroll

The \spadtype{UniversalSegment} domain generalizes \spadtype{Segment}
by allowing segments without a ‘‘hi’’ end point.
\xtc{
```

```
}{
\spadpaste{pints  := 1..        \bound{pints}}
}
\xtc{
}{
\spadpaste{nevens := (0..) by -2 \bound{nevens}}
}
\xtc{
Values of type \spadtype{Segment} are automatically converted to
type \spadtype{UniversalSegment} when appropriate.
}{
\spadpaste{useg: UniversalSegment(Integer) := 3..10 \bound{useg}}
}
\xtc{
The operation \spadfunFrom{hasHi}{UniversalSegment} is used to test
whether a segment has a \spad{hi} end point.
}{
\spadpaste{hasHi pints  \free{pints}}
}
\xtc{
}{
\spadpaste{hasHi nevens \free{nevens}}
}
\xtc{
}{
\spadpaste{hasHi useg   \free{useg}}
}
\xtc{
All operations available on type \spadtype{Segment} apply to
\spadtype{UniversalSegment}, with the proviso that expansions produce
streams rather than lists.
This is to accommodate infinite expansions.
}{
\spadpaste{expand pints \free{pints}}
}
\xtc{
}{
\spadpaste{expand nevens \free{nevens}}
}
\xtc{
}{
\spadpaste{expand [1, 3, 10..15, 100..]}
}

For more information on related topics,  see
\downlink{`Segment'}{SegmentXmpPage}\ignore{Segment},
\downlink{`SegmentBinding'}{SegmentBindingXmpPage}\ignore{SegmentBinding},
\downlink{`List'}{ListXmpPage}\ignore{List}, and
\downlink{`Stream'}{StreamXmpPage}\ignore{Stream}.
\showBlurb{UniversalSegment}
```

```
\endscroll
\autobuttons
\end{page}

\begin{patch}{UniversalSegmentXmpPagePatch1}
\begin{paste}{UniversalSegmentXmpPageFull1}{UniversalSegmentXmpPageEmpty1}
\pastebutton{UniversalSegmentXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{pints := 1..\bound{pints }}
\indentrel{3}\begin{verbatim}
   (1)  1..
                 Type: UniversalSegment PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UniversalSegmentXmpPageEmpty1}
\begin{paste}{UniversalSegmentXmpPageEmpty1}{UniversalSegmentXmpPagePatch1}
\pastebutton{UniversalSegmentXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{pints := 1..\bound{pints }}
\end{paste}\end{patch}

\begin{patch}{UniversalSegmentXmpPagePatch2}
\begin{paste}{UniversalSegmentXmpPageFull2}{UniversalSegmentXmpPageEmpty2}
\pastebutton{UniversalSegmentXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{nevens := (0..) by -2\bound{nevens }}
\indentrel{3}\begin{verbatim}
   (2)  0.. by - 2
              Type: UniversalSegment NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UniversalSegmentXmpPageEmpty2}
\begin{paste}{UniversalSegmentXmpPageEmpty2}{UniversalSegmentXmpPagePatch2}
\pastebutton{UniversalSegmentXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{nevens := (0..) by -2\bound{nevens }}
\end{paste}\end{patch}

\begin{patch}{UniversalSegmentXmpPagePatch3}
\begin{paste}{UniversalSegmentXmpPageFull3}{UniversalSegmentXmpPageEmpty3}
\pastebutton{UniversalSegmentXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{useg: UniversalSegment(Integer) := 3..10\bound{useg }}
\indentrel{3}\begin{verbatim}
   (3)  3..10
                         Type: UniversalSegment Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UniversalSegmentXmpPageEmpty3}
\begin{paste}{UniversalSegmentXmpPageEmpty3}{UniversalSegmentXmpPagePatch3}
\pastebutton{UniversalSegmentXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{useg: UniversalSegment(Integer) := 3..10\bound{useg }}
```

```
\end{paste}\end{patch}

\begin{patch}{UniversalSegmentXmpPagePatch4}
\begin{paste}{UniversalSegmentXmpPageFull4}{UniversalSegmentXmpPageEmpty4}
\pastebutton{UniversalSegmentXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{hasHi pints\free{pints }}
\indentrel{3}\begin{verbatim}
   (4)  false
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UniversalSegmentXmpPageEmpty4}
\begin{paste}{UniversalSegmentXmpPageEmpty4}{UniversalSegmentXmpPagePatch4}
\pastebutton{UniversalSegmentXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{hasHi pints\free{pints }}
\end{paste}\end{patch}

\begin{patch}{UniversalSegmentXmpPagePatch5}
\begin{paste}{UniversalSegmentXmpPageFull5}{UniversalSegmentXmpPageEmpty5}
\pastebutton{UniversalSegmentXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{hasHi nevens\free{nevens }}
\indentrel{3}\begin{verbatim}
   (5)  false
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UniversalSegmentXmpPageEmpty5}
\begin{paste}{UniversalSegmentXmpPageEmpty5}{UniversalSegmentXmpPagePatch5}
\pastebutton{UniversalSegmentXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{hasHi nevens\free{nevens }}
\end{paste}\end{patch}

\begin{patch}{UniversalSegmentXmpPagePatch6}
\begin{paste}{UniversalSegmentXmpPageFull6}{UniversalSegmentXmpPageEmpty6}
\pastebutton{UniversalSegmentXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{hasHi useg\free{useg }}
\indentrel{3}\begin{verbatim}
   (6)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UniversalSegmentXmpPageEmpty6}
\begin{paste}{UniversalSegmentXmpPageEmpty6}{UniversalSegmentXmpPagePatch6}
\pastebutton{UniversalSegmentXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{hasHi useg\free{useg }}
\end{paste}\end{patch}
```

```
\begin{patch}{UniversalSegmentXmpPagePatch7}
\begin{paste}{UniversalSegmentXmpPageFull7}{UniversalSegmentXmpPageEmpty7}
\pastebutton{UniversalSegmentXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{expand pints\free{pints }}
\indentrel{3}\begin{verbatim}
   (7)  [1,2,3,4,5,6,7,8,9,10,...]
                                  Type: Stream Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UniversalSegmentXmpPageEmpty7}
\begin{paste}{UniversalSegmentXmpPageEmpty7}{UniversalSegmentXmpPagePatch7}
\pastebutton{UniversalSegmentXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{expand pints\free{pints }}
\end{paste}\end{patch}

\begin{patch}{UniversalSegmentXmpPagePatch8}
\begin{paste}{UniversalSegmentXmpPageFull8}{UniversalSegmentXmpPageEmpty8}
\pastebutton{UniversalSegmentXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{expand nevens\free{nevens }}
\indentrel{3}\begin{verbatim}
   (8)  [0,- 2,- 4,- 6,- 8,- 10,- 12,- 14,- 16,- 18,...]
                                  Type: Stream Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UniversalSegmentXmpPageEmpty8}
\begin{paste}{UniversalSegmentXmpPageEmpty8}{UniversalSegmentXmpPagePatch8}
\pastebutton{UniversalSegmentXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{expand nevens\free{nevens }}
\end{paste}\end{patch}

\begin{patch}{UniversalSegmentXmpPagePatch9}
\begin{paste}{UniversalSegmentXmpPageFull9}{UniversalSegmentXmpPageEmpty9}
\pastebutton{UniversalSegmentXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{expand [1, 3, 10..15, 100..]}
\indentrel{3}\begin{verbatim}
   (9)  [1,3,10,11,12,13,14,15,100,101,...]
                                  Type: Stream Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UniversalSegmentXmpPageEmpty9}
\begin{paste}{UniversalSegmentXmpPageEmpty9}{UniversalSegmentXmpPagePatch9}
\pastebutton{UniversalSegmentXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{expand [1, 3, 10..15, 100..]}
\end{paste}\end{patch}
```

————

3.112 up.ht

UnivariatePolynomial

⇒ "notitle" (ugProblemFactorPage) 12 on page 2101
⇒ "notitle" (ugIntroVariablesPage) 6 on page 1566
⇒ "notitle" (ugTypesConvertPage) 7 on page 1671
⇒ "notitle" (PolynomialXmpPage) 3.88 on page 1110
⇒ "notitle" (MultivariatePolyXmpPage) 3.77 on page 1011
⇒ "notitle" (DistributedMultivariatePolyXmpPage) 3.24 on page 375

— **up.ht** —

```
\begin{page}{UnivariatePolyXmpPage}{UnivariatePolynomial}
\beginscroll

The domain constructor \spadtype{UnivariatePolynomial}
(abbreviated \spadtype{UP})
creates domains of univariate polynomials in a specified variable.
For example, the domain
\spadtype{UP(a1,POLY FRAC INT)} provides polynomials in the single
variable \spad{a1} whose coefficients are general polynomials with
rational number coefficients.

\beginImportant
\noindent {\bf Restriction:}
\texht{\begin{quotation}\noindent}{\newline\indent{5}}
Axiom does not allow you to create types where
\spadtype{UnivariatePolynomial} is contained in the coefficient type of
\spadtype{Polynomial}. Therefore,
\spadtype{UP(x,POLY INT)} is legal but \spadtype{POLY UP(x,INT)} is not.
\texht{\end{quotation}}{\indent{0}}
\endImportant

\xtc{
\spadtype{UP(x,INT)} is the domain of polynomials in the single
variable \spad{x} with integer coefficients.
}{
\spadpaste{(p,q) : UP(x,INT) \bound{pdec}\bound{qdec}}
}
\xtc{
}{
\spadpaste{p := (3*x-1)**2 * (2*x + 8) \free{pdec}\bound{p}}
}
\xtc{
}{
\spadpaste{q := (1 - 6*x + 9*x**2)**2 \free{qdec}\bound{q}}
}
\xtc{
```

```
The usual arithmetic operations are available for univariate
polynomials.
}{
\spadpaste{p**2 + p*q  \free{p q}}
}
\xtc{
The operation \spadfunFrom{leadingCoefficient}{UnivariatePolynomial}
extracts the coefficient of the term of highest degree.
}{
\spadpaste{leadingCoefficient p \free{p}}
}
\xtc{
The operation \spadfunFrom{degree}{UnivariatePolynomial} returns
the degree of the polynomial.
Since the polynomial has only one variable, the variable is not supplied
to operations like \spadfunFrom{degree}{UnivariatePolynomial}.
}{
\spadpaste{degree p \free{p}}
}
\xtc{
The reductum of the polynomial, the polynomial obtained by
subtracting the term of highest order, is returned by
\spadfunFrom{reductum}{UnivariatePolynomial}.
}{
\spadpaste{reductum p \free{p}}
}
\xtc{
The operation \spadfunFrom{gcd}{UnivariatePolynomial} computes the
greatest common divisor of two polynomials.
}{
\spadpaste{gcd(p,q) \free{p q}}
}
\xtc{
The operation \spadfunFrom{lcm}{UnivariatePolynomial} computes the
least common multiple.
}{
\spadpaste{lcm(p,q) \free{p q}}
}
\xtc{
The operation \spadfunFrom{resultant}{UnivariatePolynomial}
computes the resultant of two univariate polynomials.
In the case of \spad{p} and \spad{q}, the resultant is \spad{0} because they
share a common root.
}{
\spadpaste{resultant(p,q) \free{p q}}
}
\xtc{
To compute the derivative of a univariate polynomial with respect to its
variable, use \spadfunFrom{D}{UnivariatePolynomial}.
}{
```

```
\spadpaste{D p \free{p}}
}
\xtc{
Univariate polynomials can also be used as if they were functions.
To evaluate a univariate polynomial at some point, apply
the polynomial to the point.
}{
\spadpaste{p(2) \free{p}}
}
\xtc{
The same syntax is used for composing two univariate polynomials, i.e.
substituting one polynomial for the variable in another.
This substitutes \spad{q} for the variable in \spad{p}.
}{
\spadpaste{p(q) \free{p q}}
}
\xtc{
This substitutes \spad{p} for the variable in \spad{q}.
}{
\spadpaste{q(p) \free{p q}}
}
\xtc{
To obtain a list of coefficients of the polynomial, use
\spadfunFrom{coefficients}{UnivariatePolynomial}.
}{
\spadpaste{l := coefficients p \free{p}\bound{l}}
}
\xtc{
From this you can use \spadfunFrom{gcd}{UnivariatePolynomial}
and \spadfunFrom{reduce}{List}
to compute the content of the polynomial.
}{
\spadpaste{reduce(gcd,l) \free{l}}
}
\xtc{
Alternatively (and more easily),
you can just call \spadfunFrom{content}{UnivariatePolynomial}.
}{
\spadpaste{content p \free{p}}
}

Note that the operation \spadfunFrom{coefficients}{UnivariatePolynomial}
omits the zero coefficients from the list.
Sometimes it is useful to convert a univariate polynomial
to a vector whose \eth{\spad{i }} position contains the degree \spad{i-1}
coefficient of the polynomial.
\xtc{
}{
\spadpaste{ux := (x**4+2*x+3)::UP(x,INT) \bound{ux}}
}
```

```
\xtc{
To get a complete vector of coefficients, use the operation
\spadfunFrom{vectorise}{UnivariatePolynomial}, which takes a
univariate polynomial and an integer denoting the length of the
desired vector.
}{
\spadpaste{vectorise(ux,5) \free{ux}}
}

It is common to want to do something to every term of a polynomial,
creating a new polynomial in the process.
\xtc{
This is a function for iterating across the terms of a polynomial,
squaring each term.
}{
\begin{spadsrc}[\bound{squareTerms}]
squareTerms(p) ==
  reduce(+,[t**2 for t in monomials p])
\end{spadsrc}
}
\xtc{
Recall what \spad{p} looked like.
}{
\spadpaste{p \free{p}}
}
\xtc{
We can demonstrate \userfun{squareTerms} on \spad{p}.
}{
\spadpaste{squareTerms p \free{p}\free{squareTerms}}
}

When the coefficients of the univariate polynomial belong to a
field,\footnote{For example, when the coefficients are rational
numbers, as opposed to integers.  The important property of
a field is that non-zero elements can be divided and produce
another element. The quotient of the integers 2 and 3 is not
another integer.}
it is possible to compute quotients and remainders.
\xtc{
}{
\spadpaste{(r,s) : UP(a1,FRAC INT) \bound{rdec}\bound{sdec}}
}
\xtc{
}{
\spadpaste{r := a1**2 - 2/3  \free{rdec}\bound{r}}
}
\xtc{
}{
\spadpaste{s := a1 + 4       \free{sdec}\bound{s}}
}
```

```
\xtc{
When the coefficients are rational numbers or rational expressions, the
operation \spadfunFrom{quo}{UnivariatePolynomial} computes the quotient
of two polynomials.
}{
\spadpaste{r quo s \free{r s}}
}
\xtc{
The operation
\spadfunFrom{rem}{UnivariatePolynomial} computes the remainder.
}{
\spadpaste{r rem s \free{r s}}
}
\xtc{
The operation \spadfunFrom{divide}{UnivariatePolynomial} can be used to
return a record of both components.
}{
\spadpaste{d := divide(r, s) \free{r s}\bound{d}}
}
\xtc{
Now we check the arithmetic!
}{
\spadpaste{r - (d.quotient * s + d.remainder) \free{r s d}}
}
\xtc{
It is also possible to integrate univariate polynomials when the
coefficients belong to a field.
}{
\spadpaste{integrate r \free{r}}
}
\xtc{
}{
\spadpaste{integrate s \free{s}}
}

One application of univariate polynomials is to see expressions in terms
of a specific variable.
%
\xtc{
We start with a polynomial in \spad{a1} whose coefficients
are quotients of polynomials in \spad{b1} and \spad{b2}.
}{
\spadpaste{t : UP(a1,FRAC POLY INT) \bound{tdec}}
}
\xtc{
Since in this case we are not talking about using multivariate
polynomials in only two variables, we use \spadtype{Polynomial}.
We also use \spadtype{Fraction} because we want fractions.
}{
\spadpaste{t := a1**2 - a1/b2 + (b1**2-b1)/(b2+3) \free{tdec}\bound{t}}
```

```
}
\xtc{
We push all the variables into a single quotient of polynomials.
}{
\spadpaste{u : FRAC POLY INT := t \bound{u}\free{t}}
}
\xtc{
Alternatively, we can view this as a polynomial in the variable
This is a {\it mode-directed} conversion: you indicate
as much of the structure as you care about and let Axiom
decide on the full type and how to do the transformation.
}{
\spadpaste{u :: UP(b1,?) \free{u}}
}

See \downlink{``Polynomial Factorization''}{ugProblemFactorPage} in
Section 8.2\ignore{ugProblemFactor}
for a discussion of the factorization facilities
in Axiom for univariate polynomials.
For more information on related topics, see
\downlink{``Polynomials''}{ugIntroVariablesPage} in
Section 1.9\ignore{ugIntroVariables},
\downlink{``Conversion''}{ugTypesConvertPage} in
Section 2.7\ignore{ugTypesConvert},
\downlink{`Polynomial'}{PolynomialXmpPage}\ignore{Polynomial},
\downlink{`MultivariatePolynomial'}{MultivariatePolyXmpPage}
\ignore{MultivariatePolynomial}, and
\downlink{`DistributedMultivariatePoly'}
{DistributedMultivariatePolyXmpPage}
\ignore{DistributedMultivariatePoly}.
%
\showBlurb{UnivariatePolynomial}
\endscroll
\autobuttons
\end{page}

\begin{patch}{UnivariatePolyXmpPagePatch1}
\begin{paste}{UnivariatePolyXmpPageFull1}{UnivariatePolyXmpPageEmpty1}
\pastebutton{UnivariatePolyXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{(p,q) : UP(x,INT)\bound{pdec }\bound{qdec }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPageEmpty1}
\begin{paste}{UnivariatePolyXmpPageEmpty1}{UnivariatePolyXmpPagePatch1}
\pastebutton{UnivariatePolyXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{(p,q) : UP(x,INT)\bound{pdec }\bound{qdec }}
\end{paste}\end{patch}
```

```
\begin{patch}{UnivariatePolyXmpPagePatch2}
\begin{paste}{UnivariatePolyXmpPageFull2}{UnivariatePolyXmpPageEmpty2}
\pastebutton{UnivariatePolyXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{p := (3*x-1)**2 * (2*x + 8)\free{pdec }\bound{p }}
\indentrel{3}\begin{verbatim}
           3      2
   (2)  18x  + 60x  - 46x + 8
                  Type: UnivariatePolynomial(x,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPageEmpty2}
\begin{paste}{UnivariatePolyXmpPageEmpty2}{UnivariatePolyXmpPagePatch2}
\pastebutton{UnivariatePolyXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{p := (3*x-1)**2 * (2*x + 8)\free{pdec }\bound{p }}
\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPagePatch3}
\begin{paste}{UnivariatePolyXmpPageFull3}{UnivariatePolyXmpPageEmpty3}
\pastebutton{UnivariatePolyXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{q := (1 - 6*x + 9*x**2)**2\free{qdec }\bound{q }}
\indentrel{3}\begin{verbatim}
           4       3      2
   (3)  81x  - 108x  + 54x  - 12x + 1
                  Type: UnivariatePolynomial(x,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPageEmpty3}
\begin{paste}{UnivariatePolyXmpPageEmpty3}{UnivariatePolyXmpPagePatch3}
\pastebutton{UnivariatePolyXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{q := (1 - 6*x + 9*x**2)**2\free{qdec }\bound{q }}
\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPagePatch4}
\begin{paste}{UnivariatePolyXmpPageFull4}{UnivariatePolyXmpPageEmpty4}
\pastebutton{UnivariatePolyXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{p**2 + p*q\free{p q }}
\indentrel{3}\begin{verbatim}
   (4)
          7        6        5         4        3        2
     1458x  + 3240x  - 7074x  + 10584x  - 9282x  + 4120x
   +
     - 878x + 72
                  Type: UnivariatePolynomial(x,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPageEmpty4}
```

```
\begin{paste}{UnivariatePolyXmpPageEmpty4}{UnivariatePolyXmpPagePatch4}
\pastebutton{UnivariatePolyXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{p**2 + p*q\free{p q }}
\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPagePatch5}
\begin{paste}{UnivariatePolyXmpPageFull5}{UnivariatePolyXmpPageEmpty5}
\pastebutton{UnivariatePolyXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{leadingCoefficient p\free{p }}
\indentrel{3}\begin{verbatim}
   (5)  18
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPageEmpty5}
\begin{paste}{UnivariatePolyXmpPageEmpty5}{UnivariatePolyXmpPagePatch5}
\pastebutton{UnivariatePolyXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{leadingCoefficient p\free{p }}
\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPagePatch6}
\begin{paste}{UnivariatePolyXmpPageFull6}{UnivariatePolyXmpPageEmpty6}
\pastebutton{UnivariatePolyXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{degree p\free{p }}
\indentrel{3}\begin{verbatim}
   (6)  3
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPageEmpty6}
\begin{paste}{UnivariatePolyXmpPageEmpty6}{UnivariatePolyXmpPagePatch6}
\pastebutton{UnivariatePolyXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{degree p\free{p }}
\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPagePatch7}
\begin{paste}{UnivariatePolyXmpPageFull7}{UnivariatePolyXmpPageEmpty7}
\pastebutton{UnivariatePolyXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{reductum p\free{p }}
\indentrel{3}\begin{verbatim}
           2
   (7)  60x  - 46x + 8
                  Type: UnivariatePolynomial(x,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPageEmpty7}
\begin{paste}{UnivariatePolyXmpPageEmpty7}{UnivariatePolyXmpPagePatch7}
```

```
\pastebutton{UnivariatePolyXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{reductum p\free{p }}
\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPagePatch8}
\begin{paste}{UnivariatePolyXmpPageFull8}{UnivariatePolyXmpPageEmpty8}
\pastebutton{UnivariatePolyXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{gcd(p,q)\free{p q }}
\indentrel{3}\begin{verbatim}
          2
   (8)  9x  - 6x + 1
                  Type: UnivariatePolynomial(x,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPageEmpty8}
\begin{paste}{UnivariatePolyXmpPageEmpty8}{UnivariatePolyXmpPagePatch8}
\pastebutton{UnivariatePolyXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{gcd(p,q)\free{p q }}
\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPagePatch9}
\begin{paste}{UnivariatePolyXmpPageFull9}{UnivariatePolyXmpPageEmpty9}
\pastebutton{UnivariatePolyXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{lcm(p,q)\free{p q }}
\indentrel{3}\begin{verbatim}
            5       4       3       2
   (9)  162x  + 432x  - 756x  + 408x  - 94x + 8
                  Type: UnivariatePolynomial(x,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPageEmpty9}
\begin{paste}{UnivariatePolyXmpPageEmpty9}{UnivariatePolyXmpPagePatch9}
\pastebutton{UnivariatePolyXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{lcm(p,q)\free{p q }}
\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPagePatch10}
\begin{paste}{UnivariatePolyXmpPageFull10}{UnivariatePolyXmpPageEmpty10}
\pastebutton{UnivariatePolyXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{resultant(p,q)\free{p q }}
\indentrel{3}\begin{verbatim}
   (10)  0
                                Type: NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPageEmpty10}
\begin{paste}{UnivariatePolyXmpPageEmpty10}{UnivariatePolyXmpPagePatch10}
```

```
\pastebutton{UnivariatePolyXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{resultant(p,q)\free{p q }}
\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPagePatch11}
\begin{paste}{UnivariatePolyXmpPageFull11}{UnivariatePolyXmpPageEmpty11}
\pastebutton{UnivariatePolyXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{D p\free{p }}
\indentrel{3}\begin{verbatim}
            2
   (11)  54x  + 120x - 46
                  Type: UnivariatePolynomial(x,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPageEmpty11}
\begin{paste}{UnivariatePolyXmpPageEmpty11}{UnivariatePolyXmpPagePatch11}
\pastebutton{UnivariatePolyXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{D p\free{p }}
\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPagePatch12}
\begin{paste}{UnivariatePolyXmpPageFull12}{UnivariatePolyXmpPageEmpty12}
\pastebutton{UnivariatePolyXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{p(2)\free{p }}
\indentrel{3}\begin{verbatim}
   (12)  300
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPageEmpty12}
\begin{paste}{UnivariatePolyXmpPageEmpty12}{UnivariatePolyXmpPagePatch12}
\pastebutton{UnivariatePolyXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{p(2)\free{p }}
\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPagePatch13}
\begin{paste}{UnivariatePolyXmpPageFull13}{UnivariatePolyXmpPageEmpty13}
\pastebutton{UnivariatePolyXmpPageFull13}{\hidepaste}
\tab{5}\spadcommand{p(q)\free{p q }}
\indentrel{3}\begin{verbatim}
   (13)
             12            11             10             9
     9565938x   - 38263752x   + 70150212x   - 77944680x
   +
              8            7              6            5
     58852170x  - 32227632x  + 13349448x  - 4280688x
   +
             4          3         2
```

```
     1058184x  - 192672x  + 23328x  - 1536x + 40
                  Type: UnivariatePolynomial(x,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPageEmpty13}
\begin{paste}{UnivariatePolyXmpPageEmpty13}{UnivariatePolyXmpPagePatch13}
\pastebutton{UnivariatePolyXmpPageEmpty13}{\showpaste}
\tab{5}\spadcommand{p(q)\free{p q }}
\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPagePatch14}
\begin{paste}{UnivariatePolyXmpPageFull14}{UnivariatePolyXmpPageEmpty14}
\pastebutton{UnivariatePolyXmpPageFull14}{\hidepaste}
\tab{5}\spadcommand{q(p)\free{p q }}
\indentrel{3}\begin{verbatim}
   (14)
            12             11             10
     8503056x   + 113374080x   + 479950272x
   +
              9              8             7
     404997408x  - 1369516896x  - 626146848x
   +
               6              5              4
     2939858712x  - 2780728704x  + 1364312160x
   +
                3            2
     - 396838872x  + 69205896x  - 6716184x + 279841
                  Type: UnivariatePolynomial(x,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPageEmpty14}
\begin{paste}{UnivariatePolyXmpPageEmpty14}{UnivariatePolyXmpPagePatch14}
\pastebutton{UnivariatePolyXmpPageEmpty14}{\showpaste}
\tab{5}\spadcommand{q(p)\free{p q }}
\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPagePatch15}
\begin{paste}{UnivariatePolyXmpPageFull15}{UnivariatePolyXmpPageEmpty15}
\pastebutton{UnivariatePolyXmpPageFull15}{\hidepaste}
\tab{5}\spadcommand{l := coefficients p\free{p }\bound{l }}
\indentrel{3}\begin{verbatim}
   (15)  [18,60,- 46,8]
                                   Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPageEmpty15}
\begin{paste}{UnivariatePolyXmpPageEmpty15}{UnivariatePolyXmpPagePatch15}
```

```
\pastebutton{UnivariatePolyXmpPageEmpty15}{\showpaste}
\tab{5}\spadcommand{l := coefficients p\free{p }\bound{l }}
\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPagePatch16}
\begin{paste}{UnivariatePolyXmpPageFull16}{UnivariatePolyXmpPageEmpty16}
\pastebutton{UnivariatePolyXmpPageFull16}{\hidepaste}
\tab{5}\spadcommand{reduce(gcd,l)\free{l }}
\indentrel{3}\begin{verbatim}
   (16)  2
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPageEmpty16}
\begin{paste}{UnivariatePolyXmpPageEmpty16}{UnivariatePolyXmpPagePatch16}
\pastebutton{UnivariatePolyXmpPageEmpty16}{\showpaste}
\tab{5}\spadcommand{reduce(gcd,l)\free{l }}
\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPagePatch17}
\begin{paste}{UnivariatePolyXmpPageFull17}{UnivariatePolyXmpPageEmpty17}
\pastebutton{UnivariatePolyXmpPageFull17}{\hidepaste}
\tab{5}\spadcommand{content p\free{p }}
\indentrel{3}\begin{verbatim}
   (17)  2
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPageEmpty17}
\begin{paste}{UnivariatePolyXmpPageEmpty17}{UnivariatePolyXmpPagePatch17}
\pastebutton{UnivariatePolyXmpPageEmpty17}{\showpaste}
\tab{5}\spadcommand{content p\free{p }}
\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPagePatch18}
\begin{paste}{UnivariatePolyXmpPageFull18}{UnivariatePolyXmpPageEmpty18}
\pastebutton{UnivariatePolyXmpPageFull18}{\hidepaste}
\tab{5}\spadcommand{ux := (x**4+2*x+3)::UP(x,INT)\bound{ux }}
\indentrel{3}\begin{verbatim}
          4
   (18)  x  + 2x + 3
                  Type: UnivariatePolynomial(x,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPageEmpty18}
\begin{paste}{UnivariatePolyXmpPageEmpty18}{UnivariatePolyXmpPagePatch18}
\pastebutton{UnivariatePolyXmpPageEmpty18}{\showpaste}
```

```
\tab{5}\spadcommand{ux := (x**4+2*x+3)::UP(x,INT)\bound{ux }}
\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPagePatch19}
\begin{paste}{UnivariatePolyXmpPageFull19}{UnivariatePolyXmpPageEmpty19}
\pastebutton{UnivariatePolyXmpPageFull19}{\hidepaste}
\tab{5}\spadcommand{vectorise(ux,5)\free{ux }}
\indentrel{3}\begin{verbatim}
   (19)  [3,2,0,0,1]
                                   Type: Vector Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPageEmpty19}
\begin{paste}{UnivariatePolyXmpPageEmpty19}{UnivariatePolyXmpPagePatch19}
\pastebutton{UnivariatePolyXmpPageEmpty19}{\showpaste}
\tab{5}\spadcommand{vectorise(ux,5)\free{ux }}
\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPagePatch20}
\begin{paste}{UnivariatePolyXmpPageFull20}{UnivariatePolyXmpPageEmpty20}
\pastebutton{UnivariatePolyXmpPageFull20}{\hidepaste}
\tab{5}\spadcommand{squareTerms(p) ==
  reduce(+,[t**2 for t in monomials p])
\bound{squareTerms }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPageEmpty20}
\begin{paste}{UnivariatePolyXmpPageEmpty20}{UnivariatePolyXmpPagePatch20}
\pastebutton{UnivariatePolyXmpPageEmpty20}{\showpaste}
\tab{5}\spadcommand{squareTerms(p) ==
  reduce(+,[t**2 for t in monomials p])
\bound{squareTerms }}
\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPagePatch21}
\begin{paste}{UnivariatePolyXmpPageFull21}{UnivariatePolyXmpPageEmpty21}
\pastebutton{UnivariatePolyXmpPageFull21}{\hidepaste}
\tab{5}\spadcommand{p\free{p }}
\indentrel{3}\begin{verbatim}
           3      2
   (21)  18x  + 60x  - 46x + 8
                  Type: UnivariatePolynomial(x,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPageEmpty21}
```

```
\begin{paste}{UnivariatePolyXmpPageEmpty21}{UnivariatePolyXmpPagePatch21}
\pastebutton{UnivariatePolyXmpPageEmpty21}{\showpaste}
\tab{5}\spadcommand{p\free{p }}
\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPagePatch22}
\begin{paste}{UnivariatePolyXmpPageFull22}{UnivariatePolyXmpPageEmpty22}
\pastebutton{UnivariatePolyXmpPageFull22}{\hidepaste}
\tab{5}\spadcommand{squareTerms p\free{p }\free{squareTerms }}
\indentrel{3}\begin{verbatim}
             6        4        2
   (22)  324x  + 3600x  + 2116x  + 64
                  Type: UnivariatePolynomial(x,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPageEmpty22}
\begin{paste}{UnivariatePolyXmpPageEmpty22}{UnivariatePolyXmpPagePatch22}
\pastebutton{UnivariatePolyXmpPageEmpty22}{\showpaste}
\tab{5}\spadcommand{squareTerms p\free{p }\free{squareTerms }}
\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPagePatch23}
\begin{paste}{UnivariatePolyXmpPageFull23}{UnivariatePolyXmpPageEmpty23}
\pastebutton{UnivariatePolyXmpPageFull23}{\hidepaste}
\tab{5}\spadcommand{(r,s) : UP(a1,FRAC INT)\bound{rdec }\bound{sdec }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPageEmpty23}
\begin{paste}{UnivariatePolyXmpPageEmpty23}{UnivariatePolyXmpPagePatch23}
\pastebutton{UnivariatePolyXmpPageEmpty23}{\showpaste}
\tab{5}\spadcommand{(r,s) : UP(a1,FRAC INT)\bound{rdec }\bound{sdec }}
\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPagePatch24}
\begin{paste}{UnivariatePolyXmpPageFull24}{UnivariatePolyXmpPageEmpty24}
\pastebutton{UnivariatePolyXmpPageFull24}{\hidepaste}
\tab{5}\spadcommand{r := a1**2 - 2/3\free{rdec }\bound{r }}
\indentrel{3}\begin{verbatim}
           2   2
   (24)  a1  -
               3
        Type: UnivariatePolynomial(a1,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPageEmpty24}
```

```
\begin{paste}{UnivariatePolyXmpPageEmpty24}{UnivariatePolyXmpPagePatch24}
\pastebutton{UnivariatePolyXmpPageEmpty24}{\showpaste}
\tab{5}\spadcommand{r := a1**2 - 2/3\free{rdec }\bound{r }}
\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPagePatch25}
\begin{paste}{UnivariatePolyXmpPageFull25}{UnivariatePolyXmpPageEmpty25}
\pastebutton{UnivariatePolyXmpPageFull25}{\hidepaste}
\tab{5}\spadcommand{s := a1 + 4\free{sdec }\bound{s }}
\indentrel{3}\begin{verbatim}
   (25)  a1 + 4
        Type: UnivariatePolynomial(a1,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPageEmpty25}
\begin{paste}{UnivariatePolyXmpPageEmpty25}{UnivariatePolyXmpPagePatch25}
\pastebutton{UnivariatePolyXmpPageEmpty25}{\showpaste}
\tab{5}\spadcommand{s := a1 + 4\free{sdec }\bound{s }}
\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPagePatch26}
\begin{paste}{UnivariatePolyXmpPageFull26}{UnivariatePolyXmpPageEmpty26}
\pastebutton{UnivariatePolyXmpPageFull26}{\hidepaste}
\tab{5}\spadcommand{r quo s\free{r s }}
\indentrel{3}\begin{verbatim}
   (26)  a1 - 4
        Type: UnivariatePolynomial(a1,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPageEmpty26}
\begin{paste}{UnivariatePolyXmpPageEmpty26}{UnivariatePolyXmpPagePatch26}
\pastebutton{UnivariatePolyXmpPageEmpty26}{\showpaste}
\tab{5}\spadcommand{r quo s\free{r s }}
\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPagePatch27}
\begin{paste}{UnivariatePolyXmpPageFull27}{UnivariatePolyXmpPageEmpty27}
\pastebutton{UnivariatePolyXmpPageFull27}{\hidepaste}
\tab{5}\spadcommand{r rem s\free{r s }}
\indentrel{3}\begin{verbatim}
         46
   (27)
          3
        Type: UnivariatePolynomial(a1,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPageEmpty27}
```

```
\begin{paste}{UnivariatePolyXmpPageEmpty27}{UnivariatePolyXmpPagePatch27}
\pastebutton{UnivariatePolyXmpPageEmpty27}{\showpaste}
\tab{5}\spadcommand{r rem s\free{r s }}
\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPagePatch28}
\begin{paste}{UnivariatePolyXmpPageFull28}{UnivariatePolyXmpPageEmpty28}
\pastebutton{UnivariatePolyXmpPageFull28}{\hidepaste}
\tab{5}\spadcommand{d := divide(r, s)\free{r s }\bound{d }}
\indentrel{3}\begin{verbatim}
                                      46
   (28)  [quotient= a1 - 4,remainder= ]
                                       3
Type: Record(quotient: UnivariatePolynomial(a1,Fraction Integer),remainder: UnivariatePolynomial
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPageEmpty28}
\begin{paste}{UnivariatePolyXmpPageEmpty28}{UnivariatePolyXmpPagePatch28}
\pastebutton{UnivariatePolyXmpPageEmpty28}{\showpaste}
\tab{5}\spadcommand{d := divide(r, s)\free{r s }\bound{d }}
\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPagePatch29}
\begin{paste}{UnivariatePolyXmpPageFull29}{UnivariatePolyXmpPageEmpty29}
\pastebutton{UnivariatePolyXmpPageFull29}{\hidepaste}
\tab{5}\spadcommand{r - (d.quotient * s + d.remainder)\free{r s d }}
\indentrel{3}\begin{verbatim}
   (29)  0
        Type: UnivariatePolynomial(a1,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPageEmpty29}
\begin{paste}{UnivariatePolyXmpPageEmpty29}{UnivariatePolyXmpPagePatch29}
\pastebutton{UnivariatePolyXmpPageEmpty29}{\showpaste}
\tab{5}\spadcommand{r - (d.quotient * s + d.remainder)\free{r s d }}
\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPagePatch30}
\begin{paste}{UnivariatePolyXmpPageFull30}{UnivariatePolyXmpPageEmpty30}
\pastebutton{UnivariatePolyXmpPageFull30}{\hidepaste}
\tab{5}\spadcommand{integrate r\free{r }}
\indentrel{3}\begin{verbatim}
         1   3   2
   (30)   a1  -  a1
         3       3
        Type: UnivariatePolynomial(a1,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{UnivariatePolyXmpPageEmpty30}
\begin{paste}{UnivariatePolyXmpPageEmpty30}{UnivariatePolyXmpPagePatch30}
\pastebutton{UnivariatePolyXmpPageEmpty30}{\showpaste}
\tab{5}\spadcommand{integrate r\free{r }}
\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPagePatch31}
\begin{paste}{UnivariatePolyXmpPageFull31}{UnivariatePolyXmpPageEmpty31}
\pastebutton{UnivariatePolyXmpPageFull31}{\hidepaste}
\tab{5}\spadcommand{integrate s\free{s }}
\indentrel{3}\begin{verbatim}
         1   2
   (31)   a1  + 4a1
         2
        Type: UnivariatePolynomial(a1,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPageEmpty31}
\begin{paste}{UnivariatePolyXmpPageEmpty31}{UnivariatePolyXmpPagePatch31}
\pastebutton{UnivariatePolyXmpPageEmpty31}{\showpaste}
\tab{5}\spadcommand{integrate s\free{s }}
\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPagePatch32}
\begin{paste}{UnivariatePolyXmpPageFull32}{UnivariatePolyXmpPageEmpty32}
\pastebutton{UnivariatePolyXmpPageFull32}{\hidepaste}
\tab{5}\spadcommand{t : UP(a1,FRAC POLY INT)\bound{tdec }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPageEmpty32}
\begin{paste}{UnivariatePolyXmpPageEmpty32}{UnivariatePolyXmpPagePatch32}
\pastebutton{UnivariatePolyXmpPageEmpty32}{\showpaste}
\tab{5}\spadcommand{t : UP(a1,FRAC POLY INT)\bound{tdec }}
\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPagePatch33}
\begin{paste}{UnivariatePolyXmpPageFull33}{UnivariatePolyXmpPageEmpty33}
\pastebutton{UnivariatePolyXmpPageFull33}{\hidepaste}
\tab{5}\spadcommand{t := a1**2 - a1/b2 + (b1**2-b1)/(b2+3)\free{tdec }\bound{t }}
\indentrel{3}\begin{verbatim}
                         2
           2    1      b1  - b1
   (33)  a1  -  a1 +
               b2       b2 + 3
Type: UnivariatePolynomial(a1,Fraction Polynomial Integer)
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPageEmpty33}
\begin{paste}{UnivariatePolyXmpPageEmpty33}{UnivariatePolyXmpPagePatch33}
\pastebutton{UnivariatePolyXmpPageEmpty33}{\showpaste}
\tab{5}\spadcommand{t := a1**2 - a1/b2 + (b1**2-b1)/(b2+3)\free{tdec }\bound{t }}
\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPagePatch34}
\begin{paste}{UnivariatePolyXmpPageFull34}{UnivariatePolyXmpPageEmpty34}
\pastebutton{UnivariatePolyXmpPageFull34}{\hidepaste}
\tab{5}\spadcommand{u : FRAC POLY INT := t\bound{u }\free{t }}
\indentrel{3}\begin{verbatim}
           2  2      2           2
         a1 b2  + (b1  - b1 + 3a1  - a1)b2 - 3a1
   (34)
                           2
                         b2  + 3b2
                       Type: Fraction Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPageEmpty34}
\begin{paste}{UnivariatePolyXmpPageEmpty34}{UnivariatePolyXmpPagePatch34}
\pastebutton{UnivariatePolyXmpPageEmpty34}{\showpaste}
\tab{5}\spadcommand{u : FRAC POLY INT := t\bound{u }\free{t }}
\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPagePatch35}
\begin{paste}{UnivariatePolyXmpPageFull35}{UnivariatePolyXmpPageEmpty35}
\pastebutton{UnivariatePolyXmpPageFull35}{\hidepaste}
\tab{5}\spadcommand{u :: UP(b1,?)\free{u }}
\indentrel{3}\begin{verbatim}
                                     2
            1      2      1        a1 b2 - a1
   (35)   b1  -  b1 +
         b2 + 3       b2 + 3          b2
Type: UnivariatePolynomial(b1,Fraction Polynomial Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UnivariatePolyXmpPageEmpty35}
\begin{paste}{UnivariatePolyXmpPageEmpty35}{UnivariatePolyXmpPagePatch35}
\pastebutton{UnivariatePolyXmpPageEmpty35}{\showpaste}
\tab{5}\spadcommand{u :: UP(b1,?)\free{u }}
\end{paste}\end{patch}
```

3.113 oreup.ht

UnivariateSkewPolynomial

— **oreup.ht** —

```
\begin{page}{UnivariateSkewPolyXmpPage}{UnivariateSkewPolynomial}
\beginscroll

Skew or Ore polynomial rings provide a unified framework to
compute with differential and difference equations.
\newline
In the following, let A be an integral domain, equipped with two
endomorphisms \sigma and \delta where:
\blankline
\begin{items}
\item \sigma: A -> A is an injective ring endomorphism
\item \delta: A -> A, the pseudo-derivation with respect to
\sigma, is an additive endomorphism with
\blankline
\delta(ab) = \sigma(a)\delta(b) + \delta(a)b
\blankline
for all a,b in A
\end{items}
\blankline
The skew polynomial ring
[\Delta;\sigma,\delta] is the ring of
polynomials in \delta with coefficients in A, with the
usual addition, while the product is given by
\blankline
\delta a = \sigma(a)\delta + \delta(a) for a in A
\blankline
The two most important examples of skew polynomial rings are:
\blankline
K(x)[D,1,\delta]
\blankline
where 1 is the identity on K and \texht{$\delta$}{delta} is
the usual derviative, is the ring of differential polynomials
\blankline
  \subscriptIt{K}{n} [E,n,n->n+1,0]
\blankline
is the ring of linear recurrence operators with polynomial coefficients

\horizontalline
The UnivariateSkewPolynomialCategory (OREPCAT) provides a unified
framework for polynomial rings in a non-central indeterminate over
some coefficient ring R. The commutation relations between the
indeterminate x and the coefficient t is given by
```

```
\blankline
   x r = \sigma(r) x + \delta(r)
\blankline
where \sigma is a ring endomorphism of R
and   \delta is a \sigma-derivation of R
which is an additive map from R to R such that
\blankline
   \delta(rs) = \sigma(r) \delta(s) + \delta(r) s
\blankline
In case \sigma is the identity map on R, a \sigma-derivation of R
is just called a derivation.
\blankline
We start with a linear ordinary differential operator. First, we
define the coefficient ring to be expressions in one variable x
with fractional coefficients:
\blankline
   F := EXPR(FRAC(INT))
\blankline
Define Dx to be a derivative d/dx:
\blankline
   Dx: F->F := f+->D(f,['x])
\blankline
Define a skew polynomial ring over F with identity endomorphism as
\sigma and derivation d/dx as \delta:
\begin{verbatim}
   D0 := OREUP('d,F,1,Dx)
\end{verbatim}
\begin{verbatim}
   u:D0 := (operator 'u)(x)
\end{verbatim}
\begin{verbatim}
   d:D0 := 'd
\end{verbatim}
\begin{verbatim}
   a:D0:=u^3*d^3+u^2*d^2+u*d+1

            3 3       2 2
        u(x) d  + u(x) d  + u(x)d + 1
\end{verbatim}
\begin{verbatim}
   b:D0:=(u+1)*d^2+2*d

                   2
        (u(x) + 1)d  + 2d
\end{verbatim}
\begin{verbatim}
   r:=rightDivide(a,b)

                                3 ,           3       2
                   3      - u(x) u (x) - u(x)  + u(x)
```

```
                u(x)
   [quotient= -------- d + ---------------------------,
              u(x) + 1                2
                                u(x)  + 2u(x) + 1
                    3 ,            3
               2u(x) u (x) + 3u(x)  + u(x)

    remainder= --------------------------- d + 1]
                        2
                    u(x)  + 2u(x) + 1

\end{verbatim}
\begin{verbatim}
   r.quotient

                           3 ,           3       2
              3      - u(x) u (x) - u(x)  + u(x)
          u(x)
        -------- d + ----------------------------
        u(x) + 1                2
                          u(x)  + 2u(x) + 1

\end{verbatim}
\begin{verbatim}
   r.remainder

              3 ,           3
         2u(x) u (x) + 3u(x)  + u(x)

         --------------------------- d + 1
                  2
              u(x)  + 2u(x) + 1
\end{verbatim}

\horizontalline

   )clear all
\blankline
As a second example, we consider the so-called Weyl algebra.
\blankline
Define the coefficient ring to be an ordinary polynomial over integers
in one variable t
\blankline
\begin{verbatim}
   R := UP('t,INT)
\end{verbatim}
\blankline
Define a skew polynomial ring over R with identity map as
\sigma
and derivation d/dt as \delta.
```

```
The resulting algebra is then called a Weyl algebra:
\blankline
\begin{verbatim}
   W := OREUP('x,R,1,D)

   t:W := 't

   x:W := 'x
\end{verbatim}
\blankline
Let
\begin{verbatim}
   a:W:=(t-1)*x^4+(t^3+3*t+1)*x^2+2*t*x+t^3

                4     3            2            3
        (t - 1)x  + (t  + 3t + 1)x  + 2t x + t
\end{verbatim}
\begin{verbatim}
   b:W:=(6*t^4+2*t^2)*x^3+3*t^2*x^2

           4     2  3     2 2
        (6t  + 2t )x  + 3t x
\end{verbatim}
\blankline
Then
\begin{verbatim}
   a*b
           5     4     3     2  7       4      3      2        6
        (6t  - 6t  + 2t  - 2t )x  + (96t  - 93t  + 13t  - 16t)x
      +
           7      5     4       3       2       5
        (6t  + 20t  + 6t  + 438t  - 406t  - 24)x
      +
            6      5       4      3       2             4
        (48t  + 15t  + 152t  + 61t  + 603t  - 532t - 36)x
      +
           7      5      4       3       2               3
        (6t  + 74t  + 60t  + 226t  + 116t  + 168t - 140)x
      +
           5     3      2           2
        (3t  + 6t  + 12t  + 18t + 6)x

   a^3
         3     2           12      5     4      3      2           10
       (t  - 3t  + 3t - 1)x   + (3t  - 6t  + 12t  - 15t  + 3t + 3)x
     +
          3      2       9      7     6      5      4      3     2            8
       (6t  - 12t  + 6t)x  + (3t  - 3t  + 21t  - 18t  + 24t  - 9t  - 15t - 3)x
     +
           5      4      3      2        7
```

```
       (12t  - 12t  + 36t  - 24t  - 12t)x
     +
          9      7     6      5     4      3      2           6
       (t  + 15t  - 3t  + 45t  + 6t  + 36t  + 15t  + 9t + 1)x
     +
           7      5      3      2       5
       (6t  + 48t  + 54t  + 36t  + 6t)x
     +
           9      7     6      5      4      3      2  4
       (3t  + 21t  + 3t  + 39t  + 18t  + 39t  + 12t )x
     +
            7      5      4     3  3      9     7     6      5  2     7     9
       (12t  + 36t  + 12t  + 8t )x  + (3t  + 9t  + 3t  + 12t )x  + 6t x + t
\end{verbatim}

\horizontalline
   )clear all
\blankline
As a third example, we construct a difference operator algebra over
the ring of EXPR(INT) by using an automorphism S defined by a
"shift" operation S:EXPR(INT) -> EXPR(INT)
\blankline
\begin{verbatim}
   s(e)(n) = e(n+1)
\end{verbatim}
\blankline
and an S-derivation defined by DF:EXPR(INT) -> EXPR(INT) as
\blankline
\begin{verbatim}
   DF(e)(n) = e(n+1)-e(n)
\end{verbatim}
\blankline
Define S to be a "shift" operator, which acts on expressions with
the discrete variable n:
\blankline
\begin{verbatim}
   S:EXPR(INT)->EXPR(INT):=e+->eval(e,[n],[n+1])
\end{verbatim}
\blankline
Define DF to be a "difference" operator, which acts on expressions
with a discrete variable n:
\blankline
\begin{verbatim}
   DF:EXPR(INT)->EXPR(INT):=e+->eval(e,[n],[n+1])-e
\end{verbatim}
\blankline
Then define the difference operator algebra D0:
\blankline
\begin{verbatim}
   D0:=OREUP('D,EXPR(INT),morphism S,DF)
```

```
   u:=(operator 'u)[n]

   L:DO:='D+u

        D + u(n)

   L^2

         2                2
        D  + 2u(n)D + u(n)
\end{verbatim}

\horizontalline
   )clear all
\blankline
As a fourth example, we construct a skew polynomial ring by using an
inner derivation \delta induced by a fixed y in R:
\blankline
   \delta(r) = yr - ry
\blankline
First we should expose the constructor SquareMatrix so it is visible
in the interpreter:
\blankline
   )set expose add constructor SquareMatrix
\blankline
Define R to be the square matrix with integer entries:
\blankline
\begin{verbatim}
   R:=SQMATRIX(2,INT)

   y:R:=matrix [[1,1],[0,1]]
        +1  1+
        |    |
        +0  1+
\end{verbatim}
\blankline
Define the inner derivative \delta:
\blankline
   delta:R->R:=r+->y*r-r*y
\blankline
Define S to be a skew polynomial determined by \sigma = 1
and \delta as an inner derivative:
\blankline
\begin{verbatim}
   S:=OREUP('x,R,1,delta)

   x:S:='x

   a:S:=matrix [[2,3],[1,1]]
```

```
        +2  3+
        |    |
        +1  1+

   x^2*a
        +2  3+ 2   +2  - 2+    +0  - 2+
        |    |x  + |       |x + |       |
        +1  1+     +0  - 2+    +0   0 +
\end{verbatim}
\endscroll
\autobuttons
\end{page}
```

———-

3.114 vector.ht

Vector

⇒ "notitle" (OneDimensionalArrayXmpPage) 3.4 on page 138
⇒ "notitle" (ListXmpPage) 3.64 on page 866
⇒ "notitle" (MatrixXmpPage) 3.75 on page 984
⇒ "notitle" (OneDimensionalArrayXmpPage) 3.4 on page 138
⇒ "notitle" (SetXmpPage) 3.97 on page 1227
⇒ "notitle" (TableXmpPage) 3.106 on page 1298
⇒ "notitle" (TwoDimensionalArrayXmpPage) 3.5 on page 143

— **vector.ht** —

```
\begin{page}{VectorXmpPage}{Vector}
\beginscroll

The \spadtype{Vector} domain is used for storing data in a
one-dimensional indexed data structure.  A vector is a homogeneous
data structure in that all the components of the vector must belong to
the same Axiom domain.  Each vector has a fixed length specified by
the user; vectors are not extensible.  This domain is similar to the
\spadtype{OneDimensionalArray} domain, except that when the components
of a \spadtype{Vector} belong to a \spadtype{Ring}, arithmetic
operations are provided.  For more examples of operations that are
defined for both \spadtype{Vector} and \spadtype{OneDimensionalArray},
see \downlink{`OneDimensionalArray'}{OneDimensionalArrayXmpPage}
\ignore{OneDimensionalArray}.

As with the \spadtype{OneDimensionalArray} domain, a \spadtype{Vector}
can be created by calling the operation \spadfunFrom{new}{Vector}, its
```

```
components can be accessed by calling the operations
\spadfunFrom{elt}{Vector} and \spadfunFrom{qelt}{Vector}, and its
components can be reset by calling the operations
\spadfunFrom{setelt}{Vector} and \spadfunFromX{qsetelt}{Vector}.
\xtc{ This creates a vector of integers of length \spad{5} all of
whose components are \spad{12}.  }{ \spadpaste{u : VECTOR INT :=
new(5,12) \bound{u}} } \xtc{ This is how you create a vector from a
list of its components.  }{ \spadpaste{v : VECTOR INT :=
vector([1,2,3,4,5]) \bound{v}} }

\xtc{
Indexing for vectors begins at \spad{1}.
The last element has index equal to the length of the vector,
which is computed by \spadopFrom{\#}{Vector}.
}{
\spadpaste{\#(v) \free{v}}
}
\xtc{
This is the standard way to use \spadfunFrom{elt}{Vector} to extract an
element.
Functionally, it is the same as if you had typed \spad{elt(v,2)}.
}{
\spadpaste{v.2 \free{v}}
}
\xtc{
This is the standard way to use \spadfunFrom{setelt}{Vector} to change an
element.
It is the same as if you had typed \spad{setelt(v,3,99)}.
}{
\spadpaste{v.3 := 99 \free{v}\bound{vdelta}}
}
\xtc{
Now look at \spad{v} to see the change.  You can use
\spadfunFrom{qelt}{Vector} and \spadfunFromX{qsetelt}{Vector} (instead
of \spadfunFrom{elt}{Vector} and \spadfunFrom{setelt}{Vector},
respectively) but {\it only} when you know that the index is within
the valid range.
}{
\spadpaste{v \free{vdelta}}
}

\xtc{
When the components belong to a \spadtype{Ring}, Axiom
provides arithmetic operations for \spadtype{Vector}.
These include left and right scalar multiplication.
}{
\spadpaste{5 * v \free{vdelta}}
}
\xtc{
}{
```

```
\spadpaste{v * 7 \free{vdelta}}
}
\xtc{
}{
\spadpaste{w : VECTOR INT := vector([2,3,4,5,6]) \bound{w}}
}
\xtc{
Addition and subtraction are also available.
}{
\spadpaste{v + w \free{vdelta w}}
}
\xtc{
Of course, when adding or subtracting, the two vectors must have the same
length or an error message is displayed.
}{
\spadpaste{v - w \free{vdelta w}}
}

For more information about other aggregate domains,
see the following:
\downlink{`List'}{ListXmpPage}\ignore{List},
\downlink{`Matrix'}{MatrixXmpPage}\ignore{Matrix},
\downlink{`OneDimensionalArray'}{OneDimensionalArrayXmpPage}
\ignore{OneDimensionalArray},
\downlink{`Set'}{SetXmpPage}\ignore{Set},
\downlink{`Table'}{TableXmpPage}\ignore{Table}, and
\downlink{`TwoDimensionalArray'}{TwoDimensionalArrayXmpPage}
\ignore{TwoDimensionalArray}.
Issue the system command \spadcmd{)show Vector}
to display the full list of operations defined by
\spadtype{Vector}.

\endscroll
\autobuttons
\end{page}

\begin{patch}{VectorXmpPagePatch1}
\begin{paste}{VectorXmpPageFull1}{VectorXmpPageEmpty1}
\pastebutton{VectorXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{u : VECTOR INT := new(5,12)\bound{u }}
\indentrel{3}\begin{verbatim}
   (1)  [12,12,12,12,12]
                                   Type: Vector Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{VectorXmpPageEmpty1}
\begin{paste}{VectorXmpPageEmpty1}{VectorXmpPagePatch1}
\pastebutton{VectorXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{u : VECTOR INT := new(5,12)\bound{u }}
```

```
\end{paste}\end{patch}

\begin{patch}{VectorXmpPagePatch2}
\begin{paste}{VectorXmpPageFull2}{VectorXmpPageEmpty2}
\pastebutton{VectorXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{v : VECTOR INT := vector([1,2,3,4,5])\bound{v }}
\indentrel{3}\begin{verbatim}
   (2)  [1,2,3,4,5]
                                   Type: Vector Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{VectorXmpPageEmpty2}
\begin{paste}{VectorXmpPageEmpty2}{VectorXmpPagePatch2}
\pastebutton{VectorXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{v : VECTOR INT := vector([1,2,3,4,5])\bound{v }}
\end{paste}\end{patch}

\begin{patch}{VectorXmpPagePatch3}
\begin{paste}{VectorXmpPageFull3}{VectorXmpPageEmpty3}
\pastebutton{VectorXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{\#(v)\free{v }}
\indentrel{3}\begin{verbatim}
   (3)  5
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{VectorXmpPageEmpty3}
\begin{paste}{VectorXmpPageEmpty3}{VectorXmpPagePatch3}
\pastebutton{VectorXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{\#(v)\free{v }}
\end{paste}\end{patch}

\begin{patch}{VectorXmpPagePatch4}
\begin{paste}{VectorXmpPageFull4}{VectorXmpPageEmpty4}
\pastebutton{VectorXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{v.2\free{v }}
\indentrel{3}\begin{verbatim}
   (4)  2
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{VectorXmpPageEmpty4}
\begin{paste}{VectorXmpPageEmpty4}{VectorXmpPagePatch4}
\pastebutton{VectorXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{v.2\free{v }}
\end{paste}\end{patch}
```

```
\begin{patch}{VectorXmpPagePatch5}
\begin{paste}{VectorXmpPageFull5}{VectorXmpPageEmpty5}
\pastebutton{VectorXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{v.3 := 99\free{v }\bound{vdelta }}
\indentrel{3}\begin{verbatim}
   (5)  99
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{VectorXmpPageEmpty5}
\begin{paste}{VectorXmpPageEmpty5}{VectorXmpPagePatch5}
\pastebutton{VectorXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{v.3 := 99\free{v }\bound{vdelta }}
\end{paste}\end{patch}

\begin{patch}{VectorXmpPagePatch6}
\begin{paste}{VectorXmpPageFull6}{VectorXmpPageEmpty6}
\pastebutton{VectorXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{v\free{vdelta }}
\indentrel{3}\begin{verbatim}
   (6)  [1,2,99,4,5]
                                   Type: Vector Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{VectorXmpPageEmpty6}
\begin{paste}{VectorXmpPageEmpty6}{VectorXmpPagePatch6}
\pastebutton{VectorXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{v\free{vdelta }}
\end{paste}\end{patch}

\begin{patch}{VectorXmpPagePatch7}
\begin{paste}{VectorXmpPageFull7}{VectorXmpPageEmpty7}
\pastebutton{VectorXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{5 * v\free{vdelta }}
\indentrel{3}\begin{verbatim}
   (7)  [5,10,495,20,25]
                                   Type: Vector Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{VectorXmpPageEmpty7}
\begin{paste}{VectorXmpPageEmpty7}{VectorXmpPagePatch7}
\pastebutton{VectorXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{5 * v\free{vdelta }}
\end{paste}\end{patch}

\begin{patch}{VectorXmpPagePatch8}
\begin{paste}{VectorXmpPageFull8}{VectorXmpPageEmpty8}
```

```
\pastebutton{VectorXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{v * 7\free{vdelta }}
\indentrel{3}\begin{verbatim}
   (8)  [7,14,693,28,35]
                                   Type: Vector Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{VectorXmpPageEmpty8}
\begin{paste}{VectorXmpPageEmpty8}{VectorXmpPagePatch8}
\pastebutton{VectorXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{v * 7\free{vdelta }}
\end{paste}\end{patch}

\begin{patch}{VectorXmpPagePatch9}
\begin{paste}{VectorXmpPageFull9}{VectorXmpPageEmpty9}
\pastebutton{VectorXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{w : VECTOR INT := vector([2,3,4,5,6])\bound{w }}
\indentrel{3}\begin{verbatim}
   (9)  [2,3,4,5,6]
                                   Type: Vector Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{VectorXmpPageEmpty9}
\begin{paste}{VectorXmpPageEmpty9}{VectorXmpPagePatch9}
\pastebutton{VectorXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{w : VECTOR INT := vector([2,3,4,5,6])\bound{w }}
\end{paste}\end{patch}

\begin{patch}{VectorXmpPagePatch10}
\begin{paste}{VectorXmpPageFull10}{VectorXmpPageEmpty10}
\pastebutton{VectorXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{v + w\free{vdelta w }}
\indentrel{3}\begin{verbatim}
   (10)  [3,5,103,9,11]
                                   Type: Vector Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{VectorXmpPageEmpty10}
\begin{paste}{VectorXmpPageEmpty10}{VectorXmpPagePatch10}
\pastebutton{VectorXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{v + w\free{vdelta w }}
\end{paste}\end{patch}

\begin{patch}{VectorXmpPagePatch11}
\begin{paste}{VectorXmpPageFull11}{VectorXmpPageEmpty11}
\pastebutton{VectorXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{v - w\free{vdelta w }}
```

```
\indentrel{3}\begin{verbatim}
   (11)  [- 1,- 1,95,- 1,- 1]
                                    Type: Vector Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{VectorXmpPageEmpty11}
\begin{paste}{VectorXmpPageEmpty11}{VectorXmpPagePatch11}
\pastebutton{VectorXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{v - w\free{vdelta w }}
\end{paste}\end{patch}
```

———-

3.115 void.ht

Void

— void.ht —

```
\begin{page}{VoidXmpPage}{Void}
\beginscroll

When an expression is not in a value context, it is given
type \spadtype{Void}. For example, in the expression
\begin{verbatim}
r := (a; b; if c then d else e; f)
\end{verbatim}
values are used only from the subexpressions \spad{c} and \spad{f}:
all others are thrown away.
The subexpressions \spad{a}, \spad{b}, \spad{d} and \spad{e}
are evaluated for side-effects only and have type \spadtype{Void}.
There is a unique value of type \spadtype{Void}.

\xtc{
You will most often see results of type \spadtype{Void} when you
declare a variable.
}{
\spadpaste{a : Integer}
}
\noOutputXtc{
Usually no output is displayed for \spadtype{Void} results.
You can force the display of a rather ugly object by issuing
\spadcmd{)set message void on}.
}{
\spadpaste{)set message void on}
```

```
}
\xtc{
}{
\spadpaste{b : Fraction Integer}
}
\noOutputXtc{
}{
\spadpaste{)set message void off}
}
\xtc{
All values can be converted to type \spadtype{Void}.
}{
\spadpaste{3::Void \bound{prev}}
}
\xtc{
Once a value has been converted to \spadtype{Void}, it cannot be recovered.
}{
\spadpaste{\% :: PositiveInteger \free{prev}}
}
\endscroll
\autobuttons
\end{page}

\begin{patch}{VoidXmpPagePatch1}
\begin{paste}{VoidXmpPageFull1}{VoidXmpPageEmpty1}
\pastebutton{VoidXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{a : Integer}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{VoidXmpPageEmpty1}
\begin{paste}{VoidXmpPageEmpty1}{VoidXmpPagePatch1}
\pastebutton{VoidXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{a : Integer}
\end{paste}\end{patch}

\begin{patch}{VoidXmpPagePatch2}
\begin{paste}{VoidXmpPageFull2}{VoidXmpPageEmpty2}
\pastebutton{VoidXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{)set message void on}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{VoidXmpPageEmpty2}
\begin{paste}{VoidXmpPageEmpty2}{VoidXmpPagePatch2}
\pastebutton{VoidXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{)set message void on}
```

```
\end{paste}\end{patch}

\begin{patch}{VoidXmpPagePatch3}
\begin{paste}{VoidXmpPageFull3}{VoidXmpPageEmpty3}
\pastebutton{VoidXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{b : Fraction Integer}
\indentrel{3}\begin{verbatim}
   (2)  "()"
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{VoidXmpPageEmpty3}
\begin{paste}{VoidXmpPageEmpty3}{VoidXmpPagePatch3}
\pastebutton{VoidXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{b : Fraction Integer}
\end{paste}\end{patch}

\begin{patch}{VoidXmpPagePatch4}
\begin{paste}{VoidXmpPageFull4}{VoidXmpPageEmpty4}
\pastebutton{VoidXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{)set message void off}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{VoidXmpPageEmpty4}
\begin{paste}{VoidXmpPageEmpty4}{VoidXmpPagePatch4}
\pastebutton{VoidXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{)set message void off}
\end{paste}\end{patch}

\begin{patch}{VoidXmpPagePatch5}
\begin{paste}{VoidXmpPageFull5}{VoidXmpPageEmpty5}
\pastebutton{VoidXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{3::Void\bound{prev }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{VoidXmpPageEmpty5}
\begin{paste}{VoidXmpPageEmpty5}{VoidXmpPagePatch5}
\pastebutton{VoidXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{3::Void\bound{prev }}
\end{paste}\end{patch}

\begin{patch}{VoidXmpPagePatch6}
\begin{paste}{VoidXmpPageFull6}{VoidXmpPageEmpty6}
\pastebutton{VoidXmpPageFull6}{\hidepaste}
```

```
\tab{5}\spadcommand{\% :: PositiveInteger\free{prev }}
\indentrel{3}\begin{verbatim}
   "()"
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{VoidXmpPageEmpty6}
\begin{paste}{VoidXmpPageEmpty6}{VoidXmpPagePatch6}
\pastebutton{VoidXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{\% :: PositiveInteger\free{prev }}
\end{paste}\end{patch}
```

———-

3.116 wutset.ht

WuWenTsunTriangularSet

— wutset.ht —

```
\begin{page}{WuWenTsunTriangularSetXmpPage}{WuWenTsunTriangularSet}
\beginscroll
The \spadtype{WuWenTsunTriangularSet} domain constructor implements
the characteristic set method of Wu Wen Tsun.  This algorithm computes
a list of triangular sets from a list of polynomials such that the
algebraic variety defined by the given list of polynomials decomposes
into the union of the regular-zero sets of the computed triangular
sets.  The constructor takes four arguments.  The first one, {\bf R},
is the coefficient ring of the polynomials; it must belong to the
category \spadtype{IntegralDomain}.  The second one, {\bf E}, is the
exponent monoid of the polynomials; it must belong to the category
\spadtype{OrderedAbelianMonoidSup}.  The third one, {\bf V}, is the
ordered set of variables; it must belong to the category
\spadtype{OrderedSet}.  The last one is the polynomial ring; it must
belong to the category \spadtype{RecursivePolynomialCategory(R,E,V)}.
The abbreviation for \spadtype{WuWenTsunTriangularSet} is
\spadtype{WUTSET}.

Let us illustrate the facilities by an example.

\xtc{
Define the coefficient ring.
}{
\spadpaste{R := Integer \bound{R}}
}
\xtc{
```

```
Define the list of variables,
}{
\spadpaste{ls : List Symbol := [x,y,z,t] \bound{ls}}
}
\xtc{
and make it an ordered set;
}{
\spadpaste{V := OVAR(ls) \free{ls} \bound{V}}
}
\xtc{
then define the exponent monoid.
}{
\spadpaste{E := IndexedExponents V \free{V} \bound{E}}
}
\xtc{
Define the polynomial ring.
}{
\spadpaste{P := NSMP(R, V) \free{R} \free{V} \bound{P}}
}
\xtc{
Let the variables be polynomial.
}{
\spadpaste{x: P := 'x \free{P} \bound{x}}
}
\xtc{
}{
\spadpaste{y: P := 'y \free{P} \bound{y}}
}
\xtc{
}{
\spadpaste{z: P := 'z \free{P} \bound{z}}
}
\xtc{
}{
\spadpaste{t: P := 't \free{P} \bound{t}}
}
\xtc{
Now call the \spadtype{WuWenTsunTriangularSet} domain constructor.
}{
\spadpaste{T := WUTSET(R,E,V,P) \free{R} \free{E} \free{V}
\free{P} \bound{T} }
}
\xtc{
Define a polynomial system.
}{
\spadpaste{p1 := x ** 31 - x ** 6 - x - y \free{x} \free{y} \bound{p1}}
}
\xtc{
}{
\spadpaste{p2 := x ** 8  - z \free{x} \free{z} \bound{p2}}
```

```
}
\xtc{
}{
\spadpaste{p3 := x ** 10 - t \free{x} \free{t} \bound{p3}}
}
\xtc{
}{
\spadpaste{lp := [p1, p2, p3] \free{p1} \free{p2} \free{p3} \bound{lp}}
}
\xtc{
Compute a characteristic set of the system.
}{
\spadpaste{characteristicSet(lp)$T \free{lp} \free{T}}
}
\xtc{
Solve the system.
}{
\spadpaste{zeroSetSplit(lp)$T \free{lp} \free{T}}
}


The \spadtype{RegularTriangularSet} and
\spadtype{SquareFreeRegularTriangularSet} domain constructors, and the
\spadtype{LazardSetSolvingPackage},
\spadtype{SquareFreeRegularTriangularSet} and
\spadtype{ZeroDimensionalSolvePackage} package constructors also
provide operations to compute triangular decompositions of algebraic
varieties.  These five constructor use a special kind of
characteristic sets, called regular triangular sets.  These special
characteristic sets have better properties than the general ones.
Regular triangular sets and their related concepts are presented in
the paper "On the Theories of Triangular sets" By P. Aubry, D. Lazard
and M. Moreno Maza (to appear in the Journal of Symbolic Computation).
The decomposition algorithm (due to the third author) available in the
four above constructors provide generally better timings than the
characteristic set method.  In fact, the \spadtype{WUTSET} constructor
remains interesting for the purpose of manipulating characteristic
sets whereas the other constructors are more convenient for solving
polynomial systems.

Note that the way of understanding triangular decompositions
is detailed in the example of the \spadtype{RegularTriangularSet}
constructor.
\endscroll
\autobuttons
\end{page}

\begin{patch}{WuWenTsunTriangularSetXmpPagePatch1}
\begin{paste}{WuWenTsunTriangularSetXmpPageFull1}{WuWenTsunTriangularSetXmpPageEmpty1}
\pastebutton{WuWenTsunTriangularSetXmpPageFull1}{\hidepaste}
```

```
\tab{5}\spadcommand{R := Integer\bound{R }}
\indentrel{3}\begin{verbatim}
   (1)  Integer
                                          Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{WuWenTsunTriangularSetXmpPageEmpty1}
\begin{paste}{WuWenTsunTriangularSetXmpPageEmpty1}{WuWenTsunTriangularSetXmpPagePatch1}
\pastebutton{WuWenTsunTriangularSetXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{R := Integer\bound{R }}
\end{paste}\end{patch}

\begin{patch}{WuWenTsunTriangularSetXmpPagePatch2}
\begin{paste}{WuWenTsunTriangularSetXmpPageFull2}{WuWenTsunTriangularSetXmpPageEmpty2}
\pastebutton{WuWenTsunTriangularSetXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{ls : List Symbol := [x,y,z,t]\bound{ls }}
\indentrel{3}\begin{verbatim}
   (2)  [x,y,z,t]
                                     Type: List Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{WuWenTsunTriangularSetXmpPageEmpty2}
\begin{paste}{WuWenTsunTriangularSetXmpPageEmpty2}{WuWenTsunTriangularSetXmpPagePatch2}
\pastebutton{WuWenTsunTriangularSetXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{ls : List Symbol := [x,y,z,t]\bound{ls }}
\end{paste}\end{patch}

\begin{patch}{WuWenTsunTriangularSetXmpPagePatch3}
\begin{paste}{WuWenTsunTriangularSetXmpPageFull3}{WuWenTsunTriangularSetXmpPageEmpty3}
\pastebutton{WuWenTsunTriangularSetXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{V := OVAR(ls)\free{ls }\bound{V }}
\indentrel{3}\begin{verbatim}
   (3)  OrderedVariableList [x,y,z,t]
                                          Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{WuWenTsunTriangularSetXmpPageEmpty3}
\begin{paste}{WuWenTsunTriangularSetXmpPageEmpty3}{WuWenTsunTriangularSetXmpPagePatch3}
\pastebutton{WuWenTsunTriangularSetXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{V := OVAR(ls)\free{ls }\bound{V }}
\end{paste}\end{patch}

\begin{patch}{WuWenTsunTriangularSetXmpPagePatch4}
\begin{paste}{WuWenTsunTriangularSetXmpPageFull4}{WuWenTsunTriangularSetXmpPageEmpty4}
\pastebutton{WuWenTsunTriangularSetXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{E := IndexedExponents V\free{V }\bound{E }}
\indentrel{3}\begin{verbatim}
```

```
   (4)  IndexedExponents OrderedVariableList [x,y,z,t]
                                            Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{WuWenTsunTriangularSetXmpPageEmpty4}
\begin{paste}{WuWenTsunTriangularSetXmpPageEmpty4}{WuWenTsunTriangularSetXmpPagePatch4}
\pastebutton{WuWenTsunTriangularSetXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{E := IndexedExponents V\free{V }\bound{E }}
\end{paste}\end{patch}

\begin{patch}{WuWenTsunTriangularSetXmpPagePatch5}
\begin{paste}{WuWenTsunTriangularSetXmpPageFull5}{WuWenTsunTriangularSetXmpPageEmpty5}
\pastebutton{WuWenTsunTriangularSetXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{P := NSMP(R, V)\free{R }\free{V }\bound{P }}
\indentrel{3}\begin{verbatim}
   (5)
  NewSparseMultivariatePolynomial(Integer,OrderedVariable
  List [x,y,z,t])
                                            Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{WuWenTsunTriangularSetXmpPageEmpty5}
\begin{paste}{WuWenTsunTriangularSetXmpPageEmpty5}{WuWenTsunTriangularSetXmpPagePatch5}
\pastebutton{WuWenTsunTriangularSetXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{P := NSMP(R, V)\free{R }\free{V }\bound{P }}
\end{paste}\end{patch}

\begin{patch}{WuWenTsunTriangularSetXmpPagePatch6}
\begin{paste}{WuWenTsunTriangularSetXmpPageFull6}{WuWenTsunTriangularSetXmpPageEmpty6}
\pastebutton{WuWenTsunTriangularSetXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{x: P := 'x\free{P }\bound{x }}
\indentrel{3}\begin{verbatim}
   (6)  x
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [x,y,z,t])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{WuWenTsunTriangularSetXmpPageEmpty6}
\begin{paste}{WuWenTsunTriangularSetXmpPageEmpty6}{WuWenTsunTriangularSetXmpPagePatch6}
\pastebutton{WuWenTsunTriangularSetXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{x: P := 'x\free{P }\bound{x }}
\end{paste}\end{patch}

\begin{patch}{WuWenTsunTriangularSetXmpPagePatch7}
\begin{paste}{WuWenTsunTriangularSetXmpPageFull7}{WuWenTsunTriangularSetXmpPageEmpty7}
\pastebutton{WuWenTsunTriangularSetXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{y: P := 'y\free{P }\bound{y }}
\indentrel{3}\begin{verbatim}
```

```
   (7)  y
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [x,y,z,t])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{WuWenTsunTriangularSetXmpPageEmpty7}
\begin{paste}{WuWenTsunTriangularSetXmpPageEmpty7}{WuWenTsunTriangularSetXmpPagePatch7}
\pastebutton{WuWenTsunTriangularSetXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{y: P := 'y\free{P }\bound{y }}
\end{paste}\end{patch}

\begin{patch}{WuWenTsunTriangularSetXmpPagePatch8}
\begin{paste}{WuWenTsunTriangularSetXmpPageFull8}{WuWenTsunTriangularSetXmpPageEmpty8}
\pastebutton{WuWenTsunTriangularSetXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{z: P := 'z\free{P }\bound{z }}
\indentrel{3}\begin{verbatim}
   (8)  z
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [x,y,z,t])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{WuWenTsunTriangularSetXmpPageEmpty8}
\begin{paste}{WuWenTsunTriangularSetXmpPageEmpty8}{WuWenTsunTriangularSetXmpPagePatch8}
\pastebutton{WuWenTsunTriangularSetXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{z: P := 'z\free{P }\bound{z }}
\end{paste}\end{patch}

\begin{patch}{WuWenTsunTriangularSetXmpPagePatch9}
\begin{paste}{WuWenTsunTriangularSetXmpPageFull9}{WuWenTsunTriangularSetXmpPageEmpty9}
\pastebutton{WuWenTsunTriangularSetXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{t: P := 't\free{P }\bound{t }}
\indentrel{3}\begin{verbatim}
   (9)  t
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [x,y,z,t])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{WuWenTsunTriangularSetXmpPageEmpty9}
\begin{paste}{WuWenTsunTriangularSetXmpPageEmpty9}{WuWenTsunTriangularSetXmpPagePatch9}
\pastebutton{WuWenTsunTriangularSetXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{t: P := 't\free{P }\bound{t }}
\end{paste}\end{patch}

\begin{patch}{WuWenTsunTriangularSetXmpPagePatch10}
\begin{paste}{WuWenTsunTriangularSetXmpPageFull10}{WuWenTsunTriangularSetXmpPageEmpty10}
\pastebutton{WuWenTsunTriangularSetXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{T := WUTSET(R,E,V,P)\free{R }\free{E }\free{V }\free{P }\bound{T }}
\indentrel{3}\begin{verbatim}
   (10)
  WuWenTsunTriangularSet(Integer,IndexedExponents Ordered
```

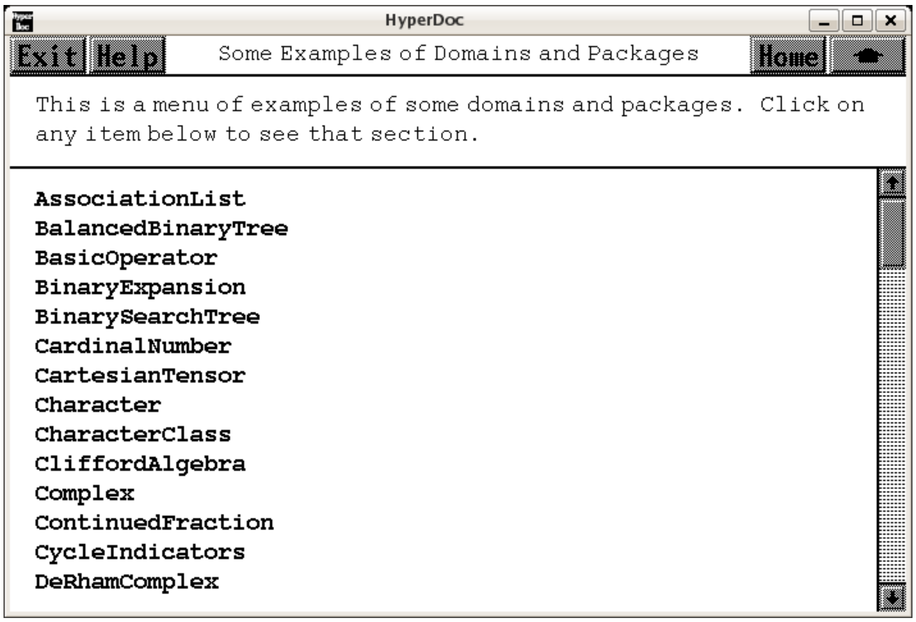
```
  VariableList [x,y,z,t],OrderedVariableList [x,y,z,t],Ne
  wSparseMultivariatePolynomial(Integer,OrderedVariableLi
  st [x,y,z,t]))
                                           Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{WuWenTsunTriangularSetXmpPageEmpty10}
\begin{paste}{WuWenTsunTriangularSetXmpPageEmpty10}{WuWenTsunTriangularSetXmpPagePatch10}
\pastebutton{WuWenTsunTriangularSetXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{T := WUTSET(R,E,V,P)\free{R }\free{E }\free{V }\free{P }\bound{T }}
\end{paste}\end{patch}

\begin{patch}{WuWenTsunTriangularSetXmpPagePatch11}
\begin{paste}{WuWenTsunTriangularSetXmpPageFull11}{WuWenTsunTriangularSetXmpPageEmpty11}
\pastebutton{WuWenTsunTriangularSetXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{p1 := x ** 31 - x ** 6 - x - y\free{x }\free{y }\bound{p1 }}
\indentrel{3}\begin{verbatim}
          31    6
   (11)  x   - x  - x - y
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [x,y,z,t])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{WuWenTsunTriangularSetXmpPageEmpty11}
\begin{paste}{WuWenTsunTriangularSetXmpPageEmpty11}{WuWenTsunTriangularSetXmpPagePatch11}
\pastebutton{WuWenTsunTriangularSetXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{p1 := x ** 31 - x ** 6 - x - y\free{x }\free{y }\bound{p1 }}
\end{paste}\end{patch}

\begin{patch}{WuWenTsunTriangularSetXmpPagePatch12}
\begin{paste}{WuWenTsunTriangularSetXmpPageFull12}{WuWenTsunTriangularSetXmpPageEmpty12}
\pastebutton{WuWenTsunTriangularSetXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{p2 := x ** 8 - z\free{x }\free{z }\bound{p2 }}
\indentrel{3}\begin{verbatim}
          8
   (12)  x  - z
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [x,y,z,t])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{WuWenTsunTriangularSetXmpPageEmpty12}
\begin{paste}{WuWenTsunTriangularSetXmpPageEmpty12}{WuWenTsunTriangularSetXmpPagePatch12}
\pastebutton{WuWenTsunTriangularSetXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{p2 := x ** 8 - z\free{x }\free{z }\bound{p2 }}
\end{paste}\end{patch}

\begin{patch}{WuWenTsunTriangularSetXmpPagePatch13}
\begin{paste}{WuWenTsunTriangularSetXmpPageFull13}{WuWenTsunTriangularSetXmpPageEmpty13}
\pastebutton{WuWenTsunTriangularSetXmpPageFull13}{\hidepaste}
```

```
\tab{5}\spadcommand{p3 := x ** 10 - t\free{x }\free{t }\bound{p3 }}
\indentrel{3}\begin{verbatim}
         10
   (13)  x   - t
Type: NewSparseMultivariatePolynomial(Integer,OrderedVariableList [x,y,z,t])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{WuWenTsunTriangularSetXmpPageEmpty13}
\begin{paste}{WuWenTsunTriangularSetXmpPageEmpty13}{WuWenTsunTriangularSetXmpPagePatch13}
\pastebutton{WuWenTsunTriangularSetXmpPageEmpty13}{\showpaste}
\tab{5}\spadcommand{p3 := x ** 10 - t\free{x }\free{t }\bound{p3 }}
\end{paste}\end{patch}

\begin{patch}{WuWenTsunTriangularSetXmpPagePatch14}
\begin{paste}{WuWenTsunTriangularSetXmpPageFull14}{WuWenTsunTriangularSetXmpPageEmpty14}
\pastebutton{WuWenTsunTriangularSetXmpPageFull14}{\hidepaste}
\tab{5}\spadcommand{lp := [p1, p2, p3]\free{p1 }\free{p2 }\free{p3 }\bound{lp }}
\indentrel{3}\begin{verbatim}
          31    6          8      10
   (14)  [x   - x  - x - y,x  - z,x   - t]
Type: List NewSparseMultivariatePolynomial(Integer,OrderedVariableList [x,y,z,t])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{WuWenTsunTriangularSetXmpPageEmpty14}
\begin{paste}{WuWenTsunTriangularSetXmpPageEmpty14}{WuWenTsunTriangularSetXmpPagePatch14}
\pastebutton{WuWenTsunTriangularSetXmpPageEmpty14}{\showpaste}
\tab{5}\spadcommand{lp := [p1, p2, p3]\free{p1 }\free{p2 }\free{p3 }\bound{lp }}
\end{paste}\end{patch}

\begin{patch}{WuWenTsunTriangularSetXmpPagePatch15}
\begin{paste}{WuWenTsunTriangularSetXmpPageFull15}{WuWenTsunTriangularSetXmpPageEmpty15}
\pastebutton{WuWenTsunTriangularSetXmpPageFull15}{\hidepaste}
\tab{5}\spadcommand{characteristicSet(lp)$T\free{lp }\free{T }}
\indentrel{3}\begin{verbatim}
   (15)
     5    4   4 2 2     3 4        7     4      6    6
   {z  - t , t z y  + 2t z y + (- t  + 2t  - t)z  + t z,
     3      3     3     3
    (t  - 1)z x - z y - t }
Type: Union(WuWenTsunTriangularSet(Integer,IndexedExponents OrderedVariableList [x,y,z,t],OrderedVariableLis
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{WuWenTsunTriangularSetXmpPageEmpty15}
\begin{paste}{WuWenTsunTriangularSetXmpPageEmpty15}{WuWenTsunTriangularSetXmpPagePatch15}
\pastebutton{WuWenTsunTriangularSetXmpPageEmpty15}{\showpaste}
\tab{5}\spadcommand{characteristicSet(lp)$T\free{lp }\free{T }}
\end{paste}\end{patch}
```

```
\begin{patch}{WuWenTsunTriangularSetXmpPagePatch16}
\begin{paste}{WuWenTsunTriangularSetXmpPageFull16}{WuWenTsunTriangularSetXmpPageEmpty16}
\pastebutton{WuWenTsunTriangularSetXmpPageFull16}{\hidepaste}
\tab{5}\spadcommand{zeroSetSplit(lp)$T\free{lp }\free{T }}
\indentrel{3}\begin{verbatim}
   (16)
                 3      5    4  3     3    2
   [{t,z,y,x}, {t  - 1,z  - t ,z y + t ,z x  - t},

       5    4
     {z  - t ,
       4 2 2     3 4        7     4      6    6
      t z y  + 2t z y + (- t  + 2t  - t)z  + t z,
        3      3     3     3
      (t  - 1)z x - z y - t }
     ]
Type: List WuWenTsunTriangularSet(Integer,IndexedExponents OrderedVariableList [x,y,z,t],Ordered
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{WuWenTsunTriangularSetXmpPageEmpty16}
\begin{paste}{WuWenTsunTriangularSetXmpPageEmpty16}{WuWenTsunTriangularSetXmpPagePatch16}
\pastebutton{WuWenTsunTriangularSetXmpPageEmpty16}{\showpaste}
\tab{5}\spadcommand{zeroSetSplit(lp)$T\free{lp }\free{T }}
\end{paste}\end{patch}
```

3.117 xmpexp.ht

Some Examples of Domains and Packages



⇐ "Reference" (TopReferencePage) 3.1 on page 123
⇒ "AssociationList" (AssociationListXmpPage) 3.3 on page 132
⇒ "BalancedBinaryTree" (BalancedBinaryTreeXmpPage) 3.7 on page 157
⇒ "BasicOperator" (BasicOperatorXmpPage) 3.10 on page 169
⇒ "BinaryExpansion" (BinaryExpansionXmpPage) 3.8 on page 163
⇒ "BinarySearchTree" (BinarySearchTreeXmpPage) 3.11 on page 178
⇒ "CardinalNumber" (CardinalNumberXmpPage) 3.12 on page 185
⇒ "CartesianTensor" (CartesianTensorXmpPage) 3.13 on page 195
⇒ "Character" (CharacterXmpPage) 3.15 on page 228
⇒ "CharacterClass" (CharacterClassXmpPage) 3.14 on page 221
⇒ "CliffordAlgebra" (CliffordAlgebraXmpPage) 3.15 on page 234
⇒ "Complex" (ComplexXmpPage) 3.16 on page 254
⇒ "ContinuedFraction" (ContinuedFractionXmpPage) 3.17 on page 262
⇒ "CycleIndicators" (CycleIndicatorsXmpPage) 3.19 on page 279
⇒ "DeRhamComplex" (DeRhamComplexXmpPage) 3.22 on page 352
⇒ "DecimalExpansion" (DecimalExpansionXmpPage) 3.21 on page 348
⇒ "DistributedMultivariatePoly" (DistributedMultivariatePolyXmpPage) 3.24 on page 375
⇒ "DoubleFloat" (DoubleFloatXmpPage) 3.23 on page 369
⇒ "EqTable" (EqTableXmpPage) 3.26 on page 386
⇒ "Equation" (EquationXmpPage) 3.25 on page 380
⇒ "Exit" (ExitXmpPage) 3.28 on page 398

⇒ “Expression” (ExpressionXmpPage) 3.33 on page 436
⇒ “Factored” (FactoredXmpPage) 3.43 on page 516
⇒ “FactoredFunctionsTwo” (FactoredFnsTwoXmpPage) 3.44 on page 539
⇒ “File” (FileXmpPage) 3.40 on page 480
⇒ “FileName” (FileNameXmpPage) 3.42 on page 507
⇒ “FlexibleArray” (FlexibleArrayXmpPage) 3.39 on page 472
⇒ “Float” (FloatXmpPage) 3.41 on page 487
⇒ “Fraction” (FractionXmpPage) 3.45 on page 543
⇒ “FullPartialFracExpansion” (FullPartialFracExpansionXmpPage) 3.46 on page 549
⇒ “GeneralSparseTable” (GeneralSparseTableXmpPage) 3.52 on page 687
⇒ “GroebnerFactorizationPkg” (GroebnerFactorizationPkgXmpPage) 3.48 on page 575
⇒ “Heap” (HeapXmpPage) 3.53 on page 690
⇒ “HexadecimalExpansion” (HexExpansionXmpPage) 3.54 on page 692
⇒ “Integer” (IntegerXmpPage) 3.55 on page 696
⇒ “IntegerLinearDependence” (IntegerLinearDependenceXmpPage) 3.122 on page 1463
⇒ “IntegerNumberTheoryFunctions” (IntNumberTheoryFnsXmpPage) 3.56 on page 722
⇒ “Kernel” (KernelXmpPage) 3.58 on page 743
⇒ “KeyedAccessFile” (KeyedAccessFileXmpPage) 3.57 on page 734
⇒ “LexTriangularPackage” (LexTriangularPkgXmpPage) 3.61 on page 784
⇒ “LazardSetSolvingPackage” (LazardSetSolvingPackageXmpPage) 3.59 on page 752
⇒ “Library” (LibraryXmpPage) 3.62 on page 840
⇒ “LieExponentials” (LieExponentialsXmpPage) 3.60 on page 778
⇒ “LiePolynomial” (LiePolynomialXmpPage) 3.68 on page 919
⇒ “LinearOrdinaryDifferentialOperator” (LinearOrdinaryDifferentialOperatorXmpPage) 3.65 on page 884
⇒ “LinearOrdinaryDifferentialOperatorOne” (LinearOrdinaryDifferentialOperatorOneXmpPage) 3.66 on page 894
⇒ “LinearODEOperatorTwo” (LinearODEOperatorTwoXmpPage) 3.67 on page 905
⇒ “List” (ListXmpPage) 3.64 on page 866
⇒ “LyndonWord” (LyndonWordXmpPage) 3.69 on page 932
⇒ “Magma” (MagmaXmpPage) 3.70 on page 942
⇒ “MakeFunction” (MakeFunctionXmpPage) 3.76 on page 1006
⇒ “MappingPackageOne” (MappingPackageOneXmpPage) 3.73 on page 965
⇒ “Matrix” (MatrixXmpPage) 3.75 on page 984
⇒ “MultiSet” (MultiSetXmpPage) 3.74 on page 978
⇒ “MultivariatePolynomial” (MultivariatePolyXmpPage) 3.77 on page 1011
⇒ “None” (NoneXmpPage) 3.79 on page 1019
⇒ “Octonion” (OctonionXmpPage) 3.81 on page 1044
⇒ “OneDimensionalArray” (OneDimensionalArrayXmpPage) 3.4 on page 138
⇒ “Operator” (OperatorXmpPage) 3.83 on page 1071
⇒ “OrderedVariableList” (OrderedVariableListXmpPage) 3.84 on page 1082
⇒ “OrderlyDifferentialPolynomial” (OrderlyDifferentialPolyXmpPage) 3.82 on page 1053
⇒ “PartialFraction” (PartialFractionXmpPage) 3.86 on page 1088
⇒ “Permanent” (PermanentXmpPage) 3.85 on page 1085
⇒ “Polynomial” (PolynomialXmpPage) 3.88 on page 1110
⇒ “Quaternion” (QuaternionXmpPage) 3.89 on page 1134

⇒ "RadixExpansion" (RadixExpansionXmpPage) 3.90 on page 1140
⇒ "RealClosure" (RealClosureXmpPage) 3.91 on page 1149
⇒ "RegularTriangularSet" (RegularTriangularSetXmpPage) 3.93 on page 1184
⇒ "RomanNumeral" (RomanNumeralXmpPage) 3.94 on page 1213
⇒ "Segment" (SegmentXmpPage) 3.95 on page 1218
⇒ "SegmentBinding" (SegmentBindingXmpPage) 3.96 on page 1224
⇒ "Set" (SetXmpPage) 3.97 on page 1227
⇒ "SingleInteger" (SingleIntegerXmpPage) 3.98 on page 1237
⇒ "SparseTable" (SparseTableXmpPage) 3.101 on page 1259
⇒ "SquareMatrix" (SqMatrixXmpPage) 3.99 on page 1243
⇒ "SquareFreeRegularTriangularSet" (SqFreeRegTriangSetXmpPage) 3.100 on page 1247
⇒ "Stream" (StreamXmpPage) 3.102 on page 1263
⇒ "String" (StringXmpPage) 3.103 on page 1269
⇒ "StringTable" (StringTableXmpPage) 3.104 on page 1284
⇒ "Symbol" (SymbolXmpPage) 3.105 on page 1286
⇒ "Table" (TableXmpPage) 3.106 on page 1298
⇒ "TextFile" (TextFileXmpPage) 3.107 on page 1307
⇒ "TwoDimensionalArray" (TwoDimensionalArrayXmpPage) 3.5 on page 143
⇒ "UnivariatePolynomial" (UnivariatePolyXmpPage) 3.112 on page 1327
⇒ "UnivariateSkewPolynomial" (UnivariateSkewPolyXmpPage) 3.113 on page 1345
⇒ "UniversalSegment" (UniversalSegmentXmpPage) 3.111 on page 1322
⇒ "Vector" (VectorXmpPage) 3.114 on page 1351
⇒ "Void" (VoidXmpPage) 3.115 on page 1357
⇒ "WuWenTsunTriangularSet" (WuWenTsunTriangularSetXmpPage) 3.116 on page 1360
⇒ "XPBWPolynomial" (XPBWPolynomialXmpPage) 3.118 on page 1374
⇒ "XPolynomial" (XPolynomialXmpPage) 3.119 on page 1395
⇒ "XPolynomialRing" (XPolynomialRingXmpPage) 3.120 on page 1402
⇒ "ZeroDimensionalSolvePackage" (ZeroDimSolvePkgXmpPage) 3.121 on page 1412

— **xmpexp.ht** —

```
\begin{page}{ExamplesExposedPage}{Some Examples of Domains and Packages}
This is a menu of examples of some domains and packages.
Click on any item below to see that section.
\beginscroll
\table{
{ \downlink{AssociationList}{AssociationListXmpPage} }
{ \downlink{BalancedBinaryTree}{BalancedBinaryTreeXmpPage} }
{ \downlink{BasicOperator}{BasicOperatorXmpPage} }
{ \downlink{BinaryExpansion}{BinaryExpansionXmpPage} }
{ \downlink{BinarySearchTree}{BinarySearchTreeXmpPage} }
{ \downlink{CardinalNumber}{CardinalNumberXmpPage} }
{ \downlink{CartesianTensor}{CartesianTensorXmpPage} }
{ \downlink{Character}{CharacterXmpPage} }
{ \downlink{CharacterClass}{CharacterClassXmpPage} }
{ \downlink{CliffordAlgebra}{CliffordAlgebraXmpPage} }
{ \downlink{Complex}{ComplexXmpPage} }
{ \downlink{ContinuedFraction}{ContinuedFractionXmpPage} }
```

```
{ \downlink{CycleIndicators}{CycleIndicatorsXmpPage} }
{ \downlink{DeRhamComplex}{DeRhamComplexXmpPage} }
{ \downlink{DecimalExpansion}{DecimalExpansionXmpPage} }
{ \downlink{DistributedMultivariatePoly}
{DistributedMultivariatePolyXmpPage} }
{ \downlink{DoubleFloat}{DoubleFloatXmpPage} }
{ \downlink{EqTable}{EqTableXmpPage} }
{ \downlink{Equation}{EquationXmpPage} }
{ \downlink{Exit}{ExitXmpPage} }
{ \downlink{Expression}{ExpressionXmpPage} }
{ \downlink{Factored}{FactoredXmpPage} }
{ \downlink{FactoredFunctions2}{FactoredFnsTwoXmpPage} }
{ \downlink{File}{FileXmpPage} }
{ \downlink{FileName}{FileNameXmpPage} }
{ \downlink{FlexibleArray}{FlexibleArrayXmpPage} }
{ \downlink{Float}{FloatXmpPage} }
{ \downlink{Fraction}{FractionXmpPage} }
{ \downlink{FullPartialFracExpansion}
{FullPartialFracExpansionXmpPage} }
{ \downlink{GeneralSparseTable}{GeneralSparseTableXmpPage} }
{ \downlink{GroebnerFactorizationPkg}
{GroebnerFactorizationPkgXmpPage} }
{ \downlink{Heap}{HeapXmpPage} }
{ \downlink{HexadecimalExpansion}{HexExpansionXmpPage} }
{ \downlink{Integer}{IntegerXmpPage} }
{ \downlink{IntegerLinearDependence}{IntegerLinearDependenceXmpPage} }
{ \downlink{IntegerNumberTheoryFunctions}
{IntNumberTheoryFnsXmpPage} }
{ \downlink{Kernel}{KernelXmpPage} }
{ \downlink{KeyedAccessFile}{KeyedAccessFileXmpPage} }
{ \downlink{LexTriangularPackage}{LexTriangularPkgXmpPage} }
{ \downlink{LazardSetSolvingPackage}{LazardSetSolvingPackageXmpPage} }
{ \downlink{Library}{LibraryXmpPage} }
{ \downlink{LieExponentials}{LieExponentialsXmpPage} }
{ \downlink{LiePolynomial}{LiePolynomialXmpPage} }
{ \downlink{LinearOrdinaryDifferentialOperator}
{LinearOrdinaryDifferentialOperatorXmpPage} }
{ \downlink{LinearOrdinaryDifferentialOperator1}
{LinearOrdinaryDifferentialOperatorOneXmpPage} }
{ \downlink{LinearOrdinaryDifferentialOperator2}
{LinearODEOperatorTwoXmpPage} }
{ \downlink{List}{ListXmpPage} }
{ \downlink{LyndonWord}{LyndonWordXmpPage} }
{ \downlink{Magma}{MagmaXmpPage} }
{ \downlink{MakeFunction}{MakeFunctionXmpPage} }
{ \downlink{MappingPackage1}{MappingPackageOneXmpPage} }
{ \downlink{Matrix}{MatrixXmpPage} }
{ \downlink{MultiSet}{MultiSetXmpPage} }
{ \downlink{MultivariatePolynomial}{MultivariatePolyXmpPage} }
{ \downlink{None}{NoneXmpPage} }
```

```
{ \downlink{Octonion}{OctonionXmpPage} }
{ \downlink{OneDimensionalArray}{OneDimensionalArrayXmpPage} }
{ \downlink{Operator}{OperatorXmpPage} }
{ \downlink{OrderedVariableList}{OrderedVariableListXmpPage} }
{ \downlink{OrderlyDifferentialPolynomial}
{OrderlyDifferentialPolyXmpPage} }
{ \downlink{PartialFraction}{PartialFractionXmpPage} }
{ \downlink{Permanent}{PermanentXmpPage} }
{ \downlink{Polynomial}{PolynomialXmpPage} }
{ \downlink{Quaternion}{QuaternionXmpPage} }
{ \downlink{RadixExpansion}{RadixExpansionXmpPage} }
{ \downlink{RealClosure}{RealClosureXmpPage} }
{ \downlink{RegularTriangularSet}{RegularTriangularSetXmpPage} }
{ \downlink{RomanNumeral}{RomanNumeralXmpPage} }
{ \downlink{Segment}{SegmentXmpPage} }
{ \downlink{SegmentBinding}{SegmentBindingXmpPage} }
{ \downlink{Set}{SetXmpPage} }
{ \downlink{SingleInteger}{SingleIntegerXmpPage} }
{ \downlink{SparseTable}{SparseTableXmpPage} }
{ \downlink{SquareMatrix}{SqMatrixXmpPage} }
{ \downlink{SquareFreeRegularTriangularSet}
{SqFreeRegTriangSetXmpPage} }
{ \downlink{Stream}{StreamXmpPage} }
{ \downlink{String}{StringXmpPage} }
{ \downlink{StringTable}{StringTableXmpPage} }
{ \downlink{Symbol}{SymbolXmpPage} }
{ \downlink{Table}{TableXmpPage} }
{ \downlink{TextFile}{TextFileXmpPage} }
{ \downlink{TwoDimensionalArray}{TwoDimensionalArrayXmpPage} }
{ \downlink{UnivariatePolynomial}{UnivariatePolyXmpPage} }
{ \downlink{UnivariateSkewPolynomial}{UnivariateSkewPolyXmpPage} }
{ \downlink{UniversalSegment}{UniversalSegmentXmpPage} }
{ \downlink{Vector}{VectorXmpPage} }
{ \downlink{Void}{VoidXmpPage} }
{ \downlink{WuWenTsunTriangularSet}{WuWenTsunTriangularSetXmpPage} }
{ \downlink{XPBWPolynomial}{XPBWPolynomialXmpPage} }
{ \downlink{XPolynomial}{XPolynomialXmpPage} }
{ \downlink{XPolynomialRing}{XPolynomialRingXmpPage} }
{ \downlink{ZeroDimensionalSolvePackage}
{ZeroDimSolvePkgXmpPage} }
}
\endscroll
\autobuttons
\end{page}
```

3.118 xpbwpoly.ht

XPBWPolynomial

— xpbwpoly.ht —

```
\begin{page}{XPBWPolynomialXmpPage}{XPBWPolynomial}
\beginscroll
Initialisations
\xtc{
}{
\spadpaste{a:Symbol := 'a \bound{a}}
}
\xtc{
}{
\spadpaste{b:Symbol := 'b \bound{b}}
}
\xtc{
}{
\spadpaste{RN    := Fraction(Integer) \bound{RN}}
}
\xtc{
}{
\spadpaste{word   := OrderedFreeMonoid Symbol \bound{word}}
}
\xtc{
}{
\spadpaste{lword := LyndonWord(Symbol) \bound{lword}}
}
\xtc{
}{
\spadpaste{base  := PoincareBirkhoffWittLyndonBasis Symbol \bound{base}}
}
\xtc{
}{
\spadpaste{dpoly := XDistributedPolynomial(Symbol, RN) \bound{dpoly}
\free{RN}}
}
\xtc{
}{
\spadpaste{rpoly := XRecursivePolynomial(Symbol, RN) \bound{rpoly}
\free{RN}}
}
\xtc{
}{
\spadpaste{lpoly := LiePolynomial(Symbol, RN) \bound{lpoly} \free{RN}}
}
\xtc{
```

```
}{
\spadpaste{poly  := XPBWPolynomial(Symbol, RN) \bound{poly} \free{RN}}
}
\xtc{
}{
\spadpaste{liste : List lword := LyndonWordsList([a,b], 6)
\bound{liste} \free{lword a b }}
}

Let's make some polynomials
\xtc{
}{
\spadpaste{0$poly \free{poly}}
}
\xtc{
}{
\spadpaste{1$poly \free{poly}}
}
\xtc{
}{
\spadpaste{p : poly := a \free{a poly} \bound{p}}
}
\xtc{
}{
\spadpaste{q : poly := b \free{b poly} \bound{q}}
}
\xtc{
}{
\spadpaste{pq: poly := p*q \free{p q poly} \bound{pq}}
}
\xtc{
Coerce to distributed polynomial
}{
\spadpaste{pq :: dpoly \free{pq dpoly}}
}

Check some polynomial operations
\xtc{
}{
\spadpaste{mirror pq \free{pq}}
}
\xtc{
}{
\spadpaste{listOfTerms pq \free{pq}}
}
\xtc{
}{
\spadpaste{reductum pq \free{pq}}
}
\xtc{
```

```
}{
\spadpaste{leadingMonomial pq \free{pq}}
}
\xtc{
}{
\spadpaste{coefficients pq \free{pq}}
}
\xtc{
}{
\spadpaste{leadingTerm pq \free{pq}}
}
\xtc{
}{
\spadpaste{degree pq \free{pq}}
}
\xtc{
}{
\spadpaste{pq4:=exp(pq,4) \bound{pq4} \free{pq}}
}
\xtc{
}{
\spadpaste{log(pq4,4) - pq \free{pq4 pq} }
}

Calculations with verification in \axiomType{XDistributedPolynomial}.
\xtc{
}{
\spadpaste{lp1 :lpoly := LiePoly liste.10 \free{liste lpoly} \bound{lp1}}
}
\xtc{
}{
\spadpaste{lp2 :lpoly := LiePoly liste.11 \free{liste lpoly} \bound{lp2}}
}
\xtc{
}{
\spadpaste{lp  :lpoly := [lp1, lp2] \free{lp1 lp2 lpoly} \bound{lp}}
}
\xtc{
}{
\spadpaste{lpd1: dpoly := lp1 \free{lp1 dpoly} \bound{lpd1}}
}
\xtc{
}{
\spadpaste{lpd2: dpoly := lp2 \free{lp2 dpoly} \bound{lpd2}}
}
\xtc{
}{
\spadpaste{lpd : dpoly := lpd1 * lpd2 - lpd2 * lpd1
\free{dpoly lpd1 lpd2} \bound{lpd}}
}
```

```
\xtc{
}{
\spadpaste{lp :: dpoly - lpd \free{lpd dpoly lp}}
}

Calculations with verification in \axiomType{XRecursivePolynomial}.
\xtc{
}{
\spadpaste{p := 3 * lp \free{lp} \bound{pp}}
}
\xtc{
}{
\spadpaste{q := lp1 \free{lp1} \bound{qq}}
}
\xtc{
}{
\spadpaste{pq:= p * q \free{pp qq} \bound{pqpq}}
}
\xtc{
}{
\spadpaste{pr:rpoly := p :: rpoly \free{rpoly pp} \bound{pr}}
}
\xtc{
}{
\spadpaste{qr:rpoly := q :: rpoly \free{rpoly qq} \bound{qr}}
}
\xtc{
}{
\spadpaste{pq :: rpoly - pr*qr \free{pr qr rpoly pqpq} }
}
\endscroll
\autobuttons
\end{page}

\begin{patch}{XPBWPolynomialXmpPagePatch1}
\begin{paste}{XPBWPolynomialXmpPageFull1}{XPBWPolynomialXmpPageEmpty1}
\pastebutton{XPBWPolynomialXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{a:Symbol := 'a\bound{a }}
\indentrel{3}\begin{verbatim}
   (1)  a
                                          Type: Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPageEmpty1}
\begin{paste}{XPBWPolynomialXmpPageEmpty1}{XPBWPolynomialXmpPagePatch1}
\pastebutton{XPBWPolynomialXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{a:Symbol := 'a\bound{a }}
\end{paste}\end{patch}
```

```
\begin{patch}{XPBWPolynomialXmpPagePatch2}
\begin{paste}{XPBWPolynomialXmpPageFull2}{XPBWPolynomialXmpPageEmpty2}
\pastebutton{XPBWPolynomialXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{b:Symbol := 'b\bound{b }}
\indentrel{3}\begin{verbatim}
   (2)  b
                                           Type: Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPageEmpty2}
\begin{paste}{XPBWPolynomialXmpPageEmpty2}{XPBWPolynomialXmpPagePatch2}
\pastebutton{XPBWPolynomialXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{b:Symbol := 'b\bound{b }}
\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPagePatch3}
\begin{paste}{XPBWPolynomialXmpPageFull3}{XPBWPolynomialXmpPageEmpty3}
\pastebutton{XPBWPolynomialXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{RN := Fraction(Integer)\bound{RN }}
\indentrel{3}\begin{verbatim}
   (3)  Fraction Integer
                                           Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPageEmpty3}
\begin{paste}{XPBWPolynomialXmpPageEmpty3}{XPBWPolynomialXmpPagePatch3}
\pastebutton{XPBWPolynomialXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{RN := Fraction(Integer)\bound{RN }}
\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPagePatch4}
\begin{paste}{XPBWPolynomialXmpPageFull4}{XPBWPolynomialXmpPageEmpty4}
\pastebutton{XPBWPolynomialXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{word := OrderedFreeMonoid Symbol\bound{word }}
\indentrel{3}\begin{verbatim}
   (4)  OrderedFreeMonoid Symbol
                                           Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPageEmpty4}
\begin{paste}{XPBWPolynomialXmpPageEmpty4}{XPBWPolynomialXmpPagePatch4}
\pastebutton{XPBWPolynomialXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{word := OrderedFreeMonoid Symbol\bound{word }}
\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPagePatch5}
\begin{paste}{XPBWPolynomialXmpPageFull5}{XPBWPolynomialXmpPageEmpty5}
```

```
\pastebutton{XPBWPolynomialXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{lword := LyndonWord(Symbol)\bound{lword }}
\indentrel{3}\begin{verbatim}
   (5)  LyndonWord Symbol
                                          Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPageEmpty5}
\begin{paste}{XPBWPolynomialXmpPageEmpty5}{XPBWPolynomialXmpPagePatch5}
\pastebutton{XPBWPolynomialXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{lword := LyndonWord(Symbol)\bound{lword }}
\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPagePatch6}
\begin{paste}{XPBWPolynomialXmpPageFull6}{XPBWPolynomialXmpPageEmpty6}
\pastebutton{XPBWPolynomialXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{base := PoincareBirkhoffWittLyndonBasis Symbol\bound{base }}
\indentrel{3}\begin{verbatim}
   (6)  PoincareBirkhoffWittLyndonBasis Symbol
                                          Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPageEmpty6}
\begin{paste}{XPBWPolynomialXmpPageEmpty6}{XPBWPolynomialXmpPagePatch6}
\pastebutton{XPBWPolynomialXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{base := PoincareBirkhoffWittLyndonBasis Symbol\bound{base }}
\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPagePatch7}
\begin{paste}{XPBWPolynomialXmpPageFull7}{XPBWPolynomialXmpPageEmpty7}
\pastebutton{XPBWPolynomialXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{dpoly := XDistributedPolynomial(Symbol, RN)\bound{dpoly }\free{RN }}
\indentrel{3}\begin{verbatim}
   (7)  XDistributedPolynomial(Symbol,Fraction Integer)
                                          Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPageEmpty7}
\begin{paste}{XPBWPolynomialXmpPageEmpty7}{XPBWPolynomialXmpPagePatch7}
\pastebutton{XPBWPolynomialXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{dpoly := XDistributedPolynomial(Symbol, RN)\bound{dpoly }\free{RN }}
\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPagePatch8}
\begin{paste}{XPBWPolynomialXmpPageFull8}{XPBWPolynomialXmpPageEmpty8}
\pastebutton{XPBWPolynomialXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{rpoly := XRecursivePolynomial(Symbol, RN)\bound{rpoly }\free{RN }}
```

```
\indentrel{3}\begin{verbatim}
   (8)  XRecursivePolynomial(Symbol,Fraction Integer)
                                          Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPageEmpty8}
\begin{paste}{XPBWPolynomialXmpPageEmpty8}{XPBWPolynomialXmpPagePatch8}
\pastebutton{XPBWPolynomialXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{rpoly := XRecursivePolynomial(Symbol, RN)\bound{rpoly }\free{RN }}
\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPagePatch9}
\begin{paste}{XPBWPolynomialXmpPageFull9}{XPBWPolynomialXmpPageEmpty9}
\pastebutton{XPBWPolynomialXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{lpoly := LiePolynomial(Symbol, RN)\bound{lpoly }\free{RN }}
\indentrel{3}\begin{verbatim}
   (9)  LiePolynomial(Symbol,Fraction Integer)
                                          Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPageEmpty9}
\begin{paste}{XPBWPolynomialXmpPageEmpty9}{XPBWPolynomialXmpPagePatch9}
\pastebutton{XPBWPolynomialXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{lpoly := LiePolynomial(Symbol, RN)\bound{lpoly }\free{RN }}
\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPagePatch10}
\begin{paste}{XPBWPolynomialXmpPageFull10}{XPBWPolynomialXmpPageEmpty10}
\pastebutton{XPBWPolynomialXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{poly := XPBWPolynomial(Symbol, RN)\bound{poly }\free{RN }}
\indentrel{3}\begin{verbatim}
   (10)  XPBWPolynomial(Symbol,Fraction Integer)
                                          Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPageEmpty10}
\begin{paste}{XPBWPolynomialXmpPageEmpty10}{XPBWPolynomialXmpPagePatch10}
\pastebutton{XPBWPolynomialXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{poly := XPBWPolynomial(Symbol, RN)\bound{poly }\free{RN }}
\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPagePatch11}
\begin{paste}{XPBWPolynomialXmpPageFull11}{XPBWPolynomialXmpPageEmpty11}
\pastebutton{XPBWPolynomialXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{liste : List lword := LyndonWordsList([a,b], 6)\bound{liste }\free{lword a b
\indentrel{3}\begin{verbatim}
   (11)
```

```
                      2        2     3      2 2
   [[a], [b], [a b], [a b], [a b ], [a b], [a b ],
        3     4      3 2     2          2 3          2
    [a b ], [a b], [a b ], [a b a b], [a b ], [a b a b ],
        4     5      4 2     3          3 3     2     2
    [a b ], [a b], [a b ], [a b a b], [a b ], [a b a b ],
      2 2       2 4           3       5
    [a b a b], [a b ], [a b a b ], [a b ]]
                          Type: List LyndonWord Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPageEmpty11}
\begin{paste}{XPBWPolynomialXmpPageEmpty11}{XPBWPolynomialXmpPagePatch11}
\pastebutton{XPBWPolynomialXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{liste : List lword := LyndonWordsList([a,b], 6)\bound{liste }\free{lword a b }}
\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPagePatch12}
\begin{paste}{XPBWPolynomialXmpPageFull12}{XPBWPolynomialXmpPageEmpty12}
\pastebutton{XPBWPolynomialXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{0$poly\free{poly }}
\indentrel{3}\begin{verbatim}
   (12)  0
          Type: XPBWPolynomial(Symbol,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPageEmpty12}
\begin{paste}{XPBWPolynomialXmpPageEmpty12}{XPBWPolynomialXmpPagePatch12}
\pastebutton{XPBWPolynomialXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{0$poly\free{poly }}
\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPagePatch13}
\begin{paste}{XPBWPolynomialXmpPageFull13}{XPBWPolynomialXmpPageEmpty13}
\pastebutton{XPBWPolynomialXmpPageFull13}{\hidepaste}
\tab{5}\spadcommand{1$poly\free{poly }}
\indentrel{3}\begin{verbatim}
   (13)  1
          Type: XPBWPolynomial(Symbol,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPageEmpty13}
\begin{paste}{XPBWPolynomialXmpPageEmpty13}{XPBWPolynomialXmpPagePatch13}
\pastebutton{XPBWPolynomialXmpPageEmpty13}{\showpaste}
\tab{5}\spadcommand{1$poly\free{poly }}
\end{paste}\end{patch}
```

```
\begin{patch}{XPBWPolynomialXmpPagePatch14}
\begin{paste}{XPBWPolynomialXmpPageFull14}{XPBWPolynomialXmpPageEmpty14}
\pastebutton{XPBWPolynomialXmpPageFull14}{\hidepaste}
\tab{5}\spadcommand{p : poly := a\free{a poly }\bound{p }}
\indentrel{3}\begin{verbatim}
   (14)  [a]
          Type: XPBWPolynomial(Symbol,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPageEmpty14}
\begin{paste}{XPBWPolynomialXmpPageEmpty14}{XPBWPolynomialXmpPagePatch14}
\pastebutton{XPBWPolynomialXmpPageEmpty14}{\showpaste}
\tab{5}\spadcommand{p : poly := a\free{a poly }\bound{p }}
\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPagePatch15}
\begin{paste}{XPBWPolynomialXmpPageFull15}{XPBWPolynomialXmpPageEmpty15}
\pastebutton{XPBWPolynomialXmpPageFull15}{\hidepaste}
\tab{5}\spadcommand{q : poly := b\free{b poly }\bound{q }}
\indentrel{3}\begin{verbatim}
   (15)  [b]
          Type: XPBWPolynomial(Symbol,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPageEmpty15}
\begin{paste}{XPBWPolynomialXmpPageEmpty15}{XPBWPolynomialXmpPagePatch15}
\pastebutton{XPBWPolynomialXmpPageEmpty15}{\showpaste}
\tab{5}\spadcommand{q : poly := b\free{b poly }\bound{q }}
\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPagePatch16}
\begin{paste}{XPBWPolynomialXmpPageFull16}{XPBWPolynomialXmpPageEmpty16}
\pastebutton{XPBWPolynomialXmpPageFull16}{\hidepaste}
\tab{5}\spadcommand{pq: poly := p*q\free{p q poly }\bound{pq }}
\indentrel{3}\begin{verbatim}
   (16)  [a b] + [b][a]
          Type: XPBWPolynomial(Symbol,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPageEmpty16}
\begin{paste}{XPBWPolynomialXmpPageEmpty16}{XPBWPolynomialXmpPagePatch16}
\pastebutton{XPBWPolynomialXmpPageEmpty16}{\showpaste}
\tab{5}\spadcommand{pq: poly := p*q\free{p q poly }\bound{pq }}
\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPagePatch17}
\begin{paste}{XPBWPolynomialXmpPageFull17}{XPBWPolynomialXmpPageEmpty17}
```

```
\pastebutton{XPBWPolynomialXmpPageFull17}{\hidepaste}
\tab{5}\spadcommand{pq :: dpoly\free{pq dpoly }}
\indentrel{3}\begin{verbatim}
   (17)  a b
  Type: XDistributedPolynomial(Symbol,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPageEmpty17}
\begin{paste}{XPBWPolynomialXmpPageEmpty17}{XPBWPolynomialXmpPagePatch17}
\pastebutton{XPBWPolynomialXmpPageEmpty17}{\showpaste}
\tab{5}\spadcommand{pq :: dpoly\free{pq dpoly }}
\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPagePatch18}
\begin{paste}{XPBWPolynomialXmpPageFull18}{XPBWPolynomialXmpPageEmpty18}
\pastebutton{XPBWPolynomialXmpPageFull18}{\hidepaste}
\tab{5}\spadcommand{mirror pq\free{pq }}
\indentrel{3}\begin{verbatim}
   (18)  [b][a]
          Type: XPBWPolynomial(Symbol,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPageEmpty18}
\begin{paste}{XPBWPolynomialXmpPageEmpty18}{XPBWPolynomialXmpPagePatch18}
\pastebutton{XPBWPolynomialXmpPageEmpty18}{\showpaste}
\tab{5}\spadcommand{mirror pq\free{pq }}
\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPagePatch19}
\begin{paste}{XPBWPolynomialXmpPageFull19}{XPBWPolynomialXmpPageEmpty19}
\pastebutton{XPBWPolynomialXmpPageFull19}{\hidepaste}
\tab{5}\spadcommand{listOfTerms pq\free{pq }}
\indentrel{3}\begin{verbatim}
   (19)  [[k= [b][a],c= 1],[k= [a b],c= 1]]
Type: List Record(k: PoincareBirkhoffWittLyndonBasis Symbol,c: Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPageEmpty19}
\begin{paste}{XPBWPolynomialXmpPageEmpty19}{XPBWPolynomialXmpPagePatch19}
\pastebutton{XPBWPolynomialXmpPageEmpty19}{\showpaste}
\tab{5}\spadcommand{listOfTerms pq\free{pq }}
\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPagePatch20}
\begin{paste}{XPBWPolynomialXmpPageFull20}{XPBWPolynomialXmpPageEmpty20}
\pastebutton{XPBWPolynomialXmpPageFull20}{\hidepaste}
\tab{5}\spadcommand{reductum pq\free{pq }}
```

```
\indentrel{3}\begin{verbatim}
   (20)  [a b]
          Type: XPBWPolynomial(Symbol,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPageEmpty20}
\begin{paste}{XPBWPolynomialXmpPageEmpty20}{XPBWPolynomialXmpPagePatch20}
\pastebutton{XPBWPolynomialXmpPageEmpty20}{\showpaste}
\tab{5}\spadcommand{reductum pq\free{pq }}
\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPagePatch21}
\begin{paste}{XPBWPolynomialXmpPageFull21}{XPBWPolynomialXmpPageEmpty21}
\pastebutton{XPBWPolynomialXmpPageFull21}{\hidepaste}
\tab{5}\spadcommand{leadingMonomial pq\free{pq }}
\indentrel{3}\begin{verbatim}
   (21)  [b][a]
           Type: PoincareBirkhoffWittLyndonBasis Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPageEmpty21}
\begin{paste}{XPBWPolynomialXmpPageEmpty21}{XPBWPolynomialXmpPagePatch21}
\pastebutton{XPBWPolynomialXmpPageEmpty21}{\showpaste}
\tab{5}\spadcommand{leadingMonomial pq\free{pq }}
\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPagePatch22}
\begin{paste}{XPBWPolynomialXmpPageFull22}{XPBWPolynomialXmpPageEmpty22}
\pastebutton{XPBWPolynomialXmpPageFull22}{\hidepaste}
\tab{5}\spadcommand{coefficients pq\free{pq }}
\indentrel{3}\begin{verbatim}
   (22)  [1,1]
                             Type: List Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPageEmpty22}
\begin{paste}{XPBWPolynomialXmpPageEmpty22}{XPBWPolynomialXmpPagePatch22}
\pastebutton{XPBWPolynomialXmpPageEmpty22}{\showpaste}
\tab{5}\spadcommand{coefficients pq\free{pq }}
\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPagePatch23}
\begin{paste}{XPBWPolynomialXmpPageFull23}{XPBWPolynomialXmpPageEmpty23}
\pastebutton{XPBWPolynomialXmpPageFull23}{\hidepaste}
\tab{5}\spadcommand{leadingTerm pq\free{pq }}
\indentrel{3}\begin{verbatim}
   (23)  [k= [b][a],c= 1]
```

```
Type: Record(k: PoincareBirkhoffWittLyndonBasis Symbol,c: Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPageEmpty23}
\begin{paste}{XPBWPolynomialXmpPageEmpty23}{XPBWPolynomialXmpPagePatch23}
\pastebutton{XPBWPolynomialXmpPageEmpty23}{\showpaste}
\tab{5}\spadcommand{leadingTerm pq\free{pq }}
\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPagePatch24}
\begin{paste}{XPBWPolynomialXmpPageFull24}{XPBWPolynomialXmpPageEmpty24}
\pastebutton{XPBWPolynomialXmpPageFull24}{\hidepaste}
\tab{5}\spadcommand{degree pq\free{pq }}
\indentrel{3}\begin{verbatim}
   (24)  2
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPageEmpty24}
\begin{paste}{XPBWPolynomialXmpPageEmpty24}{XPBWPolynomialXmpPagePatch24}
\pastebutton{XPBWPolynomialXmpPageEmpty24}{\showpaste}
\tab{5}\spadcommand{degree pq\free{pq }}
\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPagePatch25}
\begin{paste}{XPBWPolynomialXmpPageFull25}{XPBWPolynomialXmpPageEmpty25}
\pastebutton{XPBWPolynomialXmpPageFull25}{\hidepaste}
\tab{5}\spadcommand{pq4:=exp(pq,4)\bound{pq4 }\free{pq }}
\indentrel{3}\begin{verbatim}
   (25)
                          1              1     2
     1 + [a b] + [b][a] +  [a b][a b] +  [a b ][a]
                          2              2
   +
     1      2     3              1
      [b][a b] +  [b][a b][a] +  [b][b][a][a]
     2            2              2
          Type: XPBWPolynomial(Symbol,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPageEmpty25}
\begin{paste}{XPBWPolynomialXmpPageEmpty25}{XPBWPolynomialXmpPagePatch25}
\pastebutton{XPBWPolynomialXmpPageEmpty25}{\showpaste}
\tab{5}\spadcommand{pq4:=exp(pq,4)\bound{pq4 }\free{pq }}
\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPagePatch26}
```

```
\begin{paste}{XPBWPolynomialXmpPageFull26}{XPBWPolynomialXmpPageEmpty26}
\pastebutton{XPBWPolynomialXmpPageFull26}{\hidepaste}
\tab{5}\spadcommand{log(pq4,4) - pq\free{pq4 pq }}
\indentrel{3}\begin{verbatim}
   (26)  0
          Type: XPBWPolynomial(Symbol,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPageEmpty26}
\begin{paste}{XPBWPolynomialXmpPageEmpty26}{XPBWPolynomialXmpPagePatch26}
\pastebutton{XPBWPolynomialXmpPageEmpty26}{\showpaste}
\tab{5}\spadcommand{log(pq4,4) - pq\free{pq4 pq }}
\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPagePatch27}
\begin{paste}{XPBWPolynomialXmpPageFull27}{XPBWPolynomialXmpPageEmpty27}
\pastebutton{XPBWPolynomialXmpPageFull27}{\hidepaste}
\tab{5}\spadcommand{lp1 :lpoly := LiePoly liste.10\free{liste lpoly }\bound{lp1 }}
\indentrel{3}\begin{verbatim}
           3 2
   (27)  [a b ]
           Type: LiePolynomial(Symbol,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPageEmpty27}
\begin{paste}{XPBWPolynomialXmpPageEmpty27}{XPBWPolynomialXmpPagePatch27}
\pastebutton{XPBWPolynomialXmpPageEmpty27}{\showpaste}
\tab{5}\spadcommand{lp1 :lpoly := LiePoly liste.10\free{liste lpoly }\bound{lp1 }}
\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPagePatch28}
\begin{paste}{XPBWPolynomialXmpPageFull28}{XPBWPolynomialXmpPageEmpty28}
\pastebutton{XPBWPolynomialXmpPageFull28}{\hidepaste}
\tab{5}\spadcommand{lp2 :lpoly := LiePoly liste.11\free{liste lpoly }\bound{lp2 }}
\indentrel{3}\begin{verbatim}
           2
   (28)  [a b a b]
           Type: LiePolynomial(Symbol,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPageEmpty28}
\begin{paste}{XPBWPolynomialXmpPageEmpty28}{XPBWPolynomialXmpPagePatch28}
\pastebutton{XPBWPolynomialXmpPageEmpty28}{\showpaste}
\tab{5}\spadcommand{lp2 :lpoly := LiePoly liste.11\free{liste lpoly }\bound{lp2 }}
\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPagePatch29}
```

```
\begin{paste}{XPBWPolynomialXmpPageFull29}{XPBWPolynomialXmpPageEmpty29}
\pastebutton{XPBWPolynomialXmpPageFull29}{\hidepaste}
\tab{5}\spadcommand{lp :lpoly := [lp1, lp2]\free{lp1 lp2 lpoly }\bound{lp }}
\indentrel{3}\begin{verbatim}
          3 2 2
   (29)  [a b a b a b]
          Type: LiePolynomial(Symbol,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPageEmpty29}
\begin{paste}{XPBWPolynomialXmpPageEmpty29}{XPBWPolynomialXmpPagePatch29}
\pastebutton{XPBWPolynomialXmpPageEmpty29}{\showpaste}
\tab{5}\spadcommand{lp :lpoly := [lp1, lp2]\free{lp1 lp2 lpoly }\bound{lp }}
\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPagePatch30}
\begin{paste}{XPBWPolynomialXmpPageFull30}{XPBWPolynomialXmpPageEmpty30}
\pastebutton{XPBWPolynomialXmpPageFull30}{\hidepaste}
\tab{5}\spadcommand{lpd1: dpoly := lp1\free{lp1 dpoly }\bound{lpd1 }}
\indentrel{3}\begin{verbatim}
   (30)
      3 2     2         2 2                     2 2
     a b  - 2a b a b - a b a + 4a b a b a - a b a
   +
              2    2 3
     - 2b a b a  + b a
  Type: XDistributedPolynomial(Symbol,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPageEmpty30}
\begin{paste}{XPBWPolynomialXmpPageEmpty30}{XPBWPolynomialXmpPagePatch30}
\pastebutton{XPBWPolynomialXmpPageEmpty30}{\showpaste}
\tab{5}\spadcommand{lpd1: dpoly := lp1\free{lp1 dpoly }\bound{lpd1 }}
\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPagePatch31}
\begin{paste}{XPBWPolynomialXmpPageFull31}{XPBWPolynomialXmpPageEmpty31}
\pastebutton{XPBWPolynomialXmpPageFull31}{\hidepaste}
\tab{5}\spadcommand{lpd2: dpoly := lp2\free{lp2 dpoly }\bound{lpd2 }}
\indentrel{3}\begin{verbatim}
   (31)
      2         2 2          2                     2 2
     a b a b - a b a - 3a b a b + 4a b a b a - a b a
   +
         3        2             2
     2b a b - 3b a b a + b a b a
  Type: XDistributedPolynomial(Symbol,Fraction Integer)
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPageEmpty31}
\begin{paste}{XPBWPolynomialXmpPageEmpty31}{XPBWPolynomialXmpPagePatch31}
\pastebutton{XPBWPolynomialXmpPageEmpty31}{\showpaste}
\tab{5}\spadcommand{lpd2: dpoly := lp2\free{lp2 dpoly }\bound{lpd2 }}
\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPagePatch32}
\begin{paste}{XPBWPolynomialXmpPageFull32}{XPBWPolynomialXmpPageEmpty32}
\pastebutton{XPBWPolynomialXmpPageFull32}{\hidepaste}
\tab{5}\spadcommand{lpd : dpoly := lpd1 * lpd2 - lpd2 * lpd1\free{dpoly lpd1 lpd2 }\bound{lpd }}
\indentrel{3}\begin{verbatim}
   (32)
      3 2 2         3 2 2 2      3 2     2
     a b a b a b - a b a b a - 3a b a b a b
   +
       3 2             3 2   2 2     3 3 3      3 3 2
     4a b a b a b a - a b a b a  + 2a b a b - 3a b a b a
   +
      3 3     2    2       3 2     2       2 2
     a b a b a  - a b a b a b  + 3a b a b a b a
   +
       2           2       2
     6a b a b a b a b - 12a b a b a b a b a
   +
       2         2 2     2     2 3      2     2 2
     3a b a b a b a  - 4a b a b a b + 6a b a b a b a
   +
        2     3 3    2 2 4 2     2 2 3          2 2 2   2
     - a b a b a  + a b a b  - 3a b a b a b + 3a b a b a b
   +
         2 2     3      2 2     2        2 2         2
     - 2a b a b a b + 3a b a b a b a - 3a b a b a b a
   +
      2 2   2 3         2   3 2         2   2
     a b a b a  + 3a b a b a b  - 6a b a b a b a b
   +
            2   2 2          2
     - 3a b a b a b a + 12a b a b a b a b a
   +
            2     2 2         2 2     2         2 3 3
     - 3a b a b a b a  - 6a b a b a b a  + 3a b a b a
   +
                4 2             3
     - 4a b a b a b  + 12a b a b a b a b
   +
                 2   2                 3
     - 12a b a b a b a b + 8a b a b a b a b
   +
```

```
                      2                         2
     - 12a b a b a b a b a + 12a b a b a b a b a
   +
                   2 3      2 5 2       2 4
     - 4a b a b a b a  + a b a b  - 3a b a b a b
   +
         2 3   2        2 2   3        2 2   2
     3a b a b a b - 2a b a b a b + 3a b a b a b a
   +
           2 2       2      2 2 2 3       3   3 2
     - 3a b a b a b a  + a b a b a  - 2b a b a b
   +
         3   2            3   2 2        3
     4b a b a b a b + 2b a b a b a - 8b a b a b a b a
   +
         3     2 2       3 2     2       3 3 3
     2b a b a b a  + 4b a b a b a  - 2b a b a
   +
         2   4 2       2   3            2   3 2
     3b a b a b  - 6b a b a b a b - 3b a b a b a
   +
          2   2              2   2 2 2       2           2
     12b a b a b a b a - 3b a b a b a  - 6b a b a b a b a
   +
         2     2 3          5 2           4 2
     3b a b a b a  - b a b a b  + 3b a b a b a
   +
             3   2             3                  3 2 2
     6b a b a b a b - 12b a b a b a b a + 3b a b a b a
   +
               2   3            2   2             2 2 3
     - 4b a b a b a b + 6b a b a b a b a - b a b a b a
   +
      2 5         2 5 2      2 4   2      2 4
     b a b a b - b a b a - 3b a b a b + 4b a b a b a
   +
      2 4 2 2     2 3   3      2 3   2       2 3       2
   - b a b a  + 2b a b a b - 3b a b a b a + b a b a b a
  Type: XDistributedPolynomial(Symbol,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPageEmpty32}
\begin{paste}{XPBWPolynomialXmpPageEmpty32}{XPBWPolynomialXmpPagePatch32}
\pastebutton{XPBWPolynomialXmpPageEmpty32}{\showpaste}
\tab{5}\spadcommand{lpd : dpoly := lpd1 * lpd2 - lpd2 * lpd1\free{dpoly lpd1 lpd2 }\bound{lpd }}
\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPagePatch33}
\begin{paste}{XPBWPolynomialXmpPageFull33}{XPBWPolynomialXmpPageEmpty33}
```

```
\pastebutton{XPBWPolynomialXmpPageFull33}{\hidepaste}
\tab{5}\spadcommand{lp :: dpoly - lpd\free{lpd dpoly lp }}
\indentrel{3}\begin{verbatim}
   (33)  0
  Type: XDistributedPolynomial(Symbol,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPageEmpty33}
\begin{paste}{XPBWPolynomialXmpPageEmpty33}{XPBWPolynomialXmpPagePatch33}
\pastebutton{XPBWPolynomialXmpPageEmpty33}{\showpaste}
\tab{5}\spadcommand{lp :: dpoly - lpd\free{lpd dpoly lp }}
\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPagePatch34}
\begin{paste}{XPBWPolynomialXmpPageFull34}{XPBWPolynomialXmpPageEmpty34}
\pastebutton{XPBWPolynomialXmpPageFull34}{\hidepaste}
\tab{5}\spadcommand{p := 3 * lp\free{lp }\bound{pp }}
\indentrel{3}\begin{verbatim}
            3 2 2
   (34)  3[a b a b a b]
          Type: XPBWPolynomial(Symbol,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPageEmpty34}
\begin{paste}{XPBWPolynomialXmpPageEmpty34}{XPBWPolynomialXmpPagePatch34}
\pastebutton{XPBWPolynomialXmpPageEmpty34}{\showpaste}
\tab{5}\spadcommand{p := 3 * lp\free{lp }\bound{pp }}
\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPagePatch35}
\begin{paste}{XPBWPolynomialXmpPageFull35}{XPBWPolynomialXmpPageEmpty35}
\pastebutton{XPBWPolynomialXmpPageFull35}{\hidepaste}
\tab{5}\spadcommand{q := lp1\free{lp1 }\bound{qq }}
\indentrel{3}\begin{verbatim}
           3 2
   (35)  [a b ]
          Type: XPBWPolynomial(Symbol,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPageEmpty35}
\begin{paste}{XPBWPolynomialXmpPageEmpty35}{XPBWPolynomialXmpPagePatch35}
\pastebutton{XPBWPolynomialXmpPageEmpty35}{\showpaste}
\tab{5}\spadcommand{q := lp1\free{lp1 }\bound{qq }}
\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPagePatch36}
\begin{paste}{XPBWPolynomialXmpPageFull36}{XPBWPolynomialXmpPageEmpty36}
```

```
\pastebutton{XPBWPolynomialXmpPageFull36}{\hidepaste}
\tab{5}\spadcommand{pq:= p * q\free{pp qq }\bound{pqpq }}
\indentrel{3}\begin{verbatim}
            3 2 2        3 2
   (36)  3[a b a b a b][a b ]
          Type: XPBWPolynomial(Symbol,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPageEmpty36}
\begin{paste}{XPBWPolynomialXmpPageEmpty36}{XPBWPolynomialXmpPagePatch36}
\pastebutton{XPBWPolynomialXmpPageEmpty36}{\showpaste}
\tab{5}\spadcommand{pq:= p * q\free{pp qq }\bound{pqpq }}
\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPagePatch37}
\begin{paste}{XPBWPolynomialXmpPageFull37}{XPBWPolynomialXmpPageEmpty37}
\pastebutton{XPBWPolynomialXmpPageFull37}{\hidepaste}
\tab{5}\spadcommand{pr:rpoly := p :: rpoly\free{rpoly pp }\bound{pr }}
\indentrel{3}\begin{verbatim}
   (37)
       a
    *
           a
        *
               a b b
            *
                   a
                *
                     a b(a b 3 + b a(- 3))
                   +
                     b(a(a b(- 9) + b a 12) + b a a(- 3))
               +
                 b a(a(a b 6 + b a(- 9)) + b a a 3)
           +
               b
            *
                   a b
                *
                       a
                    *
                         a(a b b(- 3) + b b a 9)
                       +
                       b(a(a b 18 + b a(- 36)) + b a a 9)
                   +
                       b
                    *
                         a a(a b(- 12) + b a 18)
                       +
                         b a a a(- 3)
```

```
          +
              b a
           *
                  a
               *
                 (a(a b b 3 + b a b(- 9)) + b a a b 9)
              +
                  b
               *
                      a
                   *
                        a(a b(- 6) + b a 9)
                      +
                        b a a(- 9)
                  +
                    b a a a 3
  +
      b
   *
          a
       *
              a b
           *
                  a
               *
                      a
                   *
                        a b b 9
                      +
                        b(a b(- 18) + b a(- 9))
                  +
                    b(a b a 36 + b a a(- 9))
              +
                b(a b a a(- 18) + b a a a 9)
          +
              b a
           *
                  a
               *
                    a(a b b(- 12) + b a b 36)
                  +
                    b a a b(- 36)
              +
                  b
               *
                      a
                   *
                        a(a b 24 + b a(- 36))
                      +
                        b a a 36
```

```
                     +
                       b a a a(- 12)
         +
             b a a
          *
               a(a(a b b 3 + b a b(- 9)) + b a a b 9)
             +
                 b
              *
                   a(a(a b(- 6) + b a 9) + b a a(- 9))
                 +
                   b a a a 3
 +
     b
  *
         a
      *
             a
          *
                 a b
              *
                     a
                  *
                       a(a b b(- 6) + b(a b 12 + b a 6))
                     +
                       b(a b a(- 24) + b a a 6)
                 +
                   b(a b a a 12 + b a a a(- 6))
             +
                 b a
              *
                     a
                  *
                         a
                      *
                           a b b 9
                         +
                           b(a b(- 18) + b a(- 9))
                     +
                       b(a b a 36 + b a a(- 9))
                 +
                   b(a b a a(- 18) + b a a a 9)
         +
             b a a
          *
                 a
              *
                   a(a b b(- 3) + b b a 9)
                 +
                   b(a(a b 18 + b a(- 36)) + b a a 9)
```

```
              +
              b(a a(a b(- 12) + b a 18) + b a a a(- 3))
      +
          b a a a
       *
              a
           *
                a b(a b 3 + b a(- 3))
              +
                b(a(a b(- 9) + b a 12) + b a a(- 3))
          +
            b a(a(a b 6 + b a(- 9)) + b a a 3)
   Type: XRecursivePolynomial(Symbol,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPageEmpty37}
\begin{paste}{XPBWPolynomialXmpPageEmpty37}{XPBWPolynomialXmpPagePatch37}
\pastebutton{XPBWPolynomialXmpPageEmpty37}{\showpaste}
\tab{5}\spadcommand{pr:rpoly := p :: rpoly\free{rpoly pp }\bound{pr }}
\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPagePatch38}
\begin{paste}{XPBWPolynomialXmpPageFull38}{XPBWPolynomialXmpPageEmpty38}
\pastebutton{XPBWPolynomialXmpPageFull38}{\hidepaste}
\tab{5}\spadcommand{qr:rpoly := q :: rpoly\free{rpoly qq }\bound{qr }}
\indentrel{3}\begin{verbatim}
   (38)
       a
    *
         a(a b b 1 + b(a b(- 2) + b a(- 1)))
       +
         b(a b a 4 + b a a(- 1))
   +
     b(a b a a(- 2) + b a a a 1)
   Type: XRecursivePolynomial(Symbol,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPageEmpty38}
\begin{paste}{XPBWPolynomialXmpPageEmpty38}{XPBWPolynomialXmpPagePatch38}
\pastebutton{XPBWPolynomialXmpPageEmpty38}{\showpaste}
\tab{5}\spadcommand{qr:rpoly := q :: rpoly\free{rpoly qq }\bound{qr }}
\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPagePatch39}
\begin{paste}{XPBWPolynomialXmpPageFull39}{XPBWPolynomialXmpPageEmpty39}
\pastebutton{XPBWPolynomialXmpPageFull39}{\hidepaste}
\tab{5}\spadcommand{pq :: rpoly - pr*qr\free{pr qr rpoly pqpq }}
\indentrel{3}\begin{verbatim}
```

```
   (39)  0
    Type: XRecursivePolynomial(Symbol,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPBWPolynomialXmpPageEmpty39}
\begin{paste}{XPBWPolynomialXmpPageEmpty39}{XPBWPolynomialXmpPagePatch39}
\pastebutton{XPBWPolynomialXmpPageEmpty39}{\showpaste}
\tab{5}\spadcommand{pq :: rpoly - pr*qr\free{pr qr rpoly pqpq }}
\end{paste}\end{patch}
```

————-

3.119 xpoly.ht

XPolynomial

— xpoly.ht —

```
\begin{page}{XPolynomialXmpPage}{XPolynomial}
\beginscroll
The \spadtype{XPolynomial} domain constructor implements
multivariate polynomials
whose set of variables is \spadtype{Symbol}.
These variables do not commute.
The only parameter of this construtor is
the coefficient ring which may be non-commutative.
However, coefficients and variables commute.
The representation of the polynomials is recursive.
The abbreviation for \spadtype{XPolynomial} is \spadtype{XPOLY}.

Other constructors like \spadtype{XPolynomialRing},
\spadtype{XRecursivePolynomial}
\spadtype{XDistributedPolynomial},
\spadtype{LiePolynomial} and
\spadtype{XPBWPolynomial}
implement multivariate polynomials
in non-commutative variables.

We illustrate now some of the facilities of the \spadtype{XPOLY}
domain constructor.


\xtc{
Define a polynomial ring over the integers.
}{
```

```
\spadpaste{poly := XPolynomial(Integer) \bound{poly}}
}

\xtc{
Define a first polynomial,
}{
\spadpaste{pr: poly := 2*x + 3*y-5  \free{poly} \bound{pr}}
}


\xtc{
and a second one.
}{
\spadpaste{pr2: poly := pr*pr  \free{poly} \bound{pr2}}
}

\xtc{
Rewrite {\bf pr} in a distributive way,
}{
\spadpaste{pd  := expand pr   \free{pr} \bound{pd}}
}

\xtc{
compute its square,
}{
\spadpaste{pd2 := pd*pd   \free{pd} \bound{pd2}}
}

\xtc{
and checks that:
}{
\spadpaste{expand(pr2) - pd2 \free{pr2} \free{pd2}}
}


\xtc{
We define:
}{
\spadpaste{qr :=  pr**3 \free{pr} \bound{qr}}
}

\xtc{
and:
}{
\spadpaste{qd :=  pd**3 \free{pd} \bound{qd}}
}

\xtc{
We truncate {\bf qd} at degree {\bf 3}:
}{
```

```
\spadpaste{ trunc(qd,2) \free{qd}}
}

\xtc{
The same for {\bf qr}:
}{
\spadpaste{trunc(qr,2) \free{qr}}
}

\xtc{
We define:
}{
\spadpaste{Word := OrderedFreeMonoid Symbol \bound{Word}}
}

\xtc{
and:
}{
\spadpaste{w: Word := x*y**2 \free{Word} \bound{w}}
}

\xtc{
The we can compute the right-quotient of {\bf qr} by {\bf r}:
}{
\spadpaste{rquo(qr,w) \free{qr} \free{w}}
}

\xtc{
and the shuffle-product of {\bf pr} by {\bf r}:
}{
\spadpaste{sh(pr,w::poly) \free{pr} \free{w}}
}
\endscroll
\autobuttons
\end{page}

\begin{patch}{XPolynomialXmpPagePatch1}
\begin{paste}{XPolynomialXmpPageFull1}{XPolynomialXmpPageEmpty1}
\pastebutton{XPolynomialXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{poly := XPolynomial(Integer)\bound{poly }}
\indentrel{3}\begin{verbatim}
   (1)  XPolynomial Integer
                                          Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPolynomialXmpPageEmpty1}
\begin{paste}{XPolynomialXmpPageEmpty1}{XPolynomialXmpPagePatch1}
\pastebutton{XPolynomialXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{poly := XPolynomial(Integer)\bound{poly }}
```

```
\end{paste}\end{patch}

\begin{patch}{XPolynomialXmpPagePatch2}
\begin{paste}{XPolynomialXmpPageFull2}{XPolynomialXmpPageEmpty2}
\pastebutton{XPolynomialXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{pr: poly := 2*x + 3*y-5\free{poly }\bound{pr }}
\indentrel{3}\begin{verbatim}
   (2)  - 5 + x 2 + y 3
                             Type: XPolynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPolynomialXmpPageEmpty2}
\begin{paste}{XPolynomialXmpPageEmpty2}{XPolynomialXmpPagePatch2}
\pastebutton{XPolynomialXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{pr: poly := 2*x + 3*y-5\free{poly }\bound{pr }}
\end{paste}\end{patch}

\begin{patch}{XPolynomialXmpPagePatch3}
\begin{paste}{XPolynomialXmpPageFull3}{XPolynomialXmpPageEmpty3}
\pastebutton{XPolynomialXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{pr2: poly := pr*pr\free{poly }\bound{pr2 }}
\indentrel{3}\begin{verbatim}
   (3)  25 + x(- 20 + x 4 + y 6) + y(- 30 + x 6 + y 9)
                             Type: XPolynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPolynomialXmpPageEmpty3}
\begin{paste}{XPolynomialXmpPageEmpty3}{XPolynomialXmpPagePatch3}
\pastebutton{XPolynomialXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{pr2: poly := pr*pr\free{poly }\bound{pr2 }}
\end{paste}\end{patch}

\begin{patch}{XPolynomialXmpPagePatch4}
\begin{paste}{XPolynomialXmpPageFull4}{XPolynomialXmpPageEmpty4}
\pastebutton{XPolynomialXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{pd := expand pr\free{pr }\bound{pd }}
\indentrel{3}\begin{verbatim}
   (4)  - 5 + 2x + 3y
           Type: XDistributedPolynomial(Symbol,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPolynomialXmpPageEmpty4}
\begin{paste}{XPolynomialXmpPageEmpty4}{XPolynomialXmpPagePatch4}
\pastebutton{XPolynomialXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{pd := expand pr\free{pr }\bound{pd }}
\end{paste}\end{patch}
```

```
\begin{patch}{XPolynomialXmpPagePatch5}
\begin{paste}{XPolynomialXmpPageFull5}{XPolynomialXmpPageEmpty5}
\pastebutton{XPolynomialXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{pd2 := pd*pd\free{pd }\bound{pd2 }}
\indentrel{3}\begin{verbatim}
                           2                    2
   (5)  25 - 20x - 30y + 4x  + 6x y + 6y x + 9y
           Type: XDistributedPolynomial(Symbol,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPolynomialXmpPageEmpty5}
\begin{paste}{XPolynomialXmpPageEmpty5}{XPolynomialXmpPagePatch5}
\pastebutton{XPolynomialXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{pd2 := pd*pd\free{pd }\bound{pd2 }}
\end{paste}\end{patch}

\begin{patch}{XPolynomialXmpPagePatch6}
\begin{paste}{XPolynomialXmpPageFull6}{XPolynomialXmpPageEmpty6}
\pastebutton{XPolynomialXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{expand(pr2) - pd2\free{pr2 }\free{pd2 }}
\indentrel{3}\begin{verbatim}
   (6)  0
           Type: XDistributedPolynomial(Symbol,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPolynomialXmpPageEmpty6}
\begin{paste}{XPolynomialXmpPageEmpty6}{XPolynomialXmpPagePatch6}
\pastebutton{XPolynomialXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{expand(pr2) - pd2\free{pr2 }\free{pd2 }}
\end{paste}\end{patch}

\begin{patch}{XPolynomialXmpPagePatch7}
\begin{paste}{XPolynomialXmpPageFull7}{XPolynomialXmpPageEmpty7}
\pastebutton{XPolynomialXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{qr := pr**3\free{pr }\bound{qr }}
\indentrel{3}\begin{verbatim}
   (7)
     - 125
   +
     x(150 + x(- 60 + x 8 + y 12) + y(- 90 + x 12 + y 18))
   +
   y(225 + x(- 90 + x 12 + y 18) + y(- 135 + x 18 + y 27))
                              Type: XPolynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPolynomialXmpPageEmpty7}
\begin{paste}{XPolynomialXmpPageEmpty7}{XPolynomialXmpPagePatch7}
```

```
\pastebutton{XPolynomialXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{qr := pr**3\free{pr }\bound{qr }}
\end{paste}\end{patch}

\begin{patch}{XPolynomialXmpPagePatch8}
\begin{paste}{XPolynomialXmpPageFull8}{XPolynomialXmpPageEmpty8}
\pastebutton{XPolynomialXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{qd := pd**3\free{pd }\bound{qd }}
\indentrel{3}\begin{verbatim}
   (8)
                             2                       2
     - 125 + 150x + 225y - 60x  - 90x y - 90y x - 135y
   +
       3      2                   2        2
     8x  + 12x y + 12x y x + 18x y  + 12y x  + 18y x y
   +
        2       3
     18y x + 27y
           Type: XDistributedPolynomial(Symbol,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPolynomialXmpPageEmpty8}
\begin{paste}{XPolynomialXmpPageEmpty8}{XPolynomialXmpPagePatch8}
\pastebutton{XPolynomialXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{qd := pd**3\free{pd }\bound{qd }}
\end{paste}\end{patch}

\begin{patch}{XPolynomialXmpPagePatch9}
\begin{paste}{XPolynomialXmpPageFull9}{XPolynomialXmpPageEmpty9}
\pastebutton{XPolynomialXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{ trunc(qd,2)\free{qd }}
\indentrel{3}\begin{verbatim}
   (9)
                           2                       2
   - 125 + 150x + 225y - 60x  - 90x y - 90y x - 135y
           Type: XDistributedPolynomial(Symbol,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPolynomialXmpPageEmpty9}
\begin{paste}{XPolynomialXmpPageEmpty9}{XPolynomialXmpPagePatch9}
\pastebutton{XPolynomialXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{ trunc(qd,2)\free{qd }}
\end{paste}\end{patch}

\begin{patch}{XPolynomialXmpPagePatch10}
\begin{paste}{XPolynomialXmpPageFull10}{XPolynomialXmpPageEmpty10}
\pastebutton{XPolynomialXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{trunc(qr,2)\free{qr }}
```

```
\indentrel{3}\begin{verbatim}
   (10)
     - 125 + x(150 + x(- 60) + y(- 90))
   +
     y(225 + x(- 90) + y(- 135))
                              Type: XPolynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPolynomialXmpPageEmpty10}
\begin{paste}{XPolynomialXmpPageEmpty10}{XPolynomialXmpPagePatch10}
\pastebutton{XPolynomialXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{trunc(qr,2)\free{qr }}
\end{paste}\end{patch}

\begin{patch}{XPolynomialXmpPagePatch11}
\begin{paste}{XPolynomialXmpPageFull11}{XPolynomialXmpPageEmpty11}
\pastebutton{XPolynomialXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{Word := OrderedFreeMonoid Symbol\bound{Word }}
\indentrel{3}\begin{verbatim}
   (11)  OrderedFreeMonoid Symbol
                                           Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPolynomialXmpPageEmpty11}
\begin{paste}{XPolynomialXmpPageEmpty11}{XPolynomialXmpPagePatch11}
\pastebutton{XPolynomialXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{Word := OrderedFreeMonoid Symbol\bound{Word }}
\end{paste}\end{patch}

\begin{patch}{XPolynomialXmpPagePatch12}
\begin{paste}{XPolynomialXmpPageFull12}{XPolynomialXmpPageEmpty12}
\pastebutton{XPolynomialXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{w: Word := x*y**2\free{Word }\bound{w }}
\indentrel{3}\begin{verbatim}
            2
   (12)  x y
                         Type: OrderedFreeMonoid Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPolynomialXmpPageEmpty12}
\begin{paste}{XPolynomialXmpPageEmpty12}{XPolynomialXmpPagePatch12}
\pastebutton{XPolynomialXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{w: Word := x*y**2\free{Word }\bound{w }}
\end{paste}\end{patch}

\begin{patch}{XPolynomialXmpPagePatch13}
\begin{paste}{XPolynomialXmpPageFull13}{XPolynomialXmpPageEmpty13}
```

```
\pastebutton{XPolynomialXmpPageFull13}{\hidepaste}
\tab{5}\spadcommand{rquo(qr,w)\free{qr }\free{w }}
\indentrel{3}\begin{verbatim}
   (13)  18
                              Type: XPolynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPolynomialXmpPageEmpty13}
\begin{paste}{XPolynomialXmpPageEmpty13}{XPolynomialXmpPagePatch13}
\pastebutton{XPolynomialXmpPageEmpty13}{\showpaste}
\tab{5}\spadcommand{rquo(qr,w)\free{qr }\free{w }}
\end{paste}\end{patch}

\begin{patch}{XPolynomialXmpPagePatch14}
\begin{paste}{XPolynomialXmpPageFull14}{XPolynomialXmpPageEmpty14}
\pastebutton{XPolynomialXmpPageFull14}{\hidepaste}
\tab{5}\spadcommand{sh(pr,w::poly)\free{pr }\free{w }}
\indentrel{3}\begin{verbatim}
   (14)
   x(x y y 4 + y(x y 2 + y(- 5 + x 2 + y 9))) + y x y y 3
                              Type: XPolynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPolynomialXmpPageEmpty14}
\begin{paste}{XPolynomialXmpPageEmpty14}{XPolynomialXmpPagePatch14}
\pastebutton{XPolynomialXmpPageEmpty14}{\showpaste}
\tab{5}\spadcommand{sh(pr,w::poly)\free{pr }\free{w }}
\end{paste}\end{patch}
```

————-

3.120 xpr.ht

XPolynomialRing

— xpr.ht —

```
\begin{page}{XPolynomialRingXmpPage}{XPolynomialRing}
\beginscroll
The \spadtype{XPolynomialRing} domain constructor implements
generalized polynomials with coefficients from an arbitrary \spadtype{Ring}
(not necessarily commutative) and whose exponents are
words from an arbitrary \spadtype{OrderedMonoid}
(not necessarily commutative too).
```

```
Thus these polynomials are (finite) linear combinations of words.

This constructor takes two arguments.
The first one is a \spadtype{Ring}
and the second is an \spadtype{OrderedMonoid}.
The abbreviation for \spadtype{XPolynomialRing} is \spadtype{XPR}.

Other constructors like \spadtype{XPolynomial},
\spadtype{XRecursivePolynomial}
\spadtype{XDistributedPolynomial},
\spadtype{LiePolynomial} and
\spadtype{XPBWPolynomial}
implement multivariate polynomials
in non-commutative variables.

We illustrate now some of the facilities of the \spadtype{XPR} domain
constructor.

\xtc{
Define the free ordered monoid generated by the symbols.
}{
\spadpaste{Word := OrderedFreeMonoid(Symbol) \bound{Word}}
}

\xtc{
Define the linear combinations of these words with integer coefficients.
}{
\spadpaste{poly:= XPR(Integer,Word) \free{Word} \bound{poly}}
}


\xtc{
Then we define a first element from {\bf poly}.
}{
\spadpaste{p:poly := 2 * x - 3 * y + 1 \free{poly} \bound{p}}
}

\xtc{
And a second one.
}{
\spadpaste{q:poly := 2 * x + 1 \free{poly} \bound{q}}
}


\xtc{
We compute their sum,
}{
\spadpaste{p + q  \free{p}\free{q} }
}
```

```
\xtc{
their product,
}{
\spadpaste{p * q  \free{p}\free{q} }
}

\xtc{
and see that variables do not commute.
}{
\spadpaste{(p +q)^2 -p^2 -q^2 - 2*p*q  \free{p}\free{q} }
}



\xtc{
Now we define a ring of square matrices,
}{
\spadpaste{M := SquareMatrix(2,Fraction Integer)  \bound{M}}
}

\xtc{
and the linear combinations of words with these  matrices as coefficients.
}{
\spadpaste{poly1:= XPR(M,Word) \free{Word} \free{M} \bound{poly1}}
}


\xtc{
Define a first matrix,
}{
\spadpaste{m1:M := matrix [[i*j**2 for i in 1..2] for j in 1..2]
\free{M} \bound{m1}}
}

\xtc{
a second one,
}{
\spadpaste{m2:M := m1 - 5/4 \free{M} \free{m1} \bound{m2}}
}

\xtc{
and a third one.
}{
\spadpaste{m3: M := m2**2 \free{M} \free{m2} \bound{m3}}
}

\xtc{
Define a polynomial,
}{
\spadpaste{pm:poly1   := m1*x + m2*y + m3*z - 2/3 \free{poly1}
```

```
\free{m1} \free{m2} \free{m3} \bound{pm}}
}


\xtc{
a second one,
}{
\spadpaste{qm:poly1 := pm - m1*x \free{m1} \free{pm} \bound{qm}}
}

\xtc{
and the following power.
}{
\spadpaste{qm**3 \bound{qm}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{XPolynomialRingXmpPagePatch1}
\begin{paste}{XPolynomialRingXmpPageFull1}{XPolynomialRingXmpPageEmpty1}
\pastebutton{XPolynomialRingXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{Word := OrderedFreeMonoid(Symbol)\bound{Word }}
\indentrel{3}\begin{verbatim}
   (1)  OrderedFreeMonoid Symbol
                                          Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPolynomialRingXmpPageEmpty1}
\begin{paste}{XPolynomialRingXmpPageEmpty1}{XPolynomialRingXmpPagePatch1}
\pastebutton{XPolynomialRingXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{Word := OrderedFreeMonoid(Symbol)\bound{Word }}
\end{paste}\end{patch}

\begin{patch}{XPolynomialRingXmpPagePatch2}
\begin{paste}{XPolynomialRingXmpPageFull2}{XPolynomialRingXmpPageEmpty2}
\pastebutton{XPolynomialRingXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{poly:= XPR(Integer,Word)\free{Word }\bound{poly }}
\indentrel{3}\begin{verbatim}
   (2)
   XPolynomialRing(Integer,OrderedFreeMonoid Symbol)
                                          Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPolynomialRingXmpPageEmpty2}
\begin{paste}{XPolynomialRingXmpPageEmpty2}{XPolynomialRingXmpPagePatch2}
\pastebutton{XPolynomialRingXmpPageEmpty2}{\showpaste}
```

```
\tab{5}\spadcommand{poly:= XPR(Integer,Word)\free{Word }\bound{poly }}
\end{paste}\end{patch}

\begin{patch}{XPolynomialRingXmpPagePatch3}
\begin{paste}{XPolynomialRingXmpPageFull3}{XPolynomialRingXmpPageEmpty3}
\pastebutton{XPolynomialRingXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{p:poly := 2 * x - 3 * y + 1\free{poly }\bound{p }}
\indentrel{3}\begin{verbatim}
   (3)  1 + 2x - 3y
Type: XPolynomialRing(Integer,OrderedFreeMonoid Symbol)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPolynomialRingXmpPageEmpty3}
\begin{paste}{XPolynomialRingXmpPageEmpty3}{XPolynomialRingXmpPagePatch3}
\pastebutton{XPolynomialRingXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{p:poly := 2 * x - 3 * y + 1\free{poly }\bound{p }}
\end{paste}\end{patch}

\begin{patch}{XPolynomialRingXmpPagePatch4}
\begin{paste}{XPolynomialRingXmpPageFull4}{XPolynomialRingXmpPageEmpty4}
\pastebutton{XPolynomialRingXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{q:poly := 2 * x + 1\free{poly }\bound{q }}
\indentrel{3}\begin{verbatim}
   (4)  1 + 2x
Type: XPolynomialRing(Integer,OrderedFreeMonoid Symbol)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPolynomialRingXmpPageEmpty4}
\begin{paste}{XPolynomialRingXmpPageEmpty4}{XPolynomialRingXmpPagePatch4}
\pastebutton{XPolynomialRingXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{q:poly := 2 * x + 1\free{poly }\bound{q }}
\end{paste}\end{patch}

\begin{patch}{XPolynomialRingXmpPagePatch5}
\begin{paste}{XPolynomialRingXmpPageFull5}{XPolynomialRingXmpPageEmpty5}
\pastebutton{XPolynomialRingXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{p + q\free{p }\free{q }}
\indentrel{3}\begin{verbatim}
   (5)  2 + 4x - 3y
Type: XPolynomialRing(Integer,OrderedFreeMonoid Symbol)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPolynomialRingXmpPageEmpty5}
\begin{paste}{XPolynomialRingXmpPageEmpty5}{XPolynomialRingXmpPagePatch5}
\pastebutton{XPolynomialRingXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{p + q\free{p }\free{q }}
\end{paste}\end{patch}
```

```
\begin{patch}{XPolynomialRingXmpPagePatch6}
\begin{paste}{XPolynomialRingXmpPageFull6}{XPolynomialRingXmpPageEmpty6}
\pastebutton{XPolynomialRingXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{p * q\free{p }\free{q }}
\indentrel{3}\begin{verbatim}
                         2
   (6)  1 + 4x - 3y + 4x  - 6y x
Type: XPolynomialRing(Integer,OrderedFreeMonoid Symbol)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPolynomialRingXmpPageEmpty6}
\begin{paste}{XPolynomialRingXmpPageEmpty6}{XPolynomialRingXmpPagePatch6}
\pastebutton{XPolynomialRingXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{p * q\free{p }\free{q }}
\end{paste}\end{patch}

\begin{patch}{XPolynomialRingXmpPagePatch7}
\begin{paste}{XPolynomialRingXmpPageFull7}{XPolynomialRingXmpPageEmpty7}
\pastebutton{XPolynomialRingXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{(p +q)^2 -p^2 -q^2 - 2*p*q\free{p }\free{q }}
\indentrel{3}\begin{verbatim}
   (7)  - 6x y + 6y x
Type: XPolynomialRing(Integer,OrderedFreeMonoid Symbol)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPolynomialRingXmpPageEmpty7}
\begin{paste}{XPolynomialRingXmpPageEmpty7}{XPolynomialRingXmpPagePatch7}
\pastebutton{XPolynomialRingXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{(p +q)^2 -p^2 -q^2 - 2*p*q\free{p }\free{q }}
\end{paste}\end{patch}

\begin{patch}{XPolynomialRingXmpPagePatch8}
\begin{paste}{XPolynomialRingXmpPageFull8}{XPolynomialRingXmpPageEmpty8}
\pastebutton{XPolynomialRingXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{M := SquareMatrix(2,Fraction Integer)\bound{M }}
\indentrel{3}\begin{verbatim}
   (8)  SquareMatrix(2,Fraction Integer)
                                           Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPolynomialRingXmpPageEmpty8}
\begin{paste}{XPolynomialRingXmpPageEmpty8}{XPolynomialRingXmpPagePatch8}
\pastebutton{XPolynomialRingXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{M := SquareMatrix(2,Fraction Integer)\bound{M }}
\end{paste}\end{patch}
```

```
\begin{patch}{XPolynomialRingXmpPagePatch9}
\begin{paste}{XPolynomialRingXmpPageFull9}{XPolynomialRingXmpPageEmpty9}
\pastebutton{XPolynomialRingXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{poly1:= XPR(M,Word)\free{Word }\free{M }\bound{poly1 }}
\indentrel{3}\begin{verbatim}
   (9)
  XPolynomialRing(SquareMatrix(2,Fraction Integer),Ordere
  dFreeMonoid Symbol)
                                            Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPolynomialRingXmpPageEmpty9}
\begin{paste}{XPolynomialRingXmpPageEmpty9}{XPolynomialRingXmpPagePatch9}
\pastebutton{XPolynomialRingXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{poly1:= XPR(M,Word)\free{Word }\free{M }\bound{poly1 }}
\end{paste}\end{patch}

\begin{patch}{XPolynomialRingXmpPagePatch10}
\begin{paste}{XPolynomialRingXmpPageFull10}{XPolynomialRingXmpPageEmpty10}
\pastebutton{XPolynomialRingXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{m1:M := matrix [[i*j**2 for i in 1..2] for j in 1..2]\free{M }\bound{m1 }}
\indentrel{3}\begin{verbatim}
         1  2
   (10)
         4  8
                 Type: SquareMatrix(2,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPolynomialRingXmpPageEmpty10}
\begin{paste}{XPolynomialRingXmpPageEmpty10}{XPolynomialRingXmpPagePatch10}
\pastebutton{XPolynomialRingXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{m1:M := matrix [[i*j**2 for i in 1..2] for j in 1..2]\free{M }\bound{m1 }}
\end{paste}\end{patch}

\begin{patch}{XPolynomialRingXmpPagePatch11}
\begin{paste}{XPolynomialRingXmpPageFull11}{XPolynomialRingXmpPageEmpty11}
\pastebutton{XPolynomialRingXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{m2:M := m1 - 5/4\free{M }\free{m1 }\bound{m2 }}
\indentrel{3}\begin{verbatim}
           1
         -   2
           4
   (11)
              27
          4
               4
                 Type: SquareMatrix(2,Fraction Integer)
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPolynomialRingXmpPageEmpty11}
\begin{paste}{XPolynomialRingXmpPageEmpty11}{XPolynomialRingXmpPagePatch11}
\pastebutton{XPolynomialRingXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{m2:M := m1 - 5/4\free{M }\free{m1 }\bound{m2 }}
\end{paste}\end{patch}

\begin{patch}{XPolynomialRingXmpPagePatch12}
\begin{paste}{XPolynomialRingXmpPageFull12}{XPolynomialRingXmpPageEmpty12}
\pastebutton{XPolynomialRingXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{m3: M := m2**2\free{M }\free{m2 }\bound{m3 }}
\indentrel{3}\begin{verbatim}
         129
           13
          16
   (12)
              857
         26
               16
                 Type: SquareMatrix(2,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPolynomialRingXmpPageEmpty12}
\begin{paste}{XPolynomialRingXmpPageEmpty12}{XPolynomialRingXmpPagePatch12}
\pastebutton{XPolynomialRingXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{m3: M := m2**2\free{M }\free{m2 }\bound{m3 }}
\end{paste}\end{patch}

\begin{patch}{XPolynomialRingXmpPagePatch13}
\begin{paste}{XPolynomialRingXmpPageFull13}{XPolynomialRingXmpPageEmpty13}
\pastebutton{XPolynomialRingXmpPageFull13}{\hidepaste}
\tab{5}\spadcommand{pm:poly1 := m1*x + m2*y + m3*z - 2/3\free{poly1 }\free{m1 }\free{m2 }\free{m3 }\bound{pr
\indentrel{3}\begin{verbatim}
           2                   1        129
         -    0              -   2       13
           3        1  2      4         16
   (13)           +     x +        y +         z
              2   4  8         27         857
          0   -               4       26
              3                   4          16
Type: XPolynomialRing(SquareMatrix(2,Fraction Integer),OrderedFreeMonoid Symbol)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPolynomialRingXmpPageEmpty13}
\begin{paste}{XPolynomialRingXmpPageEmpty13}{XPolynomialRingXmpPagePatch13}
\pastebutton{XPolynomialRingXmpPageEmpty13}{\showpaste}
\tab{5}\spadcommand{pm:poly1 := m1*x + m2*y + m3*z - 2/3\free{poly1 }\free{m1 }\free{m2 }\free{m3 }\bound{pr
```

```
\end{paste}\end{patch}

\begin{patch}{XPolynomialRingXmpPagePatch14}
\begin{paste}{XPolynomialRingXmpPageFull14}{XPolynomialRingXmpPageEmpty14}
\pastebutton{XPolynomialRingXmpPageFull14}{\hidepaste}
\tab{5}\spadcommand{qm:poly1 := pm - m1*x\free{m1 }\free{pm }\bound{qm }}
\indentrel{3}\begin{verbatim}
           2          1        129
         -    0    -   2       13
           3          4         16
   (14)          +        y +         z
                2        27         857
          0   -     4       26
                3         4          16
Type: XPolynomialRing(SquareMatrix(2,Fraction Integer),OrderedFreeMonoid Symbol)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPolynomialRingXmpPageEmpty14}
\begin{paste}{XPolynomialRingXmpPageEmpty14}{XPolynomialRingXmpPagePatch14}
\pastebutton{XPolynomialRingXmpPageEmpty14}{\showpaste}
\tab{5}\spadcommand{qm:poly1 := pm - m1*x\free{m1 }\free{pm }\bound{qm }}
\end{paste}\end{patch}

\begin{patch}{XPolynomialRingXmpPagePatch15}
\begin{paste}{XPolynomialRingXmpPageFull15}{XPolynomialRingXmpPageEmpty15}
\pastebutton{XPolynomialRingXmpPageFull15}{\hidepaste}
\tab{5}\spadcommand{qm**3\bound{qm }}
\indentrel{3}\begin{verbatim}
   (15)
        8           1  8    43   52
     -    0     -
       27           3  3     4    3
                +      y +         z
              8   16        104  857
      0    -       9
             27    3         3    12
   +
       129             3199     831
     -   - 26      -    -
        8        2      32       4
                 y  +                y z
              857       831     26467
     - 52   -      -    -
               8         2        32
   +
       3199     831         103169     6409
     -    -        -    -
        32       4            128        4   2
                    z y +                   z
```

```
       831      26467          6409      820977
     -    -          -     -
        2         32              2         128
   +
     3199    831        103169    6409

      64      8    3      256       8    2
                 y   +                 y z
     831    26467       6409    820977

      4       64          4       256
   +
     103169    6409          3178239    795341

       256       8              1024       128      2
                    y z y +                     y z
      6409    820977        795341    25447787

        4       256            64        1024
   +
     103169    6409         3178239    795341

       256       8      2      1024       128
                    z y   +                    z y z
      6409    820977       795341    25447787

        4       256           64        1024
   +
     3178239    795341        98625409   12326223

       1024       128    2       4096       256    3
                       z y +                       z
     795341    25447787       12326223   788893897

       64        1024            128        4096
Type: XPolynomialRing(SquareMatrix(2,Fraction Integer),OrderedFreeMonoid Symbol)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{XPolynomialRingXmpPageEmpty15}
\begin{paste}{XPolynomialRingXmpPageEmpty15}{XPolynomialRingXmpPagePatch15}
\pastebutton{XPolynomialRingXmpPageEmpty15}{\showpaste}
\tab{5}\spadcommand{qm**3\bound{qm }}
\end{paste}\end{patch}
```

3.121 zdsolve.ht

ZeroDimensionalSolvePackage

— zdsolve.ht —

```
\begin{page}{ZeroDimSolvePkgXmpPage}{ZeroDimensionalSolvePackage}
\beginscroll
The \spadtype{ZeroDimensionalSolvePackage} package constructor
provides operations for computing symbolically the complex or real
roots of zero-dimensional algebraic systems.

The package provides {\bf no} multiplicity information (i.e. some
returned roots may be double or higher) but only distinct roots are
returned.

Complex roots are given by means of univariate representations of
irreducible regular chains.  These representations are computed by the
\axiomOpFrom{univariateSolve}{ZeroDimensionalSolvePackage} operation
(by calling the
\spadtype{InternalRationalUnivariateRepresentationPackage} package
constructor which does the job).

Real roots are given by means of tuples of coordinates lying in the
\spadtype{RealClosure} of the coefficient ring.  They are computed by
the \axiomOpFrom{realSolve}{ZeroDimensionalSolvePackage} and
\axiomOpFrom{positiveSolve}{ZeroDimensionalSolvePackage} operations.
The former computes all the solutions of the input system with real
coordinates whereas the later concentrate on the solutions with
(strictly) positive coordinates.  In both cases, the computations are
performed by the \spadtype{RealClosure} constructor.

Both computations of complex roots and real roots rely on triangular
decompositions.  These decompositions can be computed in two different
ways.  First, by a applying the
\axiomOpFrom{zeroSetSplit}{RegularTriangularSet} operation from the
\spadtype{REGSET} domain constructor.  In that case, no Groebner bases
are computed.  This strategy is used by default.  Secondly, by
applying the \axiomOpFrom{zeroSetSplit}{LexTriangularPackage} from
\spadtype{LEXTRIPK}.  To use this later strategy with the operations
\axiomOpFrom{univariateSolve}{ZeroDimensionalSolvePackage},
\axiomOpFrom{realSolve}{ZeroDimensionalSolvePackage} and
\axiomOpFrom{positiveSolve}{ZeroDimensionalSolvePackage} one just
needs to use an extra boolean argument.

Note that the way of understanding triangular decompositions
is detailed in the example of the \spadtype{RegularTriangularSet}
constructor.
```

```
The \spadtype{ZeroDimensionalSolvePackage} constructor takes three
arguments.  The first one {\bf R} is the coefficient ring; it must
belong to the categories \spadtype{OrderedRing},
\spadtype{EuclideanDomain}, \spadtype{CharacteristicZero} and
\spadtype{RealConstant}.  This means essentially that {\bf R} is
\spadtype{Integer} or \spadtype{Fraction(Integer)}.  The second
argument {\bf ls} is the list of variables involved in the systems to
solve.  The third one MUST BE {\bf concat(ls,s)} where {\bf s} is an
additional symbol used for the univariate representations.  The
abbreviation for \spadtype{ZeroDimensionalSolvePackage} is
\spadtype{ZDSOLVE}.

We illustrate now how to use the constructor \spadtype{ZDSOLVE}
by two examples: the {\em Arnborg and Lazard} system and the
{\em L-3} system (Aubry and Moreno Maza).
Note that the use of this package is also demonstrated in the example
of the \spadtype{LexTriangularPackage} constructor.

\xtc{
Define the coefficient ring.
}{
\spadpaste{R := Integer \bound{R}}
}

\xtc{
Define the lists of variables:
}{
\spadpaste{ls : List Symbol := [x,y,z,t] \bound{ls}}
}

\xtc{
and:
}{
\spadpaste{ls2 : List Symbol := [x,y,z,t,new()$Symbol] \bound{ls2}}
}

\xtc{
Call the package:
}{
\spadpaste{pack := ZDSOLVE(R,ls,ls2) \free{ls} \free{ls2}
\free{R} \bound{pack}}
}

\xtc{
Define a polynomial system (Arnborg-Lazard)
}{
\spadpaste{p1 := x**2*y*z + x*y**2*z + x*y*z**2 + x*y*z +
x*y + x*z + y*z \bound{p1}}
}
```

```
\xtc{
}{
\spadpaste{p2 := x**2*y**2*z + x*y**2*z**2 + x**2*y*z +
x*y*z + y*z + x + z \bound{p2}}
}
\xtc{
}{
\spadpaste{p3 := x**2*y**2*z**2 + x**2*y**2*z + x*y**2*z +
x*y*z + x*z + z + 1 \bound{p3}}
}
\xtc{
}{
\spadpaste{lp := [p1, p2, p3] \free{p1} \free{p2} \free{p3} \bound{lp}}
}
Note that these polynomials do not involve the variable {\bf t};
we will use it in the second example.

\xtc{
First compute a decomposition into regular chains
(i.e. regular triangular sets).
}{
\spadpaste{triangSolve(lp)$pack \free{lp} \free{pack}}
}

We can see easily from this decomposition (consisting of a single
regular chain) that the input system has 20 complex roots.

\xtc{
Then we compute a univariate representation of this regular chain.
}{
\spadpaste{univariateSolve(lp)$pack \free{lp} \free{pack}}
}

We see that the zeros of our regular chain are split into three
components.  This is due to the use of univariate polynomial
factorization.

Each of these components consist of two parts.  The first one is an
irreducible univariate polynomial {\bf p(?)} which defines a simple
algebraic extension of the field of fractions of {\bf R}.  The second
one consists of multivariate polynomials {\bf pol1(x,\%A)}, {\bf
pol2(y,\%A)} and {\bf pol3(z,\%A)}.  Each of these polynomials involve
two variables: one is an indeterminate {\bf x}, {\bf y} or {\bf z} of
the input system {\bf lp} and the other is {\bf \%A} which represents
any root of {\bf p(?)}.  Recall that this {\bf \%A} is the last
element of the third parameter of \spadtype{ZDSOLVE}.  Thus any
complex root {\bf ?} of {\bf p(?)}  leads to a solution of the input
system {\bf lp} by replacing {\bf \%A} by this {\bf ?} in
{\bf pol1(x,\%A)}, {\bf pol2(y,\%A)} and {\bf pol3(z,\%A)}.  Note that
the polynomials {\bf pol1(x,\%A)}, {\bf pol2(y,\%A)} and {\bf pol3(z,\%A)}
```

```
have degree one w.r.t. {\bf x}, {\bf y} or {\bf z} respectively.  This
is always the case for all univariate representations.  Hence the
operation {\bf univariateSolve} replaces a system of multivariate
polynomials by a list of univariate polynomials, what justifies its
name.  Another example of univariate representations illustrates the
\spadtype{LexTriangularPackage} package constructor.

\xtc{
We now compute the solutions with real coordinates:
}{
\spadpaste{lr := realSolve(lp)$pack \free{lp} \free{pack} \bound{lr}}
}

\xtc{
The number of real solutions for the input system is:
}{
\spadpaste{\# lr \free{lr}}
}

Each of these real solutions is given by a list of elements in
\spadtype{RealClosure(R)}.  In these 8 lists, the first element is a
value of {\bf z}, the second of {\bf y} and the last of {\bf x}.  This
is logical since by setting the list of variables of the package to
{\bf [x,y,z,t]} we mean that the elimination ordering on the variables
is {\bf t < z < y < x }.  Note that each system treated by the
\spadtype{ZDSOLVE} package constructor needs only to be
zero-dimensional w.r.t. the variables involved in the system it-self
and not necessarily w.r.t. all the variables used to define the
package.

\xtc{
We can approximate these real numbers as follows.
This computation takes between 30 sec. and 5 min, depending on your
machine.
}{
\spadpaste{[[approximate(r,1/1000000) for r in point] for point in lr]
\free{lr}}
}

\xtc{
We can also concentrate on the solutions with real (strictly)
positive coordinates:
}{
\spadpaste{lpr := positiveSolve(lp)$pack \free{lp} \free{pack} \bound{lpr}}
}

Thus we have checked that the input system has no solution with
strictly positive coordinates.
```

```
\xtc{
Let us define another polynomial system ({\em L-3}).
}{
\spadpaste{f0 := x**3 + y + z + t- 1 \bound{f0}}
}
\xtc{
}{
\spadpaste{f1 := x + y**3 + z + t -1 \bound{f1}}
}
\xtc{
}{
\spadpaste{f2 := x + y + z**3 + t-1 \bound{f2}}
}
\xtc{
}{
\spadpaste{f3 := x + y + z + t**3 -1 \bound{f3}}
}
\xtc{
}{
\spadpaste{lf := [f0, f1, f2, f3] \free{f0} \free{f1} \free{f2}
\free{f3} \bound{lf}}
}


\xtc{
First compute a decomposition into regular chains
(i.e. regular triangular sets).
}{
\spadpaste{lts := triangSolve(lf)$pack \free{lf} \free{pack} \bound{lts}}
}


\xtc{
Then we compute a univariate representation.
}{
\spadpaste{univariateSolve(lf)$pack \free{lf} \free{pack}}
}

Note that this computation is made from the input system {\bf lf}.
\xtc{
However it is possible to reuse a pre-computed regular chain as follows:
}{
\spadpaste{ts := lts.1 \free{lts} \bound{ts}}
}
\xtc{
}{
\spadpaste{univariateSolve(ts)$pack \free{ts} \free{pack}}
}
\xtc{
}{
```

```
\spadpaste{realSolve(ts)$pack  \free{ts} \free{pack}}
}

\xtc{
We compute now the full set of points with real coordinates:
}{
\spadpaste{lr2 := realSolve(lf)$pack \free{lf} \free{pack}  \bound{lr2}}
}


\xtc{
The number of real solutions for the input system is:
}{
\spadpaste{\#lr2 \free{lr2}}
}
Another example of computation of real solutions illustrates
the \spadtype{LexTriangularPackage} package constructor.

\xtc{
We concentrate now on the solutions with real (strictly) positive
coordinates:
}{
\spadpaste{lpr2 := positiveSolve(lf)$pack \free{lf} \free{pack}
\bound{lpr2}}
}

\xtc{
Finally, we approximate the coordinates of this point with 20 exact
digits:
}{
\spadpaste{[approximate(r,1/10**21)::Float for r in lpr2.1] \free{lpr2}}
}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ZeroDimSolvePkgXmpPagePatch1}
\begin{paste}{ZeroDimSolvePkgXmpPageFull1}{ZeroDimSolvePkgXmpPageEmpty1}
\pastebutton{ZeroDimSolvePkgXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{R := Integer\bound{R }}
\indentrel{3}\begin{verbatim}
   (1)  Integer
                                          Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPageEmpty1}
\begin{paste}{ZeroDimSolvePkgXmpPageEmpty1}{ZeroDimSolvePkgXmpPagePatch1}
\pastebutton{ZeroDimSolvePkgXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{R := Integer\bound{R }}
```

```
\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPagePatch2}
\begin{paste}{ZeroDimSolvePkgXmpPageFull2}{ZeroDimSolvePkgXmpPageEmpty2}
\pastebutton{ZeroDimSolvePkgXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{ls : List Symbol := [x,y,z,t]\bound{ls }}
\indentrel{3}\begin{verbatim}
   (2)  [x,y,z,t]
                                      Type: List Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPageEmpty2}
\begin{paste}{ZeroDimSolvePkgXmpPageEmpty2}{ZeroDimSolvePkgXmpPagePatch2}
\pastebutton{ZeroDimSolvePkgXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{ls : List Symbol := [x,y,z,t]\bound{ls }}
\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPagePatch3}
\begin{paste}{ZeroDimSolvePkgXmpPageFull3}{ZeroDimSolvePkgXmpPageEmpty3}
\pastebutton{ZeroDimSolvePkgXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{ls2 : List Symbol := [x,y,z,t,new()$Symbol]\bound{ls2 }}
\indentrel{3}\begin{verbatim}
   (3)  [x,y,z,t,%A]
                                      Type: List Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPageEmpty3}
\begin{paste}{ZeroDimSolvePkgXmpPageEmpty3}{ZeroDimSolvePkgXmpPagePatch3}
\pastebutton{ZeroDimSolvePkgXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{ls2 : List Symbol := [x,y,z,t,new()$Symbol]\bound{ls2 }}
\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPagePatch4}
\begin{paste}{ZeroDimSolvePkgXmpPageFull4}{ZeroDimSolvePkgXmpPageEmpty4}
\pastebutton{ZeroDimSolvePkgXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{pack := ZDSOLVE(R,ls,ls2)\free{ls }\free{ls2 }\free{R }\bound{pack }}
\indentrel{3}\begin{verbatim}
   (4)
  ZeroDimensionalSolvePackage(Integer,[x,y,z,t],[x,y,z,t,
  %A])
                                           Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPageEmpty4}
\begin{paste}{ZeroDimSolvePkgXmpPageEmpty4}{ZeroDimSolvePkgXmpPagePatch4}
\pastebutton{ZeroDimSolvePkgXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{pack := ZDSOLVE(R,ls,ls2)\free{ls }\free{ls2 }\free{R }\bound{pack }}
```

```
\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPagePatch5}
\begin{paste}{ZeroDimSolvePkgXmpPageFull5}{ZeroDimSolvePkgXmpPageEmpty5}
\pastebutton{ZeroDimSolvePkgXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{p1 := x**2*y*z + x*y**2*z + x*y*z**2 + x*y*z + x*y + x*z + y*z\bound{p1 }}
\indentrel{3}\begin{verbatim}
             2       2     2
   (5)  x y z  + (x y  + (x  + x + 1)y + x)z + x y
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPageEmpty5}
\begin{paste}{ZeroDimSolvePkgXmpPageEmpty5}{ZeroDimSolvePkgXmpPagePatch5}
\pastebutton{ZeroDimSolvePkgXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{p1 := x**2*y*z + x*y**2*z + x*y*z**2 + x*y*z + x*y + x*z + y*z\bound{p1 }}
\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPagePatch6}
\begin{paste}{ZeroDimSolvePkgXmpPageFull6}{ZeroDimSolvePkgXmpPageEmpty6}
\pastebutton{ZeroDimSolvePkgXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{p2 := x**2*y**2*z + x*y**2*z**2 + x**2*y*z + x*y*z + y*z + x + z\bound{p2 }}
\indentrel{3}\begin{verbatim}
           2 2     2 2     2
   (6)  x y z  + (x y  + (x  + x + 1)y + 1)z + x
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPageEmpty6}
\begin{paste}{ZeroDimSolvePkgXmpPageEmpty6}{ZeroDimSolvePkgXmpPagePatch6}
\pastebutton{ZeroDimSolvePkgXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{p2 := x**2*y**2*z + x*y**2*z**2 + x**2*y*z + x*y*z + y*z + x + z\bound{p2 }}
\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPagePatch7}
\begin{paste}{ZeroDimSolvePkgXmpPageFull7}{ZeroDimSolvePkgXmpPageEmpty7}
\pastebutton{ZeroDimSolvePkgXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{p3 := x**2*y**2*z**2 + x**2*y**2*z + x*y**2*z + x*y*z + x*z + z + 1\bound{p3 }}
\indentrel{3}\begin{verbatim}
         2 2 2      2      2
   (7)  x y z  + ((x  + x)y  + x y + x + 1)z + 1
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPageEmpty7}
\begin{paste}{ZeroDimSolvePkgXmpPageEmpty7}{ZeroDimSolvePkgXmpPagePatch7}
\pastebutton{ZeroDimSolvePkgXmpPageEmpty7}{\showpaste}
```

```
\tab{5}\spadcommand{p3 := x**2*y**2*z**2 + x**2*y**2*z + x*y**2*z + x*y*z + x*z + z + 1\bound{p3
\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPagePatch8}
\begin{paste}{ZeroDimSolvePkgXmpPageFull8}{ZeroDimSolvePkgXmpPageEmpty8}
\pastebutton{ZeroDimSolvePkgXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{lp := [p1, p2, p3]\free{p1 }\free{p2 }\free{p3 }\bound{lp }}
\indentrel{3}\begin{verbatim}
   (8)
         2       2     2
   [x y z  + (x y  + (x  + x + 1)y + x)z + x y,
       2 2     2 2     2
    x y z  + (x y  + (x  + x + 1)y + 1)z + x,
     2 2 2      2      2
    x y z  + ((x  + x)y  + x y + x + 1)z + 1]
                          Type: List Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPageEmpty8}
\begin{paste}{ZeroDimSolvePkgXmpPageEmpty8}{ZeroDimSolvePkgXmpPagePatch8}
\pastebutton{ZeroDimSolvePkgXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{lp := [p1, p2, p3]\free{p1 }\free{p2 }\free{p3 }\bound{lp }}
\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPagePatch9}
\begin{paste}{ZeroDimSolvePkgXmpPageFull9}{ZeroDimSolvePkgXmpPageEmpty9}
\pastebutton{ZeroDimSolvePkgXmpPageFull9}{\hidepaste}
\tab{5}\spadcommand{triangSolve(lp)$pack\free{lp }\free{pack }}
\indentrel{3}\begin{verbatim}
   (9)
   [
     {
         20     19      18      17       16      15
        z   - 6z   - 41z   + 71z   + 106z   + 92z
      +
           14       13       12       11       10
        197z   + 145z   + 257z   + 278z   + 201z
      +
           9       8       7       6      5       4
        278z  + 257z  + 145z  + 197z  + 92z  + 106z
      +
          3      2
        71z  - 41z  - 6z + 1
      ,

                    19             18              17
            14745844z   + 50357474z   - 130948857z
          +
                     16              15              14
```

```
        - 185261586z   - 180077775z   - 338007307z
      +
                    13             12             11
        - 275379623z   - 453190404z   - 474597456z
      +
                    10             9             8
        - 366147695z   - 481433567z  - 430613166z
      +
                    7             6             5
        - 261878358z  - 326073537z  - 163008796z
      +
                    4             3            2
        - 177213227z  - 104356755z  + 65241699z
      +
        9237732z - 1567348
   *
      y
  +
            19           18            17
    1917314z   + 6508991z   - 16973165z
  +
               16            15            14
    - 24000259z   - 23349192z   - 43786426z
  +
               13            12            11
    - 35696474z   - 58724172z   - 61480792z
  +
               10            9            8
    - 47452440z   - 62378085z  - 55776527z
  +
               7            6            5
    - 33940618z  - 42233406z  - 21122875z
  +
               4            3           2
    - 22958177z  - 13504569z  + 8448317z  + 1195888z
  +
    - 202934
  ,

          3       2       3    2               2
        (z  - 2z)y  + (- z  - z  - 2z - 1)y - z  - z
      +
        1
   *
      x
  +
     2
    z  - 1
  }
]
```

```
             Type: List RegularChain(Integer,[x,y,z,t])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPageEmpty9}
\begin{paste}{ZeroDimSolvePkgXmpPageEmpty9}{ZeroDimSolvePkgXmpPagePatch9}
\pastebutton{ZeroDimSolvePkgXmpPageEmpty9}{\showpaste}
\tab{5}\spadcommand{triangSolve(lp)$pack\free{lp }\free{pack }}
\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPagePatch10}
\begin{paste}{ZeroDimSolvePkgXmpPageFull10}{ZeroDimSolvePkgXmpPageEmpty10}
\pastebutton{ZeroDimSolvePkgXmpPageFull10}{\hidepaste}
\tab{5}\spadcommand{univariateSolve(lp)$pack\free{lp }\free{pack }}
\indentrel{3}\begin{verbatim}
   (10)
   [
     [
       complexRoots =
            12      11      10     9     8      7      6
           ?   - 12?   + 24?   + 4?  - 9?  + 27?  - 21?
         +
              5     4     3      2
           27?  - 9?  + 4?  + 24?  - 12? + 1
       ,

       coordinates =
         [
                     11        10         9        8
             63x + 62%A   - 721%A   + 1220%A  + 705%A
           +
                   7         6        5         4       3
             - 285%A  + 1512%A  - 735%A  + 1401%A  - 21%A
           +
                  2
             215%A  + 1577%A - 142
           ,

                     11        10         9        8
             63y - 75%A   + 890%A   - 1682%A  - 516%A
           +
                  7         6         5         4       3
             588%A  - 1953%A  + 1323%A  - 1815%A  + 426%A
           +
                    2
             - 243%A  - 1801%A + 679
           ,
          z - %A]
       ]
     ,
```

```
                 6    5    4    3    2
     [complexRoots= ?  + ?  + ?  + ?  + ?  + ? + 1,
                           5        3
      coordinates= [x - %A ,y - %A ,z - %A]]
     ,

                 2
     [complexRoots= ?  + 5? + 1,
      coordinates= [x - 1,y - 1,z - %A]]
     ]
Type: List Record(complexRoots: SparseUnivariatePolynomial Integer,coordinates: List Polynomial Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPageEmpty10}
\begin{paste}{ZeroDimSolvePkgXmpPageEmpty10}{ZeroDimSolvePkgXmpPagePatch10}
\pastebutton{ZeroDimSolvePkgXmpPageEmpty10}{\showpaste}
\tab{5}\spadcommand{univariateSolve(lp)$pack\free{lp }\free{pack }}
\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPagePatch11}
\begin{paste}{ZeroDimSolvePkgXmpPageFull11}{ZeroDimSolvePkgXmpPageEmpty11}
\pastebutton{ZeroDimSolvePkgXmpPageFull11}{\hidepaste}
\tab{5}\spadcommand{lr := realSolve(lp)$pack\free{lp }\free{pack }\bound{lr }}
\indentrel{3}\begin{verbatim}
   (11)
   [
     [%R1,

        1184459    19   2335702    18   5460230    17
         %R1   -  %R1   -  %R1
        1645371          548457          182819
      +
        79900378    16   43953929    15   13420192    14
         %R1   +  %R1   +  %R1
         1645371          548457           182819
      +
        553986    13   193381378    12   35978916    11
         %R1   +  %R1   +  %R1
         3731           1645371           182819
      +
        358660781    10   271667666    9   118784873    8
         %R1   +  %R1  +  %R1
         1645371           1645371           548457
      +
        337505020    7   1389370    6   688291    5
         %R1  +  %R1  +  %R1
         1645371          11193          4459
      +
```

```
     3378002    4   140671876    3   32325724    2
      %R1  +  %R1  +  %R1
      42189          1645371          548457
   +
        8270       9741532
     -  %R1 -
         343       1645371
   ,

         91729     19   487915     18   4114333     17
     -  %R1   +  %R1   +  %R1
        705159          705159          705159
   +
        1276987     16   13243117     15   16292173     14
     -  %R1   -  %R1   -  %R1
         235053          705159           705159
   +
        26536060     13   722714     12   5382578     11
     -  %R1   -  %R1   -  %R1
         705159           18081          100737
   +
        15449995     10   14279770     9   6603890     8
     -  %R1   -  %R1  -  %R1
         235053          235053          100737
   +
        409930     7   37340389    6   34893715    5
     -  %R1  -  %R1  -  %R1
         6027          705159          705159
   +
        26686318    4   801511    3   17206178    2
     -  %R1  -  %R1  -  %R1
         705159          26117          705159
   +
        4406102       377534
     -  %R1 +
         705159       705159
   ]
 ,

 [%R2,

     1184459    19   2335702    18   5460230    17
      %R2   -  %R2   -  %R2
     1645371          548457          182819
   +
     79900378    16   43953929    15   13420192    14
      %R2   +  %R2   +  %R2
      1645371          548457           182819
   +
     553986    13   193381378    12   35978916    11
```

```
   %R2    +  %R2    +  %R2
   3731            1645371            182819
 +
  358660781    10   271667666    9   118784873    8
   %R2    +  %R2   +  %R2
   1645371            1645371            548457
 +
  337505020    7   1389370    6   688291    5
   %R2   +  %R2   +  %R2
   1645371           11193           4459
 +
  3378002    4   140671876    3   32325724    2
   %R2   +  %R2   +  %R2
   42189           1645371           548457
 +
    8270        9741532
  -  %R2 -
    343        1645371
 ,

    91729    19   487915    18   4114333    17
  -  %R2    +  %R2    +  %R2
   705159          705159           705159
 +
   1276987    16   13243117    15   16292173    14
  -  %R2    -  %R2    -  %R2
    235053          705159            705159
 +
   26536060    13   722714    12   5382578    11
  -  %R2    -  %R2    -  %R2
    705159            18081           100737
 +
   15449995    10   14279770    9   6603890    8
  -  %R2    -  %R2   -  %R2
    235053            235053           100737
 +
   409930    7   37340389    6   34893715    5
  -  %R2  -  %R2  -  %R2
    6027           705159           705159
 +
   26686318    4   801511    3   17206178    2
  -  %R2  -  %R2  -  %R2
    705159          26117          705159
 +
   4406102       377534
  -  %R2 +
    705159       705159
 ]
,
```

```
[%R3,

   1184459    19   2335702    18   5460230    17
    %R3   -  %R3   -  %R3
   1645371          548457          182819
 +
   79900378    16   43953929    15   13420192    14
    %R3   +  %R3   +  %R3
    1645371          548457           182819
 +
   553986    13   193381378    12   35978916    11
    %R3   +  %R3   +  %R3
    3731           1645371           182819
 +
   358660781    10   271667666    9   118784873    8
    %R3   +  %R3  +  %R3
    1645371           1645371           548457
 +
   337505020    7   1389370    6   688291    5
    %R3  +  %R3  +  %R3
    1645371          11193          4459
 +
   3378002    4   140671876    3   32325724    2
    %R3  +  %R3  +  %R3
    42189          1645371          548457
 +
     8270       9741532
   -  %R3 -
      343       1645371
 ,

      91729    19   487915    18   4114333    17
   -  %R3   +  %R3   +  %R3
     705159         705159          705159
 +
     1276987    16   13243117    15   16292173    14
   -  %R3   -  %R3   -  %R3
      235053          705159           705159
 +
     26536060    13   722714    12   5382578    11
   -  %R3   -  %R3   -  %R3
      705159           18081          100737
 +
     15449995    10   14279770    9   6603890    8
   -  %R3   -  %R3  -  %R3
      235053           235053          100737
 +
     409930    7   37340389    6   34893715    5
   -  %R3  -  %R3  -  %R3
      6027          705159          705159
```

```
     +
         26686318     4   801511     3   17206178     2
       - %R3  -  %R3  -  %R3
          705159          26117          705159
     +
         4406102       377534
       - %R3 +
          705159       705159
     ]
   ,

   [%R4,

       1184459    19   2335702    18   5460230    17
        %R4   -  %R4   -  %R4
       1645371          548457           182819
     +
       79900378    16   43953929    15   13420192    14
        %R4   +  %R4   +  %R4
        1645371          548457            182819
     +
       553986    13   193381378    12   35978916    11
        %R4   +  %R4   +  %R4
        3731           1645371           182819
     +
       358660781    10   271667666    9   118784873    8
        %R4   +  %R4  +  %R4
        1645371           1645371           548457
     +
       337505020    7   1389370    6   688291    5
        %R4  +  %R4  +  %R4
        1645371          11193          4459
     +
       3378002    4   140671876    3   32325724    2
        %R4  +  %R4  +  %R4
        42189          1645371          548457
     +
         8270       9741532
       - %R4 -
          343       1645371
     ,

          91729    19   487915    18   4114333    17
       - %R4   +  %R4   +  %R4
         705159          705159          705159
     +
         1276987    16   13243117    15   16292173    14
       - %R4   -  %R4   -  %R4
          235053          705159            705159
     +
```

```
      26536060    13   722714    12   5382578    11
    - %R4   -  %R4   -  %R4
       705159          18081          100737
  +
      15449995    10   14279770    9   6603890    8
    - %R4   -  %R4  -  %R4
       235053          235053          100737
  +
      409930    7   37340389    6   34893715    5
    - %R4  -  %R4  -  %R4
       6027          705159          705159
  +
      26686318    4   801511    3   17206178    2
    - %R4  -  %R4  -  %R4
       705159          26117         705159
  +
      4406102       377534
    - %R4 +
       705159       705159
  ]
,

[%R5,

    1184459    19   2335702    18   5460230    17
     %R5   -  %R5   -  %R5
    1645371          548457          182819
  +
    79900378    16   43953929    15   13420192    14
     %R5   +  %R5   +  %R5
     1645371          548457           182819
  +
    553986    13   193381378    12   35978916    11
     %R5   +  %R5   +  %R5
     3731           1645371           182819
  +
    358660781    10   271667666    9   118784873    8
     %R5   +  %R5  +  %R5
     1645371          1645371           548457
  +
    337505020    7   1389370    6   688291    5
     %R5  +  %R5  +  %R5
     1645371          11193          4459
  +
    3378002    4   140671876    3   32325724    2
     %R5  +  %R5  +  %R5
     42189          1645371          548457
  +
      8270       9741532
    -  %R5 -
```

```
     343       1645371
 ,

     91729    19   487915    18   4114333    17
  -  %R5   +  %R5   +  %R5
    705159         705159          705159
 +
    1276987    16   13243117    15   16292173    14
  -  %R5   -  %R5   -  %R5
     235053          705159          705159
 +
    26536060    13   722714    12   5382578    11
  -  %R5   -  %R5   -  %R5
     705159          18081          100737
 +
    15449995    10   14279770    9   6603890    8
  -  %R5   -  %R5  -  %R5
     235053          235053          100737
 +
    409930    7   37340389    6   34893715    5
  -  %R5  -  %R5  -  %R5
     6027          705159          705159
 +
    26686318    4   801511    3   17206178    2
  -  %R5  -  %R5  -  %R5
     705159          26117         705159
 +
    4406102       377534
  -  %R5 +
     705159       705159
 ]
,

[%R6,

   1184459    19   2335702    18   5460230    17
    %R6   -  %R6   -  %R6
   1645371          548457          182819
 +
   79900378    16   43953929    15   13420192    14
    %R6   +  %R6   +  %R6
    1645371          548457          182819
 +
   553986    13   193381378    12   35978916    11
    %R6   +  %R6   +  %R6
    3731            1645371          182819
 +
   358660781    10   271667666    9   118784873    8
    %R6   +  %R6  +  %R6
    1645371          1645371          548457
```

```
  +
    337505020    7   1389370    6   688291    5
     %R6   +  %R6   +  %R6
     1645371          11193          4459
  +
    3378002    4   140671876    3   32325724    2
     %R6   +  %R6   +  %R6
     42189          1645371          548457
  +
      8270       9741532
    -  %R6  -
       343       1645371
  ,

       91729     19   487915     18   4114333    17
    -  %R6    +  %R6    +  %R6
      705159         705159          705159
  +
      1276987    16   13243117    15   16292173    14
    -  %R6    -  %R6    -  %R6
       235053          705159           705159
  +
      26536060    13   722714    12   5382578    11
    -  %R6    -  %R6    -  %R6
       705159          18081          100737
  +
      15449995    10   14279770    9   6603890    8
    -  %R6    -  %R6   -  %R6
       235053          235053          100737
  +
      409930    7   37340389    6   34893715    5
    -  %R6   -  %R6   -  %R6
       6027          705159          705159
  +
      26686318    4   801511    3   17206178    2
    -  %R6   -  %R6   -  %R6
       705159         26117         705159
  +
      4406102       377534
    -  %R6  +
       705159       705159
  ]
,

[%R7,

    1184459    19   2335702    18   5460230    17
     %R7    -  %R7    -  %R7
    1645371         548457          182819
  +
```

```
  79900378    16   43953929    15   13420192    14
   %R7    +  %R7    +  %R7
   1645371           548457           182819
+
  553986    13   193381378    12   35978916    11
   %R7    +  %R7    +  %R7
   3731            1645371            182819
+
  358660781    10   271667666    9   118784873    8
   %R7    +  %R7   +  %R7
   1645371             1645371            548457
+
  337505020    7   1389370    6   688291    5
   %R7   +  %R7   +  %R7
   1645371           11193           4459
+
  3378002    4   140671876    3   32325724    2
   %R7   +  %R7   +  %R7
   42189           1645371           548457
+
    8270        9741532
  -  %R7 -
     343        1645371
,

     91729    19   487915    18   4114333    17
  -  %R7    +  %R7    +  %R7
    705159          705159           705159
+
    1276987    16   13243117    15   16292173    14
  -  %R7    -  %R7    -  %R7
     235053           705159            705159
+
    26536060    13   722714    12   5382578    11
  -  %R7    -  %R7    -  %R7
     705159            18081           100737
+
    15449995    10   14279770    9   6603890    8
  -  %R7    -  %R7   -  %R7
     235053           235053            100737
+
    409930    7   37340389    6   34893715    5
  -  %R7   -  %R7   -  %R7
     6027            705159           705159
+
    26686318    4   801511    3   17206178    2
  -  %R7   -  %R7   -  %R7
     705159           26117           705159
+
    4406102        377534
```

```
    -  %R7 +
       705159       705159
  ]
 ,

 [%R8,

    1184459    19   2335702    18   5460230    17
     %R8   -  %R8   -  %R8
    1645371          548457          182819
  +
    79900378    16   43953929    15   13420192    14
     %R8   +  %R8   +  %R8
     1645371          548457           182819
  +
    553986    13   193381378    12   35978916    11
     %R8   +  %R8   +  %R8
     3731           1645371           182819
  +
    358660781    10   271667666    9   118784873    8
     %R8   +  %R8  +  %R8
     1645371           1645371           548457
  +
    337505020    7   1389370    6   688291    5
     %R8  +  %R8  +  %R8
     1645371          11193          4459
  +
    3378002    4   140671876    3   32325724    2
     %R8  +  %R8  +  %R8
     42189          1645371          548457
  +
      8270       9741532
    -  %R8 -
       343       1645371
  ,

       91729     19   487915    18   4114333    17
    -  %R8   +  %R8   +  %R8
      705159          705159          705159
  +
      1276987    16   13243117    15   16292173    14
    -  %R8   -  %R8   -  %R8
       235053          705159           705159
  +
      26536060    13   722714    12   5382578    11
    -  %R8   -  %R8   -  %R8
       705159           18081          100737
  +
      15449995    10   14279770    9   6603890    8
    -  %R8   -  %R8  -  %R8
```

```
          235053           235053          100737
      +
         409930    7   37340389    6   34893715    5
        -  %R8  -  %R8  -  %R8
          6027          705159          705159
      +
         26686318    4   801511    3   17206178    2
        -  %R8  -  %R8  -  %R8
          705159          26117         705159
      +
         4406102       377534
        -  %R8 +
          705159       705159
      ]
     ]
           Type: List List RealClosure Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPageEmpty11}
\begin{paste}{ZeroDimSolvePkgXmpPageEmpty11}{ZeroDimSolvePkgXmpPagePatch11}
\pastebutton{ZeroDimSolvePkgXmpPageEmpty11}{\showpaste}
\tab{5}\spadcommand{lr := realSolve(lp)$pack\free{lp }\free{pack }\bound{lr }}
\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPagePatch12}
\begin{paste}{ZeroDimSolvePkgXmpPageFull12}{ZeroDimSolvePkgXmpPageEmpty12}
\pastebutton{ZeroDimSolvePkgXmpPageFull12}{\hidepaste}
\tab{5}\spadcommand{\# lr\free{lr }}
\indentrel{3}\begin{verbatim}
   (12)  8
                                 Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPageEmpty12}
\begin{paste}{ZeroDimSolvePkgXmpPageEmpty12}{ZeroDimSolvePkgXmpPagePatch12}
\pastebutton{ZeroDimSolvePkgXmpPageEmpty12}{\showpaste}
\tab{5}\spadcommand{\# lr\free{lr }}
\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPagePatch13}
\begin{paste}{ZeroDimSolvePkgXmpPageFull13}{ZeroDimSolvePkgXmpPageEmpty13}
\pastebutton{ZeroDimSolvePkgXmpPageFull13}{\hidepaste}
\tab{5}\spadcommand{[[approximate(r,1/1000000) for r in point] for point in lr]\free{lr }}
\indentrel{3}\begin{verbatim}
   (13)
   [
       10048059
     [- ,
```

```
      2097152

     450305731698538794352439791383896641459673197621_
      176821933588120838551631405892456717609142362296_
      957774030998333607610488982289165781370943098838_
      597331137202584846939132376157019506760357601116_
      591745498681538209878909485152342039281129312661_
      413298565469771454646614954878259199411884470441_
      722440491921567263542158028061437758844364634441_
      004525302478656192316328821417 5
   /
     450305728302524548851651180698582663508310069375_
      732046528055470686564494957750991686720188943880_
      904083548179317185938627976245515189835707930448_
      774424291488708829840324189200301436123314860220_
      082144373379075531124363291986489542170422894995_
      712900161194988079570236638654430693920271489779_
      688266712323356043491523434068924275280417338557_
      48173811892770661433123966812 16
    ,
```

```
      2097152

     450305731698538794352439791383896641459673197621_
      17682193358812083855163140589245671760914236296_
      95777403099833360761048898228916578137094309838_
      59733113720258484693913237615701950676035760116_
      59174549868153820987890948515234203928112931261_
      41329856546977145464661495487825919941188447041_
      72244049192156726354215802806143775884436463441_
      0045253024786561923163288214175
   /
     450305728302524548851651180698582663508310069375_
      73204652805547068656449495775099168672018894380_
      90408354817931718593862797624551518983570793048_
      77442429148870882984032418920030143612331486020_
      08214437337907553112436329198648954217042289495_
      71290016119498807957023663865443069392027148979_
      68826671232335604349152343406892427528041733857_
      481738118927706614331239668121 6
    ,

     210626076882347507389479868048601659624960714869_
      06855387636837150206396808586496507900558895056_
      46893309447097099937802187329095325898785247249_
      02071750498366048207515661873872451468533306001_
      12029646351663813515432559822002503052839810868_
      37110614842307026091211297929876896285681830479_
      05476005638076266490561846205530604781619178201_
      158870378913898818 95
   /
     210626060949846419247211380481647417534196295329_
      64341024139031423687579676852738885855909759652_
      11778862189872881953943640246297357061959812326_
      10365979902512686325867656720234210687703171018_
      42474841814232889218376812370627084702957062184_
      85928867400771937828499200923760593314168901000_
      66637389634759811822855673103707202647449677622_
      83837629939232800768
    ]
  ,

     2563013
  [- ,
     2097152

    -
         261134617679192778969861769323775771923825996_
          306354178192275233044018989966807292833849 07_
          686235932074421259259867338159322435048092 94_
          837523030237337236806668167446173001727271 35_
```

```
      3311571242897
     /
     116522540050522253058398191600458914375722661_
      027685899000879013481991494092241375398397l3_
      940195234333204081399281531888294957554551 63_
      963417619308395977544797140231469234269034 92_
      1938055593984
   ,

   357259455027591722109658872961578827299851705467_
    560323957819814100603409173528282659062190230 44_
    669639419710389233045262733293163737574500619 78_
    989228611097699708725046623537 3
  /
   103954826934559893687707124483402605580081455112_
    017059220052236659175940965948644233914102945 29_
    502651799899601048118758225302053465051315812 43_
    901724728917386501470296630886 4
   ]
 ,

   1715967
 [- ,
   2097152

   -
      421309353378430352108483951797708239037726150_
       396958622482899843660603065607635937456481 37_
       734983766031212678225658014362069395199514 65_
       182225805246972874100225439524 91
     /
      944181414418537445864969203434922405243659747_
       096625366393064196079580588258549319984019 16_
       999176594432648246411351873835838881478673 40_
       193078576058203641958568223047 68
   ,

   763583334711264422251562542441083122534747566900_
    858933883416217250190499437634673087680904284 52_
    089199199253021057209714539189827313890725914 03_
    5
  /
   262418876408609719978429761047806663393423046789_
    585160227858097850378454920578849901964060226 69_
    660268915801035435676250390186298871412849167 56_
    48
   ]
 ,

    437701
```

```
[- ,
   2097152

   168310690863834958832217233265422591356298631318_
    195103145275016144149747345532815072136486835555_
    796467816035077771990750778352133664845336549l3_
    83623741304759
 /
   168310686809521338900170998270591363896307766873_
    122611116778518800490742522629868032588781096260_
    141402985973669842648879989083770687999984542330_
    81649008099328
  ,

   496155010983501018642268101342210873595871480100_
    376063970796809664691282670847283444311723917210_
    910424921345096631241l133
 /
   496154987275773831550919207821020902985289711861_
    109712623638404082937659261914313170254867464790_
    271836349216048244221542 4
  ]
,

  222801
[,
 2097152

  -
      899488488040242826510759512197069142713604569_
       254197827557300186521375992158813771669612630_
       491016552201951429949322991371832417058676720_
       383477
    /
      116788999866502637217776510069188858270896996_
       022993476969083575245707777941643520947376780_
       665077694058889427645877185424342556259924560_
       372224
  ,

  -
      238970488813315687832080154437380839561277150_
       920849101984745299188550954651952546783901660_
       135939996938866400362835705523211550378712910_
       458703265
    /
      535548727364509632609040328668993190598822544_
       468541143322159383368119295756283367146865420_
       903407469936562859255991176021204461834431450_
       479421952
```

```
  ]
,

  765693
[,
 2097152

   855896921981671626787324476117819808872469895861_
    66701402137657543220023032516857861186783308402_
    03328837654339523418704917749518340772512899000_
    391009630373148561
 /
   294144244553301079097642841137639349981558021594_
    58569179064525354957230138568189417023302287798_
    90141296236721138154231997238917322156711965244_
    4639331719460159488
  ,

  -
      205761823058257210124765032486024256111130258_
       15435888088439236627675493822416593627122907_
       77612800192921420574408948085193743688582762_
       2246433251878894899015
     /
      267159820332573553809795235350145022057631375_
       98908350970917225206427101987719026671839489_
       06289863714759678360292483949204616471537777_
       775324180661095366656
  ]
,

 5743879
[,
 2097152

   107628881696890684795554639477357020817145672494_
    26186140236631235747689608504342639713980725465_
    92772662158833449797698617455397887562900072984_
    76800060834355318980169340872720504761255988923_
    27575638305286889535354218094827710589175426028_
    90060941949620874083007858366669453501766248414_
    88732463225
 /
   313176895708031794664846194002355204419037661345_
    85849862285496319161966016162197817656155325322_
    94746529648276430583810894079374566460757823146_
    88858119555602920851521883888320031865840746939_
    94260632605898286123092315966691297079864813198_
    51571942927230340622934023923486703042068153044_
    0845099008
```

```
    ,

    -
        211328669918575091836412047556545843787017248_
         98654859943898281353352644446652845575264927_
         34931691731407872701432935503473348172076098_
         72054584900878007756416053431789468836611952_
         97399805029441626685500981279619504962102219_
         42878089359674925850594427768502251789758706_
         752831632503615
      /
        162761558493798758024290662434710458088914446_
         61684597180431538394083725255333098080703636_
         99585502216011211087103263609551026027769414_
         08739114812622116813978168258743807532259146_
         61319399754572005223498385689642856344480185_
         62038272378787354460106106141518010935617205_
         1706396253618176
    ]
  ,

   19739877
  [,
    2097152

    -
        299724993683270330379901580486152094921504038_
         75007071777012857667201925305794224789535660_
         24359860143101547801638082771611160372212874_
         84777803580987284314922548423836585801362934_
         17053217025823333509180096017899370239859353_
         04900460493389873837030853410347089908880814_
         85398113201846458245880061539477074169948729_
         58759602107502158919488144768548710315309312_
         95467332190133702671098200902282300510751860_
         71859284570302778073977965258138627622392869_
         96106809728023675
      /
        230843327485227859072891008119181102390650414_
         13214326461239367948739333192706089607021381_
         93417647898360620229519176632937631786851455_
         01476602720625902225250555174182368889688380_
         66366025744317604722402920931967294751602472_
         68834121141893318848728661844434927287285112_
         89708076755286489505658586403317856591038706_
         50061128015164035227410373609905560544769495_
         27059227070809593049491257519554708879259595_
         52929920110858560812556635485429471554031675_
         979542656381353984
    ,
```

```
      -
          512818926354822848909627639786894008060093841_
           06630804594079663358450092641094905204598253_
           16250084723010047035024497436523038925818959_
           28931293158470135392762143543439867426304729_
           39091228501338519906964902315660943719943337_
           95070782624011727587749989296611277318372294_
           62420711653791043655457414608288470130554391_
           26204193548854107359401577758966028223645758_
           64611831512943973974715166920465061850603762_
           87516256195847052412587282839139194642913955
        /
          228828193977843933053120879318129047118363109_
           24553689903863908242435094636442362497730806_
           47438987739144921607794682653851741189091711_
           74186814511497833728419182249767586835872948_
           66447308566225526872092037244118004814057028_
           37198310642291275676195774614443815996713502_
           62939174978359004147086012775237299648862774_
           26724876224800632688088893248918508424949343_
           47337603075939980268208482904859678177751444_
           65749979827872616963053217673201717237252096
      ]
     ]
                       Type: List List Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPageEmpty13}
\begin{paste}{ZeroDimSolvePkgXmpPageEmpty13}{ZeroDimSolvePkgXmpPagePatch13}
\pastebutton{ZeroDimSolvePkgXmpPageEmpty13}{\showpaste}
\tab{5}\spadcommand{[[approximate(r,1/1000000) for r in point] for point in lr]\free{lr }}
\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPagePatch14}
\begin{paste}{ZeroDimSolvePkgXmpPageFull14}{ZeroDimSolvePkgXmpPageEmpty14}
\pastebutton{ZeroDimSolvePkgXmpPageFull14}{\hidepaste}
\tab{5}\spadcommand{lpr := positiveSolve(lp)$pack\free{lp }\free{pack }\bound{lpr }}
\indentrel{3}\begin{verbatim}
   (14)  []
           Type: List List RealClosure Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPageEmpty14}
\begin{paste}{ZeroDimSolvePkgXmpPageEmpty14}{ZeroDimSolvePkgXmpPagePatch14}
\pastebutton{ZeroDimSolvePkgXmpPageEmpty14}{\showpaste}
\tab{5}\spadcommand{lpr := positiveSolve(lp)$pack\free{lp }\free{pack }\bound{lpr }}
\end{paste}\end{patch}
```

```
\begin{patch}{ZeroDimSolvePkgXmpPagePatch15}
\begin{paste}{ZeroDimSolvePkgXmpPageFull15}{ZeroDimSolvePkgXmpPageEmpty15}
\pastebutton{ZeroDimSolvePkgXmpPageFull15}{\hidepaste}
\tab{5}\spadcommand{f0 := x**3 + y + z + t- 1\bound{f0 }}
\indentrel{3}\begin{verbatim}
                 3
   (15)  z + y + x  + t - 1
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPageEmpty15}
\begin{paste}{ZeroDimSolvePkgXmpPageEmpty15}{ZeroDimSolvePkgXmpPagePatch15}
\pastebutton{ZeroDimSolvePkgXmpPageEmpty15}{\showpaste}
\tab{5}\spadcommand{f0 := x**3 + y + z + t- 1\bound{f0 }}
\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPagePatch16}
\begin{paste}{ZeroDimSolvePkgXmpPageFull16}{ZeroDimSolvePkgXmpPageEmpty16}
\pastebutton{ZeroDimSolvePkgXmpPageFull16}{\hidepaste}
\tab{5}\spadcommand{f1 := x + y**3 + z + t -1\bound{f1 }}
\indentrel{3}\begin{verbatim}
             3
   (16)  z + y  + x + t - 1
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPageEmpty16}
\begin{paste}{ZeroDimSolvePkgXmpPageEmpty16}{ZeroDimSolvePkgXmpPagePatch16}
\pastebutton{ZeroDimSolvePkgXmpPageEmpty16}{\showpaste}
\tab{5}\spadcommand{f1 := x + y**3 + z + t -1\bound{f1 }}
\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPagePatch17}
\begin{paste}{ZeroDimSolvePkgXmpPageFull17}{ZeroDimSolvePkgXmpPageEmpty17}
\pastebutton{ZeroDimSolvePkgXmpPageFull17}{\hidepaste}
\tab{5}\spadcommand{f2 := x + y + z**3 + t-1\bound{f2 }}
\indentrel{3}\begin{verbatim}
         3
   (17)  z  + y + x + t - 1
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPageEmpty17}
\begin{paste}{ZeroDimSolvePkgXmpPageEmpty17}{ZeroDimSolvePkgXmpPagePatch17}
\pastebutton{ZeroDimSolvePkgXmpPageEmpty17}{\showpaste}
\tab{5}\spadcommand{f2 := x + y + z**3 + t-1\bound{f2 }}
```

```
\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPagePatch18}
\begin{paste}{ZeroDimSolvePkgXmpPageFull18}{ZeroDimSolvePkgXmpPageEmpty18}
\pastebutton{ZeroDimSolvePkgXmpPageFull18}{\hidepaste}
\tab{5}\spadcommand{f3 := x + y + z + t**3 -1\bound{f3 }}
\indentrel{3}\begin{verbatim}
                      3
   (18)  z + y + x + t  - 1
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPageEmpty18}
\begin{paste}{ZeroDimSolvePkgXmpPageEmpty18}{ZeroDimSolvePkgXmpPagePatch18}
\pastebutton{ZeroDimSolvePkgXmpPageEmpty18}{\showpaste}
\tab{5}\spadcommand{f3 := x + y + z + t**3 -1\bound{f3 }}
\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPagePatch19}
\begin{paste}{ZeroDimSolvePkgXmpPageFull19}{ZeroDimSolvePkgXmpPageEmpty19}
\pastebutton{ZeroDimSolvePkgXmpPageFull19}{\hidepaste}
\tab{5}\spadcommand{lf := [f0, f1, f2, f3]\free{f0 }\free{f1 }\free{f2 }\free{f3 }\bound{lf }}
\indentrel{3}\begin{verbatim}
   (19)
             3                3
   [z + y + x  + t - 1, z + y  + x + t - 1,
     3                                3
    z  + y + x + t - 1, z + y + x + t  - 1]
                          Type: List Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPageEmpty19}
\begin{paste}{ZeroDimSolvePkgXmpPageEmpty19}{ZeroDimSolvePkgXmpPagePatch19}
\pastebutton{ZeroDimSolvePkgXmpPageEmpty19}{\showpaste}
\tab{5}\spadcommand{lf := [f0, f1, f2, f3]\free{f0 }\free{f1 }\free{f2 }\free{f3 }\bound{lf }}
\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPagePatch20}
\begin{paste}{ZeroDimSolvePkgXmpPageFull20}{ZeroDimSolvePkgXmpPageEmpty20}
\pastebutton{ZeroDimSolvePkgXmpPageFull20}{\hidepaste}
\tab{5}\spadcommand{lts := triangSolve(lf)$pack\free{lf }\free{pack }\bound{lts }}
\indentrel{3}\begin{verbatim}
   (20)
   [
      2           3        3
     {t  + t + 1, z  - z - t  + t,

                 3      2
```

```
      (3z + 3t  - 3)y
    +
         2      3           6     3            3      2
      (3z  + (6t  - 6)z + 3t  - 6t  + 3)y + (3t  - 3)z
    +
         6     3          9     6     3
      (3t  - 6t  + 3)z + t  - 3t  + 5t  - 3t
    ,
   x + y + z}
  ,

   16     13     10     7      4      2
  {t   - 6t   + 9t   + 4t  + 15t  - 54t  + 27,

                  15            14             13
          4907232t   + 40893984t   - 115013088t
        +
                   12            11             10
          22805712t   + 36330336t   + 162959040t
        +
                      9            8            7
          - 159859440t  - 156802608t  + 117168768t
        +
                    6            5            4
          126282384t  - 129351600t  + 306646992t
        +
                    3             2
          475302816t  - 1006837776t  - 237269088t
        +
          480716208
     *
        z
    +
        54       51        48      46         45
      48t   - 912t   + 8232t   - 72t   - 46848t
    +
          43          42         40          39
      1152t   + 186324t   - 3780t   - 543144t
    +
            38         37           36         35
      - 3168t   - 21384t   + 1175251t   + 41184t
    +
            34           33          32            31
      278003t   - 1843242t   - 301815t   - 1440726t
    +
             30           29           28          27
      1912012t   + 1442826t   + 4696262t   - 922481t
    +
               26            25          24
      - 4816188t   - 10583524t   - 208751t
```

```
      +
                23             22            21
        11472138t   + 16762859t   - 857663t
      +
                  20             19            18
        - 19328175t   - 18270421t   + 4914903t
      +
                17             16            15
        22483044t   + 12926517t   - 8605511t
      +
                  14            13            12
        - 17455518t   - 5014597t   + 8108814t
      +
               11           10            9            8
        8465535t   + 190542t   - 4305624t  - 2226123t
      +
               7            6           5           4
        661905t  + 1169775t  + 226260t  - 209952t
      +
                3
        - 141183t  + 27216t
      ,

               3      2
       (3z + 3t  - 3)y
      +
          2      3            6     3            3      2
       (3z  + (6t  - 6)z + 3t  - 6t  + 3)y + (3t  - 3)z
      +
          6     3           9     6     3
       (3t  - 6t  + 3)z + t  - 3t  + 5t  - 3t
      ,
                    3
     x + y + z + t  - 1}
    ,
             2                        2
   {t,z - 1,y  - 1,x + y}, {t - 1,z,y  - 1,x + y},
           2
   {t - 1,z  - 1,z y + 1,x},

     16     13     10     7      4      2
    {t   - 6t   + 9t   + 4t  + 15t  - 54t  + 27,

                  29             28              27
          4907232t   + 40893984t   - 115013088t
        +
                    26              25              24
          - 1730448t   - 168139584t   + 738024480t
        +
                      23              22               21
```

```
      - 195372288t    + 315849456t    - 2567279232t
    +
                 20              19              18
      937147968t    + 1026357696t    + 4780488240t
    +
                    17              16
      - 2893767696t    - 5617160352t
    +
                    15              14
      - 3427651728t    + 5001100848t
    +
                  13              12              11
      8720098416t    + 2331732960t    - 499046544t
    +
                     10              9
      - 16243306272t    - 9748123200t
    +
                  8               7               6
      3927244320t  + 25257280896t  + 10348032096t
    +
                     5               4             3
      - 17128672128t  - 14755488768t  + 544086720t
    +
                   2
      10848188736t  + 1423614528t - 2884297248
   *
    z
 +
       68        65         62        60           59
   - 48t    + 1152t    - 13560t    + 360t    + 103656t
 +
         57           56          54            53
   - 7560t    - 572820t    + 71316t    + 2414556t
 +
       52          51            50          49
   2736t    - 402876t    - 7985131t    - 49248t
 +
          48            47            46            45
   1431133t    + 20977409t    + 521487t    - 2697635t
 +
             44           43           42
   - 43763654t    - 3756573t    - 2093410t
 +
           41            40            39
   71546495t    + 19699032t    + 35025028t
 +
            38            37             36
   - 89623786t    - 77798760t    - 138654191t
 +
           35             34             33
```

```
      87596128t   + 235642497t   + 349607642t
    +
                 32             31             30
      - 93299834t   - 551563167t   - 630995176t
    +
                29             28             27
      186818962t   + 995427468t   + 828416204t
    +
                  26              25              24
      - 393919231t   - 1076617485t   - 1609479791t
    +
                23              22              21
      595738126t   + 1198787136t   + 4342832069t
    +
                   20              19              18
      - 2075938757t   - 4390835799t   - 4822843033t
    +
                 17              16              15
      6932747678t   + 6172196808t   + 1141517740t
    +
                   14              13              12
      - 4981677585t   - 9819815280t   - 7404299976t
    +
                  11               10               9
      - 157295760t   + 29124027630t   + 14856038208t
    +
                    8               7              6
      - 16184101410t  - 26935440354t  - 3574164258t
    +
                  5              4             3
      10271338974t  + 11191425264t  + 6869861262t
    +
                   2
      - 9780477840t  - 3586674168t + 2884297248
    ,

           3      3      2      6      3          9
          3z  + (6t  - 6)z  + (6t  - 12t  + 3)z + 2t
        +
             6    3
          - 6t  + t  + 3t
     *
        y
    +
        3      3      6      3      2
      (3t  - 3)z  + (6t  - 12t  + 6)z
    +
        9      6      3          12     9     6     3
      (4t  - 12t  + 11t  - 3)z + t   - 4t  + 5t  - 2t
    ,
```

```
               3
 x + y + z + t  - 1}
,
      2
{t - 1,z  - 1,y,x + z},

  8    7    6     5     4     3      2
 {t  + t  + t  - 2t  - 2t  - 2t  + 19t  + 19t - 8,

                7           6           5
      2395770t  + 3934440t  - 3902067t
     +
                   4           3             2
      - 10084164t  - 1010448t  + 32386932t
     +
      22413225t - 10432368
   *
     z
  +
             7           6           5           4
   - 463519t  + 3586833t  + 9494955t  - 8539305t
  +
              3            2
   - 33283098t  + 35479377t  + 46263256t - 17419896
  ,

        4      3      3       6      3      2
      3z  + (9t  - 9)z  + (12t  - 24t  + 9)z
     +
              3                     6      4      3
      (- 152t  + 219t - 67)z - 41t  + 57t  + 25t
     +
      - 57t + 16
   *
     y
  +
      3      4      6      3      3
   (3t  - 3)z  + (9t  - 18t  + 9)z
  +
          3              2
   (- 181t  + 270t - 89)z
  +
         6       4      3                    7
   (- 92t  + 135t  + 49t  - 135t + 43)z + 27t
  +
        6      4       3
   - 27t  - 54t  + 396t  - 486t + 144
  ,
               3
 x + y + z + t  - 1}
```

```
     ,
             3
    {t,z - t  + 1,y - 1,x - 1}, {t - 1,z,y,x},
    {t,z - 1,y,x}, {t,z,y - 1,x}, {t,z,y,x - 1}]
              Type: List RegularChain(Integer,[x,y,z,t])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPageEmpty20}
\begin{paste}{ZeroDimSolvePkgXmpPageEmpty20}{ZeroDimSolvePkgXmpPagePatch20}
\pastebutton{ZeroDimSolvePkgXmpPageEmpty20}{\showpaste}
\tab{5}\spadcommand{lts := triangSolve(lf)$pack\free{lf }\free{pack }\bound{lts }}
\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPagePatch21}
\begin{paste}{ZeroDimSolvePkgXmpPageFull21}{ZeroDimSolvePkgXmpPageEmpty21}
\pastebutton{ZeroDimSolvePkgXmpPageFull21}{\hidepaste}
\tab{5}\spadcommand{univariateSolve(lf)$pack\free{lf }\free{pack }}
\indentrel{3}\begin{verbatim}
   (21)
   [
     [complexRoots= ?,
      coordinates= [x - 1,y - 1,z + 1,t - %A]]
     ,
    [complexRoots= ?,coordinates= [x,y - 1,z,t - %A]],
    [complexRoots= ? - 1,coordinates= [x,y,z,t - %A]],
    [complexRoots= ?,coordinates= [x - 1,y,z,t - %A]],
    [complexRoots= ?,coordinates= [x,y,z - 1,t - %A]],

     [complexRoots= ? - 2,
      coordinates= [x - 1,y + 1,z,t - 1]]
     ,
    [complexRoots= ?,coordinates= [x + 1,y - 1,z,t - 1]],

     [complexRoots= ? - 1,
      coordinates= [x - 1,y + 1,z - 1,t]]
     ,

     [complexRoots= ? + 1,
      coordinates= [x + 1,y - 1,z - 1,t]]
     ,

                     6      3      2
     [complexRoots= ?  - 2?  + 3?  - 3,

       coordinates =
                3                   3
         [2x + %A  + %A - 1, 2y + %A  + %A - 1, z - %A,
          t - %A]
       ]
```

```
,

                5     3     2
[complexRoots= ?  + 3?  - 2?  + 3? - 3,

  coordinates =
                           3
    [x - %A,y - %A,z + %A  + 2%A - 1,t - %A]
  ]
,

                4    3     2
[complexRoots= ?  - ?  - 2?  + 3,

  coordinates =
          3                  3
    [x + %A  - %A - 1, y + %A  - %A - 1,
          3
     z - %A  + 2%A + 1, t - %A]
  ]
,

[complexRoots= ? + 1,
 coordinates= [x - 1,y - 1,z,t - %A]]
,

                6     3     2
[complexRoots= ?  + 2?  + 3?  - 3,

  coordinates =
           3                          3
    [2x - %A  - %A - 1, y + %A, 2z - %A  - %A - 1,
     t + %A]
  ]
,

                6      4      3      2
[complexRoots= ?  + 12?  + 20?  - 45?  - 42? - 953,

  coordinates =
    [
                     5       4       3        2
        12609x + 23%A  + 49%A  - 46%A  + 362%A
      +
        - 5015%A - 8239
      ,

                     5       4       3        2
        25218y + 23%A  + 49%A  - 46%A  + 362%A
      +
```

```
       7594%A - 8239
     ,

                     5       4       3        2
       25218z + 23%A  + 49%A  - 46%A  + 362%A
     +
       7594%A - 8239
     ,

                     5       4       3        2
       12609t + 23%A  + 49%A  - 46%A  + 362%A
     +
       - 5015%A - 8239
     ]
 ]
,

                5      3      2
[complexRoots= ?  + 12?  - 16?  + 48? - 96,

  coordinates =
            3
    [8x + %A  + 8%A - 8,2y - %A,2z - %A,2t - %A]
  ]
,

                5    4     3     2
[complexRoots= ?  + ?  - 5?  - 3?  + 9? + 3,

  coordinates =
            3                    3
    [2x - %A  + 2%A - 1, 2y + %A  - 4%A + 1,
            3                    3
     2z - %A  + 2%A - 1, 2t - %A  + 2%A - 1]
  ]
,

                4     3     2
[complexRoots= ?  - 3?  + 4?  - 6? + 13,

  coordinates =
             3      2
    [9x - 2%A  + 4%A  - %A + 2,
            3      2
     9y + %A  - 2%A  + 5%A - 1,
            3      2
     9z + %A  - 2%A  + 5%A - 1,
            3      2
     9t + %A  - 2%A  - 4%A - 1]
  ]
```

```
 ,

                 4      2
 [complexRoots= ?  - 11?  + 37,

   coordinates =
             2              2                    2
     [3x - %A  + 7, 6y + %A  + 3%A - 7, 3z - %A  + 7,
             2
      6t + %A  - 3%A - 7]
   ]
 ,

 [complexRoots= ? + 1,
  coordinates= [x - 1,y,z - 1,t + 1]]
 ,

 [complexRoots= ? + 2,
  coordinates= [x,y - 1,z - 1,t + 1]]
 ,

 [complexRoots= ? - 2,
  coordinates= [x,y - 1,z + 1,t - 1]]
 ,
[complexRoots= ?,coordinates= [x,y + 1,z - 1,t - 1]],

 [complexRoots= ? - 2,
  coordinates= [x - 1,y,z + 1,t - 1]]
 ,
[complexRoots= ?,coordinates= [x + 1,y,z - 1,t - 1]],

                 4     3      2
 [complexRoots= ?  + 5?  + 16?  + 30? + 57,

   coordinates =
                 3       2
     [151x + 15%A  + 54%A  + 104%A + 93,
                 3       2
      151y - 10%A  - 36%A  - 19%A - 62,
                3       2
      151z - 5%A  - 18%A  - 85%A - 31,
                3       2
      151t - 5%A  - 18%A  - 85%A - 31]
   ]
 ,

                 4    3     2
 [complexRoots= ?  - ?  - 2?  + 3,

   coordinates =
```

```
              3                   3
        [x - %A  + 2%A + 1, y + %A  - %A - 1, z - %A,
              3
         t + %A  - %A - 1]
       ]
     ,

                     4     3     2
     [complexRoots= ?  + 2?  - 8?  + 48,

       coordinates =
               3
         [8x - %A  + 4%A - 8, 2y + %A,
               3                    3
          8z + %A  - 8%A + 8, 8t - %A  + 4%A - 8]
       ]
     ,

                     5    4     3     2
     [complexRoots= ?  + ?  - 2?  - 4?  + 5? + 8,

       coordinates =
               3             3             3
         [3x + %A  - 1,3y + %A  - 1,3z + %A  - 1,t - %A]
       ]
     ,

                     3
     [complexRoots= ?  + 3? - 1,
      coordinates= [x - %A,y - %A,z - %A,t - %A]]
     ]
Type: List Record(complexRoots: SparseUnivariatePolynomial Integer,coordinates: List Polynomial Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPageEmpty21}
\begin{paste}{ZeroDimSolvePkgXmpPageEmpty21}{ZeroDimSolvePkgXmpPagePatch21}
\pastebutton{ZeroDimSolvePkgXmpPageEmpty21}{\showpaste}
\tab{5}\spadcommand{univariateSolve(lf)$pack\free{lf }\free{pack }}
\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPagePatch22}
\begin{paste}{ZeroDimSolvePkgXmpPageFull22}{ZeroDimSolvePkgXmpPageEmpty22}
\pastebutton{ZeroDimSolvePkgXmpPageFull22}{\hidepaste}
\tab{5}\spadcommand{ts := lts.1\free{lts }\bound{ts }}
\indentrel{3}\begin{verbatim}
   (22)
     2          3       3
   {t  + t + 1, z  - z - t  + t,
```

```
              3      2
       (3z + 3t  - 3)y
     +
          2      3           6     3            3      2
       (3z  + (6t  - 6)z + 3t  - 6t  + 3)y + (3t  - 3)z
     +
          6     3          9     6     3
       (3t  - 6t  + 3)z + t  - 3t  + 5t  - 3t
     ,
    x + y + z}
                 Type: RegularChain(Integer,[x,y,z,t])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPageEmpty22}
\begin{paste}{ZeroDimSolvePkgXmpPageEmpty22}{ZeroDimSolvePkgXmpPagePatch22}
\pastebutton{ZeroDimSolvePkgXmpPageEmpty22}{\showpaste}
\tab{5}\spadcommand{ts := lts.1\free{lts }\bound{ts }}
\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPagePatch23}
\begin{paste}{ZeroDimSolvePkgXmpPageFull23}{ZeroDimSolvePkgXmpPageEmpty23}
\pastebutton{ZeroDimSolvePkgXmpPageFull23}{\hidepaste}
\tab{5}\spadcommand{univariateSolve(ts)$pack\free{ts }\free{pack }}
\indentrel{3}\begin{verbatim}
   (23)
   [
                    4     3      2
     [complexRoots= ?  + 5?  + 16?  + 30? + 57,

       coordinates =
                    3       2
         [151x + 15%A  + 54%A  + 104%A + 93,
                    3       2
          151y - 10%A  - 36%A  - 19%A - 62,
                   3       2
          151z - 5%A  - 18%A  - 85%A - 31,
                   3       2
          151t - 5%A  - 18%A  - 85%A - 31]
       ]
     ,

                    4    3     2
     [complexRoots= ?  - ?  - 2?  + 3,

       coordinates =
              3                  3
         [x - %A  + 2%A + 1, y + %A  - %A - 1, z - %A,
              3
          t + %A  - %A - 1]
```

```
      ]
     ,

                     4     3     2
     [complexRoots= ?  + 2?  - 8?  + 48,

       coordinates =
                3
         [8x - %A  + 4%A - 8, 2y + %A,
                3                    3
          8z + %A  - 8%A + 8, 8t - %A  + 4%A - 8]
       ]
     ]
Type: List Record(complexRoots: SparseUnivariatePolynomial Integer,coordinates: List Polynomial Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPageEmpty23}
\begin{paste}{ZeroDimSolvePkgXmpPageEmpty23}{ZeroDimSolvePkgXmpPagePatch23}
\pastebutton{ZeroDimSolvePkgXmpPageEmpty23}{\showpaste}
\tab{5}\spadcommand{univariateSolve(ts)$pack\free{ts }\free{pack }}
\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPagePatch24}
\begin{paste}{ZeroDimSolvePkgXmpPageFull24}{ZeroDimSolvePkgXmpPageEmpty24}
\pastebutton{ZeroDimSolvePkgXmpPageFull24}{\hidepaste}
\tab{5}\spadcommand{realSolve(ts)$pack\free{ts }\free{pack }}
\indentrel{3}\begin{verbatim}
   (24)  []
           Type: List List RealClosure Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPageEmpty24}
\begin{paste}{ZeroDimSolvePkgXmpPageEmpty24}{ZeroDimSolvePkgXmpPagePatch24}
\pastebutton{ZeroDimSolvePkgXmpPageEmpty24}{\showpaste}
\tab{5}\spadcommand{realSolve(ts)$pack\free{ts }\free{pack }}
\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPagePatch25}
\begin{paste}{ZeroDimSolvePkgXmpPageFull25}{ZeroDimSolvePkgXmpPageEmpty25}
\pastebutton{ZeroDimSolvePkgXmpPageFull25}{\hidepaste}
\tab{5}\spadcommand{lr2 := realSolve(lf)$pack\free{lf }\free{pack }\bound{lr2 }}
\indentrel{3}\begin{verbatim}
   (25)
   [[0,- 1,1,1], [0,0,1,0], [1,0,0,0], [0,0,0,1],
    [0,1,0,0], [1,0,%R37,- %R37], [1,0,%R38,- %R38],
    [0,1,%R35,- %R35], [0,1,%R36,- %R36], [- 1,0,1,1],

     [%R32,
```

```
   1     15     2     14     1     13     4     12
   %R32    +  %R32    +  %R32    -  %R32
  27           27           27           27
+
    11     11     4     10     1     9    14     8
  -  %R32    -  %R32    +  %R32   +  %R32
    27           27           27          27
+
   1     7    2     6    1     5    2     4       3
   %R32  +  %R32  +  %R32  +  %R32  + %R32
  27          9          3          9
+
  4     2
   %R32  - %R32 - 2
  3
,

     1     15     1     14     1     13     2     12
  -  %R32    -  %R32    -  %R32    +  %R32
    54           27           54           27
+
  11     11     2     10     1     9     7     8
   %R32    +  %R32    -  %R32   -  %R32
  54           27           54          27
+
     1     7    1     6    1     5    1     4       3
  -  %R32  -  %R32  -  %R32  -  %R32  - %R32
    54          9          6          9
+
    2     2    1        3
  -  %R32  +  %R32 +
    3          2        2
,

     1     15     1     14     1     13     2     12
  -  %R32    -  %R32    -  %R32    +  %R32
    54           27           54           27
+
  11     11     2     10     1     9     7     8
   %R32    +  %R32    -  %R32   -  %R32
  54           27           54          27
+
     1     7    1     6    1     5    1     4       3
  -  %R32  -  %R32  -  %R32  -  %R32  - %R32
    54          9          6          9
+
    2     2    1        3
  -  %R32  +  %R32 +
    3          2        2
```

```
  ]
 ,

 [%R33,

      1     15    2     14    1     13    4     12
     %R33    +  %R33    +  %R33    -  %R33
    27           27           27           27
   +
      11     11    4     10    1     9   14     8
    -  %R33    -  %R33    +  %R33  +  %R33
      27           27           27          27
   +
     1     7   2     6   1     5   2     4       3
     %R33  +  %R33  +  %R33  +  %R33  + %R33
    27         9          3          9
   +
    4     2
     %R33  - %R33 - 2
    3
   ,

       1     15    1     14    1     13    2     12
    -  %R33    -  %R33    -  %R33    +  %R33
      54           27           54           27
   +
    11     11    2     10    1     9    7     8
     %R33    +  %R33    -  %R33  -  %R33
    54           27           54          27
   +
       1     7   1     6   1     5   1     4       3
    -  %R33  -  %R33  -  %R33  -  %R33  - %R33
      54         9          6          9
   +
      2     2   1         3
    -  %R33  +  %R33 +
      3         2         2
   ,

       1     15    1     14    1     13    2     12
    -  %R33    -  %R33    -  %R33    +  %R33
      54           27           54           27
   +
    11     11    2     10    1     9    7     8
     %R33    +  %R33    -  %R33  -  %R33
    54           27           54          27
   +
       1     7   1     6   1     5   1     4       3
    -  %R33  -  %R33  -  %R33  -  %R33  - %R33
      54         9          6          9
```

```
     +
         2      2   1         3
       - %R33  + %R33 +
         3          2         2
     ]
   ,

   [%R34,

        1      15     2      14     1      13     4      12
        %R34   +  %R34   +  %R34   -  %R34
       27            27            27            27
     +
         11      11     4      10     1      9   14      8
       - %R34   -  %R34   +  %R34  +  %R34
         27            27            27           27
     +
        1      7   2      6   1      5   2      4        3
        %R34  +  %R34  +  %R34  +  %R34  + %R34
       27          9          3          9
     +
       4      2
        %R34  - %R34 - 2
       3
     ,

          1      15     1      14     1      13     2      12
       - %R34   -  %R34   -  %R34   +  %R34
         54            27            54            27
     +
       11      11     2      10     1      9     7      8
        %R34   +  %R34   -  %R34  -  %R34
       54            27            54           27
     +
          1      7   1      6   1      5   1      4        3
       - %R34  -  %R34  -  %R34  -  %R34  - %R34
         54          9          6          9
     +
         2      2   1         3
       - %R34  + %R34 +
         3          2         2
     ,

          1      15     1      14     1      13     2      12
       - %R34   -  %R34   -  %R34   +  %R34
         54            27            54            27
     +
       11      11     2      10     1      9     7      8
        %R34   +  %R34   -  %R34  -  %R34
       54            27            54           27
```

```
      +
           1      7    1      6    1      5    1      4         3
        -  %R34  -  %R34  -  %R34  -  %R34  - %R34
          54          9          6          9
      +
          2      2    1          3
        -  %R34  +  %R34 +
          3          2          2
      ]
     ,
    [- 1,1,0,1], [- 1,1,1,0],

     [%R23,

           1       15     1       14     1       13     2       12
        -  %R23    -  %R23    -  %R23    +  %R23
          54            27            54            27
      +
        11       11     2       10     1       9     7       8
         %R23    +  %R23    -  %R23  -  %R23
        54            27            54           27
      +
           1      7    1      6    1      5    1      4         3
        -  %R23  -  %R23  -  %R23  -  %R23  - %R23
          54          9          6          9
      +
          2      2    1          3
        -  %R23  +  %R23 +
          3          2          2
      ,
      %R30,

                   1       15     1       14     1       13
        - %R30 +  %R23    +  %R23    +  %R23
                 54            27            54
      +
           2       12    11       11     2       10     1       9
        -  %R23    -  %R23    -  %R23    +  %R23
          27             54            27            54
      +
         7       8     1       7    1       6    1       5    1       4
         %R23  +  %R23  +  %R23  +  %R23  +  %R23
        27           54           9           6           9
      +
        2      2    1          1
         %R23  -  %R23 -
        3          2          2
      ]
     ,
```

```
[%R23,

     1      15     1      14     1      13     2      12
   - %R23   -  %R23   -  %R23   +  %R23
    54            27            54            27
 +
   11      11     2      10     1      9     7      8
   %R23   +  %R23   -  %R23  -  %R23
   54            27            54           27
 +
     1      7    1      6    1      5    1      4        3
   -  %R23  -  %R23  -  %R23  -  %R23  - %R23
    54          9           6           9
 +
    2      2    1          3
   -  %R23  +  %R23 +
    3          2          2
 ,
 %R31,

             1      15     1      14     1      13
   - %R31 +  %R23   +  %R23   +  %R23
            54            27            54
 +
     2      12    11      11     2      10     1      9
   -  %R23   -  %R23   -  %R23   +  %R23
    27            54            27            54
 +
   7      8     1      7    1      6    1      5    1      4
   %R23  +  %R23  +  %R23  +  %R23  +  %R23
   27           54           9           6           9
 +
   2      2    1          1
   %R23  -  %R23 -
   3          2          2
 ]
,

[%R24,

     1      15     1      14     1      13     2      12
   -  %R24   -  %R24   -  %R24   +  %R24
    54            27            54            27
 +
   11      11     2      10     1      9     7      8
   %R24   +  %R24   -  %R24  -  %R24
   54            27            54           27
 +
     1      7    1      6    1      5    1      4        3
   -  %R24  -  %R24  -  %R24  -  %R24  - %R24
```

```
      54          9           6           9
  +
      2      2    1           3
    -  %R24  +  %R24 +
      3          2           2
  ,
 %R28,

                1       15     1       14     1       13
    - %R28 +  %R24    +  %R24    +  %R24
               54             27            54
  +
       2       12    11       11     2       10     1       9
    -  %R24    -  %R24    -  %R24    +  %R24
      27             54             27             54
  +
     7       8     1       7    1       6    1       5    1       4
     %R24  +  %R24  +  %R24  +  %R24  +  %R24
    27            54            9            6            9
  +
    2      2    1           1
     %R24  -  %R24 -
    3          2           2
  ]
 ,

 [%R24,

       1       15     1       14     1       13     2       12
    -  %R24    -  %R24    -  %R24    +  %R24
      54             27             54             27
  +
    11       11     2       10     1       9     7       8
     %R24    +  %R24    -  %R24   -  %R24
    54             27             54            27
  +
       1       7    1       6    1       5    1       4         3
    -  %R24  -  %R24  -  %R24  -  %R24  - %R24
      54           9            6            9
  +
      2      2    1           3
    -  %R24  +  %R24 +
      3          2           2
  ,
 %R29,

                1       15     1       14     1       13
    - %R29 +  %R24    +  %R24    +  %R24
               54             27            54
  +
```

```
      2      12   11      11     2      10     1      9
   -  %R24   -  %R24   -  %R24   +  %R24
     27           54           27           54
 +
     7      8     1      7   1      6   1      5   1      4
    %R24  +  %R24  +  %R24  +  %R24  +  %R24
   27          54          9          6          9
 +
   2      2   1          1
    %R24  -  %R24 -
   3          2          2
 ]
,

[%R25,

      1      15     1      14     1      13     2      12
   -  %R25   -  %R25   -  %R25   +  %R25
     54           27           54           27
 +
   11      11     2      10     1      9     7      8
    %R25   +  %R25   -  %R25  -  %R25
   54           27           54          27
 +
      1      7   1      6   1      5   1      4        3
   -  %R25  -  %R25  -  %R25  -  %R25  - %R25
     54          9          6          9
 +
     2      2   1          3
   -  %R25  +  %R25 +
     3          2          2
 ,
 %R26,

              1      15     1      14     1      13
   - %R26 +  %R25   +  %R25   +  %R25
            54           27           54
 +
      2      12   11      11     2      10     1      9
   -  %R25   -  %R25   -  %R25   +  %R25
     27           54           27           54
 +
     7      8     1      7   1      6   1      5   1      4
    %R25  +  %R25  +  %R25  +  %R25  +  %R25
   27          54          9          6          9
 +
   2      2   1          1
    %R25  -  %R25 -
   3          2          2
 ]
```

```
     ,

     [%R25,

            1      15     1      14     1      13     2      12
         -  %R25    -  %R25    -  %R25    +  %R25
           54            27            54            27
       +
         11      11     2      10     1      9     7      8
          %R25    +  %R25    -  %R25   -  %R25
         54            27            54           27
       +
            1      7    1      6    1      5    1      4        3
         -  %R25  -  %R25  -  %R25  -  %R25  - %R25
           54           9           6           9
       +
           2      2    1         3
         -  %R25  +  %R25 +
           3          2          2
       ,
      %R27,

                  1      15     1      14     1      13
         - %R27 +  %R25    +  %R25    +  %R25
                 54            27            54
       +
            2      12    11      11     2      10     1      9
         -  %R25    -  %R25    -  %R25    +  %R25
           27            54            27            54
       +
          7      8     1      7    1      6    1      5    1      4
          %R25  +  %R25  +  %R25  +  %R25  +  %R25
         27           54           9           6           9
       +
         2      2    1         1
          %R25  -  %R25 -
         3          2          2
       ]
     ,
    [1,%R21,- %R21,0], [1,%R22,- %R22,0],
    [1,%R19,0,- %R19], [1,%R20,0,- %R20],
          1      3   1   1      3   1   1      3   1
    [%R17,-  %R17  + ,-  %R17  + ,-  %R17  + ],
          3          3   3          3   3          3
          1      3   1   1      3   1   1      3   1
    [%R18,-  %R18  + ,-  %R18  + ,-  %R18  + ]]
          3          3   3          3   3          3
           Type: List List RealClosure Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ZeroDimSolvePkgXmpPageEmpty25}
\begin{paste}{ZeroDimSolvePkgXmpPageEmpty25}{ZeroDimSolvePkgXmpPagePatch25}
\pastebutton{ZeroDimSolvePkgXmpPageEmpty25}{\showpaste}
\tab{5}\spadcommand{lr2 := realSolve(lf)$pack\free{lf }\free{pack }\bound{lr2 }}
\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPagePatch26}
\begin{paste}{ZeroDimSolvePkgXmpPageFull26}{ZeroDimSolvePkgXmpPageEmpty26}
\pastebutton{ZeroDimSolvePkgXmpPageFull26}{\hidepaste}
\tab{5}\spadcommand{\#lr2\free{lr2 }}
\indentrel{3}\begin{verbatim}
   (26)  27
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPageEmpty26}
\begin{paste}{ZeroDimSolvePkgXmpPageEmpty26}{ZeroDimSolvePkgXmpPagePatch26}
\pastebutton{ZeroDimSolvePkgXmpPageEmpty26}{\showpaste}
\tab{5}\spadcommand{\#lr2\free{lr2 }}
\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPagePatch27}
\begin{paste}{ZeroDimSolvePkgXmpPageFull27}{ZeroDimSolvePkgXmpPageEmpty27}
\pastebutton{ZeroDimSolvePkgXmpPageFull27}{\hidepaste}
\tab{5}\spadcommand{lpr2 := positiveSolve(lf)$pack\free{lf }\free{pack }\bound{lpr2 }}
\indentrel{3}\begin{verbatim}
   (27)
            1     3   1   1     3   1   1     3   1
   [[%R40,-  %R40  + ,-  %R40  + ,-  %R40  + ]]
            3         3   3         3   3         3
           Type: List List RealClosure Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPageEmpty27}
\begin{paste}{ZeroDimSolvePkgXmpPageEmpty27}{ZeroDimSolvePkgXmpPagePatch27}
\pastebutton{ZeroDimSolvePkgXmpPageEmpty27}{\showpaste}
\tab{5}\spadcommand{lpr2 := positiveSolve(lf)$pack\free{lf }\free{pack }\bound{lpr2 }}
\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPagePatch28}
\begin{paste}{ZeroDimSolvePkgXmpPageFull28}{ZeroDimSolvePkgXmpPageEmpty28}
\pastebutton{ZeroDimSolvePkgXmpPageFull28}{\hidepaste}
\tab{5}\spadcommand{[approximate(r,1/10**21)::Float for r in lpr2.1]\free{lpr2 }}
\indentrel{3}\begin{verbatim}
   (28)
   [0.3221853546 2608559291, 0.3221853546 2608559291,
    0.3221853546 2608559291, 0.3221853546 2608559291]
```

```
                                        Type: List Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ZeroDimSolvePkgXmpPageEmpty28}
\begin{paste}{ZeroDimSolvePkgXmpPageEmpty28}{ZeroDimSolvePkgXmpPagePatch28}
\pastebutton{ZeroDimSolvePkgXmpPageEmpty28}{\showpaste}
\tab{5}\spadcommand{[approximate(r,1/10**21)::Float for r in lpr2.1]\free{lpr2 }}
\end{paste}\end{patch}
```

——–

3.122 zlindep.ht

IntegerLinearDependence

— zlindep.ht —

```
\begin{page}{IntegerLinearDependenceXmpPage}{IntegerLinearDependence}
\beginscroll

The elements \texht{$v_1, \dots,v_n$}{\spad{v1,...,vn}}
of a module \spad{M} over a ring \spad{R} are
said to be {\it linearly dependent over \spad{R}} if there exist
\texht{$c_1,\dots,c_n$}{\spad{c1, ..., cn}} in \spad{R},
not all \smath{0},
such that
\texht{$c_1 v_1 + \dots c_n v_n = 0$}{\spad{c1*v1 + ... + cn*vn = 0}}.
If such \texht{$c_i$}{\spad{ci}}'s exist,
they form what is called a {\it linear dependence
relation over \spad{R}} for the \texht{$v_i$}{\spad{vi}}'s.

The package \spadtype{IntegerLinearDependence} provides functions
for testing whether some elements of a module over the integers are
linearly dependent over the integers, and to find the linear
dependence relations, if any.
%
\xtc{
Consider the domain of two by two square matrices with integer entries.
}{
\spadpaste{M := SQMATRIX(2,INT) \bound{M}}
}
%
%
\xtc{
Now create three such matrices.
```

```
}{
\spadpaste{m1: M := squareMatrix matrix [[1, 2], [0, -1]]
\free{M}\bound{m1}}
}
\xtc{
}{
\spadpaste{m2: M := squareMatrix matrix [[2, 3], [1, -2]]
\free{M}\bound{m2}}
}
\xtc{
}{
\spadpaste{m3: M := squareMatrix matrix [[3, 4], [2, -3]]
\free{M}\bound{m3}}
}
%
%
\xtc{
This tells you whether \spad{m1}, \spad{m2} and \spad{m3} are linearly
dependent over the integers.
}{
\spadpaste{linearlyDependentOverZ? vector [m1, m2, m3] \free{m1 m2 m3}}
}
%
%
\xtc{
Since they are linearly dependent, you can ask for the dependence
relation.
}{
\spadpaste{c := linearDependenceOverZ vector [m1, m2, m3]
\free{m1 m2 m3}\bound{c}}
}
%
%
\xtc{
This means that the following linear combination should be \spad{0}.
}{
\spadpaste{c.1 * m1 + c.2 * m2 + c.3 * m3 \free{c m1 m2 m3}}
}
%
When a given set of elements are linearly dependent over \spad{R}, this
also means that at least one of them can be rewritten as a linear
combination of the others with coefficients in the quotient field of
\spad{R}.
%
\xtc{
To express a given element in terms of other elements, use the operation
\spadfunFrom{solveLinearlyOverQ}{IntegerLinearDependence}.
}{
\spadpaste{solveLinearlyOverQ(vector [m1, m3], m2) \free{m1 m2 m3}}
}
```

```
\endscroll
\autobuttons
\end{page}

\begin{patch}{IntegerLinearDependenceXmpPagePatch1}
\begin{paste}{IntegerLinearDependenceXmpPageFull1}{IntegerLinearDependenceXmpPageEmpty1}
\pastebutton{IntegerLinearDependenceXmpPageFull1}{\hidepaste}
\tab{5}\spadcommand{M := SQMATRIX(2,INT)\bound{M }}
\indentrel{3}\begin{verbatim}
   (1)  SquareMatrix(2,Integer)
                                            Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntegerLinearDependenceXmpPageEmpty1}
\begin{paste}{IntegerLinearDependenceXmpPageEmpty1}{IntegerLinearDependenceXmpPagePatch1}
\pastebutton{IntegerLinearDependenceXmpPageEmpty1}{\showpaste}
\tab{5}\spadcommand{M := SQMATRIX(2,INT)\bound{M }}
\end{paste}\end{patch}

\begin{patch}{IntegerLinearDependenceXmpPagePatch2}
\begin{paste}{IntegerLinearDependenceXmpPageFull2}{IntegerLinearDependenceXmpPageEmpty2}
\pastebutton{IntegerLinearDependenceXmpPageFull2}{\hidepaste}
\tab{5}\spadcommand{m1: M := squareMatrix matrix [[1, 2], [0, -1]]\free{M }\bound{m1 }}
\indentrel{3}\begin{verbatim}
        1   2
   (2)
        0  - 1
                          Type: SquareMatrix(2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntegerLinearDependenceXmpPageEmpty2}
\begin{paste}{IntegerLinearDependenceXmpPageEmpty2}{IntegerLinearDependenceXmpPagePatch2}
\pastebutton{IntegerLinearDependenceXmpPageEmpty2}{\showpaste}
\tab{5}\spadcommand{m1: M := squareMatrix matrix [[1, 2], [0, -1]]\free{M }\bound{m1 }}
\end{paste}\end{patch}

\begin{patch}{IntegerLinearDependenceXmpPagePatch3}
\begin{paste}{IntegerLinearDependenceXmpPageFull3}{IntegerLinearDependenceXmpPageEmpty3}
\pastebutton{IntegerLinearDependenceXmpPageFull3}{\hidepaste}
\tab{5}\spadcommand{m2: M := squareMatrix matrix [[2, 3], [1, -2]]\free{M }\bound{m2 }}
\indentrel{3}\begin{verbatim}
        2   3
   (3)
        1  - 2
                          Type: SquareMatrix(2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{IntegerLinearDependenceXmpPageEmpty3}
\begin{paste}{IntegerLinearDependenceXmpPageEmpty3}{IntegerLinearDependenceXmpPagePatch3}
\pastebutton{IntegerLinearDependenceXmpPageEmpty3}{\showpaste}
\tab{5}\spadcommand{m2: M := squareMatrix matrix [[2, 3], [1, -2]]\free{M }\bound{m2 }}
\end{paste}\end{patch}

\begin{patch}{IntegerLinearDependenceXmpPagePatch4}
\begin{paste}{IntegerLinearDependenceXmpPageFull4}{IntegerLinearDependenceXmpPageEmpty4}
\pastebutton{IntegerLinearDependenceXmpPageFull4}{\hidepaste}
\tab{5}\spadcommand{m3: M := squareMatrix matrix [[3, 4], [2, -3]]\free{M }\bound{m3 }}
\indentrel{3}\begin{verbatim}
        3   4
   (4)
        2  - 3
                          Type: SquareMatrix(2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntegerLinearDependenceXmpPageEmpty4}
\begin{paste}{IntegerLinearDependenceXmpPageEmpty4}{IntegerLinearDependenceXmpPagePatch4}
\pastebutton{IntegerLinearDependenceXmpPageEmpty4}{\showpaste}
\tab{5}\spadcommand{m3: M := squareMatrix matrix [[3, 4], [2, -3]]\free{M }\bound{m3 }}
\end{paste}\end{patch}

\begin{patch}{IntegerLinearDependenceXmpPagePatch5}
\begin{paste}{IntegerLinearDependenceXmpPageFull5}{IntegerLinearDependenceXmpPageEmpty5}
\pastebutton{IntegerLinearDependenceXmpPageFull5}{\hidepaste}
\tab{5}\spadcommand{linearlyDependentOverZ? vector [m1, m2, m3]\free{m1 m2 m3 }}
\indentrel{3}\begin{verbatim}
   (5)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntegerLinearDependenceXmpPageEmpty5}
\begin{paste}{IntegerLinearDependenceXmpPageEmpty5}{IntegerLinearDependenceXmpPagePatch5}
\pastebutton{IntegerLinearDependenceXmpPageEmpty5}{\showpaste}
\tab{5}\spadcommand{linearlyDependentOverZ? vector [m1, m2, m3]\free{m1 m2 m3 }}
\end{paste}\end{patch}

\begin{patch}{IntegerLinearDependenceXmpPagePatch6}
\begin{paste}{IntegerLinearDependenceXmpPageFull6}{IntegerLinearDependenceXmpPageEmpty6}
\pastebutton{IntegerLinearDependenceXmpPageFull6}{\hidepaste}
\tab{5}\spadcommand{c := linearDependenceOverZ vector [m1, m2, m3]\free{m1 m2 m3 }\bound{c }}
\indentrel{3}\begin{verbatim}
   (6)  [1,- 2,1]
                        Type: Union(Vector Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

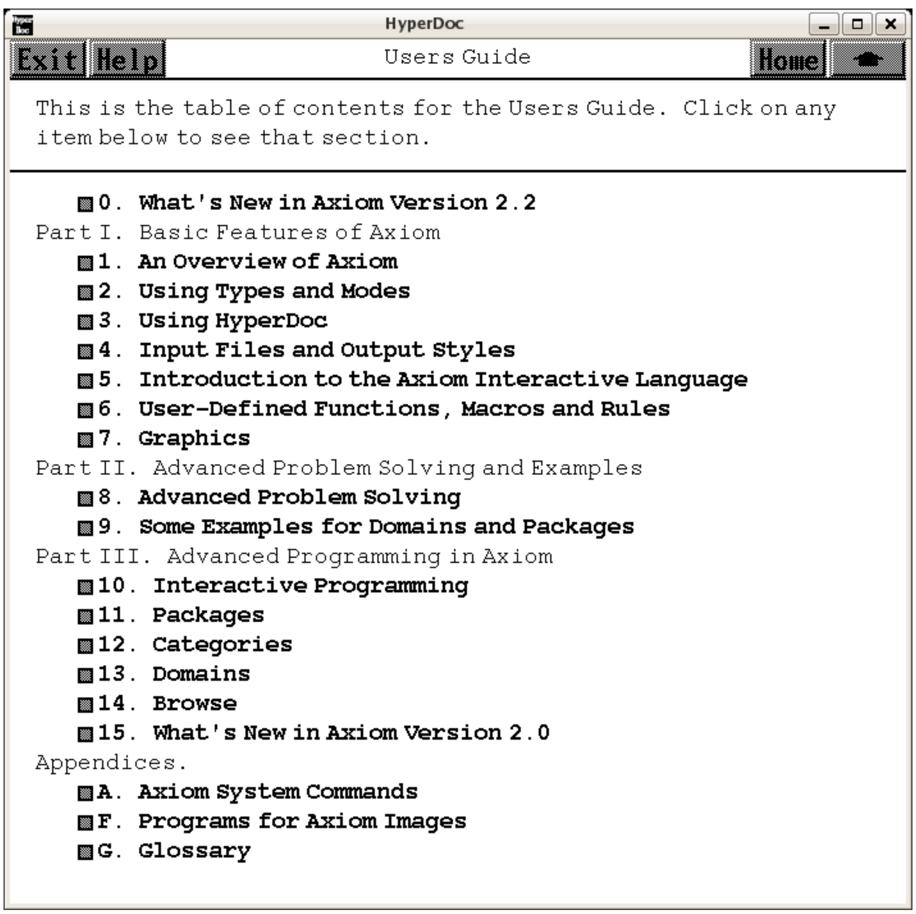
```
\begin{patch}{IntegerLinearDependenceXmpPageEmpty6}
\begin{paste}{IntegerLinearDependenceXmpPageEmpty6}{IntegerLinearDependenceXmpPagePatch6}
\pastebutton{IntegerLinearDependenceXmpPageEmpty6}{\showpaste}
\tab{5}\spadcommand{c := linearDependenceOverZ vector [m1, m2, m3]\free{m1 m2 m3 }\bound{c }}
\end{paste}\end{patch}

\begin{patch}{IntegerLinearDependenceXmpPagePatch7}
\begin{paste}{IntegerLinearDependenceXmpPageFull7}{IntegerLinearDependenceXmpPageEmpty7}
\pastebutton{IntegerLinearDependenceXmpPageFull7}{\hidepaste}
\tab{5}\spadcommand{c.1 * m1 + c.2 * m2 + c.3 * m3\free{c m1 m2 m3 }}
\indentrel{3}\begin{verbatim}
       0  0
   (7)
       0  0
                         Type: SquareMatrix(2,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntegerLinearDependenceXmpPageEmpty7}
\begin{paste}{IntegerLinearDependenceXmpPageEmpty7}{IntegerLinearDependenceXmpPagePatch7}
\pastebutton{IntegerLinearDependenceXmpPageEmpty7}{\showpaste}
\tab{5}\spadcommand{c.1 * m1 + c.2 * m2 + c.3 * m3\free{c m1 m2 m3 }}
\end{paste}\end{patch}

\begin{patch}{IntegerLinearDependenceXmpPagePatch8}
\begin{paste}{IntegerLinearDependenceXmpPageFull8}{IntegerLinearDependenceXmpPageEmpty8}
\pastebutton{IntegerLinearDependenceXmpPageFull8}{\hidepaste}
\tab{5}\spadcommand{solveLinearlyOverQ(vector [m1, m3], m2)\free{m1 m2 m3 }}
\indentrel{3}\begin{verbatim}
        1 1
   (8)  [,]
        2 2
              Type: Union(Vector Fraction Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{IntegerLinearDependenceXmpPageEmpty8}
\begin{paste}{IntegerLinearDependenceXmpPageEmpty8}{IntegerLinearDependenceXmpPagePatch8}
\pastebutton{IntegerLinearDependenceXmpPageEmpty8}{\showpaste}
\tab{5}\spadcommand{solveLinearlyOverQ(vector [m1, m3], m2)\free{m1 m2 m3 }}
\end{paste}\end{patch}
```

———

Chapter 4

Users Guide Pages (ug.ht)

Users Guide



⇐ "Reference" (TopReferencePage) 3.1 on page 123
⇒ "What's New in Axiom Version May 2008" (ugWhatsNewTwoTwoPage) 5 on page 1473
⇒ "An Overview of Axiom" (ugIntroPage) 6 on page 1483
⇒ "Using Types and Modes" (ugTypesPage) 7 on page 1613
⇒ "Using HyperDoc" (ugHyperPage) 8 on page 1707
⇒ "Input Files and Output Styles" (ugInOutPage) 9 on page 1721
⇒ "Introduction to the Axiom Interactive Language" (ugLangPage) **??** on page **??**
⇒ "User-Defined Functions, ..." (ugUserPage) 10 on page 1823
⇒ "Graphics" (ugGraphPage) 11 on page 1967
⇒ "Advanced Problem Solving" (ugProblemPage) 12 on page 2077
⇒ "Some Examples for Domains and Packages" (ExamplesExposedPage) 3.117 on page 1369
⇒ "Interactive Programming" (ugIntProgPage) 13 on page 2337
⇒ "Packages" (ugPackagesPage) 14 on page 2375
⇒ "Categories" (ugCategoriesPage) 15 on page 2403
⇒ "Domains" (ugDomainsPage) 16 on page 2425
⇒ "Browse" (ugBrowsePage) 17 on page 2471
⇒ "What's New in Axiom Version 2.0" (ugWhatsNewPage) 18 on page 2493
⇒ "Axiom System Commands" (ugSysCmdPage) 19 on page 2538
⇒ "Programs for Axiom Images" (ugAppGraphicsPage) 20 on page 2589
⇒ "Glossary" (GlossaryPage) 3.49 on page 579

— **ug.ht** —

```
\begin{page}{UsersGuidePage}{Users Guide}
This is the table of contents for the Users Guide.
Click on any item below to see that section.
\beginscroll
\indent{3}
\beginmenu
\menudownlink{{0. What's New in Axiom Version May 2008}}
{ugWhatsNewTwoTwoPage}
\endmenu
\indent{0}
Part I.   Basic Features of Axiom
\indent{3}
\beginmenu
\menudownlink{{1. An Overview of Axiom}}{ugIntroPage}
\menudownlink{{2. Using Types and Modes}}{ugTypesPage}
\menudownlink{{3. Using Hyperdoc}}{ugHyperPage}
\menudownlink{{4. Input Files and Output Styles}}{ugInOutPage}
\menudownlink{{5. Introduction to the Axiom Interactive Language}}
{ugLangPage}
\menudownlink{{6. User-Defined Functions, Macros and Rules}}{ugUserPage}
\menudownlink{{7. Graphics}}{ugGraphPage}
\endmenu
\indent{0}
Part II.  Advanced Problem Solving and Examples
\indent{3}
\beginmenu
```

```
\menudownlink{{8. Advanced Problem Solving}}{ugProblemPage}
\menudownlink{{9. Some Examples for Domains and Packages}}
{ExamplesExposedPage}
\endmenu
\indent{0}
Part III. Advanced Programming in Axiom
\indent{3}
\beginmenu
\menudownlink{{10. Interactive Programming}}{ugIntProgPage}
\menudownlink{{11. Packages}}{ugPackagesPage}
\menudownlink{{12. Categories}}{ugCategoriesPage}
\menudownlink{{13. Domains}}{ugDomainsPage}
\menudownlink{{14. Browse}}{ugBrowsePage}
\menudownlink{{15. What's New in Axiom Version 2.0}}{ugWhatsNewPage}
\endmenu
\indent{0}
Appendices.
\indent{3}
\beginmenu
\menudownlink{{A. Axiom System Commands}}{ugSysCmdPage}
\menudownlink{{F. Programs for Axiom Images}}{ugAppGraphicsPage}
\menudownlink{{G. Glossary}}{GlossaryPage}
\endmenu
\indent{0}
\endscroll
\autobuttons
\end{page}
```

Chapter 5

Users Guide Chapter 0 (ug00.ht)

What's New for May 2008

No image here because the page changes every release.
⇐ "Reference" (TopReferencePage) 3.1 on page 123
⇒ "New polynomial domains and algorithms" (ugTwoTwoPolynomialsPage) 5 on page 1474
⇒ "Enhancements to Hyperdoc..." (ugTwoTwoHyperdocPage) 5 on page 1475
⇒ "Enhancements to NAGLink" (ugTwoTwoNAGLinkPage) 5 on page 1476
⇒ "Enhancements to the Lisp system" (ugTwoTwoCCLPage) 5 on page 1476

— **ug00.ht** —

```
\begin{page}{ugWhatsNewTwoTwoPage}
{0. What's New for May 2008}
\beginscroll

\beginmenu
    \menudownlink{{0.2. New polynomial domains and algorithms}}
{ugTwoTwoPolynomialsPage}
    \menudownlink{{0.3. Enhancements to HyperDoc and Graphics}}
{ugTwoTwoHyperdocPage}
    \menudownlink{{0.4. Enhancements to NAGLink}}{ugTwoTwoNAGLinkPage}
    \menudownlink{{0.5. Enhancements to the Lisp system}}{ugTwoTwoCCLPage}
\endmenu
\endscroll
\autobuttons
\end{page}
```

———-

New polynomial domains and algorithms

— ug00.ht —

```
\begin{page}{ugTwoTwoPolynomialsPage}
{0.2. New polynomial domains and algorithms}
\beginscroll

Univariate polynomial factorisation over the integers has been
enhanced by updates to the \spadtype{GaloisGroupFactorizer} type and
friends from Frederic Lehobey (Frederic.Lehobey@lifl.fr, University of
Lille I, France).

The package constructor \spadtype{PseudoRemainderSequence} provides
efficient algorithms by Lionel Ducos
(Lionel.Ducos@mathlabo.univ-poitiers.fr, University of Poitiers,
France) for computing sub-resultants.  This leads to a speed up in
many places in Axiom where sub-resultants are computed (polynomial
system solving, algebraic factorization, integration).

Based on this package, the domain constructor
\spadtype{NewSparseUnivariatePolynomial} extends the constructor
\spadtype{SparseUnivariatePolynomial}.  In a similar way, the
\spadtype{NewSparseMultivariatePolynomial} extends the constructor
\spadtype{SparseUnivariatePolynomial}; it also provides some
additional operations related to polynomial system solving by means of
triangular sets.

Several domain constructors implement regular triangular sets (or
regular chains).  Among them \spadtype{RegularTriangularSet} and
\spadtype{SquareFreeRegularTriangularSet}.  They also implement an
algorithm by Marc Moreno Maza (marc@nag.co.uk, NAG) for computing
triangular decompositions of polynomial systems.  This method is
refined in the package \spadtype{LazardSetSolvingPackage} in order to
produce decompositions by means of Lazard triangular sets.  For the
case of polynomial systems with finitely many solutions, these
decompositions can also be computed by the package
\spadtype{LexTriangularPackage}.

The domain constructor \spadtype{RealClosure} by Renaud Rioboo
(Renaud.Rioboo@lip6.fr, University of Paris 6, France) provides the
real closure of an ordered field.  The implementation is based on
interval arithmetic.  Moreover, the design of this constructor and its
related packages allows an easy use of other codings for real
algebraic numbers.

Based on triangular decompositions and the \spadtype{RealClosure}
constructor, the package \spadtype{ZeroDimensionalSolvePackage}
provides operations for computing symbolically the real or complex
```

```
roots of polynomial systems with finitely many solutions.

Polynomial arithmetic with non-commutative variables has been improved
too by a contribution of Michel Petitot (Michel.Petitot@lifl.fr,
University of Lille I, France).  The domain constructors
\spadtype{XRecursivePolynomial} and \spadtype{XDistributedPolynomial}
provide recursive and distributed representations for these
polynomials.  They are the non-commutative equivalents for the
\spadtype{SparseMultivariatePolynomial} and
\spadtype{DistributedMultivariatePoly} constructors.  The
constructor \spadtype{LiePolynomial} implement Lie polynomials in the
Lyndon basis.  The constructor \spadtype{XPBWPolynomial} manage
polynomials with non-commutative variables in the
\texht{Poincar\'e}{Poincare\space{-1}'}-Birkhoff-Witt basis from the
Lyndon basis.  This allows to compute in the Lie Group associated with
a free nilpotent Lie algebra by using the \spadtype{LieExponentials}
domain constructor.

\endscroll
\autobuttons
\end{page}
```

———-

Enhancements to HyperDoc and Graphics

— ug00.ht —

```
\begin{page}{ugTwoTwoHyperdocPage}
{0.3. Enhancements to HyperDoc and Graphics}
\beginscroll

From this version of Axiom onwards, the pixmap format used to save
graphics images in color and to display them in HyperDoc has been
changed to the industry-standard XPM format. See
{\tt http://koala.ilog.fr/ftp/pub/xpm}.

\endscroll
\autobuttons
\end{page}
```

———-

Enhancements to NAGLink

— ug00.ht —

```
\begin{page}{ugTwoTwoNAGLinkPage}{0.4. Enhancements to NAGLink}
\beginscroll

This version of Axiom incorporates the Axiom/NAG Numerical Analyst
(ANNA) software developed in the University of Bath. ANNA is an expert
system which uses Axiom algebra functionality in choosing the most
appropriate NAG Fortran Library routine for a particular
problem. Documentation is provided on-line on the main HyperDoc page.

\endscroll
\autobuttons
\end{page}
```

———-

Enhancements to the Lisp system

— ug00.ht —

```
\begin{page}{ugTwoTwoCCLPage}{0.5. Enhancements to the Lisp system}
\beginscroll

In this version of Axiom, certain Library operations have been
accelerated by converting their Lisp implementations into kernel
operations implemented directly in C. Here is a list of the
accelerated operations. The given names encode the abbreviated type
the operation comes from, the name of the operation and, in the case
of exported functions, an encoded signature and numerical index.

\begin{verbatim}
|A1AGG-;=;2AB;28|
|A1AGG-;copy;2A;19|
|A1AGG-;sort!;M2A;2|
|ABELGRP-;*;Nni2S;3|
|ABELGRP-;-;3S;1|
|ABELMON-;*;Pi2S;2|
|ABELMON-;zero?;SB;1|
|AGG-;empty?;SB;3|
|AGG-;size?;SNniB;6|
|ALIST;keys;$L;6|
|ALIST;search;Key$U;15|
```

```
|ARR2CAT-;copy;2S;10|
|BOP;<;2$B;29|
|BOP;=;2$B;27|
|BOP;has?;$SB;9|
|BOP;is?;$SB;1|
|BOP;name;$S;2|
|BOP;properties;$Al;3|
|BOP;property;$SU;7|
|BOP;weight;$Nni;28|
|COMPCAT-;*;R2S;18|
|COMPCAT-;-;2S;17|
|COMPCAT-;=;2SB;15|
|COMPCAT-;exquo;SRU;46|
|COMPCAT-;recip;SU;48|
|COMPCAT-;rem;3S;53|
|COMPCAT-;unitNormal;SR;49|
|COMPLEX;*;3$;10|
|COMPLEX;+;3$;9|
|COMPLEX;coerce;R$;5|
|COMPLEX;exquo;2$U;11|
|COMPLEX;one?;$B;4|
|COMPLEX;zero?;$B;3|
|DIRPROD;<;2$B;18|
|DIRPROD;subtractIfCan;2$U;14|
|ELAGG-;removeDuplicates;2A;12|
|ELTAGG-;qelt;SDomIm;1|
|ELTAGG-;qsetelt!;SDom2Im;2|
|EUCDOM-;gcd;3S;5|
|EUCDOM-;unitNormalizeIdealElt|
|EXPR;*;3$;11|
|EXPR;+;3$;12|
|EXPR;-;2$;8|
|EXPR;/;3$;14|
|EXPR;=;2$B;21|
|EXPR;algkernels!0|
|EXPR;algkernels|
|EXPR;coerce;I$;10|
|EXPR;coerce;Smp$;24|
|EXPR;commonk0|
|EXPR;commonk|
|EXPR;denom;$Smp;23|
|EXPR;numer;$Smp;22|
|EXPR;reduc|
|EXPR;zero?;$B;7|
|FACUTIL;lowerPolynomial;SupSup;1|
|FAMR-;coefficients;SL;4|
|FAMR-;ground;SR;2|
|FFP;*;3$;21|
|FFP;+;3$;22|
|FFP;-;3$;23|
```

```
|FFP;=;2$B;24|
|FIELD-;/;3S;11|
|FIELD-;inv;2S;4|
|FLAGG-;sort!;2A;8|
|FLAGG-;sort;M2A;6|
|FLASORT;QuickSort|
|FLASORT;partition|
|FLASORT;quickSort;M2V;1|
|FM;*;R2$;1|
|FRAC;*;3$;18|
|FRAC;*;I2$;19|
|FRAC;+;3$;16|
|FRAC;=;2$B;22|
|FRAC;cancelGcd|
|FRAC;coerce;S$;1|
|FRAC;normalize|
|FRAC;one?;$B;23|
|FRAC;recip;$U;13|
|FRAC;zero?;$B;2|
|FSAGG-;brace;LA;3|
|GDMP;univariate;$OvlSup;21|
|HDP;<;2$B;1|
|IARRAY1;#;$Nni;1|
|IARRAY1;elt;$IS;16|
|IARRAY1;fill!;$S$;2|
|IARRAY1;map;M3$;8|
|IARRAY1;maxIndex;$I;13|
|IARRAY1;minIndex;$I;3|
|IARRAY1;new;NniS$;5|
|IARRAY1;qelt;$IS;14|
|IARRAY1;qsetelt!;$I2S;15|
|IARRAY1;setelt;$I2S;17|
|IDPAM;+;3$;4|
|IDPAM;map;M2$;7|
|IDPAM;monomial;AS$;6|
|IDPO;=;2$B;1|
|IFARRAY;#;$Nni;4|
|IFARRAY;concat!;$S$;21|
|IFARRAY;elt;$IS;17|
|IFARRAY;empty;$;3|
|IFARRAY;growAndFill|
|IFARRAY;growWith|
|IFARRAY;maxIndex;$I;6|
|IFARRAY;minIndex;$I;7|
|IFARRAY;new;NniS$;8|
|IFARRAY;removeDuplicates!;2$;30|
|IFARRAY;setelt;$I2S;18|
|IIARRAY2;elt;$2IR;11|
|IIARRAY2;empty?;$B;1|
|IIARRAY2;maxColIndex;$I;7|
```

```
|IIARRAY2;maxRowIndex;$I;6|
|IIARRAY2;minColIndex;$I;5|
|IIARRAY2;minRowIndex;$I;4|
|IIARRAY2;ncols;$Nni;9|
|IIARRAY2;nrows;$Nni;8|
|IIARRAY2;qelt;$2IR;10|
|IIARRAY2;qsetelt!;$2I2R;12|
|ILIST;concat!;3$;25|
|ILIST;copy;2$;20|
|ILIST;empty;$;6|
|ILIST;first;$S;4|
|ILIST;member?;S$B;24|
|ILIST;mergeSort|
|ILIST;minIndex;$I;18|
|ILIST;removeDuplicates!;2$;26|
|ILIST;rest;$Nni$;19|
|ILIST;sort!;M2$;27|
|ILIST;split!;$I$;29|
|IMATLIN;rowEchelon;2M;3|
|INS-;symmetricRemainder;3S;27|
|INT;exquo;2$U;44|
|INT;one?;$B;2|
|INT;positiveRemainder;3$;23|
|INT;unitNormal;$R;47|
|INTDOM-;unitCanonical;2S;2|
|ISTRING;<;2$B;6|
|KDAGG-;key?;KeySB;1|
|KERNEL;<;2$B;14|
|KERNEL;=;2$B;13|
|KERNEL;B2Z|
|KERNEL;argument;$L;6|
|KERNEL;operator;$Bo;5|
|KERNEL;position;$Nni;7|
|KERNEL;triage|
|LO;-;2$;3|
|LO;=;2$B;4|
|LSAGG-;<;2AB;25|
|LSAGG-;reduce;MA2S;16|
|LSAGG-;select!;M2A;5|
|MATCAT-;*;3S;29|
|MATCAT-;*;S2Col;32|
|MDDFACT;reduction!0|
|MDDFACT;reduction|
|MODRING;reduce;RMod$;6|
|MODRING;zero?;$B;10|
|MONOID-;one?;SB;2|
|NNI;subtractIfCan;2$U;3|
|NSMP;mvar;$VarSet;5|
|OVAR;<;2$B;10|
|PERMGRP;inv|
```

```
|PERMGRP;orbitWithSvc|
|PERMGRP;testIdentity|
|PERMGRP;times|
|PGCD;better|
|PGCD;gcd;3P;15|
|PGCD;gcdTermList|
|POLYCATQ;variables;FL;5|
|PR;*;3$;20|
|PR;addm!|
|PR;coerce;R$;12|
|PR;degree;$E;4|
|PR;leadingCoefficient;$R;6|
|PR;reductum;2$;8|
|PRIMARR;#;$Nni;1|
|PRIMARR;fill!;$S$;9|
|PRIMARR;new;NniS$;4|
|PRTITION;<;2$B;5|
|REPSQ;expt;SPiS;1|
|RING-;coerce;IS;1|
|SAE;*;3$;15|
|SAE;+;3$;13|
|SAE;-;2$;14|
|SAE;=;2$B;12|
|SAE;reduce;UP$;11|
|SCACHE;enterInCache;SMS;5|
|SET;construct;L$;19|
|SET;empty;$;4|
|SGROUP-;**;SPiS;1|
|SINT;zero?;$B;33|
|SMP;*;3$;29|
|SMP;*;R2$;25|
|SMP;+;3$;26|
|SMP;-;2$;23|
|SMP;=;2$B;28|
|SMP;coerce;R$;20|
|SMP;coerce;VarSet$;7|
|SMP;exquo;2$U;35|
|SMP;exquo;2$U;36|
|SMP;gcd;3$;41|
|SMP;gcd;3$;44|
|SMP;gcd;3$;48|
|SMP;gcd;3$;51|
|SMP;ground?;$B;15|
|SMP;leadingCoefficient;$R;73|
|SMP;mainVariable;$U;59|
|SMP;one?;$B;4|
|SMP;retract;$R;55|
|SMP;unitNormal;$R;31|
|SMP;variables;$L;58|
|SMP;zero?;$B;3|
```

```
|STAGG-;c2|
|STAGG-;concat;3A;7|
|STAGG-;elt;AIS;5|
|STAGG-;first;ANniA;3|
|STREAM2;map;MSS;2|
|STREAM2;mapp!0|
|STREAM2;mapp|
|STREAM;empty;$;33|
|STREAM;lazyEval|
|STREAM;setfrst!|
|STREAM;setrst!|
|STTAYLOR;+;3S;2|
|SUP;exquo;2$U;19|
|SUP;exquo;2$U;20|
|SUP;fmecg;$NniR2$;21|
|SUP;ground?;$B;3|
|SUP;monicDivide;2$R;28|
|URAGG-;tail;2A;16|
|ZMOD;*;3$;30|
|ZMOD;+;3$;32|
|ZMOD;-;2$;36|
|ZMOD;-;3$;33|
\end{verbatim}
\endscroll
\autobuttons
\end{page}
```

Chapter 6

Users Guide Chapter 1 (ug01.ht)

An Overview of Axiom

⇐ "Numeric Functions" (ugProblemNumericPage) 12 on page 2079
⇒ "Starting Up and Winding Down" (ugIntroStartPage) 6 on page 1484
⇒ "Typographic Conventions" (ugIntroTypoPage) 6 on page 1488
⇒ "The Axiom Language" (ugIntroExpressionsPage) 6 on page 1489
⇒ "Graphics" (ugIntroGraphicsPage) 6 on page 1510
⇒ "Numbers" (ugIntroNumbersPage) 6 on page 1513
⇒ "Data Structures" (ugIntroCollectPage) 6 on page 1532
⇒ "Expanding to Higher Dimensions" (ugIntroTwoDimPage) 6 on page 1548
⇒ "Writing Your Own Functions" (ugIntroYouPage) 6 on page 1553
⇒ "Polynomials" (ugIntroVariablesPage) 6 on page 1566
⇒ "Limits" (ugIntroCalcLimitsPage) 6 on page 1569
⇒ "Series" (ugIntroSeriesPage) 6 on page 1573
⇒ "Derivatives" (ugIntroCalcDerivPage) 6 on page 1580
⇒ "Integration" (ugIntroIntegratePage) 6 on page 1587
⇒ "Differential Equations" (ugIntroDiffEqnsPage) 6 on page 1595
⇒ "Solution of Equations" (ugIntroSolutionPage) 6 on page 1602
⇒ "System Commands" (ugIntroSysCmmandsPage) 6 on page 1606

— **ug01.ht** —

```
\begin{page}{ugIntroPage}{1. An Overview of Axiom}

\beginscroll

Welcome to the Axiom environment for interactive computation and
problem solving.  Consider this chapter a brief, whirlwind tour of the
```

```
Axiom world.  We introduce you to Axiom's graphics and the Axiom
language.  Then we give a sampling of the large variety of facilities
in the Axiom system, ranging from the various kinds of numbers, to
data types (like lists, arrays, and sets) and mathematical objects
(like matrices, integrals, and differential equations).  We conclude
with the discussion of system commands and an interactive ``undo.''

Before embarking on the tour, we need to brief those readers
working interactively with Axiom on some details.
Others can skip right immediately to
\downlink{``Typographic Conventions''}{ugIntroTypoPage} in
Section 1.2\ignore{ugIntroTypo}.

\beginmenu
    \menudownlink{{1.1. Starting Up and Winding Down}}{ugIntroStartPage}
    \menudownlink{{1.2. Typographic Conventions}}{ugIntroTypoPage}
    \menudownlink{{1.3. The Axiom Language}}{ugIntroExpressionsPage}
    \menudownlink{{1.4. Graphics}}{ugIntroGraphicsPage}
    \menudownlink{{1.5. Numbers}}{ugIntroNumbersPage}
    \menudownlink{{1.6. Data Structures}}{ugIntroCollectPage}
    \menudownlink{{1.7. Expanding to Higher Dimensions}}{ugIntroTwoDimPage}
    \menudownlink{{1.8. Writing Your Own Functions}}{ugIntroYouPage}
    \menudownlink{{1.9. Polynomials}}{ugIntroVariablesPage}
    \menudownlink{{1.10. Limits}}{ugIntroCalcLimitsPage}
    \menudownlink{{1.11. Series}}{ugIntroSeriesPage}
    \menudownlink{{1.12. Derivatives}}{ugIntroCalcDerivPage}
    \menudownlink{{1.13. Integration}}{ugIntroIntegratePage}
    \menudownlink{{1.14. Differential Equations}}{ugIntroDiffEqnsPage}
    \menudownlink{{1.15. Solution of Equations}}{ugIntroSolutionPage}
    \menudownlink{{1.16. System Commands}}{ugIntroSysCmmandsPage}
\endmenu
\endscroll
\autobuttons
\end{page}
```

———-

Starting Up and Winding Down

⇒ "notitle" (ugHyperPage) 8 on page 1707
⇒ "notitle" (ugSysCmdPage) 19 on page 2538
⇒ "notitle" (ugAvailCLEFPage) 6 on page 1487

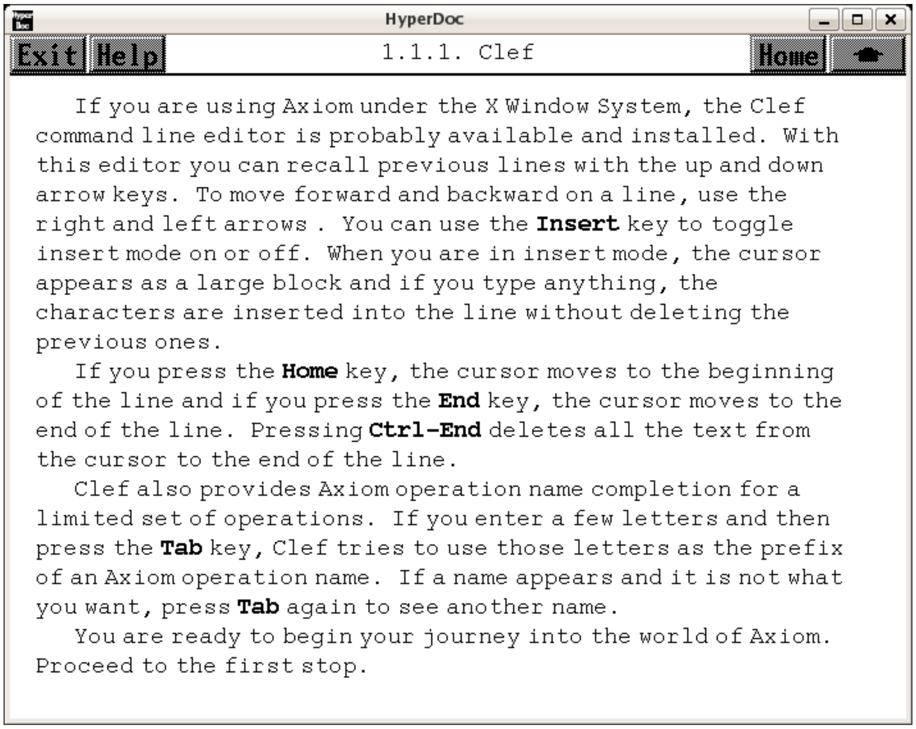
— **ug01.ht** —

```
\begin{page}{ugIntroStartPage}{1.1. Starting Up and Winding Down}
\beginscroll
```

```
You need to know how to start the Axiom system and how to stop it.
We assume that Axiom has been correctly installed on your machine (as
described in another Axiom document).

To begin using Axiom, issue the command {\bf axiom} to the operating
system shell.  There is a brief pause, some start-up messages, and
then one or more windows appear.

If you are not running Axiom under the X Window System, there is only
one window (the console).  At the lower left of the screen there is a
prompt that looks like
\begin{verbatim}
(1) ->
\end{verbatim}

When you want to enter input to Axiom, you do so on the same line
after the prompt.  The ``1'' in ``(1)'' is the computation step number
and is incremented after you enter Axiom statements.  Note, however,
that a system command such as \spadsys{)clear all} may change the step
number in other ways.  We talk about step numbers more when we discuss
system commands and the workspace history facility.

If you are running Axiom under the X Window System, there may be two
windows: the console window (as just described) and the Hyperdoc main
menu.  Hyperdoc is a multiple-window hypertext system that lets you
view Axiom documentation and examples on-line, execute Axiom
expressions, and generate graphics.  If you are in a graphical
windowing environment, it is usually started automatically when Axiom
begins.  If it is not running, issue \spadsys{)hd} to start it.  We
discuss the basics of Hyperdoc in
\downlink{``Using Hyperdoc''}{ugHyperPage} in Chapter 3\ignore{ugHyper}.

To interrupt an Axiom computation, hold down the
\texht{\fbox{\bf Ctrl}}{{\bf Ctrl}} (control) key and press
\texht{\fbox{\bf c}}{{\bf c}}.
This  brings you back to the Axiom prompt.

\beginImportant

To exit from Axiom, move to the console window, type \spadsys{)quit}
at the input prompt and press the \texht{\fbox{\bf Enter}}
{{\bf Enter}} key.  You will probably be prompted with the following
message: \centerline{{Please enter {\bf y} or {\bf yes} if you really
want to leave the }} \centerline{{interactive environment and return
to the operating system}} You should respond {\bf yes}, for example,
to exit Axiom.
\endImportant

We are purposely vague in describing exactly what your screen looks
```

```
like or what messages Axiom displays.  Axiom runs on a number of
different machines, operating systems and window environments, and
these differences all affect the physical look of the system.  You can
also change the way that Axiom behaves via \spadgloss{system commands}
described later in this chapter and in
\downlink{``Axiom System Commands''}{ugSysCmdPage} in Appendix B
\ignore{ugSysCmd}.  System commands are special
commands, like \spadcmd{)set}, that begin with a closing parenthesis
and are used to change your environment.  For example, you can set a
system variable so that you are not prompted for confirmation when you
want to leave Axiom.

\beginmenu
    \menudownlink{{1.1.1. \Clef{}}}{ugAvailCLEFPage}
\endmenu
\endscroll
\autobuttons
\end{page}
```

Clef



If you are using Axiom under the X Window System, the Clef command line editor is probably available and installed. With this editor you can recall previous lines with the up and down arrow keys. To move forward and backward on a line, use the right and left arrows . You can use the **Insert** key to toggle insert mode on or off. When you are in insert mode, the cursor appears as a large block and if you type anything, the characters are inserted into the line without deleting the previous ones.

If you press the **Home** key, the cursor moves to the beginning of the line and if you press the **End** key, the cursor moves to the end of the line. Pressing **Ctrl-End** deletes all the text from the cursor to the end of the line.

Clef also provides Axiom operation name completion for a limited set of operations. If you enter a few letters and then press the **Tab** key, Clef tries to use those letters as the prefix of an Axiom operation name. If a name appears and it is not what you want, press **Tab** again to see another name.

You are ready to begin your journey into the world of Axiom. Proceed to the first stop.

⇐ "Computation of Galois Groups" (ugProblemGaloisPage) 12 on page 2306

— **ug01.ht** —

```
\begin{page}{ugAvailCLEFPage}{1.1.1. \Clef{}}
\beginscroll
%

If you are using Axiom under the X Window System, the \Clef{} command
line editor is probably available and installed.  With this editor you
can recall previous lines with the up and down arrow keys\texht{
(\fbox{$\uparrow$} and \fbox{$\downarrow$})}{}.  To move forward and
backward on a line, use the right and left arrows
\texht{(\fbox{$\rightarrow$} and \fbox{$\leftarrow$})}{}.  You can use the
\texht{\fbox{\bf Insert}}{{\bf Insert}} key to toggle insert mode on
or off.  When you are in insert mode, the cursor appears as a large
block and if you type anything, the characters are inserted into the
line without deleting the previous ones.

If you press the \texht{\fbox{\bf Home}}{{\bf Home}} key, the cursor
moves to the beginning of the line and if you press the
\texht{\fbox{\bf End}}{{\bf End}} key, the cursor moves to the end of
the line.  Pressing
```

```
\texht{\fbox{\bf Ctrl}--\fbox{\bf End}}{{\bf Ctrl-End}}
deletes all the text from the cursor to the end of the line.

\Clef{} also provides Axiom operation name completion for a limited
set of operations.  If you enter a few letters and then press the
\texht{\fbox{\bf Tab}}{{\bf Tab}} key, \Clef{} tries to use those
letters as the prefix of an Axiom operation name.  If a name appears
and it is not what you want, press \texht{\fbox{\bf Tab}}{{\bf Tab}}
again to see another name.

You are ready to begin your journey into the world of Axiom.
Proceed to the first stop.

\endscroll
\autobuttons
\end{page}
```

———

Typographic Conventions

— ug01.ht —

```
\begin{page}{ugIntroTypoPage}{1.2. Typographic Conventions}
\beginscroll

In this book we have followed these typographical conventions:
\indent{4}
\beginitems
%
\item[-] Categories, domains and packages are displayed in
\texht{a sans-serif typeface:}{this font:}
\axiomType{Ring}, \axiomType{Integer},
\axiomType{DiophantineSolutionPackage}.
%
\item[-] Prefix operators, infix operators, and
punctuation symbols in the Axiom
language are displayed in the text like this:
\axiomOp{+}, \axiomSyntax{\$}, \axiomSyntax{+->}.
%
\item[-] Axiom expressions or expression fragments are displayed in
\texht{a mon\-o\-space typeface:}{this font:}
\axiom{inc(x) == x + 1}.
%
\item[-] For clarity of presentation, \TeX{} is often
used to format expressions\texht{: $g(x)=x^2+1.$}{.}
%
```

```
\item[-] Function names and Hyperdoc button names
are displayed in the text in
\texht{a bold typeface:}{this font:}
\axiomFun{factor}, \axiomFun{integrate},  {\bf Lighting}.
%
\item[-] Italics are used for emphasis and for words defined in the
glossary: \spadgloss{category}.
\enditems
\indent{0}

This book contains over 2500 examples of Axiom input and output.
All examples were run though Axiom and their output was
created in \texht{\TeX{}}{TeX} form for this book by the Axiom
\axiomType{TexFormat} package.
We have deleted system messages from the example output if those
messages are not important for the discussions in which the examples
appear.

\endscroll
\autobuttons
\end{page}
```

————

The Axiom Language

⇒ “notitle” (ugIntroArithmeticPage) 6 on page 1490
⇒ “notitle” (ugIntroPreviousPage) 6 on page 1492
⇒ “notitle” (ugIntroTypesPage) 6 on page 1494
⇒ “notitle” (ugIntroAssignPage) 6 on page 1497
⇒ “notitle” (ugIntroConversionPage) 6 on page 1503
⇒ “notitle” (ugIntroCallFunPage) 6 on page 1505
⇒ “notitle” (ugIntroMacrosPage) 6 on page 1508
⇒ “notitle” (ugIntroLongPage) 6 on page 1509
⇒ “notitle” (ugIntroCommentsPage) 6 on page 1510

— **ug01.ht** —

```
\begin{page}{ugIntroExpressionsPage}{1.3. The Axiom Language}
\beginscroll
%

The Axiom language is a rich language for performing interactive
computations and for building components of the Axiom library.  Here
we present only some basic aspects of the language that you need to
know for the rest of this chapter.  Our discussion here is
intentionally informal, with details unveiled on an ‘‘as needed’’
```

```
basis.  For more information on a particular construct, we suggest you
consult the index at the back of the book.

\beginmenu
    \menudownlink{{1.3.1. Arithmetic Expressions}}{ugIntroArithmeticPage}
    \menudownlink{{1.3.2. Previous Results}}{ugIntroPreviousPage}
    \menudownlink{{1.3.3. Some Types}}{ugIntroTypesPage}
    \menudownlink{
{1.3.4. Symbols, Variables, Assignments, and Declarations}}
{ugIntroAssignPage}
    \menudownlink{{1.3.5. Conversion}}{ugIntroConversionPage}
    \menudownlink{{1.3.6. Calling Functions}}{ugIntroCallFunPage}
    \menudownlink{{1.3.7. Some Predefined Macros}}{ugIntroMacrosPage}
    \menudownlink{{1.3.8. Long Lines}}{ugIntroLongPage}
    \menudownlink{{1.3.9. Comments}}{ugIntroCommentsPage}
\endmenu
\endscroll
\autobuttons
\end{page}
```

———-

Arithmetic Expressions

— ug01.ht —

```
\begin{page}{ugIntroArithmeticPage}{1.3.1. Arithmetic Expressions}
\beginscroll

For arithmetic expressions, use the \spadop{+} and \spadop{-}
\spadglossSee{operators}{operator} as in mathematics.  Use \spadop{*}
for multiplication, and \spadop{**} for exponentiation.  To create a
fraction, use \spadop{/}.  When an expression contains several
operators, those of highest \spadgloss{precedence} are evaluated
first.  For arithmetic operators, \spadop{**} has highest precedence,
\spadop{*} and \spadop{/} have the next highest precedence, and
\spadop{+} and \spadop{-} have the lowest precedence.

\xtc{
Axiom puts implicit parentheses around operations of higher
precedence, and groups those of equal precedence from left to right.
}{
\spadpaste{1 + 2 - 3 / 4 * 3 ** 2 - 1}
}
\xtc{
The above expression is equivalent to this.
}{
```

```
\spadpaste{((1 + 2) - ((3 / 4) * (3 ** 2))) - 1}
}
\xtc{
If an expression contains subexpressions enclosed in parentheses,
the parenthesized subexpressions are evaluated first (from left to
right, from inside out).
}{
\spadpaste{1 + 2 - 3/ (4 * 3 ** (2 - 1))}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugIntroArithmeticPagePatch1}
\begin{paste}{ugIntroArithmeticPageFull1}{ugIntroArithmeticPageEmpty1}
\pastebutton{ugIntroArithmeticPageFull1}{\hidepaste}
\tab{5}\spadcommand{1 + 2 - 3 / 4 * 3 ** 2 - 1}
\indentrel{3}\begin{verbatim}
          19
   (1)  -
           4
                                 Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroArithmeticPageEmpty1}
\begin{paste}{ugIntroArithmeticPageEmpty1}{ugIntroArithmeticPagePatch1}
\pastebutton{ugIntroArithmeticPageEmpty1}{\showpaste}
\tab{5}\spadcommand{1 + 2 - 3 / 4 * 3 ** 2 - 1}
\end{paste}\end{patch}

\begin{patch}{ugIntroArithmeticPagePatch2}
\begin{paste}{ugIntroArithmeticPageFull2}{ugIntroArithmeticPageEmpty2}
\pastebutton{ugIntroArithmeticPageFull2}{\hidepaste}
\tab{5}\spadcommand{((1 + 2) - ((3 / 4) * (3 ** 2))) - 1}
\indentrel{3}\begin{verbatim}
          19
   (2)  -
           4
                                 Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroArithmeticPageEmpty2}
\begin{paste}{ugIntroArithmeticPageEmpty2}{ugIntroArithmeticPagePatch2}
\pastebutton{ugIntroArithmeticPageEmpty2}{\showpaste}
\tab{5}\spadcommand{((1 + 2) - ((3 / 4) * (3 ** 2))) - 1}
\end{paste}\end{patch}
```

```
\begin{patch}{ugIntroArithmeticPagePatch3}
\begin{paste}{ugIntroArithmeticPageFull3}{ugIntroArithmeticPageEmpty3}
\pastebutton{ugIntroArithmeticPageFull3}{\hidepaste}
\tab{5}\spadcommand{1 + 2 - 3/ (4 * 3 ** (2 - 1))}
\indentrel{3}\begin{verbatim}
        11
   (3)
         4
                                 Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroArithmeticPageEmpty3}
\begin{paste}{ugIntroArithmeticPageEmpty3}{ugIntroArithmeticPagePatch3}
\pastebutton{ugIntroArithmeticPageEmpty3}{\showpaste}
\tab{5}\spadcommand{1 + 2 - 3/ (4 * 3 ** (2 - 1))}
\end{paste}\end{patch}
```

————-

Previous Results

— ug01.ht —

```
\begin{page}{ugIntroPreviousPage}{1.3.2. Previous Results}
\beginscroll

Use the percent sign (\axiomSyntax{\%}) to refer to the last
result.
Also, use \axiomSyntax{\%\%} to refer to previous results.
\axiom{\%\%(-1)} is equivalent to \axiomSyntax{\%},
\axiom{\%\%(-2)} returns the next to the last result, and so on.
\axiom{\%\%(1)} returns the result from step number 1,
\axiom{\%\%(2)} returns the result from step number 2, and so on.
\axiom{\%\%(0)} is not defined.

\xtc{
This is ten to the tenth power.
}{
\spadpaste{10 ** 10 \bound{prev}}
}
\xtc{
This is the last result minus one.
}{
\spadpaste{\% - 1 \free{prev}\bound{prev1}}
}
\xtc{
```

```
This is the last result.
}{
\spadpaste{\%\%(-1) \free{prev1}\bound{prev2}}
}
\xtc{
This is the result from step number 1.
}{
\spadpaste{\%\%(1) \free{prev2}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugIntroPreviousPagePatch1}
\begin{paste}{ugIntroPreviousPageFull1}{ugIntroPreviousPageEmpty1}
\pastebutton{ugIntroPreviousPageFull1}{\hidepaste}
\tab{5}\spadcommand{10 ** 10\bound{prev }}
\indentrel{3}\begin{verbatim}
   (1)  10000000000
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroPreviousPageEmpty1}
\begin{paste}{ugIntroPreviousPageEmpty1}{ugIntroPreviousPagePatch1}
\pastebutton{ugIntroPreviousPageEmpty1}{\showpaste}
\tab{5}\spadcommand{10 ** 10\bound{prev }}
\end{paste}\end{patch}

\begin{patch}{ugIntroPreviousPagePatch2}
\begin{paste}{ugIntroPreviousPageFull2}{ugIntroPreviousPageEmpty2}
\pastebutton{ugIntroPreviousPageFull2}{\hidepaste}
\tab{5}\spadcommand{\% - 1\free{prev }\bound{prev1 }}
\indentrel{3}\begin{verbatim}
   (2)  9999999999
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroPreviousPageEmpty2}
\begin{paste}{ugIntroPreviousPageEmpty2}{ugIntroPreviousPagePatch2}
\pastebutton{ugIntroPreviousPageEmpty2}{\showpaste}
\tab{5}\spadcommand{\% - 1\free{prev }\bound{prev1 }}
\end{paste}\end{patch}

\begin{patch}{ugIntroPreviousPagePatch3}
\begin{paste}{ugIntroPreviousPageFull3}{ugIntroPreviousPageEmpty3}
\pastebutton{ugIntroPreviousPageFull3}{\hidepaste}
\tab{5}\spadcommand{\%\%(-1)\free{prev1 }\bound{prev2 }}
```

```
\indentrel{3}\begin{verbatim}
   (3)  9999999999
                                    Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroPreviousPageEmpty3}
\begin{paste}{ugIntroPreviousPageEmpty3}{ugIntroPreviousPagePatch3}
\pastebutton{ugIntroPreviousPageEmpty3}{\showpaste}
\tab{5}\spadcommand{\%\%(-1)\free{prev1 }\bound{prev2 }}
\end{paste}\end{patch}

\begin{patch}{ugIntroPreviousPagePatch4}
\begin{paste}{ugIntroPreviousPageFull4}{ugIntroPreviousPageEmpty4}
\pastebutton{ugIntroPreviousPageFull4}{\hidepaste}
\tab{5}\spadcommand{\%\%(1)\free{prev2 }}
\indentrel{3}\begin{verbatim}
   (4)  10000000000
                                    Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroPreviousPageEmpty4}
\begin{paste}{ugIntroPreviousPageEmpty4}{ugIntroPreviousPagePatch4}
\pastebutton{ugIntroPreviousPageEmpty4}{\showpaste}
\tab{5}\spadcommand{\%\%(1)\free{prev2 }}
\end{paste}\end{patch}
```

———-

Some Types

⇒ “notitle” (ugTypesPage) 7 on page 1613

— **ug01.ht** —

```
\begin{page}{ugIntroTypesPage}{1.3.3. Some Types}
\beginscroll

Everything in Axiom has a type.
The type determines what operations you can perform on an object and
how the object can be used.
An entire chapter of this book (
\downlink{``Using Types and Modes''}{ugTypesPage}
in Chapter 2\ignore{ugTypes}) is dedicated to
the interactive use of types.
Several of the final chapters discuss how types are built and how
they are organized in the Axiom library.
```

```
\xtc{
Positive integers are given type \spadtype{PositiveInteger}.
}{
\spadpaste{8}
}
\xtc{
Negative ones are given type \spadtype{Integer}.
This fine distinction is helpful to the
Axiom interpreter.
}{
\spadpaste{-8}
}
\xtc{
Here a positive integer exponent gives a polynomial result.
}{
\spadpaste{x**8}
}
\xtc{
Here a negative integer exponent produces a fraction.
}{
\spadpaste{x**(-8)}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugIntroTypesPagePatch1}
\begin{paste}{ugIntroTypesPageFull1}{ugIntroTypesPageEmpty1}
\pastebutton{ugIntroTypesPageFull1}{\hidepaste}
\tab{5}\spadcommand{8}
\indentrel{3}\begin{verbatim}
   (1)  8
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroTypesPageEmpty1}
\begin{paste}{ugIntroTypesPageEmpty1}{ugIntroTypesPagePatch1}
\pastebutton{ugIntroTypesPageEmpty1}{\showpaste}
\tab{5}\spadcommand{8}
\end{paste}\end{patch}

\begin{patch}{ugIntroTypesPagePatch2}
\begin{paste}{ugIntroTypesPageFull2}{ugIntroTypesPageEmpty2}
\pastebutton{ugIntroTypesPageFull2}{\hidepaste}
\tab{5}\spadcommand{-8}
\indentrel{3}\begin{verbatim}
   (2)  - 8
```

```
                                               Type: Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroTypesPageEmpty2}
\begin{paste}{ugIntroTypesPageEmpty2}{ugIntroTypesPagePatch2}
\pastebutton{ugIntroTypesPageEmpty2}{\showpaste}
\tab{5}\spadcommand{-8}
\end{paste}\end{patch}

\begin{patch}{ugIntroTypesPagePatch3}
\begin{paste}{ugIntroTypesPageFull3}{ugIntroTypesPageEmpty3}
\pastebutton{ugIntroTypesPageFull3}{\hidepaste}
\tab{5}\spadcommand{x**8}
\indentrel{3}\begin{verbatim}
         8
   (3)  x
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroTypesPageEmpty3}
\begin{paste}{ugIntroTypesPageEmpty3}{ugIntroTypesPagePatch3}
\pastebutton{ugIntroTypesPageEmpty3}{\showpaste}
\tab{5}\spadcommand{x**8}
\end{paste}\end{patch}

\begin{patch}{ugIntroTypesPagePatch4}
\begin{paste}{ugIntroTypesPageFull4}{ugIntroTypesPageEmpty4}
\pastebutton{ugIntroTypesPageFull4}{\hidepaste}
\tab{5}\spadcommand{x**(-8)}
\indentrel{3}\begin{verbatim}
         1
   (4)
         8
        x
                      Type: Fraction Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroTypesPageEmpty4}
\begin{paste}{ugIntroTypesPageEmpty4}{ugIntroTypesPagePatch4}
\pastebutton{ugIntroTypesPageEmpty4}{\showpaste}
\tab{5}\spadcommand{x**(-8)}
\end{paste}\end{patch}
```

Symbols, Variables, Assignments, and Declarations

⇒ "notitle" (ugLangAssignPage) 9 on page 1745

— **ug01.ht** —

```
\begin{page}{ugIntroAssignPage}
{1.3.4. Symbols, Variables, Assignments, and Declarations}
\beginscroll

A \spadgloss{symbol} is a literal used for the input of things like
the ‘‘variables’’ in polynomials and power series.

\labelSpace{2pc}
\xtc{
We use the three symbols \axiom{x}, \axiom{y}, and \axiom{z} in
entering this polynomial.
}{
\spadpaste{(x - y*z)**2}
}

A symbol has a name beginning with an uppercase or lowercase
alphabetic character, \axiomSyntax{\%}, or \axiomSyntax{!}.
Successive characters (if any) can be any of the above, digits, or
\axiomSyntax{?}.  Case is distinguished: the symbol \axiom{points} is
different from the symbol \axiom{Points}.

A symbol can also be used in Axiom as a \spadgloss{variable}.  A
variable refers to a value.  To \spadglossSee{assign}{assignment} a
value to a variable, the operator \axiomSyntax{:=} is
used.\footnote{Axiom actually has two forms of assignment: {\it
immediate} assignment, as discussed here, and {\it delayed
assignment}. See \downlink{‘‘Immediate and Delayed Assignments’’}
{ugLangAssignPage} in Section 5.1\ignore{ugLangAssign} for details.}
A variable initially has no restrictions on the kinds of values to
which it can refer.

\xtc{
This assignment gives the value \axiom{4} (an integer) to
a variable named \axiom{x}.
}{
\spadpaste{x := 4}
}
\xtc{
This gives the value \axiom{z + 3/5} (a polynomial)  to \axiom{x}.
}{
\spadpaste{x := z + 3/5}
}
\xtc{
To restrict the types of objects that can be assigned to a variable,
```

```
use a \spadgloss{declaration}
}{
\spadpaste{y : Integer \bound{y}}
}
\xtc{
After a variable is declared to be of some type, only values
of that type can be assigned to that variable.
}{
\spadpaste{y := 89\bound{y1}\free{y}}
}
\xtc{
The declaration for \axiom{y} forces values assigned to \axiom{y} to
be converted to integer values.
}{
\spadpaste{y := sin \%pi}
}
\xtc{
If no such conversion is possible,
Axiom refuses to assign a value to \axiom{y}.
}{
\spadpaste{y := 2/3}
}
\xtc{
A type declaration can also be given together with an assignment.
The declaration can assist Axiom in choosing the correct
operations to apply.
}{
\spadpaste{f : Float := 2/3}
}

Any number of expressions can be given on input line.
Just separate them by semicolons.
Only the result of evaluating the last expression is displayed.

\xtc{
These two expressions have the same effect as
the previous single expression.
}{
\spadpaste{f : Float; f := 2/3 \bound{fff}}
}

The type of a symbol is either \axiomType{Symbol}
or \axiomType{Variable({\it name})} where {\it name} is the name
of the symbol.

\xtc{
By default, the interpreter
gives this symbol the type \axiomType{Variable(q)}.
}{
\spadpaste{q}
```

```
}
\xtc{
When multiple symbols are involved, \axiomType{Symbol} is used.
}{
\spadpaste{[q, r]}
}

\xtc{
What happens when you try to use a symbol that is the name of a variable?
}{
\spadpaste{f \free{fff}}
}
\xtc{
Use a single quote (\axiomSyntax{'}) before
the name to get the symbol.
}{
\spadpaste{'f}
}

Quoting a name creates a symbol by
preventing evaluation of the name as a variable.
Experience will teach you when you are most likely going to need to use
a quote.
We try to point out the location of such trouble spots.

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugIntroAssignPagePatch1}
\begin{paste}{ugIntroAssignPageFull1}{ugIntroAssignPageEmpty1}
\pastebutton{ugIntroAssignPageFull1}{\hidepaste}
\tab{5}\spadcommand{(x - y*z)**2}
\indentrel{3}\begin{verbatim}
         2 2            2
   (1)  y z  - 2x y z + x
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroAssignPageEmpty1}
\begin{paste}{ugIntroAssignPageEmpty1}{ugIntroAssignPagePatch1}
\pastebutton{ugIntroAssignPageEmpty1}{\showpaste}
\tab{5}\spadcommand{(x - y*z)**2}
\end{paste}\end{patch}

\begin{patch}{ugIntroAssignPagePatch2}
\begin{paste}{ugIntroAssignPageFull2}{ugIntroAssignPageEmpty2}
\pastebutton{ugIntroAssignPageFull2}{\hidepaste}
\tab{5}\spadcommand{x := 4}
```

```
\indentrel{3}\begin{verbatim}
   (2)  4
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroAssignPageEmpty2}
\begin{paste}{ugIntroAssignPageEmpty2}{ugIntroAssignPagePatch2}
\pastebutton{ugIntroAssignPageEmpty2}{\showpaste}
\tab{5}\spadcommand{x := 4}
\end{paste}\end{patch}

\begin{patch}{ugIntroAssignPagePatch3}
\begin{paste}{ugIntroAssignPageFull3}{ugIntroAssignPageEmpty3}
\pastebutton{ugIntroAssignPageFull3}{\hidepaste}
\tab{5}\spadcommand{x := z + 3/5}
\indentrel{3}\begin{verbatim}
            3
   (3)  z +
            5
                      Type: Polynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroAssignPageEmpty3}
\begin{paste}{ugIntroAssignPageEmpty3}{ugIntroAssignPagePatch3}
\pastebutton{ugIntroAssignPageEmpty3}{\showpaste}
\tab{5}\spadcommand{x := z + 3/5}
\end{paste}\end{patch}

\begin{patch}{ugIntroAssignPagePatch4}
\begin{paste}{ugIntroAssignPageFull4}{ugIntroAssignPageEmpty4}
\pastebutton{ugIntroAssignPageFull4}{\hidepaste}
\tab{5}\spadcommand{y : Integer\bound{y }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroAssignPageEmpty4}
\begin{paste}{ugIntroAssignPageEmpty4}{ugIntroAssignPagePatch4}
\pastebutton{ugIntroAssignPageEmpty4}{\showpaste}
\tab{5}\spadcommand{y : Integer\bound{y }}
\end{paste}\end{patch}

\begin{patch}{ugIntroAssignPagePatch5}
\begin{paste}{ugIntroAssignPageFull5}{ugIntroAssignPageEmpty5}
\pastebutton{ugIntroAssignPageFull5}{\hidepaste}
\tab{5}\spadcommand{y := 89\bound{y1 }\free{y }}
\indentrel{3}\begin{verbatim}
```

```
   (5)  89
                                          Type: Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroAssignPageEmpty5}
\begin{paste}{ugIntroAssignPageEmpty5}{ugIntroAssignPagePatch5}
\pastebutton{ugIntroAssignPageEmpty5}{\showpaste}
\tab{5}\spadcommand{y := 89\bound{y1 }\free{y }}
\end{paste}\end{patch}

\begin{patch}{ugIntroAssignPagePatch6}
\begin{paste}{ugIntroAssignPageFull6}{ugIntroAssignPageEmpty6}
\pastebutton{ugIntroAssignPageFull6}{\hidepaste}
\tab{5}\spadcommand{y := sin \%pi}
\indentrel{3}\begin{verbatim}
   (6)  0
                                          Type: Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroAssignPageEmpty6}
\begin{paste}{ugIntroAssignPageEmpty6}{ugIntroAssignPagePatch6}
\pastebutton{ugIntroAssignPageEmpty6}{\showpaste}
\tab{5}\spadcommand{y := sin \%pi}
\end{paste}\end{patch}

\begin{patch}{ugIntroAssignPagePatch7}
\begin{paste}{ugIntroAssignPageFull7}{ugIntroAssignPageEmpty7}
\pastebutton{ugIntroAssignPageFull7}{\hidepaste}
\tab{5}\spadcommand{y := 2/3}
\indentrel{3}\begin{verbatim}
   2

   3
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroAssignPageEmpty7}
\begin{paste}{ugIntroAssignPageEmpty7}{ugIntroAssignPagePatch7}
\pastebutton{ugIntroAssignPageEmpty7}{\showpaste}
\tab{5}\spadcommand{y := 2/3}
\end{paste}\end{patch}

\begin{patch}{ugIntroAssignPagePatch8}
\begin{paste}{ugIntroAssignPageFull8}{ugIntroAssignPageEmpty8}
\pastebutton{ugIntroAssignPageFull8}{\hidepaste}
\tab{5}\spadcommand{f : Float := 2/3}
\indentrel{3}\begin{verbatim}
   (7)  0.6666666666 6666666667
```

```
                                            Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroAssignPageEmpty8}
\begin{paste}{ugIntroAssignPageEmpty8}{ugIntroAssignPagePatch8}
\pastebutton{ugIntroAssignPageEmpty8}{\showpaste}
\tab{5}\spadcommand{f : Float := 2/3}
\end{paste}\end{patch}

\begin{patch}{ugIntroAssignPagePatch9}
\begin{paste}{ugIntroAssignPageFull9}{ugIntroAssignPageEmpty9}
\pastebutton{ugIntroAssignPageFull9}{\hidepaste}
\tab{5}\spadcommand{f : Float; f := 2/3\bound{fff }}
\indentrel{3}\begin{verbatim}
   (8)  0.6666666666 6666666667
                                            Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroAssignPageEmpty9}
\begin{paste}{ugIntroAssignPageEmpty9}{ugIntroAssignPagePatch9}
\pastebutton{ugIntroAssignPageEmpty9}{\showpaste}
\tab{5}\spadcommand{f : Float; f := 2/3\bound{fff }}
\end{paste}\end{patch}

\begin{patch}{ugIntroAssignPagePatch10}
\begin{paste}{ugIntroAssignPageFull10}{ugIntroAssignPageEmpty10}
\pastebutton{ugIntroAssignPageFull10}{\hidepaste}
\tab{5}\spadcommand{q}
\indentrel{3}\begin{verbatim}
   (9)  q
                                       Type: Variable q
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroAssignPageEmpty10}
\begin{paste}{ugIntroAssignPageEmpty10}{ugIntroAssignPagePatch10}
\pastebutton{ugIntroAssignPageEmpty10}{\showpaste}
\tab{5}\spadcommand{q}
\end{paste}\end{patch}

\begin{patch}{ugIntroAssignPagePatch11}
\begin{paste}{ugIntroAssignPageFull11}{ugIntroAssignPageEmpty11}
\pastebutton{ugIntroAssignPageFull11}{\hidepaste}
\tab{5}\spadcommand{[q, r]}
\indentrel{3}\begin{verbatim}
   (10)  [q,r]
                   Type: List OrderedVariableList [q,r]
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroAssignPageEmpty11}
\begin{paste}{ugIntroAssignPageEmpty11}{ugIntroAssignPagePatch11}
\pastebutton{ugIntroAssignPageEmpty11}{\showpaste}
\tab{5}\spadcommand{[q, r]}
\end{paste}\end{patch}

\begin{patch}{ugIntroAssignPagePatch12}
\begin{paste}{ugIntroAssignPageFull12}{ugIntroAssignPageEmpty12}
\pastebutton{ugIntroAssignPageFull12}{\hidepaste}
\tab{5}\spadcommand{f\free{fff }}
\indentrel{3}\begin{verbatim}
   (11)  0.6666666666 6666666667
                                            Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroAssignPageEmpty12}
\begin{paste}{ugIntroAssignPageEmpty12}{ugIntroAssignPagePatch12}
\pastebutton{ugIntroAssignPageEmpty12}{\showpaste}
\tab{5}\spadcommand{f\free{fff }}
\end{paste}\end{patch}

\begin{patch}{ugIntroAssignPagePatch13}
\begin{paste}{ugIntroAssignPageFull13}{ugIntroAssignPageEmpty13}
\pastebutton{ugIntroAssignPageFull13}{\hidepaste}
\tab{5}\spadcommand{'f}
\indentrel{3}\begin{verbatim}
   (12)  f
                                       Type: Variable f
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroAssignPageEmpty13}
\begin{paste}{ugIntroAssignPageEmpty13}{ugIntroAssignPagePatch13}
\pastebutton{ugIntroAssignPageEmpty13}{\showpaste}
\tab{5}\spadcommand{'f}
\end{paste}\end{patch}
```

———-

Conversion

⇒ “notitle” (ugTypesConvertPage) 7 on page 1671

— **ug01.ht** —

```
\begin{page}{ugIntroConversionPage}{1.3.5. Conversion}
\beginscroll

Objects of one type can usually be ``converted'' to objects of several
other types.  To \spadglossSee{convert}{conversion} an object to a new
type, use the \axiomSyntax{::} infix operator.\footnote{Conversion is
discussed in detail in
\downlink{``Conversion''}{ugTypesConvertPage} in Section
2.7\ignore{ugTypesConvert}.}  For example, to
display an object, it is necessary to convert the object to type
\spadtype{OutputForm}.

\xtc{
This produces a polynomial with rational number coefficients.
}{
\spadpaste{p := r**2 + 2/3 \bound{p}}
}
\xtc{
Create a quotient of polynomials with integer coefficients
by using \axiomSyntax{::}.
}{
\spadpaste{p :: Fraction Polynomial Integer \free{p}}
}

Some conversions can be performed automatically when
Axiom tries to evaluate your input.
Others conversions must be explicitly requested.

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugIntroConversionPagePatch1}
\begin{paste}{ugIntroConversionPageFull1}{ugIntroConversionPageEmpty1}
\pastebutton{ugIntroConversionPageFull1}{\hidepaste}
\tab{5}\spadcommand{p := r**2 + 2/3\bound{p }}
\indentrel{3}\begin{verbatim}
         2   2
   (1)  r  +
             3
                      Type: Polynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroConversionPageEmpty1}
\begin{paste}{ugIntroConversionPageEmpty1}{ugIntroConversionPagePatch1}
\pastebutton{ugIntroConversionPageEmpty1}{\showpaste}
\tab{5}\spadcommand{p := r**2 + 2/3\bound{p }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugIntroConversionPagePatch2}
\begin{paste}{ugIntroConversionPageFull2}{ugIntroConversionPageEmpty2}
\pastebutton{ugIntroConversionPageFull2}{\hidepaste}
\tab{5}\spadcommand{p :: Fraction Polynomial Integer\free{p }}
\indentrel{3}\begin{verbatim}
          2
        3r  + 2
   (2)
           3
                      Type: Fraction Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroConversionPageEmpty2}
\begin{paste}{ugIntroConversionPageEmpty2}{ugIntroConversionPagePatch2}
\pastebutton{ugIntroConversionPageEmpty2}{\showpaste}
\tab{5}\spadcommand{p :: Fraction Polynomial Integer\free{p }}
\end{paste}\end{patch}
```

———-

Calling Functions

— ug01.ht —

```
\begin{page}{ugIntroCallFunPage}{1.3.6. Calling Functions}
\beginscroll

As we saw earlier, when you want to add or subtract two values,
you place the arithmetic operator \spadop{+}
or \spadop{-} between the two
\spadglossSee{arguments}{argument} denoting the values.
To use most other Axiom operations, however, you use another syntax:
write the name
of the operation first, then an open parenthesis, then each of the
arguments separated by commas, and, finally, a closing parenthesis.
If the operation takes only one argument and the argument is a number
or a symbol, you can omit the parentheses.

\xtc{
This calls the operation \axiomFun{factor} with the single
integer argument \axiom{120}.
}{
\spadpaste{factor(120)}
}
\xtc{
This is a call to \axiomFun{divide} with the two integer arguments
```

```
\axiom{125} and \axiom{7}.
}{
\spadpaste{divide(125,7)}
}
\xtc{
This calls \axiomFun{quatern} with four floating-point arguments.
}{
\spadpaste{quatern(3.4,5.6,2.9,0.1)}
}
\xtc{
This is the same as \axiom{factorial(10)}.
}{
\spadpaste{factorial 10}
}

An operations that returns a \spadtype{Boolean} value (that is,
\spad{true} or \spad{false}) frequently has a name suffixed with
a question mark (``?'').  For example, the \spadfun{even?}
operation returns \spad{true} if its integer argument is an even
number, \spad{false} otherwise.

An operation that can be destructive on one or more arguments
usually has a name ending in a exclamation point (``!'').
This actually means that it is {\it allowed} to update its
arguments but it is not {\it required} to do so. For example,
the underlying representation of a collection type may not allow
the very last element to removed and so an empty object may be
returned instead. Therefore, it is important that you use the
object returned by the operation and not rely on a physical
change having occurred within the object. Usually, destructive
operations are provided for efficiency reasons.

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugIntroCallFunPagePatch1}
\begin{paste}{ugIntroCallFunPageFull1}{ugIntroCallFunPageEmpty1}
\pastebutton{ugIntroCallFunPageFull1}{\hidepaste}
\tab{5}\spadcommand{factor(120)}
\indentrel{3}\begin{verbatim}
         3
   (1)  2 3 5
                                Type: Factored Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCallFunPageEmpty1}
\begin{paste}{ugIntroCallFunPageEmpty1}{ugIntroCallFunPagePatch1}
\pastebutton{ugIntroCallFunPageEmpty1}{\showpaste}
```

```
\tab{5}\spadcommand{factor(120)}
\end{paste}\end{patch}

\begin{patch}{ugIntroCallFunPagePatch2}
\begin{paste}{ugIntroCallFunPageFull2}{ugIntroCallFunPageEmpty2}
\pastebutton{ugIntroCallFunPageFull2}{\hidepaste}
\tab{5}\spadcommand{divide(125,7)}
\indentrel{3}\begin{verbatim}
   (2)  [quotient= 17,remainder= 6]
     Type: Record(quotient: Integer,remainder: Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCallFunPageEmpty2}
\begin{paste}{ugIntroCallFunPageEmpty2}{ugIntroCallFunPagePatch2}
\pastebutton{ugIntroCallFunPageEmpty2}{\showpaste}
\tab{5}\spadcommand{divide(125,7)}
\end{paste}\end{patch}

\begin{patch}{ugIntroCallFunPagePatch3}
\begin{paste}{ugIntroCallFunPageFull3}{ugIntroCallFunPageEmpty3}
\pastebutton{ugIntroCallFunPageFull3}{\hidepaste}
\tab{5}\spadcommand{quatern(3.4,5.6,2.9,0.1)}
\indentrel{3}\begin{verbatim}
   (3)  3.4 + 5.6 i + 2.9 j + 0.1 k
                                  Type: Quaternion Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCallFunPageEmpty3}
\begin{paste}{ugIntroCallFunPageEmpty3}{ugIntroCallFunPagePatch3}
\pastebutton{ugIntroCallFunPageEmpty3}{\showpaste}
\tab{5}\spadcommand{quatern(3.4,5.6,2.9,0.1)}
\end{paste}\end{patch}

\begin{patch}{ugIntroCallFunPagePatch4}
\begin{paste}{ugIntroCallFunPageFull4}{ugIntroCallFunPageEmpty4}
\pastebutton{ugIntroCallFunPageFull4}{\hidepaste}
\tab{5}\spadcommand{factorial 10}
\indentrel{3}\begin{verbatim}
   (4)  3628800
                                   Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCallFunPageEmpty4}
\begin{paste}{ugIntroCallFunPageEmpty4}{ugIntroCallFunPagePatch4}
\pastebutton{ugIntroCallFunPageEmpty4}{\showpaste}
\tab{5}\spadcommand{factorial 10}
\end{paste}\end{patch}
```

———-

Some Predefined Macros

⇒ "notitle" (ugUserMacrosPage) 10 on page 1827
— **ug01.ht** —

```
\begin{page}{ugIntroMacrosPage}{1.3.7. Some Predefined Macros}
\beginscroll

Axiom provides several \spadglossSee{macros}{macro} for your
convenience.\footnote{See \downlink{``Macros''}{ugUserMacrosPage} in
Section 6.2\ignore{ugUserMacros} for a discussion on how to write your
own macros.}  Macros are names (or forms) that expand to larger
expressions for commonly used values.

\texht{
\centerline{{\begin{tabular}{ll}}}
\centerline{{\spadgloss{\%i} &  The square root of -1. }}
\centerline{{\spadgloss{\%e} &  The base of the natural logarithm. }}
\centerline{{\spadgloss{\%pi}            &  $\pi$. }}
\centerline{{\spadgloss{\%infinity}      &  $\infty$. }}
\centerline{{\spadgloss{\%plusInfinity}  &  $+\infty$. }}
\centerline{{\spadgloss{\%minusInfinity} &  $-\infty$.}}
\centerline{{\end{tabular}}}
}{
\indent{0}
\beginitems
\item[\axiomSyntax{\%i}]     \tab{17} The square root of -1.
\item[\axiomSyntax{\%e}]     \tab{17} The base of the natural logarithm.
\item[\axiomSyntax{\%pi}]            \tab{17} Pi.
\item[\axiomSyntax{\%infinity}]      \tab{17} Infinity.
\item[\axiomSyntax{\%plusInfinity}]  \tab{17} Plus infinity.
\item[\axiomSyntax{\%minusInfinity}] \tab{17} Minus infinity.
\enditems
\indent{0}
}

%To display all the macros (along with anything you have
%defined in the workspace), issue the system command \spadsys{)display all}.

\endscroll
\autobuttons
\end{page}
```

———-

Long Lines

⇒ "notitle" (ugInOutInPage) 9 on page 1722
⇒ "notitle" (ugLangBlocksPage) 9 on page 1753

— **ug01.ht** —

```
\begin{page}{ugIntroLongPage}{1.3.8. Long Lines}
\beginscroll

When you enter Axiom expressions from your keyboard, there
will be times when they are too long to fit on one line.
Axiom does not care how long your lines are, so you can let
them continue from the right margin to the left side of the
next line.

Alternatively, you may want to enter several shorter lines and
have Axiom glue them together.
To get this glue, put an underscore (\_) at the end of
each line you wish to continue.
\begin{verbatim}
2_
+_
3
\end{verbatim}
is the same as if you had entered
\begin{verbatim}
2+3
\end{verbatim}

If you are putting your Axiom statements in an input file
(see \downlink{``Input Files''}{ugInOutInPage}
in Section 4.1\ignore{ugInOutIn}),
you can use indentation to indicate the structure of your program.
(see \downlink{``Blocks''}{ugLangBlocksPage}
in Section 5.2\ignore{ugLangBlocks}).

\endscroll
\autobuttons
\end{page}
```

———-

Comments

— **ug01.ht** —

```
\begin{page}{ugIntroCommentsPage}{1.3.9. Comments}
\beginscroll

Comment statements begin with two consecutive hyphens or two
consecutive plus signs and continue until the end of the line.

\xtc{
The comment beginning with {\tt --} is ignored by Axiom.
}{
\spadpaste{2 + 3   -- this is rather simple, no?}
}

There is no way to write long multi-line comments
other than starting each line with \axiomSyntax{--} or
\axiomSyntax{++}.

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugIntroCommentsPagePatch1}
\begin{paste}{ugIntroCommentsPageFull1}{ugIntroCommentsPageEmpty1}
\pastebutton{ugIntroCommentsPageFull1}{\hidepaste}
\tab{5}\spadcommand{2 + 3 -- this is rather simple, no?}
\indentrel{3}\begin{verbatim}
   (1)  5
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCommentsPageEmpty1}
\begin{paste}{ugIntroCommentsPageEmpty1}{ugIntroCommentsPagePatch1}
\pastebutton{ugIntroCommentsPageEmpty1}{\showpaste}
\tab{5}\spadcommand{2 + 3 -- this is rather simple, no?}
\end{paste}\end{patch}
```

————

Graphics

⇒ "notitle" (ugProblemNumericPage) 12 on page 2079

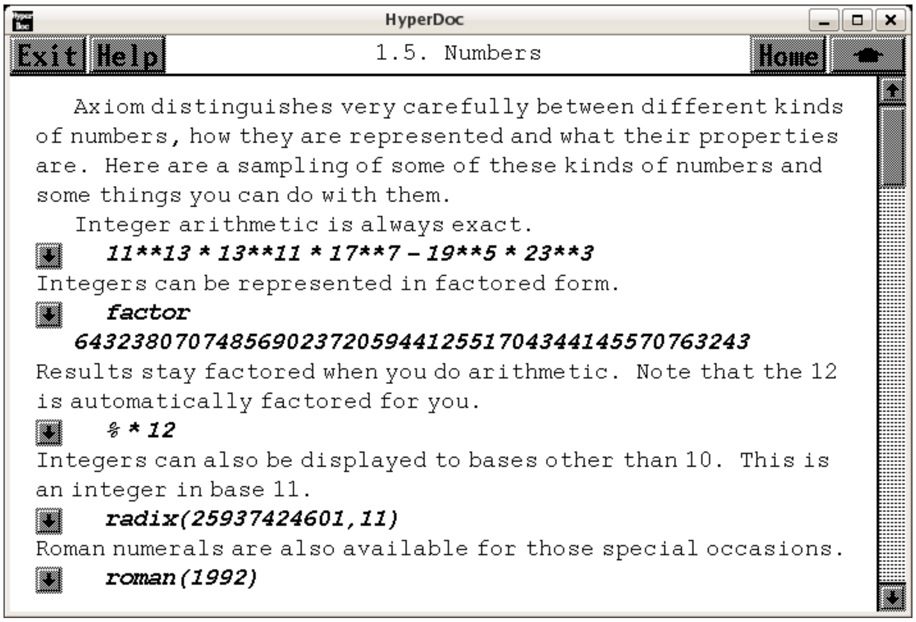
⇒ "notitle" (ugAppGraphicsPage) 20 on page 2589
⇒ "notitle" (ugGraphPage) 11 on page 1967

— **ug01.ht** —

```
\begin{page}{ugIntroGraphicsPage}{1.4. Graphics}
\beginscroll
%

Axiom has a two- and three-dimensional drawing and rendering package
that allows you to draw, shade, color, rotate, translate, map, clip,
scale and combine graphic output of Axiom computations.  The graphics
interface is capable of plotting functions of one or more variables
and plotting parametric surfaces.  Once the graphics figure appears in
a window, move your mouse to the window and click.  A control panel
appears immediately and allows you to interactively transform the
object.

\psXtc{
This is an example of Axiom's two-dimensional plotting.  From the 2D
Control Panel you can rescale the plot, turn axes and units on and off
and save the image, among other things.  This PostScript image was
produced by clicking on the
\texht{\fbox{\bf PS}}{{\bf PS}} 2D Control Panel button.
}{
\graphpaste{draw(cos(5*t/8), t=0..16*\%pi, coordinates==polar)}
}{
\epsffile[72 72 300 300]{../ps/rose-1.ps}
}

\psXtc{
This is an example of Axiom's three-dimensional plotting.  It is a
monochrome graph of the complex arctangent function.  The image
displayed was rotated and had the ``shade'' and ``outline'' display
options set from the 3D Control Panel.  The PostScript output was
produced by clicking on the \texht{\fbox{\bf save}}{{\bf save}} 3D
Control Panel button and then clicking on the \texht{\fbox{\bf
PS}}{{\bf PS}} button.  See
\downlink{``Numeric Functions''}{ugProblemNumericPage} in Section
8.1\ignore{ugProblemNumeric} for more details and
examples of Axiom's numeric and graphics capabilities.
}{
\graphpaste{draw((x,y) +-> real atan complex(x,y), -\%pi..\%pi,
-\%pi..\%pi, colorFunction == (x,y) +-> argument atan complex(x,y))}
}{
\epsffile[72 72 285 285]{../ps/atan-1.ps}
}

An exhibit of \Gallery{} is given in the center section of this book.
For a description of the commands and programs that produced these
figures, see
```

```
\downlink{``Programs for Axiom Images''}{ugAppGraphicsPage}
in Appendix G.\ignore{ugAppGraphics}.
PostScript output is available so that Axiom images can be
printed.\footnote{PostScript is a trademark of Adobe Systems
Incorporated, registered in the United States.}  See
\downlink{``Graphics''}{ugGraphPage} in Chapter 7
\ignore{ugGraph} for more examples and details about
using Axiom's graphics facilities.

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugIntroGraphicsPagePatch1}
\begin{paste}{ugIntroGraphicsPageFull1}{ugIntroGraphicsPageEmpty1}
\pastebutton{ugIntroGraphicsPageFull1}{\hidepaste}
\tab{5}\spadgraph{draw(cos(5*t/8), t=0..16*\%pi, coordinates==polar)}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/ugintrographicspage1.view/image}}{vie
\end{paste}\end{patch}

\begin{patch}{ugIntroGraphicsPageEmpty1}
\begin{paste}{ugIntroGraphicsPageEmpty1}{ugIntroGraphicsPagePatch1}
\pastebutton{ugIntroGraphicsPageEmpty1}{\showpaste}
\tab{5}\spadgraph{draw(cos(5*t/8), t=0..16*\%pi, coordinates==polar)}
\end{paste}\end{patch}

\begin{patch}{ugIntroGraphicsPagePatch2}
\begin{paste}{ugIntroGraphicsPageFull2}{ugIntroGraphicsPageEmpty2}
\pastebutton{ugIntroGraphicsPageFull2}{\hidepaste}
\tab{5}\spadgraph{draw((x,y) +-> real atan complex(x,y), -\%pi..\%pi, -\%pi..\%pi, colorFunction
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/ugintrographicspage2.view/image}}{vie
\end{paste}\end{patch}

\begin{patch}{ugIntroGraphicsPageEmpty2}
\begin{paste}{ugIntroGraphicsPageEmpty2}{ugIntroGraphicsPagePatch2}
\pastebutton{ugIntroGraphicsPageEmpty2}{\showpaste}
\tab{5}\spadgraph{draw((x,y) +-> real atan complex(x,y), -\%pi..\%pi, -\%pi..\%pi, colorFunction
\end{paste}\end{patch}
```

Numbers



⇐ "Integer" (IntegerXmpPage) 3.55 on page 696
⇒ "Float" (FloatXmpPage) 3.41 on page 487
⇒ "DoubleFloat" (DoubleFloatXmpPage) 3.23 on page 369
⇒ "DecimalExpansion" (DecimalExpansionXmpPage) 3.21 on page 348
⇒ "ContinuedFraction" (ContinuedFractionXmpPage) 3.17 on page 262
⇒ "PartialFraction" (PartialFractionXmpPage) 3.86 on page 1088
⇒ "RadixExpansion" (RadixExpansionXmpPage) 3.90 on page 1140
⇒ "Modular Arithmetic and Prime Fields" (ugxProblemFinitePrimePage) 12 on page 2237

— ug01.ht —

```
\begin{page}{ugIntroNumbersPage}{1.5. Numbers}
\beginscroll
%

Axiom distinguishes very carefully between different kinds
of numbers, how they are represented and what their properties
are.
Here are a sampling of some of these kinds of numbers and some
things you can do with them.

\xtc{
Integer arithmetic is always exact.
}{
\spadpaste{11**13 * 13**11 * 17**7 - 19**5 * 23**3}
}
```

```
\xtc{
Integers can be represented in factored form.
}{
\spadpaste{factor 643238070748569023720594412551704344145570763243
\bound{ex1}}
}
\xtc{
Results stay factored when you do arithmetic.
Note that the \axiom{12} is automatically factored for you.
}{
\spadpaste{\% * 12 \free{ex1}}
}
\xtc{
Integers can also be displayed to bases other than 10.
This is an integer in base 11.
}{
\spadpaste{radix(25937424601,11)}
}
\xtc{
Roman numerals are also available for those special occasions.
}{
\spadpaste{roman(1992)}
}
\xtc{
Rational number arithmetic is also exact.
}{
\spadpaste{r := 10 + 9/2 + 8/3 + 7/4 + 6/5 + 5/6 + 4/7 + 3/8 + 2/9
\bound{r}}
}
\xtc{
To factor fractions, you have to
map \axiomFun{factor} onto the numerator and denominator.
}{
\spadpaste{map(factor,r) \free{r}}
}
\xtc{
Type \spadtype{SingleInteger} refers to machine word-length
integers.
In English, this expression means ‘‘\axiom{11} as a small
integer’’.
}{
\spadpaste{11@SingleInteger}
}
\xtc{
Machine double-precision floating-point numbers are also
available for numeric and graphical
applications.
}{
\spadpaste{123.21@DoubleFloat}
}
```

```
The normal floating-point type in Axiom, \spadtype{Float},
is a software implementation of floating-point numbers in which
the exponent and the mantissa may have any number of
digits.\footnote{See
\downlink{`Float'}{FloatXmpPage}\ignore{Float} and
\downlink{`DoubleFloat'}{DoubleFloatXmpPage}\ignore{DoubleFloat} for
additional information on floating-point types.}
The types \spadtype{Complex(Float)} and
\spadtype{Complex(DoubleFloat)} are the corresponding software
implementations of complex floating-point numbers.

\xtc{
This is a floating-point approximation to about twenty digits.
The \axiomSyntax{::}
is used here to change from one kind of object
(here, a rational number) to another (a floating-point number).
}{
\spadpaste{r :: Float \free{r}}
}
\xtc{
Use \spadfunFrom{digits}{Float} to change the number of digits in
the representation.
This operation returns the previous value so you can reset it
later.
}{
\spadpaste{digits(22) \bound{fewerdigits}}
}
\xtc{
To \axiom{22} digits of precision, the number
\texht{$e^{\pi {\sqrt {163.0}}}$}{\axiom{exp(\%pi * sqrt 163.0)}}
appears to be an integer.
}{
\spadpaste{exp(\%pi * sqrt 163.0) \free{fewerdigits}}
}
\xtc{
Increase the precision to forty digits and try again.
}{
\spadpaste{digits(40);  exp(\%pi * sqrt 163.0) \free{moredigits}}
}
\xtc{
Here are complex numbers with rational numbers as real and
imaginary parts.
}{
\spadpaste{(2/3 + \%i)**3 \bound{gaussint}}
}
\xtc{
The standard operations on complex numbers are available.
}{
\spadpaste{conjugate \% \free{gaussint}}
```

```
}
\xtc{
You can factor complex integers.
}{
\spadpaste{factor(89 - 23 * \%i)}
}
\xtc{
Complex numbers with floating point parts are also available.
}{
\spadpaste{exp(\%pi/4.0 * \%i)}
}
\xtc{
Every rational number has an exact representation as a
repeating decimal expansion
(see \downlink{`DecimalExpansion'}{DecimalExpansionXmpPage}
\ignore{DecimalExpansion}).
}{
\spadpaste{decimal(1/352)}
}
\xtc{
A rational number can also be expressed as a continued fraction (see
\downlink{`ContinuedFraction'}{ContinuedFractionXmpPage}
\ignore{ContinuedFraction}).
}{
\spadpaste{continuedFraction(6543/210)}
}
\xtc{
Also, partial fractions can be used and can be displayed in a
compact \ldots
}{
\spadpaste{partialFraction(1,factorial(10)) \bound{partfrac}}
}
\xtc{
or expanded format (see
\downlink{`PartialFraction'}{PartialFractionXmpPage}
\ignore{PartialFraction}).
}{
\spadpaste{padicFraction(\%) \free{partfrac}}
}
\xtc{
Like integers, bases (radices) other than ten can be used for rational
numbers (see
\downlink{`RadixExpansion'}{RadixExpansionXmpPage}
\ignore{RadixExpansion}).
Here we use base eight.
}{
\spadpaste{radix(4/7, 8)\bound{rad}}
}
\xtc{
Of course, there are complex versions of these as well.
```

```
Axiom decides to make the result a complex rational number.
}{
\spadpaste{\% + 2/3*\%i\free{rad}}
}
\xtc{
You can also use Axiom to manipulate fractional powers.
}{
\spadpaste{(5 + sqrt 63 + sqrt 847)**(1/3)}
}
\xtc{
You can also compute with integers modulo a prime.
}{
\spadpaste{x : PrimeField 7 := 5 \bound{x}}
}
\xtc{
Arithmetic is then done modulo \mathOrSpad{7}.
}{
\spadpaste{x**3 \free{x}}
}
\xtc{
Since \mathOrSpad{7} is prime, you can invert nonzero values.
}{
\spadpaste{1/x \free{x}}
}
\xtc{
You can also compute modulo an integer that is not a prime.
}{
\spadpaste{y : IntegerMod 6 := 5 \bound{y}}
}
\xtc{
All of the usual arithmetic operations are available.
}{
\spadpaste{y**3 \free{y}}
}
\xtc{
Inversion is not available if the modulus is not a prime
number.
Modular arithmetic and prime fields are discussed in
\downlink{``Modular Arithmetic and Prime Fields''}
{ugxProblemFinitePrimePage}
in Section 8.11.1\ignore{ugxProblemFinitePrime}.
}{
\spadpaste{1/y \free{y}}
}
\xtc{
This defines \axiom{a} to be an algebraic number, that is,
a root of a polynomial equation.
}{
\spadpaste{a := rootOf(a**5 + a**3 + a**2 + 3,a) \bound{a}}
}
```

```
\xtc{
Computations with \axiom{a} are reduced according
to the polynomial equation.
}{
\spadpaste{(a + 1)**10\free{a}}
}
\xtc{
Define \axiom{b} to be an algebraic number involving \axiom{a}.
}{
\spadpaste{b := rootOf(b**4 + a,b) \bound{b}\free{a}}
}
\xtc{
Do some arithmetic.
}{
\spadpaste{2/(b - 1) \free{b}\bound{check}}
}
\xtc{
To expand and simplify this, call \axiomFun{ratDenom}
to rationalize the denominator.
}{
\spadpaste{ratDenom(\%) \free{check}\bound{check1}}
}
\xtc{
If we do this, we should get \axiom{b}.
}{
\spadpaste{2/\%+1 \free{check1}\bound{check2}}
}
\xtc{
But we need to rationalize the denominator again.
}{
\spadpaste{ratDenom(\%) \free{check2}}
}
\xtc{
Types \spadtype{Quaternion} and \spadtype{Octonion} are also available.
Multiplication of quaternions is non-commutative, as expected.
}{
\spadpaste{q:=quatern(1,2,3,4)*quatern(5,6,7,8) -
quatern(5,6,7,8)*quatern(1,2,3,4)}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugIntroNumbersPagePatch1}
\begin{paste}{ugIntroNumbersPageFull1}{ugIntroNumbersPageEmpty1}
\pastebutton{ugIntroNumbersPageFull1}{\hidepaste}
\tab{5}\spadcommand{11**13 * 13**11 * 17**7 - 19**5 * 23**3}
\indentrel{3}\begin{verbatim}
   (1)  25387751112538918594666224484237298
```

```
                                    Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPageEmpty1}
\begin{paste}{ugIntroNumbersPageEmpty1}{ugIntroNumbersPagePatch1}
\pastebutton{ugIntroNumbersPageEmpty1}{\showpaste}
\tab{5}\spadcommand{11**13 * 13**11 * 17**7 - 19**5 * 23**3}
\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPagePatch2}
\begin{paste}{ugIntroNumbersPageFull2}{ugIntroNumbersPageEmpty2}
\pastebutton{ugIntroNumbersPageFull2}{\hidepaste}
\tab{5}\spadcommand{factor 643238070748569023720594412551704344145570763243\bound{ex1 }}
\indentrel{3}\begin{verbatim}
          13  11  7  5  3  2
   (2)  11  13  17 19 23 29
                                 Type: Factored Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPageEmpty2}
\begin{paste}{ugIntroNumbersPageEmpty2}{ugIntroNumbersPagePatch2}
\pastebutton{ugIntroNumbersPageEmpty2}{\showpaste}
\tab{5}\spadcommand{factor 643238070748569023720594412551704344145570763243\bound{ex1 }}
\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPagePatch3}
\begin{paste}{ugIntroNumbersPageFull3}{ugIntroNumbersPageEmpty3}
\pastebutton{ugIntroNumbersPageFull3}{\hidepaste}
\tab{5}\spadcommand{\% * 12\free{ex1 }}
\indentrel{3}\begin{verbatim}
         2    13  11  7  5  3  2
   (3)  2 3 11  13  17 19 23 29
                                 Type: Factored Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPageEmpty3}
\begin{paste}{ugIntroNumbersPageEmpty3}{ugIntroNumbersPagePatch3}
\pastebutton{ugIntroNumbersPageEmpty3}{\showpaste}
\tab{5}\spadcommand{\% * 12\free{ex1 }}
\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPagePatch4}
\begin{paste}{ugIntroNumbersPageFull4}{ugIntroNumbersPageEmpty4}
\pastebutton{ugIntroNumbersPageFull4}{\hidepaste}
\tab{5}\spadcommand{radix(25937424601,11)}
\indentrel{3}\begin{verbatim}
   (4)  10000000000
```

```
                                Type: RadixExpansion 11
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPageEmpty4}
\begin{paste}{ugIntroNumbersPageEmpty4}{ugIntroNumbersPagePatch4}
\pastebutton{ugIntroNumbersPageEmpty4}{\showpaste}
\tab{5}\spadcommand{radix(25937424601,11)}
\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPagePatch5}
\begin{paste}{ugIntroNumbersPageFull5}{ugIntroNumbersPageEmpty5}
\pastebutton{ugIntroNumbersPageFull5}{\hidepaste}
\tab{5}\spadcommand{roman(1992)}
\indentrel{3}\begin{verbatim}
   (5)  MCMXCII
                                     Type: RomanNumeral
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPageEmpty5}
\begin{paste}{ugIntroNumbersPageEmpty5}{ugIntroNumbersPagePatch5}
\pastebutton{ugIntroNumbersPageEmpty5}{\showpaste}
\tab{5}\spadcommand{roman(1992)}
\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPagePatch6}
\begin{paste}{ugIntroNumbersPageFull6}{ugIntroNumbersPageEmpty6}
\pastebutton{ugIntroNumbersPageFull6}{\hidepaste}
\tab{5}\spadcommand{r := 10 + 9/2 + 8/3 + 7/4 + 6/5 + 5/6 + 4/7 + 3/8 + 2/9\bound{r }}
\indentrel{3}\begin{verbatim}
        55739
   (6)
         2520
                                 Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPageEmpty6}
\begin{paste}{ugIntroNumbersPageEmpty6}{ugIntroNumbersPagePatch6}
\pastebutton{ugIntroNumbersPageEmpty6}{\showpaste}
\tab{5}\spadcommand{r := 10 + 9/2 + 8/3 + 7/4 + 6/5 + 5/6 + 4/7 + 3/8 + 2/9\bound{r }}
\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPagePatch7}
\begin{paste}{ugIntroNumbersPageFull7}{ugIntroNumbersPageEmpty7}
\pastebutton{ugIntroNumbersPageFull7}{\hidepaste}
\tab{5}\spadcommand{map(factor,r)\free{r }}
\indentrel{3}\begin{verbatim}
        139 401
```

```
   (7)
         3 2
        2 3 5 7
                        Type: Fraction Factored Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPageEmpty7}
\begin{paste}{ugIntroNumbersPageEmpty7}{ugIntroNumbersPagePatch7}
\pastebutton{ugIntroNumbersPageEmpty7}{\showpaste}
\tab{5}\spadcommand{map(factor,r)\free{r }}
\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPagePatch8}
\begin{paste}{ugIntroNumbersPageFull8}{ugIntroNumbersPageEmpty8}
\pastebutton{ugIntroNumbersPageFull8}{\hidepaste}
\tab{5}\spadcommand{11@SingleInteger}
\indentrel{3}\begin{verbatim}
   (8)  11
                                    Type: SingleInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPageEmpty8}
\begin{paste}{ugIntroNumbersPageEmpty8}{ugIntroNumbersPagePatch8}
\pastebutton{ugIntroNumbersPageEmpty8}{\showpaste}
\tab{5}\spadcommand{11@SingleInteger}
\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPagePatch9}
\begin{paste}{ugIntroNumbersPageFull9}{ugIntroNumbersPageEmpty9}
\pastebutton{ugIntroNumbersPageFull9}{\hidepaste}
\tab{5}\spadcommand{123.21@DoubleFloat}
\indentrel{3}\begin{verbatim}
   (9)  123.21000000000001
                                      Type: DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPageEmpty9}
\begin{paste}{ugIntroNumbersPageEmpty9}{ugIntroNumbersPagePatch9}
\pastebutton{ugIntroNumbersPageEmpty9}{\showpaste}
\tab{5}\spadcommand{123.21@DoubleFloat}
\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPagePatch10}
\begin{paste}{ugIntroNumbersPageFull10}{ugIntroNumbersPageEmpty10}
\pastebutton{ugIntroNumbersPageFull10}{\hidepaste}
\tab{5}\spadcommand{r :: Float\free{r }}
\indentrel{3}\begin{verbatim}
```

```
   (10)  22.1186507936 50793651
                                             Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPageEmpty10}
\begin{paste}{ugIntroNumbersPageEmpty10}{ugIntroNumbersPagePatch10}
\pastebutton{ugIntroNumbersPageEmpty10}{\showpaste}
\tab{5}\spadcommand{r :: Float\free{r }}
\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPagePatch11}
\begin{paste}{ugIntroNumbersPageFull11}{ugIntroNumbersPageEmpty11}
\pastebutton{ugIntroNumbersPageFull11}{\hidepaste}
\tab{5}\spadcommand{digits(22)\bound{fewerdigits }}
\indentrel{3}\begin{verbatim}
   (11)  20
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPageEmpty11}
\begin{paste}{ugIntroNumbersPageEmpty11}{ugIntroNumbersPagePatch11}
\pastebutton{ugIntroNumbersPageEmpty11}{\showpaste}
\tab{5}\spadcommand{digits(22)\bound{fewerdigits }}
\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPagePatch12}
\begin{paste}{ugIntroNumbersPageFull12}{ugIntroNumbersPageEmpty12}
\pastebutton{ugIntroNumbersPageFull12}{\hidepaste}
\tab{5}\spadcommand{exp(\%pi * sqrt 163.0)\free{fewerdigits }}
\indentrel{3}\begin{verbatim}
   (12)  26253741 2640768744.0
                                             Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPageEmpty12}
\begin{paste}{ugIntroNumbersPageEmpty12}{ugIntroNumbersPagePatch12}
\pastebutton{ugIntroNumbersPageEmpty12}{\showpaste}
\tab{5}\spadcommand{exp(\%pi * sqrt 163.0)\free{fewerdigits }}
\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPagePatch13}
\begin{paste}{ugIntroNumbersPageFull13}{ugIntroNumbersPageEmpty13}
\pastebutton{ugIntroNumbersPageFull13}{\hidepaste}
\tab{5}\spadcommand{digits(40); exp(\%pi * sqrt 163.0)\free{moredigits }}
\indentrel{3}\begin{verbatim}
   (13)  26253741 2640768743.9999999999 9925007259 76
                                             Type: Float
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPageEmpty13}
\begin{paste}{ugIntroNumbersPageEmpty13}{ugIntroNumbersPagePatch13}
\pastebutton{ugIntroNumbersPageEmpty13}{\showpaste}
\tab{5}\spadcommand{digits(40); exp(\%pi * sqrt 163.0)\free{moredigits }}
\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPagePatch14}
\begin{paste}{ugIntroNumbersPageFull14}{ugIntroNumbersPageEmpty14}
\pastebutton{ugIntroNumbersPageFull14}{\hidepaste}
\tab{5}\spadcommand{(2/3 + \%i)**3\bound{gaussint }}
\indentrel{3}\begin{verbatim}
          46   1
   (14)  -  +  %i
          27   3
                         Type: Complex Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPageEmpty14}
\begin{paste}{ugIntroNumbersPageEmpty14}{ugIntroNumbersPagePatch14}
\pastebutton{ugIntroNumbersPageEmpty14}{\showpaste}
\tab{5}\spadcommand{(2/3 + \%i)**3\bound{gaussint }}
\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPagePatch15}
\begin{paste}{ugIntroNumbersPageFull15}{ugIntroNumbersPageEmpty15}
\pastebutton{ugIntroNumbersPageFull15}{\hidepaste}
\tab{5}\spadcommand{conjugate \%\free{gaussint }}
\indentrel{3}\begin{verbatim}
          46   1
   (15)  -  -  %i
          27   3
                         Type: Complex Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPageEmpty15}
\begin{paste}{ugIntroNumbersPageEmpty15}{ugIntroNumbersPagePatch15}
\pastebutton{ugIntroNumbersPageEmpty15}{\showpaste}
\tab{5}\spadcommand{conjugate \%\free{gaussint }}
\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPagePatch16}
\begin{paste}{ugIntroNumbersPageFull16}{ugIntroNumbersPageEmpty16}
\pastebutton{ugIntroNumbersPageFull16}{\hidepaste}
\tab{5}\spadcommand{factor(89 - 23 * \%i)}
\indentrel{3}\begin{verbatim}
```

```
                           2         2
   (16)  - (1 + %i)(2 + %i) (3 + 2%i)
                         Type: Factored Complex Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPageEmpty16}
\begin{paste}{ugIntroNumbersPageEmpty16}{ugIntroNumbersPagePatch16}
\pastebutton{ugIntroNumbersPageEmpty16}{\showpaste}
\tab{5}\spadcommand{factor(89 - 23 * \%i)}
\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPagePatch17}
\begin{paste}{ugIntroNumbersPageFull17}{ugIntroNumbersPageEmpty17}
\pastebutton{ugIntroNumbersPageFull17}{\hidepaste}
\tab{5}\spadcommand{exp(\%pi/4.0 * \%i)}
\indentrel{3}\begin{verbatim}
   (17)
     0.7071067811 8654752440 0844362104 8490392849
   +
     0.7071067811 8654752440 0844362104 8490392848 %i
                                    Type: Complex Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPageEmpty17}
\begin{paste}{ugIntroNumbersPageEmpty17}{ugIntroNumbersPagePatch17}
\pastebutton{ugIntroNumbersPageEmpty17}{\showpaste}
\tab{5}\spadcommand{exp(\%pi/4.0 * \%i)}
\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPagePatch18}
\begin{paste}{ugIntroNumbersPageFull18}{ugIntroNumbersPageEmpty18}
\pastebutton{ugIntroNumbersPageFull18}{\hidepaste}
\tab{5}\spadcommand{decimal(1/352)}
\indentrel{3}\begin{verbatim}
                  --
   (18)  0.0028409
                                 Type: DecimalExpansion
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPageEmpty18}
\begin{paste}{ugIntroNumbersPageEmpty18}{ugIntroNumbersPagePatch18}
\pastebutton{ugIntroNumbersPageEmpty18}{\showpaste}
\tab{5}\spadcommand{decimal(1/352)}
\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPagePatch19}
\begin{paste}{ugIntroNumbersPageFull19}{ugIntroNumbersPageEmpty19}
```

```
\pastebutton{ugIntroNumbersPageFull19}{\hidepaste}
\tab{5}\spadcommand{continuedFraction(6543/210)}
\indentrel{3}\begin{verbatim}
                1      1      1      1
   (19)  31 +  +  +  +
               6      2      1      3
                         Type: ContinuedFraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPageEmpty19}
\begin{paste}{ugIntroNumbersPageEmpty19}{ugIntroNumbersPagePatch19}
\pastebutton{ugIntroNumbersPageEmpty19}{\showpaste}
\tab{5}\spadcommand{continuedFraction(6543/210)}
\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPagePatch20}
\begin{paste}{ugIntroNumbersPageFull20}{ugIntroNumbersPageEmpty20}
\pastebutton{ugIntroNumbersPageFull20}{\hidepaste}
\tab{5}\spadcommand{partialFraction(1,factorial(10))\bound{partfrac }}
\indentrel{3}\begin{verbatim}
         159   23   12   1
   (20)   -  -  +
           8    4    2   7
          2    3    5
                           Type: PartialFraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPageEmpty20}
\begin{paste}{ugIntroNumbersPageEmpty20}{ugIntroNumbersPagePatch20}
\pastebutton{ugIntroNumbersPageEmpty20}{\showpaste}
\tab{5}\spadcommand{partialFraction(1,factorial(10))\bound{partfrac }}
\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPagePatch21}
\begin{paste}{ugIntroNumbersPageFull21}{ugIntroNumbersPageEmpty21}
\pastebutton{ugIntroNumbersPageFull21}{\hidepaste}
\tab{5}\spadcommand{padicFraction(\%)\free{partfrac }}
\indentrel{3}\begin{verbatim}
   (21)
   1    1    1    1    1    1    2    1    2   2    2   1
    +  +  +  +  +  -  -  -  -  -  +
   2    4    5    6    7    8    2    3    4   5    2   7
       2    2    2    2    2    3    3    3        5
                           Type: PartialFraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPageEmpty21}
```

```
\begin{paste}{ugIntroNumbersPageEmpty21}{ugIntroNumbersPagePatch21}
\pastebutton{ugIntroNumbersPageEmpty21}{\showpaste}
\tab{5}\spadcommand{padicFraction(\%)\free{partfrac }}
\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPagePatch22}
\begin{paste}{ugIntroNumbersPageFull22}{ugIntroNumbersPageEmpty22}
\pastebutton{ugIntroNumbersPageFull22}{\hidepaste}
\tab{5}\spadcommand{radix(4/7, 8)\bound{rad }}
\indentrel{3}\begin{verbatim}
           -
   (22)  0.4
                                 Type: RadixExpansion 8
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPageEmpty22}
\begin{paste}{ugIntroNumbersPageEmpty22}{ugIntroNumbersPagePatch22}
\pastebutton{ugIntroNumbersPageEmpty22}{\showpaste}
\tab{5}\spadcommand{radix(4/7, 8)\bound{rad }}
\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPagePatch23}
\begin{paste}{ugIntroNumbersPageFull23}{ugIntroNumbersPageEmpty23}
\pastebutton{ugIntroNumbersPageFull23}{\hidepaste}
\tab{5}\spadcommand{\% + 2/3*\%i\free{rad }}
\indentrel{3}\begin{verbatim}
         4   2
   (23)   +  %i
         7   3
                         Type: Complex Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPageEmpty23}
\begin{paste}{ugIntroNumbersPageEmpty23}{ugIntroNumbersPagePatch23}
\pastebutton{ugIntroNumbersPageEmpty23}{\showpaste}
\tab{5}\spadcommand{\% + 2/3*\%i\free{rad }}
\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPagePatch24}
\begin{paste}{ugIntroNumbersPageFull24}{ugIntroNumbersPageEmpty24}
\pastebutton{ugIntroNumbersPageFull24}{\hidepaste}
\tab{5}\spadcommand{(5 + sqrt 63 + sqrt 847)**(1/3)}
\indentrel{3}\begin{verbatim}

         3
   (24)  \14\7  + 5
                                  Type: AlgebraicNumber
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPageEmpty24}
\begin{paste}{ugIntroNumbersPageEmpty24}{ugIntroNumbersPagePatch24}
\pastebutton{ugIntroNumbersPageEmpty24}{\showpaste}
\tab{5}\spadcommand{(5 + sqrt 63 + sqrt 847)**(1/3)}
\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPagePatch25}
\begin{paste}{ugIntroNumbersPageFull25}{ugIntroNumbersPageEmpty25}
\pastebutton{ugIntroNumbersPageFull25}{\hidepaste}
\tab{5}\spadcommand{x : PrimeField 7 := 5\bound{x }}
\indentrel{3}\begin{verbatim}
   (25)  5
                                     Type: PrimeField 7
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPageEmpty25}
\begin{paste}{ugIntroNumbersPageEmpty25}{ugIntroNumbersPagePatch25}
\pastebutton{ugIntroNumbersPageEmpty25}{\showpaste}
\tab{5}\spadcommand{x : PrimeField 7 := 5\bound{x }}
\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPagePatch26}
\begin{paste}{ugIntroNumbersPageFull26}{ugIntroNumbersPageEmpty26}
\pastebutton{ugIntroNumbersPageFull26}{\hidepaste}
\tab{5}\spadcommand{x**3\free{x }}
\indentrel{3}\begin{verbatim}
   (26)  6
                                     Type: PrimeField 7
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPageEmpty26}
\begin{paste}{ugIntroNumbersPageEmpty26}{ugIntroNumbersPagePatch26}
\pastebutton{ugIntroNumbersPageEmpty26}{\showpaste}
\tab{5}\spadcommand{x**3\free{x }}
\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPagePatch27}
\begin{paste}{ugIntroNumbersPageFull27}{ugIntroNumbersPageEmpty27}
\pastebutton{ugIntroNumbersPageFull27}{\hidepaste}
\tab{5}\spadcommand{1/x\free{x }}
\indentrel{3}\begin{verbatim}
   (27)  3
                                     Type: PrimeField 7
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugIntroNumbersPageEmpty27}
\begin{paste}{ugIntroNumbersPageEmpty27}{ugIntroNumbersPagePatch27}
\pastebutton{ugIntroNumbersPageEmpty27}{\showpaste}
\tab{5}\spadcommand{1/x\free{x }}
\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPagePatch28}
\begin{paste}{ugIntroNumbersPageFull28}{ugIntroNumbersPageEmpty28}
\pastebutton{ugIntroNumbersPageFull28}{\hidepaste}
\tab{5}\spadcommand{y : IntegerMod 6 := 5\bound{y }}
\indentrel{3}\begin{verbatim}
   (28)  5
                                    Type: IntegerMod 6
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPageEmpty28}
\begin{paste}{ugIntroNumbersPageEmpty28}{ugIntroNumbersPagePatch28}
\pastebutton{ugIntroNumbersPageEmpty28}{\showpaste}
\tab{5}\spadcommand{y : IntegerMod 6 := 5\bound{y }}
\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPagePatch29}
\begin{paste}{ugIntroNumbersPageFull29}{ugIntroNumbersPageEmpty29}
\pastebutton{ugIntroNumbersPageFull29}{\hidepaste}
\tab{5}\spadcommand{y**3\free{y }}
\indentrel{3}\begin{verbatim}
   (29)  5
                                    Type: IntegerMod 6
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPageEmpty29}
\begin{paste}{ugIntroNumbersPageEmpty29}{ugIntroNumbersPagePatch29}
\pastebutton{ugIntroNumbersPageEmpty29}{\showpaste}
\tab{5}\spadcommand{y**3\free{y }}
\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPagePatch30}
\begin{paste}{ugIntroNumbersPageFull30}{ugIntroNumbersPageEmpty30}
\pastebutton{ugIntroNumbersPageFull30}{\hidepaste}
\tab{5}\spadcommand{1/y\free{y }}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPageEmpty30}
\begin{paste}{ugIntroNumbersPageEmpty30}{ugIntroNumbersPagePatch30}
\pastebutton{ugIntroNumbersPageEmpty30}{\showpaste}
\tab{5}\spadcommand{1/y\free{y }}
```

```
\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPagePatch31}
\begin{paste}{ugIntroNumbersPageFull31}{ugIntroNumbersPageEmpty31}
\pastebutton{ugIntroNumbersPageFull31}{\hidepaste}
\tab{5}\spadcommand{a := rootOf(a**5 + a**3 + a**2 + 3,a)\bound{a }}
\indentrel{3}\begin{verbatim}
   (30)  a
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPageEmpty31}
\begin{paste}{ugIntroNumbersPageEmpty31}{ugIntroNumbersPagePatch31}
\pastebutton{ugIntroNumbersPageEmpty31}{\showpaste}
\tab{5}\spadcommand{a := rootOf(a**5 + a**3 + a**2 + 3,a)\bound{a }}
\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPagePatch32}
\begin{paste}{ugIntroNumbersPageFull32}{ugIntroNumbersPageEmpty32}
\pastebutton{ugIntroNumbersPageFull32}{\hidepaste}
\tab{5}\spadcommand{(a + 1)**10\free{a }}
\indentrel{3}\begin{verbatim}
             4       3       2
   (31)  - 85a  - 264a  - 378a  - 458a - 287
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPageEmpty32}
\begin{paste}{ugIntroNumbersPageEmpty32}{ugIntroNumbersPagePatch32}
\pastebutton{ugIntroNumbersPageEmpty32}{\showpaste}
\tab{5}\spadcommand{(a + 1)**10\free{a }}
\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPagePatch33}
\begin{paste}{ugIntroNumbersPageFull33}{ugIntroNumbersPageEmpty33}
\pastebutton{ugIntroNumbersPageFull33}{\hidepaste}
\tab{5}\spadcommand{b := rootOf(b**4 + a,b)\bound{b }\free{a }}
\indentrel{3}\begin{verbatim}
   (32)  b
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPageEmpty33}
\begin{paste}{ugIntroNumbersPageEmpty33}{ugIntroNumbersPagePatch33}
\pastebutton{ugIntroNumbersPageEmpty33}{\showpaste}
\tab{5}\spadcommand{b := rootOf(b**4 + a,b)\bound{b }\free{a }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugIntroNumbersPagePatch34}
\begin{paste}{ugIntroNumbersPageFull34}{ugIntroNumbersPageEmpty34}
\pastebutton{ugIntroNumbersPageFull34}{\hidepaste}
\tab{5}\spadcommand{2/(b - 1)\free{b }\bound{check }}
\indentrel{3}\begin{verbatim}
          2
   (33)
        b - 1
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPageEmpty34}
\begin{paste}{ugIntroNumbersPageEmpty34}{ugIntroNumbersPagePatch34}
\pastebutton{ugIntroNumbersPageEmpty34}{\showpaste}
\tab{5}\spadcommand{2/(b - 1)\free{b }\bound{check }}
\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPagePatch35}
\begin{paste}{ugIntroNumbersPageFull35}{ugIntroNumbersPageEmpty35}
\pastebutton{ugIntroNumbersPageFull35}{\hidepaste}
\tab{5}\spadcommand{ratDenom(\%)\free{check }\bound{check1 }}
\indentrel{3}\begin{verbatim}
   (34)
       4    3     2          3     4    3     2          2
     (a  - a  + 2a  - a + 1)b  + (a  - a  + 2a  - a + 1)b
   +
       4    3     2              4    3     2
     (a  - a  + 2a  - a + 1)b + a  - a  + 2a  - a + 1
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPageEmpty35}
\begin{paste}{ugIntroNumbersPageEmpty35}{ugIntroNumbersPagePatch35}
\pastebutton{ugIntroNumbersPageEmpty35}{\showpaste}
\tab{5}\spadcommand{ratDenom(\%)\free{check }\bound{check1 }}
\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPagePatch36}
\begin{paste}{ugIntroNumbersPageFull36}{ugIntroNumbersPageEmpty36}
\pastebutton{ugIntroNumbersPageFull36}{\hidepaste}
\tab{5}\spadcommand{2/\%+1\free{check1 }\bound{check2 }}
\indentrel{3}\begin{verbatim}
   (35)
         4    3     2          3
       (a  - a  + 2a  - a + 1)b
     +
         4    3     2          2
```

```
       (a  - a  + 2a  - a + 1)b
     +
         4    3     2              4    3     2
       (a  - a  + 2a  - a + 1)b + a  - a  + 2a  - a + 3
  /
         4    3     2          3
       (a  - a  + 2a  - a + 1)b
     +
         4    3     2          2
       (a  - a  + 2a  - a + 1)b
     +
         4    3     2              4    3     2
       (a  - a  + 2a  - a + 1)b + a  - a  + 2a  - a + 1
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPageEmpty36}
\begin{paste}{ugIntroNumbersPageEmpty36}{ugIntroNumbersPagePatch36}
\pastebutton{ugIntroNumbersPageEmpty36}{\showpaste}
\tab{5}\spadcommand{2/\%+1\free{check1 }\bound{check2 }}
\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPagePatch37}
\begin{paste}{ugIntroNumbersPageFull37}{ugIntroNumbersPageEmpty37}
\pastebutton{ugIntroNumbersPageFull37}{\hidepaste}
\tab{5}\spadcommand{ratDenom(\%)\free{check2 }}
\indentrel{3}\begin{verbatim}
   (36)  b
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPageEmpty37}
\begin{paste}{ugIntroNumbersPageEmpty37}{ugIntroNumbersPagePatch37}
\pastebutton{ugIntroNumbersPageEmpty37}{\showpaste}
\tab{5}\spadcommand{ratDenom(\%)\free{check2 }}
\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPagePatch38}
\begin{paste}{ugIntroNumbersPageFull38}{ugIntroNumbersPageEmpty38}
\pastebutton{ugIntroNumbersPageFull38}{\hidepaste}
\tab{5}\spadcommand{q:=quatern(1,2,3,4)*quatern(5,6,7,8) - quatern(5,6,7,8)*quatern(1,2,3,4)}
\indentrel{3}\begin{verbatim}
   (37)  - 8i + 16j - 8k
                               Type: Quaternion Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroNumbersPageEmpty38}
```

```
\begin{paste}{ugIntroNumbersPageEmpty38}{ugIntroNumbersPagePatch38}
\pastebutton{ugIntroNumbersPageEmpty38}{\showpaste}
\tab{5}\spadcommand{q:=quatern(1,2,3,4)*quatern(5,6,7,8) - quatern(5,6,7,8)*quatern(1,2,3,4)}
\end{paste}\end{patch}
```

———-

Data Structures

⇒ "notitle" (ListXmpPage) 3.64 on page 866
⇒ "notitle" (ugLangItsPage) 9 on page 1808
⇒ "notitle" (StreamXmpPage) 3.102 on page 1263
⇒ "notitle" (ugLangItsPage) 9 on page 1808
⇒ "notitle" (OneDimensionalArrayXmpPage) 3.4 on page 138
⇒ "notitle" (FlexibleArrayXmpPage) 3.39 on page 472
⇒ "notitle" (HeapXmpPage) 3.53 on page 690
⇒ "notitle" (BinarySearchTreeXmpPage) 3.11 on page 178
⇒ "notitle" (BalancedBinaryTreeXmpPage) 3.7 on page 157
⇒ "notitle" (SetXmpPage) 3.97 on page 1227
⇒ "notitle" (MultiSetXmpPage) 3.74 on page 978
⇒ "notitle" (AssociationListXmpPage) 3.3 on page 132
⇒ "notitle" (KeyedAccessFileXmpPage) 3.57 on page 734
⇒ "notitle" (LibraryXmpPage) 3.62 on page 840
⇒ "notitle" (SparseTableXmpPage) 3.101 on page 1259
⇒ "notitle" (StringTableXmpPage) 3.104 on page 1284
⇒ "notitle" (TableXmpPage) 3.106 on page 1298
⇒ "notitle" (ugTypesRecordsPage) 7 on page 1647
⇒ "notitle" (ugTypesUnionsPage) 7 on page 1656
⇒ "notitle" (ugDomainsPage) 16 on page 2425

— **ug01.ht** —

```
\begin{page}{ugIntroCollectPage}{1.6. Data Structures}
\beginscroll
%

Axiom has a large variety of data structures available.
Many data structures are particularly useful for interactive
computation and others are useful for building applications.
The data structures of Axiom are organized into
\spadglossSee{category hierarchies}{hierarchy} as shown on
the inside back cover.

A \spadgloss{list} is the most commonly used data structure in
Axiom for holding objects all of the same
type.\footnote{Lists are discussed in
```

```
\downlink{`List'}{ListXmpPage}\ignore{List} and in
\downlink{``Creating Lists and Streams with Iterators''}
{ugLangItsPage} in Section 5.5\ignore{ugLangIts}.}
The name {\it list} is short for ``linked-list of nodes.'' Each
node consists of a value (\spadfunFrom{first}{List}) and a link
(\spadfunFrom{rest}{List}) that
\spadglossSee{points}{pointer} to the next node, or to a
distinguished value denoting the empty list.
To get to, say, the third element, Axiom starts at the front
of the list, then traverses across two links to the third node.

\xtc{
Write a list of elements using
square brackets with commas separating the elements.
}{
\spadpaste{u := [1,-7,11] \bound{u}}
}
\xtc{
This is the value at the third node.
Alternatively, you can say \axiom{u.3}.
}{
\spadpaste{first rest rest u\free{u}}
}

Many operations are defined on lists, such as:
\axiomFun{empty?}, to test that a list has no elements;
\axiomFun{cons}\axiom{(x,l)}, to create a new list with
\axiomFun{first} element \axiom{x} and \axiomFun{rest} \axiom{l};
\axiomFun{reverse}, to create a new list with elements in reverse
order; and \axiomFun{sort}, to arrange elements in order.

An important point about lists is that they are ``mutable'': their
constituent elements and links can be changed ``in place.''
To do this, use any of the operations whose names end with the
character \axiomSyntax{!}.

\xtc{
The operation \spadfunFromX{concat}{List}\axiom{(u,v)}
replaces the last link of the list
\axiom{u} to point to some other list \axiom{v}.
Since \axiom{u} refers to the original list,
this change is seen by \axiom{u}.
}{
\spadpaste{concat!(u,[9,1,3,-4]); u\free{u}\bound{u1}}
}
\xtc{
A {\it cyclic list} is a list with a ``cycle'':
a link pointing back to an earlier node of the list.
To create a cycle, first get a node somewhere down
the list.
```

```
}{
\spadpaste{lastnode := rest(u,3)\free{u1}\bound{u2}}
}
\xtc{
Use \spadfunFromX{setrest}{List} to
change the link emanating from that node to point back to an
earlier part of the list.
}{
\spadpaste{setrest!(lastnode,rest(u,2)); u\free{u2}}
}

A \spadgloss{stream}
is a structure that (potentially) has an infinite number of
distinct elements.\footnote{Streams are discussed in
\downlink{`Stream'}{StreamXmpPage}\ignore{Stream} and in
\downlink{``Creating Lists and Streams with Iterators''}
{ugLangItsPage} in Section 5.5\ignore{ugLangIts}.}
Think of a stream as an ``infinite list'' where elements are
computed successively.

\xtc{
Create an infinite stream of factored integers.
Only a certain number of initial elements are computed
and displayed.
}{
\spadpaste{[factor(i) for i in 2.. by 2] \bound{stream1}}
}
\xtc{
Axiom represents streams by a collection of already-computed
elements together with a function to compute the next element
``on demand.''
Asking for the \eth{\axiom{n}} element causes elements \axiom{1} through
\axiom{n} to be evaluated.
}{
\spadpaste{\%.36 \free{stream1}}
}

Streams can also be finite or cyclic.
They are implemented by a linked list structure similar to lists
and have many of the same operations.
For example, \axiomFun{first} and \axiomFun{rest} are used to access
elements and successive nodes of a stream.
%%> reverse and sort do not exist for streams
%%Don't try to reverse or sort a stream: the
%%operation will generally run forever!

A \spadgloss{one-dimensional array} is another data structure used
to hold objects of the same type.\footnote{See
\downlink{`OneDimensionalArray'}{OneDimensionalArrayXmpPage}
\ignore{OneDimensionalArray} for
```

```
details.}
Unlike lists, one-dimensional arrays are inflexible---they are
implemented using a fixed block of storage.
Their advantage is that they give quick and equal access time to
any element.

\xtc{
A simple way to create a one-dimensional array is to apply the
operation \axiomFun{oneDimensionalArray} to a list of elements.
}{
\spadpaste{a := oneDimensionalArray [1, -7, 3, 3/2]\bound{a}}
}
\xtc{
One-dimensional arrays are also mutable:
you can change their constituent elements ``in place.''
}{
\spadpaste{a.3 := 11; a\bound{a1}\free{a}}
}
\xtc{
However, one-dimensional arrays are not flexible structures.
You cannot destructively \spadfunX{concat} them together.
}{
\spadpaste{concat!(a,oneDimensionalArray [1,-2])\free{a1}}
}

Examples of datatypes similar to \spadtype{OneDimensionalArray}
are: \spadtype{Vector} (vectors are mathematical structures
implemented by one-dimensional arrays), \spadtype{String} (arrays
of ``characters,'' represented by byte vectors), and
\spadtype{Bits} (represented by ``bit vectors'').

\xtc{
A vector of 32 bits,
each representing the \spadtype{Boolean} value \axiom{true}.
}{
\spadpaste{bits(32,true)}
}

A \spadgloss{flexible array} is a cross between a list
and a one-dimensional array.\footnote{See
\downlink{`FlexibleArray'}{FlexibleArrayXmpPage}\ignore{FlexibleArray} for
details.}
Like a one-dimensional array, a flexible array occupies a fixed
block of storage.
Its block of storage, however, has room to expand!
When it gets full, it grows (a new, larger block of storage is
allocated); when it has too much room, it contracts.

\xtc{
Create a flexible array of three elements.
```

```
}{
\spadpaste{f := flexibleArray [2, 7, -5]\bound{f}}
}
\xtc{
Insert some elements between the second and third elements.
}{
\spadpaste{insert!(flexibleArray [11, -3],f,2)\free{f}}
}

Flexible arrays are used to implement ‘‘heaps.’’ A
\spadgloss{heap} is an example of a data structure called a
\spadgloss{priority queue}, where elements are ordered with
respect to one another.\footnote{See
\downlink{‘Heap’}{HeapXmpPage}\ignore{Heap} for more details.
Heaps are also examples of data structures called
\spadglossSee{bags}{bag}.
Other bag data structures are \spadtype{Stack}, \spadtype{Queue},
and \spadtype{Dequeue}.}
A heap is organized
so as to optimize insertion and extraction of maximum elements.
The \spadfunX{extract} operation
returns the maximum element of the heap, after destructively
removing that element and
reorganizing the heap
so that the next maximum element is ready to be delivered.

\xtc{
An easy way to create a heap is to apply the
operation \spadfun{heap} to a list of values.
}{
\spadpaste{h := heap [-4,7,11,3,4,-7]\bound{h}}
}
\xtc{
This loop extracts elements one-at-a-time from \spad{h}
until the heap is exhausted, returning the elements
as a list in the order they were extracted.
}{
\spadpaste{[extract!(h) while not empty?(h)]\free{h}}
}

A \spadgloss{binary tree} is a ‘‘tree’’ with at most two branches per
node: it is either empty, or else is a node consisting of a value, and
a left and right subtree (again, binary trees).\footnote{Example of
binary tree types are \spadtype{BinarySearchTree} (see
\downlink{‘BinarySearchTree’}{BinarySearchTreeXmpPage}
\ignore{BinarySearchTree}, \spadtype{PendantTree},
\spadtype{TournamentTree}, and \spadtype{BalancedBinaryTree} (see
\downlink{‘BalancedBinaryTree’}{BalancedBinaryTreeXmpPage}
\ignore{BalancedBinaryTree}).}
```

```
\xtc{
A {\it binary search tree} is a binary tree such that,
for each node, the value of the node is
greater than all values (if any) in the left subtree,
and less than or equal all values (if any) in the right subtree.
}{
\spadpaste{binarySearchTree [5,3,2,9,4,7,11]}
}

\xtc{
A {\it balanced binary tree} is useful for doing modular computations.
Given a list \axiom{lm} of moduli,
\axiomFun{modTree}\axiom{(a,lm)} produces a balanced binary
tree with the values \texht{$a \bmod m$}{a {\tt mod} m}
at its leaves.
}{
\spadpaste{modTree(8,[2,3,5,7])}
}

A \spadgloss{set} is a collection of elements where duplication
and order is irrelevant.\footnote{See
\downlink{`Set'}{SetXmpPage}\ignore{Set} for more
details.}
Sets are always finite and have no corresponding
structure like streams for infinite collections.

\xtc{
%Create sets using braces (\axiomSyntax{\{} and \axiomSyntax{\}})
%rather than brackets.
}{
\spadpaste{fs := set [1/3,4/5,-1/3,4/5] \bound{fs}}
}

A \spadgloss{multiset}
is a set that keeps track of the number
of duplicate values.\footnote{See
\downlink{`MultiSet'}{MultiSetXmpPage}\ignore{MultiSet} for details.}
\xtc{
For all the primes \axiom{p} between 2 and 1000, find the
distribution of \texht{$p \bmod 5$}{p mod 5}.
}{
\spadpaste{multiset [x rem 5 for x in primes(2,1000)]}
}

A \spadgloss{table}
is conceptually a set of ``key--value'' pairs and
is a generalization of a multiset.\footnote{For examples of tables, see
\spadtype{AssociationList} (
\downlink{`AssociationList'}{AssociationListXmpPage}
\ignore{AssociationList}),
```

```
\spadtype{HashTable},
\spadtype{KeyedAccessFile} (
\downlink{`KeyedAccessFile'}{KeyedAccessFileXmpPage}
\ignore{KeyedAccessFile}),
\spadtype{Library} (\downlink{`Library'}{LibraryXmpPage}
\ignore{Library}),
\spadtype{SparseTable} (
\downlink{`SparseTable'}{SparseTableXmpPage}\ignore{SparseTable}),
\spadtype{StringTable} (
\downlink{`StringTable'}{StringTableXmpPage}\ignore{StringTable}),
and \spadtype{Table} (\downlink{`Table'}{TableXmpPage}\ignore{Table}).}
The domain \spadtype{Table(Key, Entry)} provides a general-purpose
type for tables with {\it values} of type \axiom{Entry} indexed
by {\it keys} of type \axiom{Key}.

\xtc{
Compute the above distribution of primes using tables.
First, let \axiom{t} denote an empty table of keys and values,
each of type \spadtype{Integer}.
}{
\spadpaste{t : Table(Integer,Integer) := empty()\bound{t}}
}

We define a function \userfun{howMany} to return the number
of values of a given modulus \axiom{k} seen so far.
It calls \axiomFun{search}\axiom{(k,t)} which returns the number of
values stored under the key \axiom{k} in table \axiom{t}, or
\axiom{"failed"} if no such value is yet stored in \axiom{t} under
\axiom{k}.

\xtc{
In English, this says ``Define \axiom{howMany(k)} as follows.
First, let \smath{n} be the value of \axiomFun{search}\smath{(k,t)}.
Then, if \smath{n} has the value \smath{"failed"}, return the value
\smath{1}; otherwise return \smath{n + 1}.''
}{
\spadpaste{howMany(k) == (n:=search(k,t); n case "failed" => 1; n+1)
\bound{how}}
}
\xtc{
Run through the primes to create the table, then print the table.
The expression \axiom{t.m := howMany(m)} updates the value in table \axiom{t}
stored under key \axiom{m}.
}{
\spadpaste{for p in primes(2,1000) repeat (m:= p rem 5; t.m:= howMany(m));
t\free{how t}}
}

A {\it record}
is an example of an inhomogeneous collection
```

```
of objects.\footnote{See
\downlink{``Records''}{ugTypesRecordsPage}
in Section 2.4\ignore{ugTypesRecords} for details.}
A record consists of a set of named {\it selectors} that
can be used to access its components.

\xtc{
Declare that \axiom{daniel} can only be
assigned a record with two prescribed fields.
}{
\spadpaste{daniel : Record(age : Integer, salary : Float) \bound{danieldec}}
}
\xtc{
Give \axiom{daniel} a value, using square brackets to enclose the values of
the fields.
}{
\spadpaste{daniel := [28, 32005.12] \free{danieldec}\bound{daniel}}
}
\xtc{
Give \axiom{daniel} a raise.
}{
\spadpaste{daniel.salary := 35000; daniel \free{daniel}}
}

A {\it union}
is a data structure used when objects
have multiple types.\footnote{See
\downlink{``Unions''}{ugTypesUnionsPage}
in Section 2.5\ignore{ugTypesUnions} for details.}

\xtc{
Let \axiom{dog} be either an integer or a string value.
}{
\spadpaste{dog: Union(licenseNumber: Integer, name: String)\bound{xint}}
}
\xtc{
Give \axiom{dog} a name.
}{
\spadpaste{dog := "Whisper"\free{xint}}
}

All told, there are over forty different data structures in
Axiom.
Using the domain constructors described in
\downlink{``Domains''}{ugDomainsPage}
in Chapter 13\ignore{ugDomains}, you
can add your own data structure or extend an existing one.
Choosing the right data structure for your application may be the key
to obtaining good performance.
```

```
\endscroll
\autobuttons
\end{page}

\begin{patch}{ugIntroCollectPagePatch1}
\begin{paste}{ugIntroCollectPageFull1}{ugIntroCollectPageEmpty1}
\pastebutton{ugIntroCollectPageFull1}{\hidepaste}
\tab{5}\spadcommand{u := [1,-7,11]\bound{u }}
\indentrel{3}\begin{verbatim}
   (1)  [1,- 7,11]
                                     Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPageEmpty1}
\begin{paste}{ugIntroCollectPageEmpty1}{ugIntroCollectPagePatch1}
\pastebutton{ugIntroCollectPageEmpty1}{\showpaste}
\tab{5}\spadcommand{u := [1,-7,11]\bound{u }}
\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPagePatch2}
\begin{paste}{ugIntroCollectPageFull2}{ugIntroCollectPageEmpty2}
\pastebutton{ugIntroCollectPageFull2}{\hidepaste}
\tab{5}\spadcommand{first rest rest u\free{u }}
\indentrel{3}\begin{verbatim}
   (2)  11
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPageEmpty2}
\begin{paste}{ugIntroCollectPageEmpty2}{ugIntroCollectPagePatch2}
\pastebutton{ugIntroCollectPageEmpty2}{\showpaste}
\tab{5}\spadcommand{first rest rest u\free{u }}
\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPagePatch3}
\begin{paste}{ugIntroCollectPageFull3}{ugIntroCollectPageEmpty3}
\pastebutton{ugIntroCollectPageFull3}{\hidepaste}
\tab{5}\spadcommand{concat!(u,[9,1,3,-4]); u\free{u }\bound{u1 }}
\indentrel{3}\begin{verbatim}
   (3)  [1,- 7,11,9,1,3,- 4]
                                     Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPageEmpty3}
\begin{paste}{ugIntroCollectPageEmpty3}{ugIntroCollectPagePatch3}
\pastebutton{ugIntroCollectPageEmpty3}{\showpaste}
\tab{5}\spadcommand{concat!(u,[9,1,3,-4]); u\free{u }\bound{u1 }}
```

```
\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPagePatch4}
\begin{paste}{ugIntroCollectPageFull4}{ugIntroCollectPageEmpty4}
\pastebutton{ugIntroCollectPageFull4}{\hidepaste}
\tab{5}\spadcommand{lastnode := rest(u,3)\free{u1 }\bound{u2 }}
\indentrel{3}\begin{verbatim}
   (4)  [9,1,3,- 4]
                                    Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPageEmpty4}
\begin{paste}{ugIntroCollectPageEmpty4}{ugIntroCollectPagePatch4}
\pastebutton{ugIntroCollectPageEmpty4}{\showpaste}
\tab{5}\spadcommand{lastnode := rest(u,3)\free{u1 }\bound{u2 }}
\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPagePatch5}
\begin{paste}{ugIntroCollectPageFull5}{ugIntroCollectPageEmpty5}
\pastebutton{ugIntroCollectPageFull5}{\hidepaste}
\tab{5}\spadcommand{setrest!(lastnode,rest(u,2)); u\free{u2 }}
\indentrel{3}\begin{verbatim}
               ----
   (5)  [1,- 7,11,9]
                                    Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPageEmpty5}
\begin{paste}{ugIntroCollectPageEmpty5}{ugIntroCollectPagePatch5}
\pastebutton{ugIntroCollectPageEmpty5}{\showpaste}
\tab{5}\spadcommand{setrest!(lastnode,rest(u,2)); u\free{u2 }}
\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPagePatch6}
\begin{paste}{ugIntroCollectPageFull6}{ugIntroCollectPageEmpty6}
\pastebutton{ugIntroCollectPageFull6}{\hidepaste}
\tab{5}\spadcommand{[factor(i) for i in 2.. by 2]\bound{stream1 }}
\indentrel{3}\begin{verbatim}
            2      3      2       4    2  2
   (6)  [2,2 ,2 3,2 ,2 5,2 3,2 7,2 ,2 3 ,2 5,...]
                           Type: Stream Factored Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPageEmpty6}
\begin{paste}{ugIntroCollectPageEmpty6}{ugIntroCollectPagePatch6}
\pastebutton{ugIntroCollectPageEmpty6}{\showpaste}
\tab{5}\spadcommand{[factor(i) for i in 2.. by 2]\bound{stream1 }}
```

```
\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPagePatch7}
\begin{paste}{ugIntroCollectPageFull7}{ugIntroCollectPageEmpty7}
\pastebutton{ugIntroCollectPageFull7}{\hidepaste}
\tab{5}\spadcommand{\%.36\free{stream1 }}
\indentrel{3}\begin{verbatim}
        3 2
   (7)  2 3
                                 Type: Factored Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPageEmpty7}
\begin{paste}{ugIntroCollectPageEmpty7}{ugIntroCollectPagePatch7}
\pastebutton{ugIntroCollectPageEmpty7}{\showpaste}
\tab{5}\spadcommand{\%.36\free{stream1 }}
\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPagePatch8}
\begin{paste}{ugIntroCollectPageFull8}{ugIntroCollectPageEmpty8}
\pastebutton{ugIntroCollectPageFull8}{\hidepaste}
\tab{5}\spadcommand{a := oneDimensionalArray [1, -7, 3, 3/2]\bound{a }}
\indentrel{3}\begin{verbatim}
                3
   (8)  [1,- 7,3,]
                2
             Type: OneDimensionalArray Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPageEmpty8}
\begin{paste}{ugIntroCollectPageEmpty8}{ugIntroCollectPagePatch8}
\pastebutton{ugIntroCollectPageEmpty8}{\showpaste}
\tab{5}\spadcommand{a := oneDimensionalArray [1, -7, 3, 3/2]\bound{a }}
\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPagePatch9}
\begin{paste}{ugIntroCollectPageFull9}{ugIntroCollectPageEmpty9}
\pastebutton{ugIntroCollectPageFull9}{\hidepaste}
\tab{5}\spadcommand{a.3 := 11; a\bound{a1 }\free{a }}
\indentrel{3}\begin{verbatim}
                 3
   (9)  [1,- 7,11,]
                 2
             Type: OneDimensionalArray Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPageEmpty9}
```

```
\begin{paste}{ugIntroCollectPageEmpty9}{ugIntroCollectPagePatch9}
\pastebutton{ugIntroCollectPageEmpty9}{\showpaste}
\tab{5}\spadcommand{a.3 := 11; a\bound{a1 }\free{a }}
\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPagePatch10}
\begin{paste}{ugIntroCollectPageFull10}{ugIntroCollectPageEmpty10}
\pastebutton{ugIntroCollectPageFull10}{\hidepaste}
\tab{5}\spadcommand{concat!(a,oneDimensionalArray [1,-2])\free{a1 }}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPageEmpty10}
\begin{paste}{ugIntroCollectPageEmpty10}{ugIntroCollectPagePatch10}
\pastebutton{ugIntroCollectPageEmpty10}{\showpaste}
\tab{5}\spadcommand{concat!(a,oneDimensionalArray [1,-2])\free{a1 }}
\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPagePatch11}
\begin{paste}{ugIntroCollectPageFull11}{ugIntroCollectPageEmpty11}
\pastebutton{ugIntroCollectPageFull11}{\hidepaste}
\tab{5}\spadcommand{bits(32,true)}
\indentrel{3}\begin{verbatim}
   (10)  "11111111111111111111111111111111"
                                           Type: Bits
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPageEmpty11}
\begin{paste}{ugIntroCollectPageEmpty11}{ugIntroCollectPagePatch11}
\pastebutton{ugIntroCollectPageEmpty11}{\showpaste}
\tab{5}\spadcommand{bits(32,true)}
\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPagePatch12}
\begin{paste}{ugIntroCollectPageFull12}{ugIntroCollectPageEmpty12}
\pastebutton{ugIntroCollectPageFull12}{\hidepaste}
\tab{5}\spadcommand{f := flexibleArray [2, 7, -5]\bound{f }}
\indentrel{3}\begin{verbatim}
   (11)  [2,7,- 5]
                           Type: FlexibleArray Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPageEmpty12}
\begin{paste}{ugIntroCollectPageEmpty12}{ugIntroCollectPagePatch12}
\pastebutton{ugIntroCollectPageEmpty12}{\showpaste}
\tab{5}\spadcommand{f := flexibleArray [2, 7, -5]\bound{f }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugIntroCollectPagePatch13}
\begin{paste}{ugIntroCollectPageFull13}{ugIntroCollectPageEmpty13}
\pastebutton{ugIntroCollectPageFull13}{\hidepaste}
\tab{5}\spadcommand{insert!(flexibleArray [11, -3],f,2)\free{f }}
\indentrel{3}\begin{verbatim}
   (12)  [2,11,- 3,7,- 5]
                            Type: FlexibleArray Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPageEmpty13}
\begin{paste}{ugIntroCollectPageEmpty13}{ugIntroCollectPagePatch13}
\pastebutton{ugIntroCollectPageEmpty13}{\showpaste}
\tab{5}\spadcommand{insert!(flexibleArray [11, -3],f,2)\free{f }}
\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPagePatch14}
\begin{paste}{ugIntroCollectPageFull14}{ugIntroCollectPageEmpty14}
\pastebutton{ugIntroCollectPageFull14}{\hidepaste}
\tab{5}\spadcommand{h := heap [-4,7,11,3,4,-7]\bound{h }}
\indentrel{3}\begin{verbatim}
   (13)  [11,4,7,- 4,3,- 7]
                                     Type: Heap Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPageEmpty14}
\begin{paste}{ugIntroCollectPageEmpty14}{ugIntroCollectPagePatch14}
\pastebutton{ugIntroCollectPageEmpty14}{\showpaste}
\tab{5}\spadcommand{h := heap [-4,7,11,3,4,-7]\bound{h }}
\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPagePatch15}
\begin{paste}{ugIntroCollectPageFull15}{ugIntroCollectPageEmpty15}
\pastebutton{ugIntroCollectPageFull15}{\hidepaste}
\tab{5}\spadcommand{[extract!(h) while not empty?(h)]\free{h }}
\indentrel{3}\begin{verbatim}
   (14)  [11,7,4,3,- 4,- 7]
                                     Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPageEmpty15}
\begin{paste}{ugIntroCollectPageEmpty15}{ugIntroCollectPagePatch15}
\pastebutton{ugIntroCollectPageEmpty15}{\showpaste}
\tab{5}\spadcommand{[extract!(h) while not empty?(h)]\free{h }}
\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPagePatch16}
```

```
\begin{paste}{ugIntroCollectPageFull16}{ugIntroCollectPageEmpty16}
\pastebutton{ugIntroCollectPageFull16}{\hidepaste}
\tab{5}\spadcommand{binarySearchTree [5,3,2,9,4,7,11]}
\indentrel{3}\begin{verbatim}
   (15)  [[2,3,4],5,[7,9,11]]
                 Type: BinarySearchTree PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPageEmpty16}
\begin{paste}{ugIntroCollectPageEmpty16}{ugIntroCollectPagePatch16}
\pastebutton{ugIntroCollectPageEmpty16}{\showpaste}
\tab{5}\spadcommand{binarySearchTree [5,3,2,9,4,7,11]}
\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPagePatch17}
\begin{paste}{ugIntroCollectPageFull17}{ugIntroCollectPageEmpty17}
\pastebutton{ugIntroCollectPageFull17}{\hidepaste}
\tab{5}\spadcommand{modTree(8,[2,3,5,7])}
\indentrel{3}\begin{verbatim}
   (16)  [0,2,3,1]
                                    Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPageEmpty17}
\begin{paste}{ugIntroCollectPageEmpty17}{ugIntroCollectPagePatch17}
\pastebutton{ugIntroCollectPageEmpty17}{\showpaste}
\tab{5}\spadcommand{modTree(8,[2,3,5,7])}
\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPagePatch18}
\begin{paste}{ugIntroCollectPageFull18}{ugIntroCollectPageEmpty18}
\pastebutton{ugIntroCollectPageFull18}{\hidepaste}
\tab{5}\spadcommand{fs := set[1/3,4/5,-1/3,4/5]\bound{fs }}
\indentrel{3}\begin{verbatim}
           1 1 4
   (17)  {- ,,}
           3 3 5
                            Type: Set Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPageEmpty18}
\begin{paste}{ugIntroCollectPageEmpty18}{ugIntroCollectPagePatch18}
\pastebutton{ugIntroCollectPageEmpty18}{\showpaste}
\tab{5}\spadcommand{fs := set[1/3,4/5,-1/3,4/5]\bound{fs }}
\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPagePatch19}
```

```
\begin{paste}{ugIntroCollectPageFull19}{ugIntroCollectPageEmpty19}
\pastebutton{ugIntroCollectPageFull19}{\hidepaste}
\tab{5}\spadcommand{multiset [x rem 5 for x in primes(2,1000)]}
\indentrel{3}\begin{verbatim}
   (18)  {0,42: 3,40: 1,38: 4,47: 2}
                                 Type: Multiset Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPageEmpty19}
\begin{paste}{ugIntroCollectPageEmpty19}{ugIntroCollectPagePatch19}
\pastebutton{ugIntroCollectPageEmpty19}{\showpaste}
\tab{5}\spadcommand{multiset [x rem 5 for x in primes(2,1000)]}
\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPagePatch20}
\begin{paste}{ugIntroCollectPageFull20}{ugIntroCollectPageEmpty20}
\pastebutton{ugIntroCollectPageFull20}{\hidepaste}
\tab{5}\spadcommand{t : Table(Integer,Integer) := empty()\bound{t }}
\indentrel{3}\begin{verbatim}
   (19)  table()
                           Type: Table(Integer,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPageEmpty20}
\begin{paste}{ugIntroCollectPageEmpty20}{ugIntroCollectPagePatch20}
\pastebutton{ugIntroCollectPageEmpty20}{\showpaste}
\tab{5}\spadcommand{t : Table(Integer,Integer) := empty()\bound{t }}
\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPagePatch21}
\begin{paste}{ugIntroCollectPageFull21}{ugIntroCollectPageEmpty21}
\pastebutton{ugIntroCollectPageFull21}{\hidepaste}
\tab{5}\spadcommand{howMany(k) == (n:=search(k,t); n case "failed" => 1; n+1)\bound{how }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPageEmpty21}
\begin{paste}{ugIntroCollectPageEmpty21}{ugIntroCollectPagePatch21}
\pastebutton{ugIntroCollectPageEmpty21}{\showpaste}
\tab{5}\spadcommand{howMany(k) == (n:=search(k,t); n case "failed" => 1; n+1)\bound{how }}
\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPagePatch22}
\begin{paste}{ugIntroCollectPageFull22}{ugIntroCollectPageEmpty22}
\pastebutton{ugIntroCollectPageFull22}{\hidepaste}
\tab{5}\spadcommand{for p in primes(2,1000) repeat (m:= p rem 5; t.m:= howMany(m)); t\free{how t
```

```
\indentrel{3}\begin{verbatim}
   (21)  table(2= 47,4= 38,1= 40,3= 42,0= 1)
                              Type: Table(Integer,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPageEmpty22}
\begin{paste}{ugIntroCollectPageEmpty22}{ugIntroCollectPagePatch22}
\pastebutton{ugIntroCollectPageEmpty22}{\showpaste}
\tab{5}\spadcommand{for p in primes(2,1000) repeat (m:= p rem 5; t.m:= howMany(m)); t\free{how t }}
\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPagePatch23}
\begin{paste}{ugIntroCollectPageFull23}{ugIntroCollectPageEmpty23}
\pastebutton{ugIntroCollectPageFull23}{\hidepaste}
\tab{5}\spadcommand{daniel : Record(age : Integer, salary : Float)\bound{danieldec }}
\indentrel{3}\begin{verbatim}
                                                Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPageEmpty23}
\begin{paste}{ugIntroCollectPageEmpty23}{ugIntroCollectPagePatch23}
\pastebutton{ugIntroCollectPageEmpty23}{\showpaste}
\tab{5}\spadcommand{daniel : Record(age : Integer, salary : Float)\bound{danieldec }}
\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPagePatch24}
\begin{paste}{ugIntroCollectPageFull24}{ugIntroCollectPageEmpty24}
\pastebutton{ugIntroCollectPageFull24}{\hidepaste}
\tab{5}\spadcommand{daniel := [28, 32005.12]\free{danieldec }\bound{daniel }}
\indentrel{3}\begin{verbatim}
   (23)  [age= 28,salary= 32005.12]
               Type: Record(age: Integer,salary: Float)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPageEmpty24}
\begin{paste}{ugIntroCollectPageEmpty24}{ugIntroCollectPagePatch24}
\pastebutton{ugIntroCollectPageEmpty24}{\showpaste}
\tab{5}\spadcommand{daniel := [28, 32005.12]\free{danieldec }\bound{daniel }}
\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPagePatch25}
\begin{paste}{ugIntroCollectPageFull25}{ugIntroCollectPageEmpty25}
\pastebutton{ugIntroCollectPageFull25}{\hidepaste}
\tab{5}\spadcommand{daniel.salary := 35000; daniel\free{daniel }}
\indentrel{3}\begin{verbatim}
   (24)  [age= 28,salary= 35000.0]
               Type: Record(age: Integer,salary: Float)
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPageEmpty25}
\begin{paste}{ugIntroCollectPageEmpty25}{ugIntroCollectPagePatch25}
\pastebutton{ugIntroCollectPageEmpty25}{\showpaste}
\tab{5}\spadcommand{daniel.salary := 35000; daniel\free{daniel }}
\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPagePatch26}
\begin{paste}{ugIntroCollectPageFull26}{ugIntroCollectPageEmpty26}
\pastebutton{ugIntroCollectPageFull26}{\hidepaste}
\tab{5}\spadcommand{dog: Union(licenseNumber: Integer, name: String)\bound{xint }}
\indentrel{3}\begin{verbatim}
                                              Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPageEmpty26}
\begin{paste}{ugIntroCollectPageEmpty26}{ugIntroCollectPagePatch26}
\pastebutton{ugIntroCollectPageEmpty26}{\showpaste}
\tab{5}\spadcommand{dog: Union(licenseNumber: Integer, name: String)\bound{xint }}
\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPagePatch27}
\begin{paste}{ugIntroCollectPageFull27}{ugIntroCollectPageEmpty27}
\pastebutton{ugIntroCollectPageFull27}{\hidepaste}
\tab{5}\spadcommand{dog := "Whisper"\free{xint }}
\indentrel{3}\begin{verbatim}
   (26)  "Whisper"
                         Type: Union(name: String,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCollectPageEmpty27}
\begin{paste}{ugIntroCollectPageEmpty27}{ugIntroCollectPagePatch27}
\pastebutton{ugIntroCollectPageEmpty27}{\showpaste}
\tab{5}\spadcommand{dog := "Whisper"\free{xint }}
\end{paste}\end{patch}
```

Expanding to Higher Dimensions

⇒ "notitle" (TwoDimensionalArrayXmpPage) 3.5 on page 143
⇒ "notitle" (MatrixXmpPage) 3.75 on page 984
⇒ "notitle" (PermanentXmpPage) 3.85 on page 1085

⇒ "notitle" (SqMatrixXmpPage) 3.99 on page 1243
⇒ "notitle" (VectorXmpPage) 3.114 on page 1351
⇒ "notitle" (ugProblemEigenPage) 12 on page 2123
⇒ "notitle" (ugProblemLinPolEqnPage) 12 on page 2130
⇒ "notitle" (ugLangItsPage) 9 on page 1808

— **ug01.ht** —

```
\begin{page}{ugIntroTwoDimPage}{1.7. Expanding to Higher Dimensions}
\beginscroll
%

To get higher dimensional aggregates, you can create one-dimensional
aggregates with elements that are themselves aggregates, for example,
lists of lists, one-dimensional arrays of lists of multisets, and so
on.  For applications requiring two-dimensional homogeneous
aggregates, you will likely find {\it two-dimensional arrays} and {\it
matrices} most useful.

The entries in \spadtype{TwoDimensionalArray} and
\spadtype{Matrix} objects
are all the same type, except that those for
\spadtype{Matrix} must belong to a \spadtype{Ring}.
You create and access elements in roughly the same way.
Since matrices have an understood algebraic structure, certain algebraic
operations are available for matrices but not for arrays.
Because of this, we limit our discussion here to
\spadtype{Matrix}, that can be regarded as an extension of
\spadtype{TwoDimensionalArray}.\footnote{See
\downlink{`TwoDimensionalArray'}{TwoDimensionalArrayXmpPage}
\ignore{TwoDimensionalArray} for more information about arrays.
For more information about Axiom's linear algebra
facilities, see \downlink{`Matrix'}{MatrixXmpPage}\ignore{Matrix},
\downlink{`Permanent'}{PermanentXmpPage}\ignore{Permanent},
\downlink{`SquareMatrix'}{SqMatrixXmpPage}\ignore{SquareMatrix},
\downlink{`Vector'}{VectorXmpPage}\ignore{Vector},
\downlink{``Computation of Eigenvalues and Eigenvectors''}
{ugProblemEigenPage} in Section 8.4\ignore{ugProblemEigen}
\texht{(computation of eigenvalues and
eigenvectors)}{}, and
\downlink{``Solution of Linear and Polynomial Equations''}
{ugProblemLinPolEqnPage} in Section 8.5\ignore{ugProblemLinPolEqn}
\texht{(solution of linear and
polynomial equations)}{}.}

\xtc{
You can create a matrix from a list of lists,
where each of the inner lists represents a row of the matrix.
}{
\spadpaste{m := matrix([[1,2], [3,4]]) \bound{m}}
```

```
}
\xtc{
The ``collections'' construct (see
\downlink{``Creating Lists and Streams with Iterators''}
{ugLangItsPage} in Section 5.5\ignore{ugLangIts}) is
useful for creating matrices whose entries are given by formulas.
}{
\spadpaste{matrix([[1/(i + j - x) for i in 1..4] for j in 1..4])
\bound{hilb}}
}
\xtc{
Let \axiom{vm} denote the three by three Vandermonde matrix.
}{
\spadpaste{vm := matrix [[1,1,1], [x,y,z], [x*x,y*y,z*z]] \bound{vm}}
}
\xtc{
Use this syntax to extract an entry in the matrix.
}{
\spadpaste{vm(3,3) \free{vm}}
}
\xtc{
You can also pull out a \axiomFun{row} or a \axiom{column}.
}{
\spadpaste{column(vm,2) \free{vm}}
}
\xtc{
You can do arithmetic.
}{
\spadpaste{vm * vm \free{vm}}
}
\xtc{
You can perform operations such as
\axiomFun{transpose}, \axiomFun{trace}, and \axiomFun{determinant}.
}{
\spadpaste{factor determinant vm \free{vm}\bound{d}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugIntroTwoDimPagePatch1}
\begin{paste}{ugIntroTwoDimPageFull1}{ugIntroTwoDimPageEmpty1}
\pastebutton{ugIntroTwoDimPageFull1}{\hidepaste}
\tab{5}\spadcommand{m := matrix([[1,2], [3,4]])\bound{m }}
\indentrel{3}\begin{verbatim}
        1  2
   (1)
        3  4
                                   Type: Matrix Integer
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroTwoDimPageEmpty1}
\begin{paste}{ugIntroTwoDimPageEmpty1}{ugIntroTwoDimPagePatch1}
\pastebutton{ugIntroTwoDimPageEmpty1}{\showpaste}
\tab{5}\spadcommand{m := matrix([[1,2], [3,4]])\bound{m }}
\end{paste}\end{patch}

\begin{patch}{ugIntroTwoDimPagePatch2}
\begin{paste}{ugIntroTwoDimPageFull2}{ugIntroTwoDimPageEmpty2}
\pastebutton{ugIntroTwoDimPageFull2}{\hidepaste}
\tab{5}\spadcommand{matrix([[1/(i + j - x) for i in 1..4] for j in 1..4])\bound{hilb }}
\indentrel{3}\begin{verbatim}
            1        1        1        1
        -   -   -   -
          x - 2    x - 3    x - 4    x - 5

            1        1        1        1
        -   -   -   -
          x - 3    x - 4    x - 5    x - 6
   (2)
            1        1        1        1
        -   -   -   -
          x - 4    x - 5    x - 6    x - 7

            1        1        1        1
        -   -   -   -
          x - 5    x - 6    x - 7    x - 8
               Type: Matrix Fraction Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroTwoDimPageEmpty2}
\begin{paste}{ugIntroTwoDimPageEmpty2}{ugIntroTwoDimPagePatch2}
\pastebutton{ugIntroTwoDimPageEmpty2}{\showpaste}
\tab{5}\spadcommand{matrix([[1/(i + j - x) for i in 1..4] for j in 1..4])\bound{hilb }}
\end{paste}\end{patch}

\begin{patch}{ugIntroTwoDimPagePatch3}
\begin{paste}{ugIntroTwoDimPageFull3}{ugIntroTwoDimPageEmpty3}
\pastebutton{ugIntroTwoDimPageFull3}{\hidepaste}
\tab{5}\spadcommand{vm := matrix [[1,1,1], [x,y,z], [x*x,y*y,z*z]]\bound{vm }}
\indentrel{3}\begin{verbatim}
        1   1   1

   (3)  x   y   z

         2   2   2
        x   y   z
```

```
                        Type: Matrix Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroTwoDimPageEmpty3}
\begin{paste}{ugIntroTwoDimPageEmpty3}{ugIntroTwoDimPagePatch3}
\pastebutton{ugIntroTwoDimPageEmpty3}{\showpaste}
\tab{5}\spadcommand{vm := matrix [[1,1,1], [x,y,z], [x*x,y*y,z*z]]\bound{vm }}
\end{paste}\end{patch}

\begin{patch}{ugIntroTwoDimPagePatch4}
\begin{paste}{ugIntroTwoDimPageFull4}{ugIntroTwoDimPageEmpty4}
\pastebutton{ugIntroTwoDimPageFull4}{\hidepaste}
\tab{5}\spadcommand{vm(3,3)\free{vm }}
\indentrel{3}\begin{verbatim}
         2
   (4)  z
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroTwoDimPageEmpty4}
\begin{paste}{ugIntroTwoDimPageEmpty4}{ugIntroTwoDimPagePatch4}
\pastebutton{ugIntroTwoDimPageEmpty4}{\showpaste}
\tab{5}\spadcommand{vm(3,3)\free{vm }}
\end{paste}\end{patch}

\begin{patch}{ugIntroTwoDimPagePatch5}
\begin{paste}{ugIntroTwoDimPageFull5}{ugIntroTwoDimPageEmpty5}
\pastebutton{ugIntroTwoDimPageFull5}{\hidepaste}
\tab{5}\spadcommand{column(vm,2)\free{vm }}
\indentrel{3}\begin{verbatim}
              2
   (5)  [1,y,y ]
                        Type: Vector Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroTwoDimPageEmpty5}
\begin{paste}{ugIntroTwoDimPageEmpty5}{ugIntroTwoDimPagePatch5}
\pastebutton{ugIntroTwoDimPageEmpty5}{\showpaste}
\tab{5}\spadcommand{column(vm,2)\free{vm }}
\end{paste}\end{patch}

\begin{patch}{ugIntroTwoDimPagePatch6}
\begin{paste}{ugIntroTwoDimPageFull6}{ugIntroTwoDimPageEmpty6}
\pastebutton{ugIntroTwoDimPageFull6}{\hidepaste}
\tab{5}\spadcommand{vm * vm\free{vm }}
\indentrel{3}\begin{verbatim}
   (6)
```

```
       2                 2               2
      x  + x + 1        y  + y + 1      z  + z + 1

     2                  2     2        3
    x z + x y + x      y z + y  + x   z  + y z + x

    2 2       2    2   2 2    3    2   4    2     2
   x z  + x y  + x   y z  + y  + x   z  + y z + x
                        Type: Matrix Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroTwoDimPageEmpty6}
\begin{paste}{ugIntroTwoDimPageEmpty6}{ugIntroTwoDimPagePatch6}
\pastebutton{ugIntroTwoDimPageEmpty6}{\showpaste}
\tab{5}\spadcommand{vm * vm\free{vm }}
\end{paste}\end{patch}

\begin{patch}{ugIntroTwoDimPagePatch7}
\begin{paste}{ugIntroTwoDimPageFull7}{ugIntroTwoDimPageEmpty7}
\pastebutton{ugIntroTwoDimPageFull7}{\hidepaste}
\tab{5}\spadcommand{factor determinant vm\free{vm }\bound{d }}
\indentrel{3}\begin{verbatim}
   (7)  (y - x)(z - y)(z - x)
                      Type: Factored Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroTwoDimPageEmpty7}
\begin{paste}{ugIntroTwoDimPageEmpty7}{ugIntroTwoDimPagePatch7}
\pastebutton{ugIntroTwoDimPageEmpty7}{\showpaste}
\tab{5}\spadcommand{factor determinant vm\free{vm }\bound{d }}
\end{paste}\end{patch}
```

———-

Writing Your Own Functions

⇒ “notitle” (ugUserPage) 10 on page 1823
⇒ “notitle” (ugInOutInPage) 9 on page 1722

— **ug01.ht** —

```
\begin{page}{ugIntroYouPage}{1.8. Writing Your Own Functions}
\beginscroll
%
```

```
Axiom provides you with a very large library of predefined operations
and objects to compute with.  You can use the Axiom library of
constructors to create new objects dynamically of quite arbitrary
complexity.  For example, you can make lists of matrices of fractions
of polynomials with complex floating point numbers as coefficients.
Moreover, the library provides a wealth of operations that allow you
to create and manipulate these objects.

For many applications, you need to interact with the interpreter and
write some Axiom programs to tackle your application.  Axiom allows
you to write functions interactively, thereby effectively extending
the system library.  Here we give a few simple examples, leaving the
details to \downlink{``User-Defined Functions, Macros and Rules''}
{ugUserPage} in Chapter 6\ignore{ugUser}.

We begin by looking at several ways that you can define the
``factorial'' function in Axiom.  The first way is to give a
piece-wise definition of the function.  This method is best for a
general recurrence relation since the pieces are gathered together and
compiled into an efficient iterative function.  Furthermore, enough
previously computed values are automatically saved so that a
subsequent call to the function can pick up from where it left off.

\xtc{
Define the value of \userfun{fact} at \axiom{0}.
}{
\spadpaste{fact(0) == 1 \bound{fact}}
}
\xtc{
Define the value of \axiom{fact(n)} for general \axiom{n}.
}{
\spadpaste{fact(n) == n*fact(n-1)\bound{facta}\free{fact}}
}
\xtc{
Ask for the value at \axiom{50}.
The resulting function created by Axiom
computes the value by iteration.
}{
\spadpaste{fact(50) \free{facta}}
}
\xtc{
A second definition uses an \axiom{if-then-else} and recursion.
}{
\spadpaste{fac(n) == if n < 3 then n else n * fac(n - 1) \bound{fac}}
}
\xtc{
This function is less efficient than the previous version since
each iteration involves a recursive function call.
}{
\spadpaste{fac(50) \free{fac}}
```

```
}
\xtc{
A third version directly uses iteration.
}{
\spadpaste{fa(n) == (a := 1; for i in 2..n repeat a := a*i; a) \bound{fa}}
}
\xtc{
This is the least space-consumptive version.
}{
\spadpaste{fa(50) \free{fa}}
}
\xtc{
A final version appears to construct a large list and then reduces over
it with multiplication.
}{
\spadpaste{f(n) == reduce(*,[i for i in 2..n]) \bound{f}}
}
\xtc{
In fact, the resulting computation is optimized into an efficient
iteration loop equivalent to that of the third version.
}{
\spadpaste{f(50) \free{f}}
}
\xtc{
The library version uses an algorithm that is different from the four
above because it highly optimizes the recurrence relation definition of
\axiomFun{factorial}.
}{
\spadpaste{factorial(50)}
}

You are not limited to one-line functions in Axiom.
If you place your function definitions in {\bf .input} files
(see \downlink{``Input Files''}{ugInOutInPage}
in Section 4.1\ignore{ugInOutIn}), you can have
multi-line functions that use indentation for grouping.

Given \axiom{n} elements, \axiomFun{diagonalMatrix} creates an
\axiom{n} by \axiom{n} matrix with those elements down the diagonal.
This function uses a permutation matrix
that interchanges the \axiom{i}th and \axiom{j}th rows of a matrix
by which it is right-multiplied.

\xtc{
This function definition shows a style of definition that can be used
in {\bf .input} files.
Indentation is used to create \spadglossSee{blocks}{block}\texht{\/}{}:
sequences of expressions that are evaluated in sequence except as
modified by control statements such as \axiom{if-then-else} and \axiom{return}.
}{
```

```
\begin{spadsrc}[\bound{permMat}]
permMat(n, i, j) ==
  m := diagonalMatrix
    [(if i = k or j = k then 0 else 1)
      for k in 1..n]
  m(i,j) := 1
  m(j,i) := 1
  m
\end{spadsrc}
}
\xtc{
This creates a four by four matrix that interchanges the second and third
rows.
}{
\spadpaste{p := permMat(4,2,3) \free{permMat}\bound{p}}
}
\xtc{
Create an example matrix to permute.
}{
\spadpaste{m := matrix [[4*i + j for j in 1..4] for i in 0..3]\bound{m}}
}
\xtc{
Interchange the second and third rows of m.
}{
\spadpaste{permMat(4,2,3) * m \free{p m}}
}

A function can also be passed as an argument to another function,
which then applies the function or passes it off to some other
function that does.
You often have to declare the type of a function that has
functional arguments.

\xtc{
This declares \userfun{t} to be a two-argument function that
returns a \spadtype{Float}.
The first argument is a function that takes one \spadtype{Float}
argument and returns a \spadtype{Float}.
}{
\spadpaste{t : (Float -> Float, Float) -> Float \bound{tdecl}}
}
\xtc{
This is the definition of \userfun{t}.
}{
\spadpaste{t(fun, x) == fun(x)**2 + sin(x)**2 \free{tdecl}\bound{t}}
}
\xtc{
We have not defined a \axiomFun{cos} in the workspace. The one from the
Axiom library will do.
}{
```

```
\spadpaste{t(cos, 5.2058) \free{t}}
}
\xtc{
Here we define our own (user-defined) function.
}{
\spadpaste{cosinv(y) == cos(1/y) \bound{cosinv}}
}
\xtc{
Pass this function as an argument to \userfun{t}.
}{
\spadpaste{t(cosinv, 5.2058) \free{t}\free{cosinv}}
}

Axiom also has pattern matching capabilities for
simplification
of expressions and for defining new functions by rules.
For example, suppose that you want to apply regularly a transformation
that groups together products of radicals:
\texht{$$\sqrt{a}\:\sqrt{b} \mapsto \sqrt{ab}, \quad
(\forall a)(\forall b)$$}
{\axiom{sqrt(a) * sqrt(b) by sqrt(a*b)} for any \axiom{a} and \axiom{b}}
Note that such a transformation is not generally correct.
Axiom never uses it automatically.

\xtc{
Give this rule the name \userfun{groupSqrt}.
}{
\spadpaste{groupSqrt := rule(sqrt(a) * sqrt(b) == sqrt(a*b)) \bound{g}}
}
\xtc{
Here is a test expression.
}{
\spadpaste{a := (sqrt(x) + sqrt(y) + sqrt(z))**4 \bound{sxy}}
}
\xtc{
The rule
\userfun{groupSqrt} successfully simplifies the expression.
}{
\spadpaste{groupSqrt a \free{sxy} \free{g}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugIntroYouPagePatch1}
\begin{paste}{ugIntroYouPageFull1}{ugIntroYouPageEmpty1}
\pastebutton{ugIntroYouPageFull1}{\hidepaste}
\tab{5}\spadcommand{fact(0) == 1\bound{fact }}
\indentrel{3}\begin{verbatim}
```

```
                                          Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroYouPageEmpty1}
\begin{paste}{ugIntroYouPageEmpty1}{ugIntroYouPagePatch1}
\pastebutton{ugIntroYouPageEmpty1}{\showpaste}
\tab{5}\spadcommand{fact(0) == 1\bound{fact }}
\end{paste}\end{patch}

\begin{patch}{ugIntroYouPagePatch2}
\begin{paste}{ugIntroYouPageFull2}{ugIntroYouPageEmpty2}
\pastebutton{ugIntroYouPageFull2}{\hidepaste}
\tab{5}\spadcommand{fact(n) == n*fact(n-1)\bound{facta }\free{fact }}
\indentrel{3}\begin{verbatim}
                                          Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroYouPageEmpty2}
\begin{paste}{ugIntroYouPageEmpty2}{ugIntroYouPagePatch2}
\pastebutton{ugIntroYouPageEmpty2}{\showpaste}
\tab{5}\spadcommand{fact(n) == n*fact(n-1)\bound{facta }\free{fact }}
\end{paste}\end{patch}

\begin{patch}{ugIntroYouPagePatch3}
\begin{paste}{ugIntroYouPageFull3}{ugIntroYouPageEmpty3}
\pastebutton{ugIntroYouPageFull3}{\hidepaste}
\tab{5}\spadcommand{fact(50)\free{facta }}
\indentrel{3}\begin{verbatim}
   (3)
  30414093201713378043612608166064768844377641568960512 0_
   00000000000
                               Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroYouPageEmpty3}
\begin{paste}{ugIntroYouPageEmpty3}{ugIntroYouPagePatch3}
\pastebutton{ugIntroYouPageEmpty3}{\showpaste}
\tab{5}\spadcommand{fact(50)\free{facta }}
\end{paste}\end{patch}

\begin{patch}{ugIntroYouPagePatch4}
\begin{paste}{ugIntroYouPageFull4}{ugIntroYouPageEmpty4}
\pastebutton{ugIntroYouPageFull4}{\hidepaste}
\tab{5}\spadcommand{fac(n) == if n < 3 then n else n * fac(n - 1)\bound{fac }}
\indentrel{3}\begin{verbatim}
                                          Type: Void
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroYouPageEmpty4}
\begin{paste}{ugIntroYouPageEmpty4}{ugIntroYouPagePatch4}
\pastebutton{ugIntroYouPageEmpty4}{\showpaste}
\tab{5}\spadcommand{fac(n) == if n < 3 then n else n * fac(n - 1)\bound{fac }}
\end{paste}\end{patch}

\begin{patch}{ugIntroYouPagePatch5}
\begin{paste}{ugIntroYouPageFull5}{ugIntroYouPageEmpty5}
\pastebutton{ugIntroYouPageFull5}{\hidepaste}
\tab{5}\spadcommand{fac(50)\free{fac }}
\indentrel{3}\begin{verbatim}
   (5)
  30414093201713378043612608166064768844377641568960512_
   00000000000
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroYouPageEmpty5}
\begin{paste}{ugIntroYouPageEmpty5}{ugIntroYouPagePatch5}
\pastebutton{ugIntroYouPageEmpty5}{\showpaste}
\tab{5}\spadcommand{fac(50)\free{fac }}
\end{paste}\end{patch}

\begin{patch}{ugIntroYouPagePatch6}
\begin{paste}{ugIntroYouPageFull6}{ugIntroYouPageEmpty6}
\pastebutton{ugIntroYouPageFull6}{\hidepaste}
\tab{5}\spadcommand{fa(n) == (a := 1; for i in 2..n repeat a := a*i; a)\bound{fa }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroYouPageEmpty6}
\begin{paste}{ugIntroYouPageEmpty6}{ugIntroYouPagePatch6}
\pastebutton{ugIntroYouPageEmpty6}{\showpaste}
\tab{5}\spadcommand{fa(n) == (a := 1; for i in 2..n repeat a := a*i; a)\bound{fa }}
\end{paste}\end{patch}

\begin{patch}{ugIntroYouPagePatch7}
\begin{paste}{ugIntroYouPageFull7}{ugIntroYouPageEmpty7}
\pastebutton{ugIntroYouPageFull7}{\hidepaste}
\tab{5}\spadcommand{fa(50)\free{fa }}
\indentrel{3}\begin{verbatim}
   (7)
  30414093201713378043612608166064768844377641568960512_
   00000000000
                                  Type: PositiveInteger
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroYouPageEmpty7}
\begin{paste}{ugIntroYouPageEmpty7}{ugIntroYouPagePatch7}
\pastebutton{ugIntroYouPageEmpty7}{\showpaste}
\tab{5}\spadcommand{fa(50)\free{fa }}
\end{paste}\end{patch}

\begin{patch}{ugIntroYouPagePatch8}
\begin{paste}{ugIntroYouPageFull8}{ugIntroYouPageEmpty8}
\pastebutton{ugIntroYouPageFull8}{\hidepaste}
\tab{5}\spadcommand{f(n) == reduce(*,[i for i in 2..n])\bound{f }}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroYouPageEmpty8}
\begin{paste}{ugIntroYouPageEmpty8}{ugIntroYouPagePatch8}
\pastebutton{ugIntroYouPageEmpty8}{\showpaste}
\tab{5}\spadcommand{f(n) == reduce(*,[i for i in 2..n])\bound{f }}
\end{paste}\end{patch}

\begin{patch}{ugIntroYouPagePatch9}
\begin{paste}{ugIntroYouPageFull9}{ugIntroYouPageEmpty9}
\pastebutton{ugIntroYouPageFull9}{\hidepaste}
\tab{5}\spadcommand{f(50)\free{f }}
\indentrel{3}\begin{verbatim}
   (9)
  30414093201713378043612608166064768844377641568960512_
   00000000000
                                 Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroYouPageEmpty9}
\begin{paste}{ugIntroYouPageEmpty9}{ugIntroYouPagePatch9}
\pastebutton{ugIntroYouPageEmpty9}{\showpaste}
\tab{5}\spadcommand{f(50)\free{f }}
\end{paste}\end{patch}

\begin{patch}{ugIntroYouPagePatch10}
\begin{paste}{ugIntroYouPageFull10}{ugIntroYouPageEmpty10}
\pastebutton{ugIntroYouPageFull10}{\hidepaste}
\tab{5}\spadcommand{factorial(50)}
\indentrel{3}\begin{verbatim}
   (10)
  30414093201713378043612608166064768844377641568960512_
   00000000000
```

```
                                    Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroYouPageEmpty10}
\begin{paste}{ugIntroYouPageEmpty10}{ugIntroYouPagePatch10}
\pastebutton{ugIntroYouPageEmpty10}{\showpaste}
\tab{5}\spadcommand{factorial(50)}
\end{paste}\end{patch}

\begin{patch}{ugIntroYouPagePatch11}
\begin{paste}{ugIntroYouPageFull11}{ugIntroYouPageEmpty11}
\pastebutton{ugIntroYouPageFull11}{\hidepaste}
\tab{5}\spadcommand{permMat(n, i, j) ==
  m := diagonalMatrix
    [(if i = k or j = k then 0 else 1)
      for k in 1..n]
  m(i,j) := 1
  m(j,i) := 1
  m
\bound{permMat }}
\indentrel{3}\begin{verbatim}
                                               Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroYouPageEmpty11}
\begin{paste}{ugIntroYouPageEmpty11}{ugIntroYouPagePatch11}
\pastebutton{ugIntroYouPageEmpty11}{\showpaste}
\tab{5}\spadcommand{permMat(n, i, j) ==
  m := diagonalMatrix
    [(if i = k or j = k then 0 else 1)
      for k in 1..n]
  m(i,j) := 1
  m(j,i) := 1
  m
\bound{permMat }}
\end{paste}\end{patch}

\begin{patch}{ugIntroYouPagePatch12}
\begin{paste}{ugIntroYouPageFull12}{ugIntroYouPageEmpty12}
\pastebutton{ugIntroYouPageFull12}{\hidepaste}
\tab{5}\spadcommand{p := permMat(4,2,3)\free{permMat }\bound{p }}
\indentrel{3}\begin{verbatim}
         1  0  0  0

         0  0  1  0
   (12)
         0  1  0  0
```

```
        0  0  0  1
                                   Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroYouPageEmpty12}
\begin{paste}{ugIntroYouPageEmpty12}{ugIntroYouPagePatch12}
\pastebutton{ugIntroYouPageEmpty12}{\showpaste}
\tab{5}\spadcommand{p := permMat(4,2,3)\free{permMat }\bound{p }}
\end{paste}\end{patch}

\begin{patch}{ugIntroYouPagePatch13}
\begin{paste}{ugIntroYouPageFull13}{ugIntroYouPageEmpty13}
\pastebutton{ugIntroYouPageFull13}{\hidepaste}
\tab{5}\spadcommand{m := matrix [[4*i + j for j in 1..4] for i in 0..3]\bound{m }}
\indentrel{3}\begin{verbatim}
        1   2   3   4

        5   6   7   8
   (13)
        9   10  11  12

        13  14  15  16
                                   Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroYouPageEmpty13}
\begin{paste}{ugIntroYouPageEmpty13}{ugIntroYouPagePatch13}
\pastebutton{ugIntroYouPageEmpty13}{\showpaste}
\tab{5}\spadcommand{m := matrix [[4*i + j for j in 1..4] for i in 0..3]\bound{m }}
\end{paste}\end{patch}

\begin{patch}{ugIntroYouPagePatch14}
\begin{paste}{ugIntroYouPageFull14}{ugIntroYouPageEmpty14}
\pastebutton{ugIntroYouPageFull14}{\hidepaste}
\tab{5}\spadcommand{permMat(4,2,3) * m\free{p m }}
\indentrel{3}\begin{verbatim}
        1   2   3   4

        9   10  11  12
   (14)
        5   6   7   8

        13  14  15  16
                                   Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroYouPageEmpty14}
```

```
\begin{paste}{ugIntroYouPageEmpty14}{ugIntroYouPagePatch14}
\pastebutton{ugIntroYouPageEmpty14}{\showpaste}
\tab{5}\spadcommand{permMat(4,2,3) * m\free{p m }}
\end{paste}\end{patch}

\begin{patch}{ugIntroYouPagePatch15}
\begin{paste}{ugIntroYouPageFull15}{ugIntroYouPageEmpty15}
\pastebutton{ugIntroYouPageFull15}{\hidepaste}
\tab{5}\spadcommand{t : (Float -> Float, Float) -> Float\bound{tdecl }}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroYouPageEmpty15}
\begin{paste}{ugIntroYouPageEmpty15}{ugIntroYouPagePatch15}
\pastebutton{ugIntroYouPageEmpty15}{\showpaste}
\tab{5}\spadcommand{t : (Float -> Float, Float) -> Float\bound{tdecl }}
\end{paste}\end{patch}

\begin{patch}{ugIntroYouPagePatch16}
\begin{paste}{ugIntroYouPageFull16}{ugIntroYouPageEmpty16}
\pastebutton{ugIntroYouPageFull16}{\hidepaste}
\tab{5}\spadcommand{t(fun, x) == fun(x)**2 + sin(x)**2\free{tdecl }\bound{t }}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroYouPageEmpty16}
\begin{paste}{ugIntroYouPageEmpty16}{ugIntroYouPagePatch16}
\pastebutton{ugIntroYouPageEmpty16}{\showpaste}
\tab{5}\spadcommand{t(fun, x) == fun(x)**2 + sin(x)**2\free{tdecl }\bound{t }}
\end{paste}\end{patch}

\begin{patch}{ugIntroYouPagePatch17}
\begin{paste}{ugIntroYouPageFull17}{ugIntroYouPageEmpty17}
\pastebutton{ugIntroYouPageFull17}{\hidepaste}
\tab{5}\spadcommand{t(cos, 5.2058)\free{t }}
\indentrel{3}\begin{verbatim}
   (17)  1.0
                                           Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroYouPageEmpty17}
\begin{paste}{ugIntroYouPageEmpty17}{ugIntroYouPagePatch17}
\pastebutton{ugIntroYouPageEmpty17}{\showpaste}
\tab{5}\spadcommand{t(cos, 5.2058)\free{t }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugIntroYouPagePatch18}
\begin{paste}{ugIntroYouPageFull18}{ugIntroYouPageEmpty18}
\pastebutton{ugIntroYouPageFull18}{\hidepaste}
\tab{5}\spadcommand{cosinv(y) == cos(1/y)\bound{cosinv }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroYouPageEmpty18}
\begin{paste}{ugIntroYouPageEmpty18}{ugIntroYouPagePatch18}
\pastebutton{ugIntroYouPageEmpty18}{\showpaste}
\tab{5}\spadcommand{cosinv(y) == cos(1/y)\bound{cosinv }}
\end{paste}\end{patch}

\begin{patch}{ugIntroYouPagePatch19}
\begin{paste}{ugIntroYouPageFull19}{ugIntroYouPageEmpty19}
\pastebutton{ugIntroYouPageFull19}{\hidepaste}
\tab{5}\spadcommand{t(cosinv, 5.2058)\free{t }\free{cosinv }}
\indentrel{3}\begin{verbatim}
   (19)  1.7392237241 800516493
                                            Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroYouPageEmpty19}
\begin{paste}{ugIntroYouPageEmpty19}{ugIntroYouPagePatch19}
\pastebutton{ugIntroYouPageEmpty19}{\showpaste}
\tab{5}\spadcommand{t(cosinv, 5.2058)\free{t }\free{cosinv }}
\end{paste}\end{patch}

\begin{patch}{ugIntroYouPagePatch20}
\begin{paste}{ugIntroYouPageFull20}{ugIntroYouPageEmpty20}
\pastebutton{ugIntroYouPageFull20}{\hidepaste}
\tab{5}\spadcommand{groupSqrt := rule(sqrt(a) * sqrt(b) == sqrt(a*b))\bound{g }}
\indentrel{3}\begin{verbatim}

   (20)  %BD\a \b  == %BD\a b
  Type: RewriteRule(Integer,Integer,Expression Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroYouPageEmpty20}
\begin{paste}{ugIntroYouPageEmpty20}{ugIntroYouPagePatch20}
\pastebutton{ugIntroYouPageEmpty20}{\showpaste}
\tab{5}\spadcommand{groupSqrt := rule(sqrt(a) * sqrt(b) == sqrt(a*b))\bound{g }}
\end{paste}\end{patch}

\begin{patch}{ugIntroYouPagePatch21}
```

```
\begin{paste}{ugIntroYouPageFull21}{ugIntroYouPageEmpty21}
\pastebutton{ugIntroYouPageFull21}{\hidepaste}
\tab{5}\spadcommand{a := (sqrt(x) + sqrt(y) + sqrt(z))**4\bound{sxy }}
\indentrel{3}\begin{verbatim}
   (21)

     ((4z + 4y + 12x)\y  + (4z + 12y + 4x)\x )\z
   +
                          2                 2
     (12z + 4y + 4x)\x \y  + z  + (6y + 6x)z + y  + 6x y
   +
      2
     x
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroYouPageEmpty21}
\begin{paste}{ugIntroYouPageEmpty21}{ugIntroYouPagePatch21}
\pastebutton{ugIntroYouPageEmpty21}{\showpaste}
\tab{5}\spadcommand{a := (sqrt(x) + sqrt(y) + sqrt(z))**4\bound{sxy }}
\end{paste}\end{patch}

\begin{patch}{ugIntroYouPagePatch22}
\begin{paste}{ugIntroYouPageFull22}{ugIntroYouPageEmpty22}
\pastebutton{ugIntroYouPageFull22}{\hidepaste}
\tab{5}\spadcommand{groupSqrt a\free{sxy }\free{g }}
\indentrel{3}\begin{verbatim}
   (22)

     (4z + 4y + 12x)\y z  + (4z + 12y + 4x)\x z
   +
                         2                 2
     (12z + 4y + 4x)\x y  + z  + (6y + 6x)z + y  + 6x y
   +
      2
     x
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroYouPageEmpty22}
\begin{paste}{ugIntroYouPageEmpty22}{ugIntroYouPagePatch22}
\pastebutton{ugIntroYouPageEmpty22}{\showpaste}
\tab{5}\spadcommand{groupSqrt a\free{sxy }\free{g }}
\end{paste}\end{patch}
```

Polynomials

— **ug01.ht** —

```
\begin{page}{ugIntroVariablesPage}{1.9. Polynomials}
\beginscroll
%

Polynomials are the commonly used algebraic types in symbolic
computation.
Interactive users of Axiom generally only see one type
of polynomial: \spadtype{Polynomial(R)}.
This type represents polynomials in any number of unspecified
variables over a particular coefficient domain \axiom{R}.
This type represents its coefficients
\spadglossSee{sparsely}{sparse}: only terms with non-zero
coefficients are represented.

In building applications, many other kinds of polynomial
representations are useful.
Polynomials may have one variable or multiple variables, the
variables can be named or unnamed, the coefficients can be stored
sparsely or densely.
So-called ``distributed multivariate polynomials'' store
polynomials as coefficients paired with vectors of exponents.
This type is particularly efficient for use in algorithms for
solving systems of non-linear polynomial equations.

\xtc{
The polynomial constructor most familiar to the interactive user
is \spadtype{Polynomial}.
}{
\spadpaste{(x**2 - x*y**3 +3*y)**2}
}
\xtc{
If you wish to restrict the variables used,
\spadtype{UnivariatePolynomial}
provides polynomials in one variable.
}{
\spadpaste{p: UP(x,INT) := (3*x-1)**2 * (2*x + 8)}
}
\xtc{
The constructor
\spadtype{MultivariatePolynomial} provides polynomials in one or more
specified variables.
}{
\spadpaste{m: MPOLY([x,y],INT) := (x**2-x*y**3+3*y)**2 \bound{m}}
}
\xtc{
```

```
You can change the way the polynomial appears by modifying the variable
ordering in the explicit list.
}{
\spadpaste{m :: MPOLY([y,x],INT) \free{m}}
}
\xtc{
The constructor
\spadtype{DistributedMultivariatePoly} provides
polynomials in one or more specified variables with the monomials
ordered lexicographically.
}{
\spadpaste{m :: DMP([y,x],INT) \free{m}}
}
\xtc{
The constructor
\spadtype{HomogeneousDistributedMultivariatePoly} is similar
except that the monomials are ordered by total order refined by
reverse lexicographic order.
}{
\spadpaste{m :: HDMP([y,x],INT) \free{m}}
}

More generally, the domain constructor
\spadtype{GeneralDistributedMultivariatePoly} allows the
user to provide an arbitrary predicate to define his own term ordering.
These last three constructors are typically used in
\texht{Gr\"{o}bner}{Groebner} basis
applications and when a flat (that is, non-recursive) display is
wanted and the term ordering is critical for controlling the computation.

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugIntroVariablesPagePatch1}
\begin{paste}{ugIntroVariablesPageFull1}{ugIntroVariablesPageEmpty1}
\pastebutton{ugIntroVariablesPageFull1}{\hidepaste}
\tab{5}\spadcommand{(x**2 - x*y**3 +3*y)**2}
\indentrel{3}\begin{verbatim}
         2 6       4     3 3     2     2     4
   (1)  x y  - 6x y  - 2x y  + 9y  + 6x y + x
                              Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroVariablesPageEmpty1}
\begin{paste}{ugIntroVariablesPageEmpty1}{ugIntroVariablesPagePatch1}
\pastebutton{ugIntroVariablesPageEmpty1}{\showpaste}
\tab{5}\spadcommand{(x**2 - x*y**3 +3*y)**2}
\end{paste}\end{patch}
```

```
\begin{patch}{ugIntroVariablesPagePatch2}
\begin{paste}{ugIntroVariablesPageFull2}{ugIntroVariablesPageEmpty2}
\pastebutton{ugIntroVariablesPageFull2}{\hidepaste}
\tab{5}\spadcommand{p: UP(x,INT) := (3*x-1)**2 * (2*x + 8)}
\indentrel{3}\begin{verbatim}
           3      2
   (2)  18x  + 60x  - 46x + 8
                  Type: UnivariatePolynomial(x,Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroVariablesPageEmpty2}
\begin{paste}{ugIntroVariablesPageEmpty2}{ugIntroVariablesPagePatch2}
\pastebutton{ugIntroVariablesPageEmpty2}{\showpaste}
\tab{5}\spadcommand{p: UP(x,INT) := (3*x-1)**2 * (2*x + 8)}
\end{paste}\end{patch}

\begin{patch}{ugIntroVariablesPagePatch3}
\begin{paste}{ugIntroVariablesPageFull3}{ugIntroVariablesPageEmpty3}
\pastebutton{ugIntroVariablesPageFull3}{\hidepaste}
\tab{5}\spadcommand{m: MPOLY([x,y],INT) := (x**2-x*y**3+3*y)**2\bound{m }}
\indentrel{3}\begin{verbatim}
         4     3 3     6       2     4      2
   (3)  x  - 2y x  + (y  + 6y)x  - 6y x + 9y
            Type: MultivariatePolynomial([x,y],Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroVariablesPageEmpty3}
\begin{paste}{ugIntroVariablesPageEmpty3}{ugIntroVariablesPagePatch3}
\pastebutton{ugIntroVariablesPageEmpty3}{\showpaste}
\tab{5}\spadcommand{m: MPOLY([x,y],INT) := (x**2-x*y**3+3*y)**2\bound{m }}
\end{paste}\end{patch}

\begin{patch}{ugIntroVariablesPagePatch4}
\begin{paste}{ugIntroVariablesPageFull4}{ugIntroVariablesPageEmpty4}
\pastebutton{ugIntroVariablesPageFull4}{\hidepaste}
\tab{5}\spadcommand{m :: MPOLY([y,x],INT)\free{m }}
\indentrel{3}\begin{verbatim}
         2 6       4     3 3     2     2     4
   (4)  x y  - 6x y  - 2x y  + 9y  + 6x y + x
            Type: MultivariatePolynomial([y,x],Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroVariablesPageEmpty4}
\begin{paste}{ugIntroVariablesPageEmpty4}{ugIntroVariablesPagePatch4}
\pastebutton{ugIntroVariablesPageEmpty4}{\showpaste}
\tab{5}\spadcommand{m :: MPOLY([y,x],INT)\free{m }}
```

```
\end{paste}\end{patch}

\begin{patch}{ugIntroVariablesPagePatch5}
\begin{paste}{ugIntroVariablesPageFull5}{ugIntroVariablesPageEmpty5}
\pastebutton{ugIntroVariablesPageFull5}{\hidepaste}
\tab{5}\spadcommand{m :: DMP([y,x],INT)\free{m }}
\indentrel{3}\begin{verbatim}
         6 2     4      3 3     2       2    4
   (5)  y x  - 6y x - 2y x  + 9y  + 6y x  + x
 Type: DistributedMultivariatePolynomial([y,x],Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroVariablesPageEmpty5}
\begin{paste}{ugIntroVariablesPageEmpty5}{ugIntroVariablesPagePatch5}
\pastebutton{ugIntroVariablesPageEmpty5}{\showpaste}
\tab{5}\spadcommand{m :: DMP([y,x],INT)\free{m }}
\end{paste}\end{patch}

\begin{patch}{ugIntroVariablesPagePatch6}
\begin{paste}{ugIntroVariablesPageFull6}{ugIntroVariablesPageEmpty6}
\pastebutton{ugIntroVariablesPageFull6}{\hidepaste}
\tab{5}\spadcommand{m :: HDMP([y,x],INT)\free{m }}
\indentrel{3}\begin{verbatim}
         6 2     3 3     4     4       2     2
   (6)  y x  - 2y x  - 6y x + x  + 6y x  + 9y
Type: HomogeneousDistributedMultivariatePolynomial([y,x],Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroVariablesPageEmpty6}
\begin{paste}{ugIntroVariablesPageEmpty6}{ugIntroVariablesPagePatch6}
\pastebutton{ugIntroVariablesPageEmpty6}{\showpaste}
\tab{5}\spadcommand{m :: HDMP([y,x],INT)\free{m }}
\end{paste}\end{patch}
```

———-

Limits

⇒ “notitle” (ugProblemLimitsPage) 12 on page 2145

— **ug01.ht** —

```
\begin{page}{ugIntroCalcLimitsPage}{1.10. Limits}
\beginscroll
%
```

```
Axiom's \axiomFun{limit} function is usually used to
evaluate limits of quotients where the numerator and denominator
both tend to zero or both tend to infinity.
To find the limit of an expression \axiom{f} as a real variable
\axiom{x} tends to a limit value \axiom{a}, enter \axiom{limit(f, x=a)}.
Use \axiomFun{complexLimit} if the variable is complex.
Additional information and examples of limits are in
\downlink{``Limits''}{ugProblemLimitsPage}
in Section 8.6\ignore{ugProblemLimits}.

\xtc{
You can take limits of functions with parameters.
}{
\spadpaste{g := csc(a*x) / csch(b*x) \bound{g}}
}
\xtc{
As you can see, the limit is expressed in terms of the parameters.
}{
\spadpaste{limit(g,x=0) \free{g}}
}
%
\xtc{
A variable may also approach plus or minus infinity:
}{
\spadpaste{h := (1 + k/x)**x \bound{h}}
}
\xtc{
\texht{Use \axiom{\%plusInfinity} and \axiom{\%minusInfinity} to
denote $\infty$ and $-\infty$.}{}
}{
\spadpaste{limit(h,x=\%plusInfinity) \free{h}}
}
\xtc{
A function can be defined on both sides of a particular value, but may
tend to different limits as its variable approaches that value from
the left and from the right.
}{
\spadpaste{limit(sqrt(y**2)/y,y = 0)}
}
\xtc{
As \axiom{x} approaches \axiom{0} along the real axis, \axiom{exp(-1/x**2)}
tends to \axiom{0}.
}{
\spadpaste{limit(exp(-1/x**2),x = 0)}
}
\xtc{
However, if \axiom{x} is allowed to approach \axiom{0} along any path
in the complex plane, the limiting value of \axiom{exp(-1/x**2)}
depends on the path taken because the function has an essential
singularity at \axiom{x=0}.  This is reflected in the error message
```

```
returned by the function.
}{
\spadpaste{complexLimit(exp(-1/x**2),x = 0)}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugIntroCalcLimitsPagePatch1}
\begin{paste}{ugIntroCalcLimitsPageFull1}{ugIntroCalcLimitsPageEmpty1}
\pastebutton{ugIntroCalcLimitsPageFull1}{\hidepaste}
\tab{5}\spadcommand{g := csc(a*x) / csch(b*x)\bound{g }}
\indentrel{3}\begin{verbatim}
         csc(a x)
   (1)
        csch(b x)
                              Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCalcLimitsPageEmpty1}
\begin{paste}{ugIntroCalcLimitsPageEmpty1}{ugIntroCalcLimitsPagePatch1}
\pastebutton{ugIntroCalcLimitsPageEmpty1}{\showpaste}
\tab{5}\spadcommand{g := csc(a*x) / csch(b*x)\bound{g }}
\end{paste}\end{patch}

\begin{patch}{ugIntroCalcLimitsPagePatch2}
\begin{paste}{ugIntroCalcLimitsPageFull2}{ugIntroCalcLimitsPageEmpty2}
\pastebutton{ugIntroCalcLimitsPageFull2}{\hidepaste}
\tab{5}\spadcommand{limit(g,x=0)\free{g }}
\indentrel{3}\begin{verbatim}
        b
   (2)
        a
  Type: Union(OrderedCompletion Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCalcLimitsPageEmpty2}
\begin{paste}{ugIntroCalcLimitsPageEmpty2}{ugIntroCalcLimitsPagePatch2}
\pastebutton{ugIntroCalcLimitsPageEmpty2}{\showpaste}
\tab{5}\spadcommand{limit(g,x=0)\free{g }}
\end{paste}\end{patch}

\begin{patch}{ugIntroCalcLimitsPagePatch3}
\begin{paste}{ugIntroCalcLimitsPageFull3}{ugIntroCalcLimitsPageEmpty3}
\pastebutton{ugIntroCalcLimitsPageFull3}{\hidepaste}
\tab{5}\spadcommand{h := (1 + k/x)**x\bound{h }}
\indentrel{3}\begin{verbatim}
```

```
         x + k x
   (3)  ()
           x
                                Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCalcLimitsPageEmpty3}
\begin{paste}{ugIntroCalcLimitsPageEmpty3}{ugIntroCalcLimitsPagePatch3}
\pastebutton{ugIntroCalcLimitsPageEmpty3}{\showpaste}
\tab{5}\spadcommand{h := (1 + k/x)**x\bound{h }}
\end{paste}\end{patch}

\begin{patch}{ugIntroCalcLimitsPagePatch4}
\begin{paste}{ugIntroCalcLimitsPageFull4}{ugIntroCalcLimitsPageEmpty4}
\pastebutton{ugIntroCalcLimitsPageFull4}{\hidepaste}
\tab{5}\spadcommand{limit(h,x=\%plusInfinity)\free{h }}
\indentrel{3}\begin{verbatim}
          k
   (4)  %e
  Type: Union(OrderedCompletion Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCalcLimitsPageEmpty4}
\begin{paste}{ugIntroCalcLimitsPageEmpty4}{ugIntroCalcLimitsPagePatch4}
\pastebutton{ugIntroCalcLimitsPageEmpty4}{\showpaste}
\tab{5}\spadcommand{limit(h,x=\%plusInfinity)\free{h }}
\end{paste}\end{patch}

\begin{patch}{ugIntroCalcLimitsPagePatch5}
\begin{paste}{ugIntroCalcLimitsPageFull5}{ugIntroCalcLimitsPageEmpty5}
\pastebutton{ugIntroCalcLimitsPageFull5}{\hidepaste}
\tab{5}\spadcommand{limit(sqrt(y**2)/y,y = 0)}
\indentrel{3}\begin{verbatim}
   (5)  [leftHandLimit= - 1,rightHandLimit= 1]
Type: Union(Record(leftHandLimit: Union(OrderedCompletion Expression Integer,"failed"),rightHand
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCalcLimitsPageEmpty5}
\begin{paste}{ugIntroCalcLimitsPageEmpty5}{ugIntroCalcLimitsPagePatch5}
\pastebutton{ugIntroCalcLimitsPageEmpty5}{\showpaste}
\tab{5}\spadcommand{limit(sqrt(y**2)/y,y = 0)}
\end{paste}\end{patch}

\begin{patch}{ugIntroCalcLimitsPagePatch6}
\begin{paste}{ugIntroCalcLimitsPageFull6}{ugIntroCalcLimitsPageEmpty6}
\pastebutton{ugIntroCalcLimitsPageFull6}{\hidepaste}
\tab{5}\spadcommand{limit(exp(-1/x**2),x = 0)}
```

```
\indentrel{3}\begin{verbatim}
   (6)  0
  Type: Union(OrderedCompletion Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCalcLimitsPageEmpty6}
\begin{paste}{ugIntroCalcLimitsPageEmpty6}{ugIntroCalcLimitsPagePatch6}
\pastebutton{ugIntroCalcLimitsPageEmpty6}{\showpaste}
\tab{5}\spadcommand{limit(exp(-1/x**2),x = 0)}
\end{paste}\end{patch}

\begin{patch}{ugIntroCalcLimitsPagePatch7}
\begin{paste}{ugIntroCalcLimitsPageFull7}{ugIntroCalcLimitsPageEmpty7}
\pastebutton{ugIntroCalcLimitsPageFull7}{\hidepaste}
\tab{5}\spadcommand{complexLimit(exp(-1/x**2),x = 0)}
\indentrel{3}\begin{verbatim}
   (7)  "failed"
                              Type: Union("failed",...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCalcLimitsPageEmpty7}
\begin{paste}{ugIntroCalcLimitsPageEmpty7}{ugIntroCalcLimitsPagePatch7}
\pastebutton{ugIntroCalcLimitsPageEmpty7}{\showpaste}
\tab{5}\spadcommand{complexLimit(exp(-1/x**2),x = 0)}
\end{paste}\end{patch}
```

——–

Series

⇒ “notitle” (ugProblemSeriesPage) 12 on page 2164

— **ug01.ht** —

```
\begin{page}{ugIntroSeriesPage}{1.11. Series}
\beginscroll
%

Axiom also provides power series.
By default, Axiom tries to compute and display the first ten elements
of a series.
Use \spadsys{)set streams calculate} to change the default value
to something else.
For the purposes of this book, we have used this system command to display
fewer than ten terms.
For more information about working with series, see
```

```
\downlink{‘‘Working with Power Series’’}{ugProblemSeriesPage}
in Section 8.9\ignore{ugProblemSeries}.

\xtc{
You can convert a functional expression to a power series by using the
operation \axiomFun{series}.
In this example,
\axiom{sin(a*x)} is expanded in powers of \axiom{(x - 0)},
that is, in powers of \axiom{x}.
}{
\spadpaste{series(sin(a*x),x = 0)}
}
\xtc{
This expression expands
\axiom{sin(a*x)} in powers of \axiom{(x - \%pi/4)}.
}{
\spadpaste{series(sin(a*x),x = \%pi/4)}
}
\xtc{
Axiom provides
{\it Puiseux series:}
series with rational number exponents.
The first argument to \axiomFun{series} is an in-place function that
computes the \eth{\axiom{n}} coefficient.
(Recall that
the \axiomSyntax{+->} is an infix operator meaning ‘‘maps to.’’)
}{
\spadpaste{
series(n +-> (-1)**((3*n - 4)/6)/factorial(n - 1/3),x = 0,4/3..,2)}
}
\xtc{
Once you have created a power series, you can perform arithmetic operations
on that series.
We compute the Taylor expansion of \axiom{1/(1-x)}.
}{
\spadpaste{f := series(1/(1-x),x = 0) \bound{f}}
}
\xtc{
Compute the square of the series.
}{
\spadpaste{f ** 2 \free{f}}
}
\xtc{
The usual elementary functions
(\axiomFun{log}, \axiomFun{exp}, trigonometric functions, and so on)
are defined for power series.
}{
\spadpaste{f := series(1/(1-x),x = 0) \bound{f1}}
}
\xtc{
```

```
}{
\spadpaste{g := log(f) \free{f1}\bound{g}}
}
\xtc{
}{
\spadpaste{exp(g) \free{g}}
}
\xtc{
Here is a way to obtain numerical approximations of
\axiom{e} from the Taylor series expansion of \axiom{exp(x)}.
First create the desired Taylor expansion.
}{
\spadpaste{f := taylor(exp(x)) \bound{f2}}
}
\xtc{
Evaluate the series at the value \axiom{1.0}.
As you see, you get a sequence of partial sums.
}{
\spadpaste{eval(f,1.0) \free{f2}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugIntroSeriesPagePatch1}
\begin{paste}{ugIntroSeriesPageFull1}{ugIntroSeriesPageEmpty1}
\pastebutton{ugIntroSeriesPageFull1}{\hidepaste}
\tab{5}\spadcommand{series(sin(a*x),x = 0)}
\indentrel{3}\begin{verbatim}
   (1)
            3        5        7          9
           a   3    a   5    a    7     a     9
     a x -  x  +  x  -  x  +  x
            6      120      5040      362880
   +
           11
          a      11      12
     -  x   + O(x  )
       39916800
  Type: UnivariatePuiseuxSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroSeriesPageEmpty1}
\begin{paste}{ugIntroSeriesPageEmpty1}{ugIntroSeriesPagePatch1}
\pastebutton{ugIntroSeriesPageEmpty1}{\showpaste}
\tab{5}\spadcommand{series(sin(a*x),x = 0)}
\end{paste}\end{patch}
```

```
\begin{patch}{ugIntroSeriesPagePatch2}
\begin{paste}{ugIntroSeriesPageFull2}{ugIntroSeriesPageEmpty2}
\pastebutton{ugIntroSeriesPageFull2}{\hidepaste}
\tab{5}\spadcommand{series(sin(a*x),x = \%pi/4)}
\indentrel{3}\begin{verbatim}
   (2)
         a %pi          a %pi      %pi
     sin() + a cos()(x - )
           4             4         4
   +
        2    a %pi                3    a %pi
       a sin()              a cos()
               4         %pi 2           4         %pi 3
     -  (x - )  -  (x - )
             2             4            6            4
   +
      4    a %pi                 5    a %pi
     a sin()              a cos()
             4         %pi 4           4         %pi 5
      (x - )  +  (x - )
          24            4           120           4
   +
        6    a %pi                7    a %pi
       a sin()              a cos()
               4         %pi 6           4         %pi 7
     -  (x - )  -  (x - )
           720           4          5040           4
   +
      8    a %pi                 9    a %pi
     a sin()              a cos()
             4         %pi 8           4         %pi 9
      (x - )  +  (x - )
         40320          4         362880          4
   +
        10    a %pi
       a  sin()
                4         %pi 10          %pi 11
     -  (x - )   + O((x - )  )
          3628800         4              4
Type: UnivariatePuiseuxSeries(Expression Integer,x,pi/4)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroSeriesPageEmpty2}
\begin{paste}{ugIntroSeriesPageEmpty2}{ugIntroSeriesPagePatch2}
\pastebutton{ugIntroSeriesPageEmpty2}{\showpaste}
\tab{5}\spadcommand{series(sin(a*x),x = \%pi/4)}
\end{paste}\end{patch}

\begin{patch}{ugIntroSeriesPagePatch3}
```

```
\begin{paste}{ugIntroSeriesPageFull3}{ugIntroSeriesPageEmpty3}
\pastebutton{ugIntroSeriesPageFull3}{\hidepaste}
\tab{5}\spadcommand{series(n +-> (-1)**((3*n - 4)/6)/factorial(n - 1/3),x = 0,4/3..,2)}
\indentrel{3}\begin{verbatim}
         4      10

         3   1   3      5
   (3)  x  -  x   + O(x )
             6
  Type: UnivariatePuiseuxSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroSeriesPageEmpty3}
\begin{paste}{ugIntroSeriesPageEmpty3}{ugIntroSeriesPagePatch3}
\pastebutton{ugIntroSeriesPageEmpty3}{\showpaste}
\tab{5}\spadcommand{series(n +-> (-1)**((3*n - 4)/6)/factorial(n - 1/3),x = 0,4/3..,2)}
\end{paste}\end{patch}

\begin{patch}{ugIntroSeriesPagePatch4}
\begin{paste}{ugIntroSeriesPageFull4}{ugIntroSeriesPageEmpty4}
\pastebutton{ugIntroSeriesPageFull4}{\hidepaste}
\tab{5}\spadcommand{f := series(1/(1-x),x = 0)\bound{f }}
\indentrel{3}\begin{verbatim}
   (4)
             2    3    4    5    6    7    8    9    10
     1 + x + x  + x  + x  + x  + x  + x  + x  + x  + x
   +
        11
     O(x  )
  Type: UnivariatePuiseuxSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroSeriesPageEmpty4}
\begin{paste}{ugIntroSeriesPageEmpty4}{ugIntroSeriesPagePatch4}
\pastebutton{ugIntroSeriesPageEmpty4}{\showpaste}
\tab{5}\spadcommand{f := series(1/(1-x),x = 0)\bound{f }}
\end{paste}\end{patch}

\begin{patch}{ugIntroSeriesPagePatch5}
\begin{paste}{ugIntroSeriesPageFull5}{ugIntroSeriesPageEmpty5}
\pastebutton{ugIntroSeriesPageFull5}{\hidepaste}
\tab{5}\spadcommand{f ** 2\free{f }}
\indentrel{3}\begin{verbatim}
   (5)
               2     3     4     5     6     7     8
     1 + 2x + 3x  + 4x  + 5x  + 6x  + 7x  + 8x  + 9x
   +
        9      10      11
```

```
     10x  + 11x   + O(x  )
  Type: UnivariatePuiseuxSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroSeriesPageEmpty5}
\begin{paste}{ugIntroSeriesPageEmpty5}{ugIntroSeriesPagePatch5}
\pastebutton{ugIntroSeriesPageEmpty5}{\showpaste}
\tab{5}\spadcommand{f ** 2\free{f }}
\end{paste}\end{patch}

\begin{patch}{ugIntroSeriesPagePatch6}
\begin{paste}{ugIntroSeriesPageFull6}{ugIntroSeriesPageEmpty6}
\pastebutton{ugIntroSeriesPageFull6}{\hidepaste}
\tab{5}\spadcommand{f := series(1/(1-x),x = 0)\bound{f1 }}
\indentrel{3}\begin{verbatim}
   (6)
              2    3    4    5    6    7    8    9    10
     1 + x + x  + x  + x  + x  + x  + x  + x  + x  + x
   +
        11
     O(x  )
  Type: UnivariatePuiseuxSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroSeriesPageEmpty6}
\begin{paste}{ugIntroSeriesPageEmpty6}{ugIntroSeriesPagePatch6}
\pastebutton{ugIntroSeriesPageEmpty6}{\showpaste}
\tab{5}\spadcommand{f := series(1/(1-x),x = 0)\bound{f1 }}
\end{paste}\end{patch}

\begin{patch}{ugIntroSeriesPagePatch7}
\begin{paste}{ugIntroSeriesPageFull7}{ugIntroSeriesPageEmpty7}
\pastebutton{ugIntroSeriesPageFull7}{\hidepaste}
\tab{5}\spadcommand{g := log(f)\free{f1 }\bound{g }}
\indentrel{3}\begin{verbatim}
   (7)
        1  2   1  3   1  4   1  5   1  6   1  7   1  8
     x +  x  +  x  +  x  +  x  +  x  +  x  +  x
        2      3      4      5      6      7      8
   +
     1  9    1  10    1  11      12
      x  +  x   +  x   + O(x  )
     9      10       11
  Type: UnivariatePuiseuxSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroSeriesPageEmpty7}
```

```
\begin{paste}{ugIntroSeriesPageEmpty7}{ugIntroSeriesPagePatch7}
\pastebutton{ugIntroSeriesPageEmpty7}{\showpaste}
\tab{5}\spadcommand{g := log(f)\free{f1 }\bound{g }}
\end{paste}\end{patch}

\begin{patch}{ugIntroSeriesPagePatch8}
\begin{paste}{ugIntroSeriesPageFull8}{ugIntroSeriesPageEmpty8}
\pastebutton{ugIntroSeriesPageFull8}{\hidepaste}
\tab{5}\spadcommand{exp(g)\free{g }}
\indentrel{3}\begin{verbatim}
   (8)
              2    3    4    5    6    7    8    9    10
     1 + x + x  + x  + x  + x  + x  + x  + x  + x  + x
   +
        11
     O(x  )
  Type: UnivariatePuiseuxSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroSeriesPageEmpty8}
\begin{paste}{ugIntroSeriesPageEmpty8}{ugIntroSeriesPagePatch8}
\pastebutton{ugIntroSeriesPageEmpty8}{\showpaste}
\tab{5}\spadcommand{exp(g)\free{g }}
\end{paste}\end{patch}

\begin{patch}{ugIntroSeriesPagePatch9}
\begin{paste}{ugIntroSeriesPageFull9}{ugIntroSeriesPageEmpty9}
\pastebutton{ugIntroSeriesPageFull9}{\hidepaste}
\tab{5}\spadcommand{f := taylor(exp(x))\bound{f2 }}
\indentrel{3}\begin{verbatim}
   (9)
             1  2   1  3    1  4    1   5    1   6
     1 + x +  x  +  x  +  x  +  x  +  x
             2      6      24      120      720
   +
     1   7     1    8      1    9      1     10      11
    x  +  x  +  x  +  x   + O(x  )
   5040      40320      362880      3628800
   Type: UnivariateTaylorSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroSeriesPageEmpty9}
\begin{paste}{ugIntroSeriesPageEmpty9}{ugIntroSeriesPagePatch9}
\pastebutton{ugIntroSeriesPageEmpty9}{\showpaste}
\tab{5}\spadcommand{f := taylor(exp(x))\bound{f2 }}
\end{paste}\end{patch}

\begin{patch}{ugIntroSeriesPagePatch10}
```

```
\begin{paste}{ugIntroSeriesPageFull10}{ugIntroSeriesPageEmpty10}
\pastebutton{ugIntroSeriesPageFull10}{\hidepaste}
\tab{5}\spadcommand{eval(f,1.0)\free{f2 }}
\indentrel{3}\begin{verbatim}
   (10)
   [1.0, 2.0, 2.5, 2.6666666666 666666667,
    2.7083333333 333333333, 2.7166666666 666666667,
    2.7180555555 555555556, 2.7182539682 53968254,
    2.7182787698 412698413, 2.7182815255 731922399, ...]
                          Type: Stream Expression Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroSeriesPageEmpty10}
\begin{paste}{ugIntroSeriesPageEmpty10}{ugIntroSeriesPagePatch10}
\pastebutton{ugIntroSeriesPageEmpty10}{\showpaste}
\tab{5}\spadcommand{eval(f,1.0)\free{f2 }}
\end{paste}\end{patch}
```

———-

Derivatives

— ug01.ht —

```
\begin{page}{ugIntroCalcDerivPage}{1.12. Derivatives}
\beginscroll
%
Use the Axiom function \axiomFun{D} to differentiate an
expression.

\texht{\vskip 2pc}{}
\xtc{
To find the derivative of an expression \axiom{f} with respect to a
variable \axiom{x}, enter \axiom{D(f, x)}.
}{
\spadpaste{f := exp exp x \bound{f}}
}
\xtc{
}{
\spadpaste{D(f, x) \free{f}}
}
\xtc{
An optional third argument \axiom{n} in \axiomFun{D} asks
Axiom for the \eth{\axiom{n}} derivative of \axiom{f}.
This finds the fourth derivative of \axiom{f} with respect to \axiom{x}.
}{
```

```
\spadpaste{D(f, x, 4) \free{f}}
}
\xtc{
You can also compute partial derivatives by specifying the order of
differentiation.
}{
\spadpaste{g := sin(x**2 + y) \bound{g}}
}
\xtc{
}{
\spadpaste{D(g, y) \free{g}}
}
\xtc{
}{
\spadpaste{D(g, [y, y, x, x]) \free{g}}
}

Axiom can manipulate the derivatives (partial and iterated) of
expressions involving formal operators.
All the dependencies must be explicit.
\xtc{
This returns \axiom{0} since \axiom{F} (so far)
does not explicitly depend on \axiom{x}.
}{
\spadpaste{D(F,x)}
}
Suppose that we have \axiom{F} a function of \axiom{x},
\axiom{y}, and \axiom{z}, where \axiom{x} and \axiom{y} are themselves
functions of \axiom{z}.
\xtc{
Start by declaring that \axiom{F}, \axiom{x}, and \axiom{y}
are operators.
}{
\spadpaste{
F := operator 'F; x := operator 'x; y := operator 'y\bound{F x y}}
}
\xtc{
You can use \axiom{F}, \axiom{x}, and \axiom{y} in expressions.
}{
\spadpaste{
a := F(x z, y z, z**2) + x y(z+1) \bound{a}\free{F}\free{x}\free{y}}
}
\xtc{
Differentiate formally with respect to \axiom{z}.  The formal
derivatives appearing in \axiom{dadz} are not just formal symbols, but
do represent the derivatives of \axiom{x}, \axiom{y}, and \axiom{F}.
}{
\spadpaste{dadz := D(a, z)\bound{da}\free{a}}
}
\xtc{
```

```
You can evaluate the above for particular functional values of
\axiom{F}, \axiom{x}, and \axiom{y}.  If \axiom{x(z)} is
\axiom{exp(z)} and \axiom{y(z)} is \axiom{log(z+1)}, then this
evaluates \axiom{dadz}.
}{
\spadpaste{eval(eval(dadz, 'x, z +-> exp z), 'y, z +-> log(z+1))\free{da}}
}
\xtc{
You obtain the same result by first evaluating \axiom{a} and
then differentiating.
}{
\spadpaste{eval(eval(a, 'x, z +-> exp z), 'y, z +-> log(z+1)) \free{a}\bound{eva}}
}
\xtc{
}{
\spadpaste{D(\%, z)\free{eva}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugIntroCalcDerivPagePatch1}
\begin{paste}{ugIntroCalcDerivPageFull1}{ugIntroCalcDerivPageEmpty1}
\pastebutton{ugIntroCalcDerivPageFull1}{\hidepaste}
\tab{5}\spadcommand{f := exp exp x\bound{f }}
\indentrel{3}\begin{verbatim}
            x
          %e
   (1)  %e
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCalcDerivPageEmpty1}
\begin{paste}{ugIntroCalcDerivPageEmpty1}{ugIntroCalcDerivPagePatch1}
\pastebutton{ugIntroCalcDerivPageEmpty1}{\showpaste}
\tab{5}\spadcommand{f := exp exp x\bound{f }}
\end{paste}\end{patch}

\begin{patch}{ugIntroCalcDerivPagePatch2}
\begin{paste}{ugIntroCalcDerivPageFull2}{ugIntroCalcDerivPageEmpty2}
\pastebutton{ugIntroCalcDerivPageFull2}{\hidepaste}
\tab{5}\spadcommand{D(f, x)\free{f }}
\indentrel{3}\begin{verbatim}
               x
          x  %e
   (2)  %e %e
                               Type: Expression Integer
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCalcDerivPageEmpty2}
\begin{paste}{ugIntroCalcDerivPageEmpty2}{ugIntroCalcDerivPagePatch2}
\pastebutton{ugIntroCalcDerivPageEmpty2}{\showpaste}
\tab{5}\spadcommand{D(f, x)\free{f }}
\end{paste}\end{patch}

\begin{patch}{ugIntroCalcDerivPagePatch3}
\begin{paste}{ugIntroCalcDerivPageFull3}{ugIntroCalcDerivPageEmpty3}
\pastebutton{ugIntroCalcDerivPageFull3}{\hidepaste}
\tab{5}\spadcommand{D(f, x, 4)\free{f }}
\indentrel{3}\begin{verbatim}
                                                x
            x 4       x 3       x 2     x   %e
   (3)  ((%e )  + 6(%e )  + 7(%e )  + %e )%e
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCalcDerivPageEmpty3}
\begin{paste}{ugIntroCalcDerivPageEmpty3}{ugIntroCalcDerivPagePatch3}
\pastebutton{ugIntroCalcDerivPageEmpty3}{\showpaste}
\tab{5}\spadcommand{D(f, x, 4)\free{f }}
\end{paste}\end{patch}

\begin{patch}{ugIntroCalcDerivPagePatch4}
\begin{paste}{ugIntroCalcDerivPageFull4}{ugIntroCalcDerivPageEmpty4}
\pastebutton{ugIntroCalcDerivPageFull4}{\hidepaste}
\tab{5}\spadcommand{g := sin(x**2 + y)\bound{g }}
\indentrel{3}\begin{verbatim}
                 2
   (4)  sin(y + x )
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCalcDerivPageEmpty4}
\begin{paste}{ugIntroCalcDerivPageEmpty4}{ugIntroCalcDerivPagePatch4}
\pastebutton{ugIntroCalcDerivPageEmpty4}{\showpaste}
\tab{5}\spadcommand{g := sin(x**2 + y)\bound{g }}
\end{paste}\end{patch}

\begin{patch}{ugIntroCalcDerivPagePatch5}
\begin{paste}{ugIntroCalcDerivPageFull5}{ugIntroCalcDerivPageEmpty5}
\pastebutton{ugIntroCalcDerivPageFull5}{\hidepaste}
\tab{5}\spadcommand{D(g, y)\free{g }}
\indentrel{3}\begin{verbatim}
                 2
   (5)  cos(y + x )
```

```
                              Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCalcDerivPageEmpty5}
\begin{paste}{ugIntroCalcDerivPageEmpty5}{ugIntroCalcDerivPagePatch5}
\pastebutton{ugIntroCalcDerivPageEmpty5}{\showpaste}
\tab{5}\spadcommand{D(g, y)\free{g }}
\end{paste}\end{patch}

\begin{patch}{ugIntroCalcDerivPagePatch6}
\begin{paste}{ugIntroCalcDerivPageFull6}{ugIntroCalcDerivPageEmpty6}
\pastebutton{ugIntroCalcDerivPageFull6}{\hidepaste}
\tab{5}\spadcommand{D(g, [y, y, x, x])\free{g }}
\indentrel{3}\begin{verbatim}
         2         2              2
   (6)  4x sin(y + x ) - 2cos(y + x )
                              Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCalcDerivPageEmpty6}
\begin{paste}{ugIntroCalcDerivPageEmpty6}{ugIntroCalcDerivPagePatch6}
\pastebutton{ugIntroCalcDerivPageEmpty6}{\showpaste}
\tab{5}\spadcommand{D(g, [y, y, x, x])\free{g }}
\end{paste}\end{patch}

\begin{patch}{ugIntroCalcDerivPagePatch7}
\begin{paste}{ugIntroCalcDerivPageFull7}{ugIntroCalcDerivPageEmpty7}
\pastebutton{ugIntroCalcDerivPageFull7}{\hidepaste}
\tab{5}\spadcommand{D(F,x)}
\indentrel{3}\begin{verbatim}
   (7)  0
                              Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCalcDerivPageEmpty7}
\begin{paste}{ugIntroCalcDerivPageEmpty7}{ugIntroCalcDerivPagePatch7}
\pastebutton{ugIntroCalcDerivPageEmpty7}{\showpaste}
\tab{5}\spadcommand{D(F,x)}
\end{paste}\end{patch}

\begin{patch}{ugIntroCalcDerivPagePatch8}
\begin{paste}{ugIntroCalcDerivPageFull8}{ugIntroCalcDerivPageEmpty8}
\pastebutton{ugIntroCalcDerivPageFull8}{\hidepaste}
\tab{5}\spadcommand{F := operator 'F; x := operator 'x; y := operator 'y\bound{F x y }}
\indentrel{3}\begin{verbatim}
   (8)  y
                                   Type: BasicOperator
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCalcDerivPageEmpty8}
\begin{paste}{ugIntroCalcDerivPageEmpty8}{ugIntroCalcDerivPagePatch8}
\pastebutton{ugIntroCalcDerivPageEmpty8}{\showpaste}
\tab{5}\spadcommand{F := operator 'F; x := operator 'x; y := operator 'y\bound{F x y }}
\end{paste}\end{patch}

\begin{patch}{ugIntroCalcDerivPagePatch9}
\begin{paste}{ugIntroCalcDerivPageFull9}{ugIntroCalcDerivPageEmpty9}
\pastebutton{ugIntroCalcDerivPageFull9}{\hidepaste}
\tab{5}\spadcommand{a := F(x z, y z, z**2) + x y(z+1)\bound{a }\free{F }\free{x }\free{y }}
\indentrel{3}\begin{verbatim}
                                  2
   (9)  x(y(z + 1)) + F(x(z),y(z),z )
                              Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCalcDerivPageEmpty9}
\begin{paste}{ugIntroCalcDerivPageEmpty9}{ugIntroCalcDerivPagePatch9}
\pastebutton{ugIntroCalcDerivPageEmpty9}{\showpaste}
\tab{5}\spadcommand{a := F(x z, y z, z**2) + x y(z+1)\bound{a }\free{F }\free{x }\free{y }}
\end{paste}\end{patch}

\begin{patch}{ugIntroCalcDerivPagePatch10}
\begin{paste}{ugIntroCalcDerivPageFull10}{ugIntroCalcDerivPageEmpty10}
\pastebutton{ugIntroCalcDerivPageFull10}{\hidepaste}
\tab{5}\spadcommand{dadz := D(a, z)\bound{da }\free{a }}
\indentrel{3}\begin{verbatim}
   (10)
                     2     ,                    2
     2zF  (x(z),y(z),z ) + y (z)F  (x(z),y(z),z )
        ,3                       ,2
   +
      ,                 2     ,            ,
     x (z)F  (x(z),y(z),z ) + x (y(z + 1))y (z + 1)
           ,1
                              Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCalcDerivPageEmpty10}
\begin{paste}{ugIntroCalcDerivPageEmpty10}{ugIntroCalcDerivPagePatch10}
\pastebutton{ugIntroCalcDerivPageEmpty10}{\showpaste}
\tab{5}\spadcommand{dadz := D(a, z)\bound{da }\free{a }}
\end{paste}\end{patch}

\begin{patch}{ugIntroCalcDerivPagePatch11}
```

```
\begin{paste}{ugIntroCalcDerivPageFull11}{ugIntroCalcDerivPageEmpty11}
\pastebutton{ugIntroCalcDerivPageFull11}{\hidepaste}
\tab{5}\spadcommand{eval(eval(dadz, 'x, z +-> exp z), 'y, z +-> log(z+1))\free{da }}
\indentrel{3}\begin{verbatim}
   (11)
         2             z              2
       (2z  + 2z)F  (%e ,log(z + 1),z )
                  ,3
     +
             z              2
       F  (%e ,log(z + 1),z )
        ,2
     +
                z      z              2
       (z + 1)%e F  (%e ,log(z + 1),z ) + z + 1
                  ,1
  /
     z + 1
                                Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCalcDerivPageEmpty11}
\begin{paste}{ugIntroCalcDerivPageEmpty11}{ugIntroCalcDerivPagePatch11}
\pastebutton{ugIntroCalcDerivPageEmpty11}{\showpaste}
\tab{5}\spadcommand{eval(eval(dadz, 'x, z +-> exp z), 'y, z +-> log(z+1))\free{da }}
\end{paste}\end{patch}

\begin{patch}{ugIntroCalcDerivPagePatch12}
\begin{paste}{ugIntroCalcDerivPageFull12}{ugIntroCalcDerivPageEmpty12}
\pastebutton{ugIntroCalcDerivPageFull12}{\hidepaste}
\tab{5}\spadcommand{eval(eval(a, 'x, z +-> exp z), 'y, z +-> log(z+1))\free{a }\bound{eva }}
\indentrel{3}\begin{verbatim}
             z              2
   (12)  F(%e ,log(z + 1),z ) + z + 2
                                Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCalcDerivPageEmpty12}
\begin{paste}{ugIntroCalcDerivPageEmpty12}{ugIntroCalcDerivPagePatch12}
\pastebutton{ugIntroCalcDerivPageEmpty12}{\showpaste}
\tab{5}\spadcommand{eval(eval(a, 'x, z +-> exp z), 'y, z +-> log(z+1))\free{a }\bound{eva }}
\end{paste}\end{patch}

\begin{patch}{ugIntroCalcDerivPagePatch13}
\begin{paste}{ugIntroCalcDerivPageFull13}{ugIntroCalcDerivPageEmpty13}
\pastebutton{ugIntroCalcDerivPageFull13}{\hidepaste}
\tab{5}\spadcommand{D(\%, z)\free{eva }}
\indentrel{3}\begin{verbatim}
```

```
   (13)
          2           z            2
      (2z  + 2z)F  (%e ,log(z + 1),z )
                 ,3
    +
           z             2
      F  (%e ,log(z + 1),z )
       ,2
    +
              z      z            2
      (z + 1)%e F  (%e ,log(z + 1),z ) + z + 1
                 ,1
  /
    z + 1
                                 Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroCalcDerivPageEmpty13}
\begin{paste}{ugIntroCalcDerivPageEmpty13}{ugIntroCalcDerivPagePatch13}
\pastebutton{ugIntroCalcDerivPageEmpty13}{\showpaste}
\tab{5}\spadcommand{D(\%, z)\free{eva }}
\end{paste}\end{patch}
```

———-

Integration

⇒ "notitle" (ugProblemIntegrationPage) 12 on page 2157

— **ug01.ht** —

```
\begin{page}{ugIntroIntegratePage}{1.13. Integration}
\beginscroll
%

Axiom has extensive library facilities for integration.

The first example is the integration of a fraction with denominator
that factors into a quadratic and a quartic irreducible polynomial.
The usual partial fraction approach used by most other computer
algebra systems either fails or introduces expensive unneeded
algebraic numbers.

\xtc{
We use a factorization-free algorithm.
}{
\spadpaste{integrate((x**2+2*x+1)/((x+1)**6+1),x)}
```

```
}

When real parameters are present, the form of the integral can depend on
the signs of some expressions.

\xtc{
Rather than query the user or make sign assumptions, Axiom returns
all possible answers.
}{
\spadpaste{integrate(1/(x**2 + a),x)}
}

The \axiomFun{integrate} operation generally assumes that all
parameters are real.
The only exception is when the integrand has complex valued
quantities.

\xtc{
If the parameter is complex instead of real, then the notion of sign
is undefined and there is a unique answer.  You can request this
answer by ``prepending'' the word ``complex'' to the command name:
}{
\spadpaste{complexIntegrate(1/(x**2 + a),x)}
}

The following two examples illustrate the limitations of table-based
approaches.  The two integrands are very similar, but the answer to
one of them requires the addition of two new algebraic numbers.

\xtc{
This one is the easy one.
The next one looks very similar
but the answer is much more complicated.
}{
\spadpaste{integrate(x**3 / (a+b*x)**(1/3),x)}
}
\xtc{
Only an algorithmic approach
is guaranteed to find what new constants must be added in order to
find a solution.
}{
\spadpaste{integrate(1 / (x**3 * (a+b*x)**(1/3)),x)}
}

Some computer algebra systems use heuristics or table-driven
approaches to integration.  When these systems cannot determine the
answer to an integration problem, they reply ``I don't know.'' Axiom
uses an algorithm for integration.  that conclusively proves that an
integral cannot be expressed in terms of elementary functions.
```

```
\xtc{
When Axiom returns an integral sign, it has proved
that no answer exists as an elementary function.
}{
\spadpaste{integrate(log(1 + sqrt(a*x + b)) / x,x)}
}
Axiom can handle complicated mixed functions much beyond what you
can find in tables.
\xtc{
Whenever possible, Axiom tries to express the answer using the functions
present in the integrand.
}{
\spadpaste{integrate((sinh(1+sqrt(x+b))+2*sqrt(x+b)) /
(sqrt(x+b) * (x + cosh(1+sqrt(x + b)))), x)}
}
\xtc{
A strong structure-checking algorithm in Axiom finds hidden algebraic
relationships between functions.
}{
\spadpaste{integrate(tan(atan(x)/3),x)}
}
\noindent
%%--> Bob---> please make these formulas in this section smaller.
The discovery of this algebraic relationship is necessary for correct
integration of this function.
Here are the details:
\indent{4}
\beginitems
\item[1. ]
If \texht{$x=\tan t$}{\axiom{x=tan(t)}} and
\texht{$g=\tan (t/3)$}{\axiom{g=tan(t/3)}} then the following
algebraic relation is true:
\texht{$${g^3-3xg^2-3g+x=0}$$}
{\centerline{\axiom{g**3 - 3*x*g**2 - 3*g + x = 0}}}
\item[2. ]
Integrate \axiom{g} using this algebraic relation; this produces:
\texht{$${%
{(24g^2 - 8)\log(3g^2 - 1) + (81x^2 + 24)g^2 + 72xg - 27x^2 - 16}
\over{54g^2 - 18}}$$}{\centerline{\axiom{(24g**2 - 8)log(3g**2 - 1) +
(81x**2 + 24)g**2 + 72xg - 27x**2 - 16/ (54g**2 - 18)}}}
\item[3. ]
Rationalize the denominator, producing:
\texht{\narrowDisplay{{8\log(3g^2-1) - 3g^2 + 18xg + 16} \over
{18}}}{\centerline{\axiom{(8*log(3*g**2-1) - 3*g**2 + 18*x*g + 16)/18}}}
Replace \axiom{g} by the initial definition
\texht{$g = \tan(\arctan(x)/3)$}{\axiom{g = tan(arctan(x)/3)}}
to produce the final result.
\enditems
\indent{0}
```

```
\xtc{
This is an example of a mixed function where
the algebraic layer is over the transcendental one.
}{
\spadpaste{integrate((x + 1) / (x*(x + log x) ** (3/2)), x)}
}
\xtc{
While incomplete for non-elementary functions, Axiom can
handle some of them.
}{
\spadpaste{integrate(exp(-x**2) * erf(x) /
(erf(x)**3 - erf(x)**2 - erf(x) + 1),x)}
}

More examples of Axiom's integration capabilities are discussed in
\downlink{``Integration''}{ugProblemIntegrationPage}
in Section 8.8\ignore{ugProblemIntegration}.

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugIntroIntegratePagePatch1}
\begin{paste}{ugIntroIntegratePageFull1}{ugIntroIntegratePageEmpty1}
\pastebutton{ugIntroIntegratePageFull1}{\hidepaste}
\tab{5}\spadcommand{integrate((x**2+2*x+1)/((x+1)**6+1),x)}
\indentrel{3}\begin{verbatim}
              3     2
        atan(x  + 3x  + 3x + 1)
   (1)
                   3
                    Type: Union(Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroIntegratePageEmpty1}
\begin{paste}{ugIntroIntegratePageEmpty1}{ugIntroIntegratePagePatch1}
\pastebutton{ugIntroIntegratePageEmpty1}{\showpaste}
\tab{5}\spadcommand{integrate((x**2+2*x+1)/((x+1)**6+1),x)}
\end{paste}\end{patch}

\begin{patch}{ugIntroIntegratePagePatch2}
\begin{paste}{ugIntroIntegratePageFull2}{ugIntroIntegratePageEmpty2}
\pastebutton{ugIntroIntegratePageFull2}{\hidepaste}
\tab{5}\spadcommand{integrate(1/(x**2 + a),x)}
\indentrel{3}\begin{verbatim}
               2
             (x  - a)\- a  + 2a x
         log()      x\a
                     2            atan()
```

```
                  x  + a                 a
   (2)  [,]

                2\- a               \a
              Type: Union(List Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroIntegratePageEmpty2}
\begin{paste}{ugIntroIntegratePageEmpty2}{ugIntroIntegratePagePatch2}
\pastebutton{ugIntroIntegratePageEmpty2}{\showpaste}
\tab{5}\spadcommand{integrate(1/(x**2 + a),x)}
\end{paste}\end{patch}

\begin{patch}{ugIntroIntegratePagePatch3}
\begin{paste}{ugIntroIntegratePageFull3}{ugIntroIntegratePageEmpty3}
\pastebutton{ugIntroIntegratePageFull3}{\hidepaste}
\tab{5}\spadcommand{complexIntegrate(1/(x**2 + a),x)}
\indentrel{3}\begin{verbatim}

           x\- a  + a        x\- a  - a
        log() - log()

              \- a              \- a
   (3)

                     2\- a
                              Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroIntegratePageEmpty3}
\begin{paste}{ugIntroIntegratePageEmpty3}{ugIntroIntegratePagePatch3}
\pastebutton{ugIntroIntegratePageEmpty3}{\showpaste}
\tab{5}\spadcommand{complexIntegrate(1/(x**2 + a),x)}
\end{paste}\end{patch}

\begin{patch}{ugIntroIntegratePagePatch4}
\begin{paste}{ugIntroIntegratePageFull4}{ugIntroIntegratePageEmpty4}
\pastebutton{ugIntroIntegratePageFull4}{\hidepaste}
\tab{5}\spadcommand{integrate(x**3 / (a+b*x)**(1/3),x)}
\indentrel{3}\begin{verbatim}
   (4)
        3 3         2 2       2          3 32
   (120b x  - 135a b x  + 162a b x - 243a )\b x + a

                              4
                          440b
                    Type: Union(Expression Integer,...)
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroIntegratePageEmpty4}
\begin{paste}{ugIntroIntegratePageEmpty4}{ugIntroIntegratePagePatch4}
\pastebutton{ugIntroIntegratePageEmpty4}{\showpaste}
\tab{5}\spadcommand{integrate(x**3 / (a+b*x)**(1/3),x)}
\end{paste}\end{patch}

\begin{patch}{ugIntroIntegratePagePatch5}
\begin{paste}{ugIntroIntegratePageFull5}{ugIntroIntegratePageEmpty5}
\pastebutton{ugIntroIntegratePageFull5}{\hidepaste}
\tab{5}\spadcommand{integrate(1 / (x**3 * (a+b*x)**(1/3)),x)}
\indentrel{3}\begin{verbatim}
   (5)
       -
              2 2
            2b x \3
         *
                332   32 3
            log(\a \b x + a   + \a   \b x + a + a)
     +
         2 2     32 3
       4b x \3 log(\a   \b x + a - a)
     +
                   32 3
          2 2     2\3 \a   \b x + a + a\3
       12b x atan()
                             3a
     +
                   332
       (12b x - 9a)\3 \a \b x + a
  /
        2 2 3
     18a x \3 \a
                    Type: Union(Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroIntegratePageEmpty5}
\begin{paste}{ugIntroIntegratePageEmpty5}{ugIntroIntegratePagePatch5}
\pastebutton{ugIntroIntegratePageEmpty5}{\showpaste}
\tab{5}\spadcommand{integrate(1 / (x**3 * (a+b*x)**(1/3)),x)}
\end{paste}\end{patch}

\begin{patch}{ugIntroIntegratePagePatch6}
\begin{paste}{ugIntroIntegratePageFull6}{ugIntroIntegratePageEmpty6}
\pastebutton{ugIntroIntegratePageFull6}{\hidepaste}
\tab{5}\spadcommand{integrate(log(1 + sqrt(a*x + b)) / x,x)}
\indentrel{3}\begin{verbatim}
           x
```

```
          log(\b + %S a  + 1)
   (6)      d%S
                   %S
                   Type: Union(Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroIntegratePageEmpty6}
\begin{paste}{ugIntroIntegratePageEmpty6}{ugIntroIntegratePagePatch6}
\pastebutton{ugIntroIntegratePageEmpty6}{\showpaste}
\tab{5}\spadcommand{integrate(log(1 + sqrt(a*x + b)) / x,x)}
\end{paste}\end{patch}

\begin{patch}{ugIntroIntegratePagePatch7}
\begin{paste}{ugIntroIntegratePageFull7}{ugIntroIntegratePageEmpty7}
\pastebutton{ugIntroIntegratePageFull7}{\hidepaste}
\tab{5}\spadcommand{integrate((sinh(1+sqrt(x+b))+2*sqrt(x+b)) / (sqrt(x+b) * (x + cosh(1+sqrt(x + b)))), x)
\indentrel{3}\begin{verbatim}
   (7)

              - 2cosh(\x + b  + 1) - 2x
     2log()

          sinh(\x + b  + 1) - cosh(\x + b  + 1)
   +

     - 2\x + b
                   Type: Union(Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroIntegratePageEmpty7}
\begin{paste}{ugIntroIntegratePageEmpty7}{ugIntroIntegratePagePatch7}
\pastebutton{ugIntroIntegratePageEmpty7}{\showpaste}
\tab{5}\spadcommand{integrate((sinh(1+sqrt(x+b))+2*sqrt(x+b)) / (sqrt(x+b) * (x + cosh(1+sqrt(x + b)))), x)
\end{paste}\end{patch}

\begin{patch}{ugIntroIntegratePagePatch8}
\begin{paste}{ugIntroIntegratePageFull8}{ugIntroIntegratePageEmpty8}
\pastebutton{ugIntroIntegratePageFull8}{\hidepaste}
\tab{5}\spadcommand{integrate(tan(atan(x)/3),x)}
\indentrel{3}\begin{verbatim}
   (8)
              atan(x) 2            atan(x) 2
       8log(3tan()  - 1) - 3tan()
                   3                    3
     +
             atan(x)
       18x tan()
                3
```

```
  /
     18
                    Type: Union(Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroIntegratePageEmpty8}
\begin{paste}{ugIntroIntegratePageEmpty8}{ugIntroIntegratePagePatch8}
\pastebutton{ugIntroIntegratePageEmpty8}{\showpaste}
\tab{5}\spadcommand{integrate(tan(atan(x)/3),x)}
\end{paste}\end{patch}

\begin{patch}{ugIntroIntegratePagePatch9}
\begin{paste}{ugIntroIntegratePageFull9}{ugIntroIntegratePageEmpty9}
\pastebutton{ugIntroIntegratePageFull9}{\hidepaste}
\tab{5}\spadcommand{integrate((x + 1) / (x*(x + log x) ** (3/2)), x)}
\indentrel{3}\begin{verbatim}

          2\log(x) + x
   (9)  -
            log(x) + x
                    Type: Union(Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroIntegratePageEmpty9}
\begin{paste}{ugIntroIntegratePageEmpty9}{ugIntroIntegratePagePatch9}
\pastebutton{ugIntroIntegratePageEmpty9}{\showpaste}
\tab{5}\spadcommand{integrate((x + 1) / (x*(x + log x) ** (3/2)), x)}
\end{paste}\end{patch}

\begin{patch}{ugIntroIntegratePagePatch10}
\begin{paste}{ugIntroIntegratePageFull10}{ugIntroIntegratePageEmpty10}
\pastebutton{ugIntroIntegratePageFull10}{\hidepaste}
\tab{5}\spadcommand{integrate(exp(-x**2) * erf(x) / (erf(x)**3 - erf(x)**2 - erf(x) + 1),x)}
\indentrel{3}\begin{verbatim}
                          erf(x) - 1
         (erf(x) - 1)\%pi log() - 2\%pi
                               erf(x) + 1
   (10)
                         8erf(x) - 8
                    Type: Union(Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroIntegratePageEmpty10}
\begin{paste}{ugIntroIntegratePageEmpty10}{ugIntroIntegratePagePatch10}
\pastebutton{ugIntroIntegratePageEmpty10}{\showpaste}
\tab{5}\spadcommand{integrate(exp(-x**2) * erf(x) / (erf(x)**3 - erf(x)**2 - erf(x) + 1),x)}
\end{paste}\end{patch}
```

———-

Differential Equations

— ug01.ht —

```
\begin{page}{ugIntroDiffEqnsPage}{1.14. Differential Equations}
\beginscroll
%
The general approach used in integration also carries over to the
solution of linear differential equations.

\labelSpace{2pc}
\xtc{
Let's solve some differential equations.
Let \axiom{y} be the unknown function in terms of \axiom{x}.
}{
\spadpaste{y := operator 'y \bound{y}}
}
\xtc{
Here we solve a third order equation with polynomial coefficients.
}{
\spadpaste{deq := x**3 * D(y x, x, 3) + x**2 * D(y x, x, 2) - 2 *
x * D(y x, x) + 2 * y x = 2 * x**4 \bound{e3}\free{y}}
}
\xtc{
}{
\spadpaste{solve(deq, y, x) \free{e3}\free{y}}
}
\xtc{
Here we find all the algebraic function solutions of the equation.
}{
\spadpaste{
deq := (x**2 + 1) * D(y x, x, 2) + 3 * x * D(y x, x) + y x = 0
\bound{e5}\free{y}}
}
\xtc{
}{
\spadpaste{solve(deq, y, x) \free{e5}\free{y}}
}

Coefficients of differential equations can come from arbitrary
constant fields.
For example, coefficients can contain algebraic numbers.

\xtc{
```

```
This example has solutions
whose logarithmic derivative is an algebraic function of
degree two.
}{
\spadpaste{eq := 2*x**3 * D(y x,x,2) + 3*x**2 * D(y x,x) - 2 * y x
\bound{eq}\free{y}}
}
\xtc{
}{
\spadpaste{solve(eq,y,x).basis\free{eq}}
}

\xtc{
Here's another differential equation to solve.
}{
\spadpaste{deq := D(y x, x) = y(x) / (x + y(x) * log y x)
\bound{deqi}\free{y}}
}
\xtc{
}{
\spadpaste{solve(deq, y, x) \free{deqi y}}
}

Rather than attempting to get a closed form solution of
a differential equation, you instead might want to find an
approximate solution in the form of a series.

\xtc{
Let's solve a system of nonlinear first order equations and get a
solution in power series.
Tell Axiom that \axiom{x} is also an operator.
}{
\spadpaste{x := operator 'x\bound{x}}
}
\xtc{
Here are the two equations forming our system.
}{
\spadpaste{eq1 := D(x(t), t) = 1 + x(t)**2\free{x}\free{y}\bound{eq1}}
}
\xtc{
}{
\spadpaste{eq2 := D(y(t), t) = x(t) * y(t)\free{x}\free{y}\bound{eq2}}
}
\xtc{
We can solve the system around \axiom{t = 0} with the initial conditions
\axiom{x(0) = 0} and \axiom{y(0) = 1}.
Notice that since we give the unknowns in the
order \axiom{[x, y]}, the answer is a list of two series in the order
\axiom{[series for x(t), series for y(t)]}.
}{
```

```
\spadpaste{seriesSolve([eq2, eq1], [x, y], t = 0,
[y(0) = 1, x(0) = 0])\free{x}\free{y}\free{eq1}\free{eq2}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugIntroDiffEqnsPagePatch1}
\begin{paste}{ugIntroDiffEqnsPageFull1}{ugIntroDiffEqnsPageEmpty1}
\pastebutton{ugIntroDiffEqnsPageFull1}{\hidepaste}
\tab{5}\spadcommand{y := operator 'y\bound{y }}
\indentrel{3}\begin{verbatim}
   (1)  y
                                   Type: BasicOperator
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroDiffEqnsPageEmpty1}
\begin{paste}{ugIntroDiffEqnsPageEmpty1}{ugIntroDiffEqnsPagePatch1}
\pastebutton{ugIntroDiffEqnsPageEmpty1}{\showpaste}
\tab{5}\spadcommand{y := operator 'y\bound{y }}
\end{paste}\end{patch}

\begin{patch}{ugIntroDiffEqnsPagePatch2}
\begin{paste}{ugIntroDiffEqnsPageFull2}{ugIntroDiffEqnsPageEmpty2}
\pastebutton{ugIntroDiffEqnsPageFull2}{\hidepaste}
\tab{5}\spadcommand{deq := x**3 * D(y x, x, 3) + x**2 * D(y x, x, 2) - 2 * x * D(y x, x) + 2 * y x = 2 * x*
\indentrel{3}\begin{verbatim}
         3 ,,,       2 ,,          ,                4
   (2)  x y   (x) + x y  (x) - 2xy (x) + 2y(x)= 2x

                      Type: Equation Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroDiffEqnsPageEmpty2}
\begin{paste}{ugIntroDiffEqnsPageEmpty2}{ugIntroDiffEqnsPagePatch2}
\pastebutton{ugIntroDiffEqnsPageEmpty2}{\showpaste}
\tab{5}\spadcommand{deq := x**3 * D(y x, x, 3) + x**2 * D(y x, x, 2) - 2 * x * D(y x, x) + 2 * y x = 2 * x*
\end{paste}\end{patch}

\begin{patch}{ugIntroDiffEqnsPagePatch3}
\begin{paste}{ugIntroDiffEqnsPageFull3}{ugIntroDiffEqnsPageEmpty3}
\pastebutton{ugIntroDiffEqnsPageFull3}{\hidepaste}
\tab{5}\spadcommand{solve(deq, y, x)\free{e3 }\free{y }}
\indentrel{3}\begin{verbatim}
   (3)
                 5      3      2
                x  - 10x  + 20x  + 4
```

```
   [particular= ,
                      15x
            3     2      3      3     2
          2x  - 3x  + 1 x  - 1 x  - 3x  - 1
    basis= [,,]]
                  x          x         x
Type: Union(Record(particular: Expression Integer,basis: List Expression Integer),...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroDiffEqnsPageEmpty3}
\begin{paste}{ugIntroDiffEqnsPageEmpty3}{ugIntroDiffEqnsPagePatch3}
\pastebutton{ugIntroDiffEqnsPageEmpty3}{\showpaste}
\tab{5}\spadcommand{solve(deq, y, x)\free{e3 }\free{y }}
\end{paste}\end{patch}

\begin{patch}{ugIntroDiffEqnsPagePatch4}
\begin{paste}{ugIntroDiffEqnsPageFull4}{ugIntroDiffEqnsPageEmpty4}
\pastebutton{ugIntroDiffEqnsPageFull4}{\hidepaste}
\tab{5}\spadcommand{deq := (x**2 + 1) * D(y x, x, 2) + 3 * x * D(y x, x) + y x = 0\bound{e5 }\fr
\indentrel{3}\begin{verbatim}
         2      ,,         ,
   (4)  (x  + 1)y  (x) + 3xy (x) + y(x)= 0

                      Type: Equation Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroDiffEqnsPageEmpty4}
\begin{paste}{ugIntroDiffEqnsPageEmpty4}{ugIntroDiffEqnsPagePatch4}
\pastebutton{ugIntroDiffEqnsPageEmpty4}{\showpaste}
\tab{5}\spadcommand{deq := (x**2 + 1) * D(y x, x, 2) + 3 * x * D(y x, x) + y x = 0\bound{e5 }\fr
\end{paste}\end{patch}

\begin{patch}{ugIntroDiffEqnsPagePatch5}
\begin{paste}{ugIntroDiffEqnsPageFull5}{ugIntroDiffEqnsPageEmpty5}
\pastebutton{ugIntroDiffEqnsPageFull5}{\hidepaste}
\tab{5}\spadcommand{solve(deq, y, x)\free{e5 }\free{y }}
\indentrel{3}\begin{verbatim}
   (5)

                                            2
                              1     log(\x  + 1  - x)
   [particular= 0,basis= [,]]

                            2             2
                          \x  + 1       \x  + 1
Type: Union(Record(particular: Expression Integer,basis: List Expression Integer),...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugIntroDiffEqnsPageEmpty5}
\begin{paste}{ugIntroDiffEqnsPageEmpty5}{ugIntroDiffEqnsPagePatch5}
\pastebutton{ugIntroDiffEqnsPageEmpty5}{\showpaste}
\tab{5}\spadcommand{solve(deq, y, x)\free{e5 }\free{y }}
\end{paste}\end{patch}

\begin{patch}{ugIntroDiffEqnsPagePatch6}
\begin{paste}{ugIntroDiffEqnsPageFull6}{ugIntroDiffEqnsPageEmpty6}
\pastebutton{ugIntroDiffEqnsPageFull6}{\hidepaste}
\tab{5}\spadcommand{eq := 2*x**3 * D(y x,x,2) + 3*x**2 * D(y x,x) - 2 * y x\bound{eq }\free{y }}
\indentrel{3}\begin{verbatim}
          3 ,,        2 ,
   (6)  2x y  (x) + 3x y (x) - 2y(x)

                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroDiffEqnsPageEmpty6}
\begin{paste}{ugIntroDiffEqnsPageEmpty6}{ugIntroDiffEqnsPagePatch6}
\pastebutton{ugIntroDiffEqnsPageEmpty6}{\showpaste}
\tab{5}\spadcommand{eq := 2*x**3 * D(y x,x,2) + 3*x**2 * D(y x,x) - 2 * y x\bound{eq }\free{y }}
\end{paste}\end{patch}

\begin{patch}{ugIntroDiffEqnsPagePatch7}
\begin{paste}{ugIntroDiffEqnsPageFull7}{ugIntroDiffEqnsPageEmpty7}
\pastebutton{ugIntroDiffEqnsPageFull7}{\hidepaste}
\tab{5}\spadcommand{solve(eq,y,x).basis\free{eq }}
\indentrel{3}\begin{verbatim}
               2      2
           -


             \x    \x
   (7)  [%e      ,%e    ]
                          Type: List Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroDiffEqnsPageEmpty7}
\begin{paste}{ugIntroDiffEqnsPageEmpty7}{ugIntroDiffEqnsPagePatch7}
\pastebutton{ugIntroDiffEqnsPageEmpty7}{\showpaste}
\tab{5}\spadcommand{solve(eq,y,x).basis\free{eq }}
\end{paste}\end{patch}

\begin{patch}{ugIntroDiffEqnsPagePatch8}
\begin{paste}{ugIntroDiffEqnsPageFull8}{ugIntroDiffEqnsPageEmpty8}
\pastebutton{ugIntroDiffEqnsPageFull8}{\hidepaste}
\tab{5}\spadcommand{deq := D(y x, x) = y(x) / (x + y(x) * log y x)\bound{deqi }\free{y }}
\indentrel{3}\begin{verbatim}
```

```
        ,            y(x)
   (8)  y (x)=
              y(x)log(y(x)) + x
                     Type: Equation Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroDiffEqnsPageEmpty8}
\begin{paste}{ugIntroDiffEqnsPageEmpty8}{ugIntroDiffEqnsPagePatch8}
\pastebutton{ugIntroDiffEqnsPageEmpty8}{\showpaste}
\tab{5}\spadcommand{deq := D(y x, x) = y(x) / (x + y(x) * log y x)\bound{deqi }\free{y }}
\end{paste}\end{patch}

\begin{patch}{ugIntroDiffEqnsPagePatch9}
\begin{paste}{ugIntroDiffEqnsPageFull9}{ugIntroDiffEqnsPageEmpty9}
\pastebutton{ugIntroDiffEqnsPageFull9}{\hidepaste}
\tab{5}\spadcommand{solve(deq, y, x)\free{deqi y }}
\indentrel{3}\begin{verbatim}
                     2
        y(x)log(y(x))  - 2x
   (9)
              2y(x)
                   Type: Union(Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroDiffEqnsPageEmpty9}
\begin{paste}{ugIntroDiffEqnsPageEmpty9}{ugIntroDiffEqnsPagePatch9}
\pastebutton{ugIntroDiffEqnsPageEmpty9}{\showpaste}
\tab{5}\spadcommand{solve(deq, y, x)\free{deqi y }}
\end{paste}\end{patch}

\begin{patch}{ugIntroDiffEqnsPagePatch10}
\begin{paste}{ugIntroDiffEqnsPageFull10}{ugIntroDiffEqnsPageEmpty10}
\pastebutton{ugIntroDiffEqnsPageFull10}{\hidepaste}
\tab{5}\spadcommand{x := operator 'x\bound{x }}
\indentrel{3}\begin{verbatim}
   (10)  x
                                   Type: BasicOperator
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroDiffEqnsPageEmpty10}
\begin{paste}{ugIntroDiffEqnsPageEmpty10}{ugIntroDiffEqnsPagePatch10}
\pastebutton{ugIntroDiffEqnsPageEmpty10}{\showpaste}
\tab{5}\spadcommand{x := operator 'x\bound{x }}
\end{paste}\end{patch}

\begin{patch}{ugIntroDiffEqnsPagePatch11}
\begin{paste}{ugIntroDiffEqnsPageFull11}{ugIntroDiffEqnsPageEmpty11}
```

```
\pastebutton{ugIntroDiffEqnsPageFull11}{\hidepaste}
\tab{5}\spadcommand{eq1 := D(x(t), t) = 1 + x(t)**2\free{x }\free{y }\bound{eq1 }}
\indentrel{3}\begin{verbatim}
         ,         2
   (11)  x (t)= x(t)  + 1

                      Type: Equation Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroDiffEqnsPageEmpty11}
\begin{paste}{ugIntroDiffEqnsPageEmpty11}{ugIntroDiffEqnsPagePatch11}
\pastebutton{ugIntroDiffEqnsPageEmpty11}{\showpaste}
\tab{5}\spadcommand{eq1 := D(x(t), t) = 1 + x(t)**2\free{x }\free{y }\bound{eq1 }}
\end{paste}\end{patch}

\begin{patch}{ugIntroDiffEqnsPagePatch12}
\begin{paste}{ugIntroDiffEqnsPageFull12}{ugIntroDiffEqnsPageEmpty12}
\pastebutton{ugIntroDiffEqnsPageFull12}{\hidepaste}
\tab{5}\spadcommand{eq2 := D(y(t), t) = x(t) * y(t)\free{x }\free{y }\bound{eq2 }}
\indentrel{3}\begin{verbatim}
         ,
   (12)  y (t)= x(t)y(t)

                      Type: Equation Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroDiffEqnsPageEmpty12}
\begin{paste}{ugIntroDiffEqnsPageEmpty12}{ugIntroDiffEqnsPagePatch12}
\pastebutton{ugIntroDiffEqnsPageEmpty12}{\showpaste}
\tab{5}\spadcommand{eq2 := D(y(t), t) = x(t) * y(t)\free{x }\free{y }\bound{eq2 }}
\end{paste}\end{patch}

\begin{patch}{ugIntroDiffEqnsPagePatch13}
\begin{paste}{ugIntroDiffEqnsPageFull13}{ugIntroDiffEqnsPageEmpty13}
\pastebutton{ugIntroDiffEqnsPageFull13}{\hidepaste}
\tab{5}\spadcommand{seriesSolve([eq2, eq1], [x, y], t = 0, [y(0) = 1, x(0) = 0])\free{x }\free{y }\free{eq1
\indentrel{3}\begin{verbatim}
   (13)
        1  3    2  5    17  7    62   9      11
   [t + - t  +  -- t  + --- t  + ---- t  + O(t  ),
        3       15      315      2835

           1  2    5  4    61  6    277  8    50521   10
      1 +  - t  +  -- t  + --- t  + ---- t  + ------- t
           2       24      720      8064      3628800
    +
          11
      O(t  )
```

```
     ]
Type: List UnivariateTaylorSeries(Expression Integer,t,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroDiffEqnsPageEmpty13}
\begin{paste}{ugIntroDiffEqnsPageEmpty13}{ugIntroDiffEqnsPagePatch13}
\pastebutton{ugIntroDiffEqnsPageEmpty13}{\showpaste}
\tab{5}\spadcommand{seriesSolve([eq2, eq1], [x, y], t = 0, [y(0) = 1, x(0) = 0])\free{x }\free{y
\end{paste}\end{patch}
```

———-

Solution of Equations

— ug01.ht —

```
\begin{page}{ugIntroSolutionPage}{1.15. Solution of Equations}
\beginscroll
%

Axiom also has state-of-the-art algorithms for the solution
of systems of polynomial equations.
When the number of equations and unknowns is the same, and you
have no symbolic coefficients, you can use \spadfun{solve} for
real roots and \spadfun{complexSolve} for complex roots.
In each case, you tell Axiom how accurate you want your
result to be.
All operations in the \spadfun{solve} family return answers in
the form of a list of solution sets, where each solution set is a
list of equations.

\xtc{
A system of two equations involving a symbolic
parameter \axiom{t}.
}{
\spadpaste{S(t) == [x**2-2*y**2 - t,x*y-y-5*x + 5]\bound{S1}}
}
\xtc{
Find the real roots of \spad{S(19)} with
rational arithmetic, correct to within \smath{1/10^{20}}.
}{
\spadpaste{solve(S(19),1/10**20)\free{S1}}
}
\xtc{
Find the complex roots of \spad{S(19)} with floating
point coefficients to \spad{20} digits accuracy in the mantissa.
```

```
}{
\spadpaste{complexSolve(S(19),10.e-20)\free{S1}}
}
\xtc{
If a system of equations has symbolic coefficients and you want
a solution in radicals, try \spadfun{radicalSolve}.
}{
\spadpaste{radicalSolve(S(a),[x,y])\free{S1}}
}
For systems of equations with symbolic coefficients, you can apply
\spadfun{solve}, listing the variables that you want Axiom to solve
for.  For polynomial equations, a solution cannot usually be expressed
solely in terms of the other variables.  Instead, the solution is
presented as a ``triangular'' system of equations, where each
polynomial has coefficients involving only the succeeding
variables. This is analogous to converting a linear system of
equations to ``triangular form''.
\xtc{
A system of three equations in five variables.
}{
\spadpaste{eqns := [x**2 - y + z,x**2*z + x**4 - b*y, y**2 *z - a - b*x]
\bound{e}}
}
\xtc{
Solve the system for unknowns \smath{[x,y,z]},
reducing the solution to triangular form.
}{
\spadpaste{solve(eqns,[x,y,z])\free{e}}
}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ugIntroSolutionPagePatch1}
\begin{paste}{ugIntroSolutionPageFull1}{ugIntroSolutionPageEmpty1}
\pastebutton{ugIntroSolutionPageFull1}{\hidepaste}
\tab{5}\spadcommand{S(t) == [x**2-2*y**2 - t,x*y-y-5*x + 5]\bound{S1 }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroSolutionPageEmpty1}
\begin{paste}{ugIntroSolutionPageEmpty1}{ugIntroSolutionPagePatch1}
\pastebutton{ugIntroSolutionPageEmpty1}{\showpaste}
\tab{5}\spadcommand{S(t) == [x**2-2*y**2 - t,x*y-y-5*x + 5]\bound{S1 }}
\end{paste}\end{patch}

\begin{patch}{ugIntroSolutionPagePatch2}
\begin{paste}{ugIntroSolutionPageFull2}{ugIntroSolutionPageEmpty2}
```

```
\pastebutton{ugIntroSolutionPageFull2}{\hidepaste}
\tab{5}\spadcommand{solve(S(19),1/10**20)\free{S1 }}
\indentrel{3}\begin{verbatim}
   (2)
               2451682632253093442511
   [[y= 5,x= - ],
                295147905179352825856
             2451682632253093442511
    [y= 5,x= ]]
              295147905179352825856
   Type: List List Equation Polynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroSolutionPageEmpty2}
\begin{paste}{ugIntroSolutionPageEmpty2}{ugIntroSolutionPagePatch2}
\pastebutton{ugIntroSolutionPageEmpty2}{\showpaste}
\tab{5}\spadcommand{solve(S(19),1/10**20)\free{S1 }}
\end{paste}\end{patch}

\begin{patch}{ugIntroSolutionPagePatch3}
\begin{paste}{ugIntroSolutionPageFull3}{ugIntroSolutionPageEmpty3}
\pastebutton{ugIntroSolutionPageFull3}{\hidepaste}
\tab{5}\spadcommand{complexSolve(S(19),10.e-20)\free{S1 }}
\indentrel{3}\begin{verbatim}
   (3)
   [
     [y= 5.0,
      x= 8.3066238629 1807485256 1669055295 290320373]
     ,

     [y= 5.0,
      x= - 8.3066238629 1807485256 1669055295 290320373]
     ,
    [y= - 3.0 %i,x= 1.0], [y= 3.0 %i,x= 1.0]]
      Type: List List Equation Polynomial Complex Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroSolutionPageEmpty3}
\begin{paste}{ugIntroSolutionPageEmpty3}{ugIntroSolutionPagePatch3}
\pastebutton{ugIntroSolutionPageEmpty3}{\showpaste}
\tab{5}\spadcommand{complexSolve(S(19),10.e-20)\free{S1 }}
\end{paste}\end{patch}

\begin{patch}{ugIntroSolutionPagePatch4}
\begin{paste}{ugIntroSolutionPageFull4}{ugIntroSolutionPageEmpty4}
\pastebutton{ugIntroSolutionPageFull4}{\hidepaste}
\tab{5}\spadcommand{radicalSolve(S(a),[x,y])\free{S1 }}
\indentrel{3}\begin{verbatim}
```

```
   (4)

   [[x= - \a + 50 ,y= 5], [x= \a + 50 ,y= 5],

              - a + 1                - a + 1
    [x= 1,y=   ], [x= 1,y= -   ]]
             \   2                  \   2
           Type: List List Equation Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroSolutionPageEmpty4}
\begin{paste}{ugIntroSolutionPageEmpty4}{ugIntroSolutionPagePatch4}
\pastebutton{ugIntroSolutionPageEmpty4}{\showpaste}
\tab{5}\spadcommand{radicalSolve(S(a),[x,y])\free{S1 }}
\end{paste}\end{patch}

\begin{patch}{ugIntroSolutionPagePatch5}
\begin{paste}{ugIntroSolutionPageFull5}{ugIntroSolutionPageEmpty5}
\pastebutton{ugIntroSolutionPageFull5}{\hidepaste}
\tab{5}\spadcommand{eqns := [x**2 - y + z,x**2*z + x**4 - b*y, y**2 *z - a - b*x]\bound{e }}
\indentrel{3}\begin{verbatim}
                 2  2           4  2
   (5)  [z - y + x ,x z - b y + x ,y z - b x - a]
                          Type: List Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroSolutionPageEmpty5}
\begin{paste}{ugIntroSolutionPageEmpty5}{ugIntroSolutionPagePatch5}
\pastebutton{ugIntroSolutionPageEmpty5}{\showpaste}
\tab{5}\spadcommand{eqns := [x**2 - y + z,x**2*z + x**4 - b*y, y**2 *z - a - b*x]\bound{e }}
\end{paste}\end{patch}

\begin{patch}{ugIntroSolutionPagePatch6}
\begin{paste}{ugIntroSolutionPageFull6}{ugIntroSolutionPageEmpty6}
\pastebutton{ugIntroSolutionPageFull6}{\hidepaste}
\tab{5}\spadcommand{solve(eqns,[x,y,z])\free{e }}
\indentrel{3}\begin{verbatim}
   (6)
                    2
         a          a
   [[x= - ,y= 0,z= - ],
         b          2
                    b

         3       2    2
        z  + 2b z  + b z - a
     [x= , y= z + b,
                 b
```

```
          6       5     2 4      3       3     4         2
         z  + 4b z  + 6b z  + (4b  - 2a)z  + (b  - 4a b)z
       +
              2     3    2
         - 2a b z - b  + a
         =
         0
       ]
     ]
   Type: List List Equation Fraction Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroSolutionPageEmpty6}
\begin{paste}{ugIntroSolutionPageEmpty6}{ugIntroSolutionPagePatch6}
\pastebutton{ugIntroSolutionPageEmpty6}{\showpaste}
\tab{5}\spadcommand{solve(eqns,[x,y,z])\free{e }}
\end{paste}\end{patch}
```

———-

System Commands

⇒ "notitle" (ugSysCmdPage) 19 on page 2538

— ug01.ht —

```
\begin{page}{ugIntroSysCmmandsPage}{1.16. System Commands}
\beginscroll
%

We conclude our tour of Axiom with a brief discussion of
\spadgloss{system commands}.  System commands are special statements
that start with a closing parenthesis (\axiomSyntax{)}). They are used
to control or display your Axiom environment, start the Hyperdoc
system, issue operating system commands and leave Axiom.  For example,
\spadsys{)system} is used to issue commands to the operating system
from Axiom.  Here is a brief description of some of these commands.
For more information on specific commands, see
\downlink{``Axiom System Commands''}{ugSysCmdPage} in Appendix B
\ignore{ugSysCmd}.

Perhaps the most important user command is the \spadsys{)clear all}
command that initializes your environment.  Every section and
subsection in this book has an invisible \spadsys{)clear all} that is
read prior to the examples given in the section.  \spadsys{)clear all}
gives you a fresh, empty environment with no user variables defined
```

```
and the step number reset to \axiom{1}.  The \spadsys{)clear} command
can also be used to selectively clear values and properties of system
variables.

Another useful system command is \spadsys{)read}.  A preferred way to
develop an application in Axiom is to put your interactive commands
into a file, say {\bf my.input} file.  To get Axiom to read this file,
you use the system command \spadsys{)read my.input}.  If you need to
make changes to your approach or definitions, go into your favorite
editor, change {\bf my.input}, then \spadsys{)read my.input} again.

Other system commands include: \spadsys{)history}, to display
previous input and/or output lines; \spadsys{)display}, to display
properties and values of workspace variables; and \spadsys{)what}.

\xtc{
Issue \spadsys{)what} to get a list of Axiom objects that
contain a given substring in their name.
}{
\spadpaste{)what operations integrate}
}

%\head{subsection}{Undo}{ugIntroUndo}

A useful system command is \spadcmd{)undo}.
Sometimes while computing interactively with Axiom, you make
a mistake and enter an incorrect definition or assignment.
Or perhaps you
need to try one of several alternative approaches, one after
another, to find the best way to approach an application.
For this, you will find the \spadgloss{undo} facility of
Axiom helpful.

System command \spadsys{)undo n} means ``undo back to step
\axiom{n}''; it restores the values of user variables to those that
existed immediately after input expression \axiom{n} was evaluated.
Similarly, \spadsys{)undo -n} undoes changes caused by the last
\axiom{n} input expressions.  Once you have done an \spadsys{)undo},
you can continue on from there, or make a change and {\bf redo} all
your input expressions from the point of the \spadsys{)undo} forward.
The \spadsys{)undo} is completely general: it changes the environment
like any user expression.  Thus you can \spadsys{)undo} any previous
undo.

Here is a sample dialogue between user and Axiom.
\xtc{
``Let me define
two mutually dependent functions \axiom{f} and \axiom{g} piece-wise.''
}{
\spadpaste{f(0) == 1; g(0) == 1\bound{u1}}
```

```
}
\xtc{
``Here is the general term for \axiom{f}.''
}{
\spadpaste{f(n) == e/2*f(n-1) - x*g(n-1)\bound{u2}\free{u1}}
}
\xtc{
``And here is the general term for \axiom{g}.''
}{
\spadpaste{g(n) == -x*f(n-1) + d/3*g(n-1)\bound{u3}\free{u2}}
}
\xtc{
``What is value of \axiom{f(3)}?''
}{
\spadpaste{f(3)\bound{u4}\free{u3}}
}
\noOutputXtc{
``Hmm, I think I want to define \axiom{f} differently.
Undo to the environment right after I defined \axiom{f}.''
}{
\spadpaste{)undo 2\bound{u5}\free{u4}}
}
\xtc{
``Here is how I think I want \axiom{f} to be defined instead.''
}{
\spadpaste{f(n) == d/3*f(n-1) - x*g(n-1)\bound{u6}\free{u5}}
}
\noOutputXtc{
Redo the computation from expression \axiom{3} forward.
}{
\spadpaste{)undo )redo\bound{u7}\free{u6}}
}
\noOutputXtc{
``I want my old definition of
\axiom{f} after all. Undo the undo and restore
the environment to that immediately after \axiom{(4)}.''
}{
\spadpaste{)undo 4\bound{u8}\free{u7}}
}
\xtc{
``Check that the value of \axiom{f(3)} is restored.''
}{
\spadpaste{f(3)\bound{u9}\free{u8}}
}

After you have gone off on several tangents, then backtracked to
previous points in your conversation using \spadsys{)undo}, you might
want to save all the ``correct'' input commands you issued,
disregarding those undone.  The system command \spadsys{)history
)write mynew.input} writes a clean straight-line program onto the file
```

```
{\bf mynew.input} on your disk.

This concludes your tour of Axiom.
To disembark, issue the system command \spadsys{)quit} to leave Axiom
and return to the operating system.
\endscroll
\autobuttons
\end{page}

\begin{patch}{ugIntroSysCmmandsPagePatch1}
\begin{paste}{ugIntroSysCmmandsPageFull1}{ugIntroSysCmmandsPageEmpty1}
\pastebutton{ugIntroSysCmmandsPageFull1}{\hidepaste}
\tab{5}\spadcommand{)what operations integrate}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroSysCmmandsPageEmpty1}
\begin{paste}{ugIntroSysCmmandsPageEmpty1}{ugIntroSysCmmandsPagePatch1}
\pastebutton{ugIntroSysCmmandsPageEmpty1}{\showpaste}
\tab{5}\spadcommand{)what operations integrate}
\end{paste}\end{patch}

\begin{patch}{ugIntroSysCmmandsPagePatch2}
\begin{paste}{ugIntroSysCmmandsPageFull2}{ugIntroSysCmmandsPageEmpty2}
\pastebutton{ugIntroSysCmmandsPageFull2}{\hidepaste}
\tab{5}\spadcommand{f(0) == 1; g(0) == 1\bound{u1 }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroSysCmmandsPageEmpty2}
\begin{paste}{ugIntroSysCmmandsPageEmpty2}{ugIntroSysCmmandsPagePatch2}
\pastebutton{ugIntroSysCmmandsPageEmpty2}{\showpaste}
\tab{5}\spadcommand{f(0) == 1; g(0) == 1\bound{u1 }}
\end{paste}\end{patch}

\begin{patch}{ugIntroSysCmmandsPagePatch3}
\begin{paste}{ugIntroSysCmmandsPageFull3}{ugIntroSysCmmandsPageEmpty3}
\pastebutton{ugIntroSysCmmandsPageFull3}{\hidepaste}
\tab{5}\spadcommand{f(n) == e/2*f(n-1) - x*g(n-1)\bound{u2 }\free{u1 }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroSysCmmandsPageEmpty3}
\begin{paste}{ugIntroSysCmmandsPageEmpty3}{ugIntroSysCmmandsPagePatch3}
\pastebutton{ugIntroSysCmmandsPageEmpty3}{\showpaste}
```

```
\tab{5}\spadcommand{f(n) == e/2*f(n-1) - x*g(n-1)\bound{u2 }\free{u1 }}
\end{paste}\end{patch}

\begin{patch}{ugIntroSysCmmandsPagePatch4}
\begin{paste}{ugIntroSysCmmandsPageFull4}{ugIntroSysCmmandsPageEmpty4}
\pastebutton{ugIntroSysCmmandsPageFull4}{\hidepaste}
\tab{5}\spadcommand{g(n) == -x*f(n-1) + d/3*g(n-1)\bound{u3 }\free{u2 }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroSysCmmandsPageEmpty4}
\begin{paste}{ugIntroSysCmmandsPageEmpty4}{ugIntroSysCmmandsPagePatch4}
\pastebutton{ugIntroSysCmmandsPageEmpty4}{\showpaste}
\tab{5}\spadcommand{g(n) == -x*f(n-1) + d/3*g(n-1)\bound{u3 }\free{u2 }}
\end{paste}\end{patch}

\begin{patch}{ugIntroSysCmmandsPagePatch5}
\begin{paste}{ugIntroSysCmmandsPageFull5}{ugIntroSysCmmandsPageEmpty5}
\pastebutton{ugIntroSysCmmandsPageFull5}{\hidepaste}
\tab{5}\spadcommand{f(3)\bound{u4 }\free{u3 }}
\indentrel{3}\begin{verbatim}
   (4)
      3        1    2      1  2   1       1  2     1  3
   - x  + (e +  d)x  + (-  e  -  d e -  d )x +  e
               3           4      6       9       8
                      Type: Polynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroSysCmmandsPageEmpty5}
\begin{paste}{ugIntroSysCmmandsPageEmpty5}{ugIntroSysCmmandsPagePatch5}
\pastebutton{ugIntroSysCmmandsPageEmpty5}{\showpaste}
\tab{5}\spadcommand{f(3)\bound{u4 }\free{u3 }}
\end{paste}\end{patch}

\begin{patch}{ugIntroSysCmmandsPagePatch6}
\begin{paste}{ugIntroSysCmmandsPageFull6}{ugIntroSysCmmandsPageEmpty6}
\pastebutton{ugIntroSysCmmandsPageFull6}{\hidepaste}
\tab{5}\spadcommand{)undo 2\bound{u5 }\free{u4 }}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroSysCmmandsPageEmpty6}
\begin{paste}{ugIntroSysCmmandsPageEmpty6}{ugIntroSysCmmandsPagePatch6}
\pastebutton{ugIntroSysCmmandsPageEmpty6}{\showpaste}
\tab{5}\spadcommand{)undo 2\bound{u5 }\free{u4 }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugIntroSysCmmandsPagePatch7}
\begin{paste}{ugIntroSysCmmandsPageFull7}{ugIntroSysCmmandsPageEmpty7}
\pastebutton{ugIntroSysCmmandsPageFull7}{\hidepaste}
\tab{5}\spadcommand{f(n) == d/3*f(n-1) - x*g(n-1)\bound{u6 }\free{u5 }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroSysCmmandsPageEmpty7}
\begin{paste}{ugIntroSysCmmandsPageEmpty7}{ugIntroSysCmmandsPagePatch7}
\pastebutton{ugIntroSysCmmandsPageEmpty7}{\showpaste}
\tab{5}\spadcommand{f(n) == d/3*f(n-1) - x*g(n-1)\bound{u6 }\free{u5 }}
\end{paste}\end{patch}

\begin{patch}{ugIntroSysCmmandsPagePatch8}
\begin{paste}{ugIntroSysCmmandsPageFull8}{ugIntroSysCmmandsPageEmpty8}
\pastebutton{ugIntroSysCmmandsPageFull8}{\hidepaste}
\tab{5}\spadcommand{)undo )redo\bound{u7 }\free{u6 }}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroSysCmmandsPageEmpty8}
\begin{paste}{ugIntroSysCmmandsPageEmpty8}{ugIntroSysCmmandsPagePatch8}
\pastebutton{ugIntroSysCmmandsPageEmpty8}{\showpaste}
\tab{5}\spadcommand{)undo )redo\bound{u7 }\free{u6 }}
\end{paste}\end{patch}

\begin{patch}{ugIntroSysCmmandsPagePatch9}
\begin{paste}{ugIntroSysCmmandsPageFull9}{ugIntroSysCmmandsPageEmpty9}
\pastebutton{ugIntroSysCmmandsPageFull9}{\hidepaste}
\tab{5}\spadcommand{)undo 4\bound{u8 }\free{u7 }}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroSysCmmandsPageEmpty9}
\begin{paste}{ugIntroSysCmmandsPageEmpty9}{ugIntroSysCmmandsPagePatch9}
\pastebutton{ugIntroSysCmmandsPageEmpty9}{\showpaste}
\tab{5}\spadcommand{)undo 4\bound{u8 }\free{u7 }}
\end{paste}\end{patch}

\begin{patch}{ugIntroSysCmmandsPagePatch10}
\begin{paste}{ugIntroSysCmmandsPageFull10}{ugIntroSysCmmandsPageEmpty10}
\pastebutton{ugIntroSysCmmandsPageFull10}{\hidepaste}
\tab{5}\spadcommand{f(3)\bound{u9 }\free{u8 }}
\indentrel{3}\begin{verbatim}
   (6)
```

```
     3        1    2      1  2   1       1  2     1  3
  - x  + (e +  d)x  + (-  e  -  d e -  d )x +  e
              3           4      6       9        8
                     Type: Polynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntroSysCmmandsPageEmpty10}
\begin{paste}{ugIntroSysCmmandsPageEmpty10}{ugIntroSysCmmandsPagePatch10}
\pastebutton{ugIntroSysCmmandsPageEmpty10}{\showpaste}
\tab{5}\spadcommand{f(3)\bound{u9 }\free{u8 }}
\end{paste}\end{patch}
```

Chapter 7

Users Guide Chapter 2 (ug02.ht)

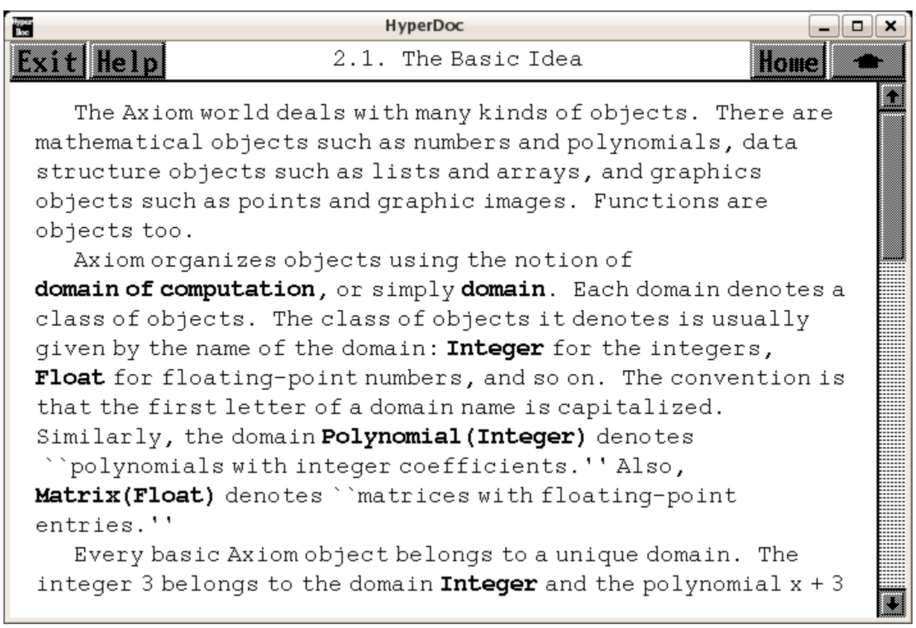
Using Types and Modes

— **ug02.ht** —

```
\begin{page}{ugTypesPage}{2. Using Types and Modes}
\beginscroll

In this chapter we look at the key notion of \spadgloss{type} and its
generalization \spadgloss{mode}.  We show that every Axiom object has
a type that determines what you can do with the object.  In
particular, we explain how to use types to call specific functions
from particular parts of the library and how types and modes can be
used to create new objects from old.  We also look at
\pspadtype{Record} and \pspadtype{Union} types and the special type
\axiomType{Any}.  Finally, we give you an idea of how Axiom
```

```
manipulates types and modes internally to resolve ambiguities.

\beginmenu
    \menudownlink{{2.1. The Basic Idea}}{ugTypesBasicPage}
    \menudownlink{{2.2. Writing Types and Modes}}{ugTypesWritingPage}
    \menudownlink{{2.3. Declarations}}{ugTypesDeclarePage}
    \menudownlink{{2.4. Records}}{ugTypesRecordsPage}
    \menudownlink{{2.5. Unions}}{ugTypesUnionsPage}
    \menudownlink{{2.6. The ‘‘Any’’ Domain}}{ugTypesAnyNonePage}
    \menudownlink{{2.7. Conversion}}{ugTypesConvertPage}
    \menudownlink{{2.8. Subdomains Again}}{ugTypesSubdomainsPage}
    \menudownlink{{2.9. Package Calling and Target Types}}
{ugTypesPkgCallPage}
    \menudownlink{{2.10. Resolving Types}}{ugTypesResolvePage}
    \menudownlink{{2.11. Exposing Domains and Packages}}
{ugTypesExposePage}
    \menudownlink{{2.12. Commands for Snooping}}{ugAvailSnoopPage}
\endmenu
\endscroll
\autobuttons
\end{page}
```

The Basic Idea



⇐ "Conversion" (ugTypesConvertPage) 7 on page 1671
⇒ "Domain Constructors" (ugTypesBasicDomainConsPage) 7 on page 1619
— **ug02.ht** —

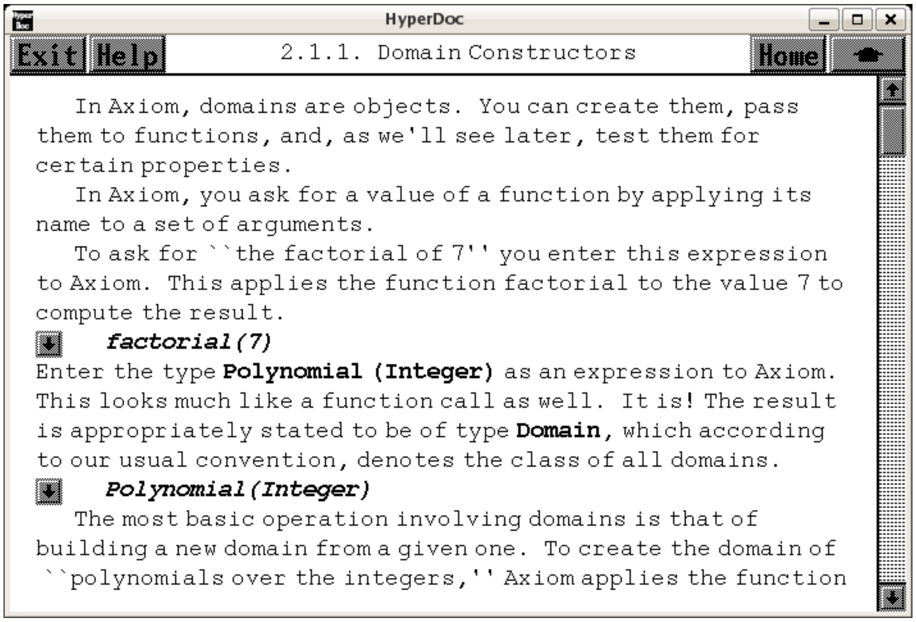
```
\begin{page}{ugTypesBasicPage}{2.1. The Basic Idea}
\beginscroll

The Axiom world deals with many kinds of objects.
There are mathematical objects such as numbers and polynomials,
data structure objects such as lists and arrays, and graphics
objects such as points and graphic images.
Functions are objects too.

Axiom organizes objects using the notion of \spadglossSee{domain of
computation}{domain}, or simply \spadgloss{domain}.  Each domain
denotes a class of objects.  The class of objects it denotes is
usually given by the name of the domain: \axiomType{Integer} for the
integers, \axiomType{Float} for floating-point numbers, and so on.
The convention is that the first letter of a domain name is
capitalized.  Similarly, the domain \axiomType{Polynomial(Integer)}
denotes ``polynomials with integer coefficients.''  Also,
\axiomType{Matrix(Float)} denotes ``matrices with floating-point
entries.''

Every basic Axiom object belongs to a unique domain.  The integer
\axiom{3} belongs to the domain \axiomType{Integer} and the polynomial
\axiom{x + 3} belongs to the domain \axiomType{Polynomial(Integer)}.
The domain of an object is also called its \spadgloss{type}.  Thus we
speak of ``the type \axiomType{Integer}'' and ``the type
\axiomType{Polynomial(Integer)}.''

\xtc{
After an Axiom computation, the type is displayed toward the
right-hand side of the page (or screen).
}{
\spadpaste{-3}
}
\xtc{
Here we create a rational number but it looks like the last result.
The type however tells you it is different.
You cannot identify the type of an object by how Axiom
displays the object.
}{
\spadpaste{-3/1}
}
\xtc{
When a computation produces a result of a simpler type, Axiom leaves
the type unsimplified.
Thus no information is lost.
}{
```

```
\spadpaste{x + 3 - x \bound{three}}
}
\xtc{
This seldom matters since Axiom retracts the answer to the
simpler type if it is necessary.
}{
\spadpaste{factorial(\%) \free{three}}
}
\xtc{
When you issue a positive number, the type \axiomType{PositiveInteger}
is printed.  Surely, \axiom{3} also has type \axiomType{Integer}!  The
curious reader may now have two questions.  First, is the type of an
object not unique?  Second, how is \axiomType{PositiveInteger} related
to \axiomType{Integer}?  Read on!
}{
\spadpaste{3}
}

Any domain can be refined to a \spadgloss{subdomain} by a membership
\spadgloss{predicate}.\footnote{A predicate is a function that, when
applied to an object of the domain, returns either \axiom{true} or
\axiom{false}.}  For example, the domain \axiomType{Integer} can be
refined to the subdomain \axiomType{PositiveInteger}, the set of
integers \axiom{x} such that \axiom{x > 0}, by giving the Axiom
predicate \axiom{x +-> x > 0}.  Similarly, Axiom can define subdomains
such as ``the subdomain of diagonal matrices,'' ``the subdomain of
lists of length two,'' ``the subdomain of monic irreducible
polynomials in \axiom{x},'' and so on.  Trivially, any domain is a
subdomain of itself.

While an object belongs to a unique domain, it can belong to any
number of subdomains.  Any subdomain of the domain of an object can be
used as the {\it type} of that object.  The type of \axiom{3} is
indeed both \axiomType{Integer} and \axiomType{PositiveInteger} as
well as any other subdomain of integer whose predicate is satisfied,
such as ``the prime integers,'' ``the odd positive integers between 3
and 17,'' and so on.

\beginmenu
    \menudownlink{{2.1.1. Domain Constructors}}{ugTypesBasicDomainConsPage}
\endmenu
\endscroll
\autobuttons
\end{page}

\begin{patch}{ugTypesBasicPagePatch1}
\begin{paste}{ugTypesBasicPageFull1}{ugTypesBasicPageEmpty1}
\pastebutton{ugTypesBasicPageFull1}{\hidepaste}
\tab{5}\spadcommand{-3}
\indentrel{3}\begin{verbatim}
```

```
   (1)  - 3
                                          Type: Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesBasicPageEmpty1}
\begin{paste}{ugTypesBasicPageEmpty1}{ugTypesBasicPagePatch1}
\pastebutton{ugTypesBasicPageEmpty1}{\showpaste}
\tab{5}\spadcommand{-3}
\end{paste}\end{patch}

\begin{patch}{ugTypesBasicPagePatch2}
\begin{paste}{ugTypesBasicPageFull2}{ugTypesBasicPageEmpty2}
\pastebutton{ugTypesBasicPageFull2}{\hidepaste}
\tab{5}\spadcommand{-3/1}
\indentrel{3}\begin{verbatim}
   (2)  - 3
                                 Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesBasicPageEmpty2}
\begin{paste}{ugTypesBasicPageEmpty2}{ugTypesBasicPagePatch2}
\pastebutton{ugTypesBasicPageEmpty2}{\showpaste}
\tab{5}\spadcommand{-3/1}
\end{paste}\end{patch}

\begin{patch}{ugTypesBasicPagePatch3}
\begin{paste}{ugTypesBasicPageFull3}{ugTypesBasicPageEmpty3}
\pastebutton{ugTypesBasicPageFull3}{\hidepaste}
\tab{5}\spadcommand{x + 3 - x\bound{three }}
\indentrel{3}\begin{verbatim}
   (3)  3
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesBasicPageEmpty3}
\begin{paste}{ugTypesBasicPageEmpty3}{ugTypesBasicPagePatch3}
\pastebutton{ugTypesBasicPageEmpty3}{\showpaste}
\tab{5}\spadcommand{x + 3 - x\bound{three }}
\end{paste}\end{patch}

\begin{patch}{ugTypesBasicPagePatch4}
\begin{paste}{ugTypesBasicPageFull4}{ugTypesBasicPageEmpty4}
\pastebutton{ugTypesBasicPageFull4}{\hidepaste}
\tab{5}\spadcommand{factorial(\%)\free{three }}
\indentrel{3}\begin{verbatim}
   (4)  6
                               Type: Expression Integer
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesBasicPageEmpty4}
\begin{paste}{ugTypesBasicPageEmpty4}{ugTypesBasicPagePatch4}
\pastebutton{ugTypesBasicPageEmpty4}{\showpaste}
\tab{5}\spadcommand{factorial(\%)\free{three }}
\end{paste}\end{patch}

\begin{patch}{ugTypesBasicPagePatch5}
\begin{paste}{ugTypesBasicPageFull5}{ugTypesBasicPageEmpty5}
\pastebutton{ugTypesBasicPageFull5}{\hidepaste}
\tab{5}\spadcommand{3}
\indentrel{3}\begin{verbatim}
   (5)  3
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesBasicPageEmpty5}
\begin{paste}{ugTypesBasicPageEmpty5}{ugTypesBasicPagePatch5}
\pastebutton{ugTypesBasicPageEmpty5}{\showpaste}
\tab{5}\spadcommand{3}
\end{paste}\end{patch}
```

Domain Constructors



⇐ "The Basic Idea" (ugTypesBasicPage) 7 on page 1614
⇒ "Categories" (ugCategoriesPage) 15 on page 2403
⇒ "Conversion" (ugTypesConvertPage) 7 on page 1671

— **ug02.ht** —

```
\begin{page}{ugTypesBasicDomainConsPage}{2.1.1. Domain Constructors}
\beginscroll

In Axiom, domains are objects.  You can create them, pass them to
functions, and, as we'll see later, test them for certain properties.

In Axiom, you ask for a value of a function by applying its name
to a set of arguments.

\xtc{
To ask for ``the factorial of 7'' you enter this expression to Axiom.
This applies the function \axiom{factorial} to the value \axiom{7} to
compute the result.
}{
\spadpaste{factorial(7)}
}
\xtc{
Enter the type \axiomType{Polynomial (Integer)} as an expression to
Axiom.  This looks much like a function call as well.  It is!  The
result is appropriately stated to be of type \axiomType{Domain}, which
```

```
according to our usual convention, denotes the class of all domains.
}{
\spadpaste{Polynomial(Integer)}
}

The most basic operation involving domains is that of building a new
domain from a given one.  To create the domain of ``polynomials over
the integers,'' Axiom applies the function \axiomType{Polynomial} to
the domain \axiomType{Integer}.  A function like
\axiomType{Polynomial} is called a \spadgloss{domain constructor} or,
more simply, a \spadgloss{constructor}.  A domain constructor is a
function that creates a domain.  An argument to a domain constructor
can be another domain or, in general, an arbitrary kind of object.
\axiomType{Polynomial} takes a single domain argument while
\axiomType{SquareMatrix} takes a positive integer as an argument to
give its dimension and a domain argument to give the type of its
components.

What kinds of domains can you use as the argument to
\axiomType{Polynomial} or \axiomType{SquareMatrix} or
\axiomType{List}?  Well, the first two are mathematical in nature.
You want to be able to perform algebraic operations like \axiomOp{+}
and \axiomOp{*} on polynomials and square matrices, and operations
such as \axiomFun{determinant} on square matrices.  So you want to
allow polynomials of integers {\it and} polynomials of square matrices
with complex number coefficients and, in general, anything that
``makes sense.'' At the same time, you don't want Axiom to be able to
build nonsense domains such as ``polynomials of strings!''

In contrast to algebraic structures, data structures can hold any kind
of object.  Operations on lists such as \axiomFunFrom{insert}{List},
\axiomFunFrom{delete}{List}, and \axiomFunFrom{concat}{List} just
manipulate the list itself without changing or operating on its
elements.  Thus you can build \axiomType{List} over almost any
datatype, including itself.
\xtc{
Create a complicated algebraic domain.
}{
\spadpaste{
List (List (Matrix (Polynomial (Complex (Fraction (Integer))))))}
}
\xtc{
Try to create a meaningless domain.
}{
\spadpaste{Polynomial(String)}
}
Evidently from our last example, Axiom has some mechanism
that tells what a constructor can use as an argument.
This brings us to the notion of \spadgloss{category}.
As domains are objects, they too have a domain.
```

```
The domain of a domain is a category.
A category is simply a type whose members are domains.

A common algebraic category is \axiomType{Ring}, the class of all
domains that are ``rings.''  A ring is an algebraic structure with
constants \axiom{0} and \axiom{1} and operations
\axiomOpFrom{+}{Ring}, \axiomOpFrom{-}{Ring}, and
\axiomOpFrom{*}{Ring}.  These operations are assumed ``closed'' with
respect to the domain, meaning that they take two objects of the
domain and produce a result object also in the domain.  The operations
are understood to satisfy certain ``axioms,'' certain mathematical
principles providing the algebraic foundation for rings.  For example,
the {\it additive inverse axiom} for rings states: \centerline{{Every
element \axiom{x} has an additive inverse \axiom{y} such}}
\centerline{{that \axiom{x + y = 0}.}}  The prototypical example of a
domain that is a ring is the integers.  Keep them in mind whenever we
mention \axiomType{Ring}.

Many algebraic domain constructors such as \axiomType{Complex},
\axiomType{Polynomial}, \axiomType{Fraction}, take rings as arguments
and return rings as values.  You can use the infix operator
``\axiom{has}'' \spadkey{has} to ask a domain if it belongs to a
particular category.

\xtc{
All numerical types are rings.  Domain constructor
\axiomType{Polynomial} builds ``the ring of polynomials over any other
ring.''
}{
\spadpaste{Polynomial(Integer) has Ring}
}
\xtc{
Constructor \axiomType{List} never produces a ring.
}{
\spadpaste{List(Integer) has Ring}
}
\xtc{
The constructor \axiomType{Matrix(R)} builds ``the domain of all
matrices over the ring \axiom{R}.'' This domain is never a ring since
the operations \axiomSyntax{+}, \axiomSyntax{-}, and \axiomSyntax{*}
on matrices of arbitrary shapes are undefined.
}{
\spadpaste{Matrix(Integer) has Ring}
}
\xtc{
Thus you can never build polynomials over matrices.
}{
\spadpaste{Polynomial(Matrix(Integer))}
}
\xtc{
```

```
Use \axiomType{SquareMatrix(n,R)} instead.
For any positive integer \axiom{n}, it builds ``the ring of \axiom{n} by
\axiom{n} matrices over \axiom{R}.''
}{
\spadpaste{Polynomial(SquareMatrix(7,Complex(Integer)))}
}

Another common category is \axiomType{Field}, the class of all fields.
A field is a ring with additional operations.  For example, a field
has commutative multiplication and a closed operation
\axiomOpFrom{/}{Field} for the division of two elements.
\axiomType{Integer} is not a field since, for example, \axiom{3/2}
does not have an integer result.  The prototypical example of a field
is the rational numbers, that is, the domain
\axiomType{Fraction(Integer)}.  In general, the constructor
\axiomType{Fraction} takes a ring as an argument and returns a
field.\footnote{Actually, the argument domain must have some
additional properties so as to belong to category
\axiomType{IntegralDomain}.}  Other domain constructors, such as
\axiomType{Complex}, build fields only if their argument domain is a
field.

\xtc{
The complex integers (often called the ``Gaussian integers'') do not
form a field.
}{
\spadpaste{Complex(Integer) has Field}
}
\xtc{
But fractions of complex integers do.
}{
\spadpaste{Fraction(Complex(Integer)) has Field}
}
\xtc{
The algebraically equivalent domain of complex rational numbers is a
field since domain constructor \axiomType{Complex} produces a field
whenever its argument is a field.
}{
\spadpaste{Complex(Fraction(Integer)) has Field}
}

The most basic category is \axiomType{Type}.  It denotes the class of
all domains and subdomains.\footnote{\axiomType{Type} does not denote
the class of all types.  The type of all categories is
\axiomType{Category}.  The type of \axiomType{Type} itself is
undefined.}  Domain constructor \axiomType{List} is able to build
``lists of elements from domain \axiom{D}'' for arbitrary \axiom{D}
simply by requiring that \axiom{D} belong to category
\axiomType{Type}.
```

```
Now, you may ask, what exactly is a category?
Like domains, categories can be defined in the Axiom language.
A category is defined by three components:
%
\indent{4}
\beginitems
\item[1. ] a name (for example, \axiomType{Ring}),
used to refer to the class of domains that the category represents;
\item[2. ] a set of operations, used to refer to the operations that
the domains of this class support
(for example, \axiomOp{+}, \axiomOp{-}, and \axiomOp{*} for rings); and
\item[3. ] an optional list of other categories that this category extends.
\enditems
\indent{0}
%

This last component is a new idea.  And it is key to the design of
Axiom!  Because categories can extend one another, they form
hierarchies.  \texht{Detailed charts showing the category hierarchies
in Axiom are displayed in the endpages of this book.  There you see
that all categories are extensions of \axiomType{Type} and that
\axiomType{Field} is an extension of \axiomType{Ring}.}{}

The operations supported by the domains of a category are called the
\spadglossSee{exports}{export} of that category because these are the
operations made available for system-wide use.  The exports of a
domain of a given category are not only the ones explicitly mentioned
by the category.  Since a category extends other categories, the
operations of these other categories---and all categories these other
categories extend---are also exported by the domains.

For example, polynomial domains belong to
\axiomType{PolynomialCategory}.  This category explicitly mentions
some twenty-nine operations on polynomials, but it extends eleven
other categories (including \axiomType{Ring}).  As a result, the
current system has over one hundred operations on polynomials.

If a domain belongs to a category that extends, say, \axiomType{Ring},
it is convenient to say that the domain exports \axiomType{Ring}.  The
name of the category thus provides a convenient shorthand for the list
of operations exported by the category.  Rather than listing
operations such as \axiomOpFrom{+}{Ring} and \axiomOpFrom{*}{Ring} of
\axiomType{Ring} each time they are needed, the definition of a type
simply asserts that it exports category \axiomType{Ring}.

The category name, however, is more than a shorthand.
The name \axiomType{Ring}, in fact, implies that the operations exported by
rings are required to satisfy a set of ``axioms'' associated with the name
\axiomType{Ring}.\footnote{This subtle
but important feature distinguishes Axiom from
```

```
other abstract datatype designs.}

Why is it not correct to assume that some type is a ring if it exports
all of the operations of \axiomType{Ring}?  Here is why.  Some
languages such as {\bf APL} denote the \axiomType{Boolean} constants
\axiom{true} and \axiom{false} by the integers \axiom{1} and \axiom{0}
respectively, then use \axiomOp{+} and \axiomOp{*} to denote the
logical operators \axiomFun{or} and \axiomFun{and}.  But with these
definitions \axiomType{Boolean} is not a ring since the additive
inverse axiom is violated.\footnote{There is no inverse element
\axiom{a} such that \axiom{1 + a = 0}, or, in the usual terms:
\axiom{true or a = false}.}  This alternative definition of
\axiomType{Boolean} can be easily and correctly implemented in Axiom,
since \axiomType{Boolean} simply does not assert that it is of
category \axiomType{Ring}.  This prevents the system from building
meaningless domains such as \axiomType{Polynomial(Boolean)} and then
wrongfully applying algorithms that presume that the ring axioms hold.


Enough on categories. To learn more about them, see
\downlink{``Categories''}{ugCategoriesPage}
in Chapter 12\ignore{ugCategories}.
We now return to our discussion of domains.

Domains \spadgloss{export} a set of operations to make them available
for system-wide use.  \axiomType{Integer}, for example, exports the
operations \axiomOpFrom{+}{Integer} and \axiomOpFrom{=}{Integer} given
by the \spadglossSee{signatures}{signature} \axiomOpFrom{+}{Integer}:
\spadsig{(Integer,Integer)}{Integer} and \axiomOpFrom{=}{Integer}:
\spadsig{(Integer,Integer)}{Boolean}, respectively.  Each of these
operations takes two \axiomType{Integer} arguments.  The
\axiomOpFrom{+}{Integer} operation also returns an \axiomType{Integer}
but \axiomOpFrom{=}{Integer} returns a \axiomType{Boolean}:
\axiom{true} or \axiom{false}.  The operations exported by a domain
usually manipulate objects of the domain---but not always.

The operations of a domain may actually take as arguments, and return
as values, objects from any domain.  For example, \axiomType{Fraction
(Integer)} exports the operations \axiomOpFrom{/}{Fraction}:
\spadsig{(Integer,Integer)}{Fraction(Integer)} and
\axiomFunFrom{characteristic}{Fraction}:
\spadsig{}{NonNegativeInteger}.

Suppose all operations of a domain take as arguments and return as
values, only objects from {\it other} domains.  This kind of domain is
what Axiom calls a \spadgloss{package}.

A package does not designate a class of objects at all.  Rather, a
package is just a collection of operations.  Actually the bulk of the
Axiom library of algorithms consists of packages.  The facilities for
```

```
factorization; integration; solution of linear, polynomial, and
differential equations; computation of limits; and so on, are all
defined in packages.  Domains needed by algorithms can be passed to a
package as arguments or used by name if they are not ``variable.''
Packages are useful for defining operations that convert objects of
one type to another, particularly when these types have different
parameterizations.  As an example, the package
\axiomType{PolynomialFunction2(R,S)} defines operations that convert
polynomials over a domain \axiom{R} to polynomials over \axiom{S}.  To
convert an object from \axiomType{Polynomial(Integer)} to
\axiomType{Polynomial(Float)}, Axiom builds the package
\axiomType{PolynomialFunctions2(Integer,Float)} in order to create the
required conversion function.  (This happens ``behind the scenes'' for
you: see \downlink{``Conversion''}{ugTypesConvertPage} in
Section 2.7\ignore{ugTypesConvert} for details on
how to convert objects.)

Axiom categories, domains and packages and all their contained
functions are written in the Axiom programming language and have
been compiled into machine code.
This is what comprises the Axiom \spadgloss{library}.
In the rest of this book we show you how to use these domains and
their functions and how to write your own functions.

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugTypesBasicDomainConsPagePatch1}
\begin{paste}{ugTypesBasicDomainConsPageFull1}{ugTypesBasicDomainConsPageEmpty1}
\pastebutton{ugTypesBasicDomainConsPageFull1}{\hidepaste}
\tab{5}\spadcommand{factorial(7)}
\indentrel{3}\begin{verbatim}
   (1)  5040
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesBasicDomainConsPageEmpty1}
\begin{paste}{ugTypesBasicDomainConsPageEmpty1}{ugTypesBasicDomainConsPagePatch1}
\pastebutton{ugTypesBasicDomainConsPageEmpty1}{\showpaste}
\tab{5}\spadcommand{factorial(7)}
\end{paste}\end{patch}

\begin{patch}{ugTypesBasicDomainConsPagePatch2}
\begin{paste}{ugTypesBasicDomainConsPageFull2}{ugTypesBasicDomainConsPageEmpty2}
\pastebutton{ugTypesBasicDomainConsPageFull2}{\hidepaste}
\tab{5}\spadcommand{Polynomial(Integer)}
\indentrel{3}\begin{verbatim}
   (2)  Polynomial Integer
```

```
                                           Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesBasicDomainConsPageEmpty2}
\begin{paste}{ugTypesBasicDomainConsPageEmpty2}{ugTypesBasicDomainConsPagePatch2}
\pastebutton{ugTypesBasicDomainConsPageEmpty2}{\showpaste}
\tab{5}\spadcommand{Polynomial(Integer)}
\end{paste}\end{patch}

\begin{patch}{ugTypesBasicDomainConsPagePatch3}
\begin{paste}{ugTypesBasicDomainConsPageFull3}{ugTypesBasicDomainConsPageEmpty3}
\pastebutton{ugTypesBasicDomainConsPageFull3}{\hidepaste}
\tab{5}\spadcommand{List (List (Matrix (Polynomial (Complex (Fraction (Integer))))))}
\indentrel{3}\begin{verbatim}
   (3)
   List List Matrix Polynomial Complex Fraction Integer
                                           Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesBasicDomainConsPageEmpty3}
\begin{paste}{ugTypesBasicDomainConsPageEmpty3}{ugTypesBasicDomainConsPagePatch3}
\pastebutton{ugTypesBasicDomainConsPageEmpty3}{\showpaste}
\tab{5}\spadcommand{List (List (Matrix (Polynomial (Complex (Fraction (Integer))))))}
\end{paste}\end{patch}

\begin{patch}{ugTypesBasicDomainConsPagePatch4}
\begin{paste}{ugTypesBasicDomainConsPageFull4}{ugTypesBasicDomainConsPageEmpty4}
\pastebutton{ugTypesBasicDomainConsPageFull4}{\hidepaste}
\tab{5}\spadcommand{Polynomial(String)}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesBasicDomainConsPageEmpty4}
\begin{paste}{ugTypesBasicDomainConsPageEmpty4}{ugTypesBasicDomainConsPagePatch4}
\pastebutton{ugTypesBasicDomainConsPageEmpty4}{\showpaste}
\tab{5}\spadcommand{Polynomial(String)}
\end{paste}\end{patch}

\begin{patch}{ugTypesBasicDomainConsPagePatch5}
\begin{paste}{ugTypesBasicDomainConsPageFull5}{ugTypesBasicDomainConsPageEmpty5}
\pastebutton{ugTypesBasicDomainConsPageFull5}{\hidepaste}
\tab{5}\spadcommand{Polynomial(Integer) has Ring}
\indentrel{3}\begin{verbatim}
   (4)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugTypesBasicDomainConsPageEmpty5}
\begin{paste}{ugTypesBasicDomainConsPageEmpty5}{ugTypesBasicDomainConsPagePatch5}
\pastebutton{ugTypesBasicDomainConsPageEmpty5}{\showpaste}
\tab{5}\spadcommand{Polynomial(Integer) has Ring}
\end{paste}\end{patch}

\begin{patch}{ugTypesBasicDomainConsPagePatch6}
\begin{paste}{ugTypesBasicDomainConsPageFull6}{ugTypesBasicDomainConsPageEmpty6}
\pastebutton{ugTypesBasicDomainConsPageFull6}{\hidepaste}
\tab{5}\spadcommand{List(Integer) has Ring}
\indentrel{3}\begin{verbatim}
   (5)  false
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesBasicDomainConsPageEmpty6}
\begin{paste}{ugTypesBasicDomainConsPageEmpty6}{ugTypesBasicDomainConsPagePatch6}
\pastebutton{ugTypesBasicDomainConsPageEmpty6}{\showpaste}
\tab{5}\spadcommand{List(Integer) has Ring}
\end{paste}\end{patch}

\begin{patch}{ugTypesBasicDomainConsPagePatch7}
\begin{paste}{ugTypesBasicDomainConsPageFull7}{ugTypesBasicDomainConsPageEmpty7}
\pastebutton{ugTypesBasicDomainConsPageFull7}{\hidepaste}
\tab{5}\spadcommand{Matrix(Integer) has Ring}
\indentrel{3}\begin{verbatim}
   (6)  false
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesBasicDomainConsPageEmpty7}
\begin{paste}{ugTypesBasicDomainConsPageEmpty7}{ugTypesBasicDomainConsPagePatch7}
\pastebutton{ugTypesBasicDomainConsPageEmpty7}{\showpaste}
\tab{5}\spadcommand{Matrix(Integer) has Ring}
\end{paste}\end{patch}

\begin{patch}{ugTypesBasicDomainConsPagePatch8}
\begin{paste}{ugTypesBasicDomainConsPageFull8}{ugTypesBasicDomainConsPageEmpty8}
\pastebutton{ugTypesBasicDomainConsPageFull8}{\hidepaste}
\tab{5}\spadcommand{Polynomial(Matrix(Integer))}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesBasicDomainConsPageEmpty8}
\begin{paste}{ugTypesBasicDomainConsPageEmpty8}{ugTypesBasicDomainConsPagePatch8}
\pastebutton{ugTypesBasicDomainConsPageEmpty8}{\showpaste}
```

```
\tab{5}\spadcommand{Polynomial(Matrix(Integer))}
\end{paste}\end{patch}

\begin{patch}{ugTypesBasicDomainConsPagePatch9}
\begin{paste}{ugTypesBasicDomainConsPageFull9}{ugTypesBasicDomainConsPageEmpty9}
\pastebutton{ugTypesBasicDomainConsPageFull9}{\hidepaste}
\tab{5}\spadcommand{Polynomial(SquareMatrix(7,Complex(Integer)))}
\indentrel{3}\begin{verbatim}
   (7)  Polynomial SquareMatrix(7,Complex Integer)
                                          Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesBasicDomainConsPageEmpty9}
\begin{paste}{ugTypesBasicDomainConsPageEmpty9}{ugTypesBasicDomainConsPagePatch9}
\pastebutton{ugTypesBasicDomainConsPageEmpty9}{\showpaste}
\tab{5}\spadcommand{Polynomial(SquareMatrix(7,Complex(Integer)))}
\end{paste}\end{patch}

\begin{patch}{ugTypesBasicDomainConsPagePatch10}
\begin{paste}{ugTypesBasicDomainConsPageFull10}{ugTypesBasicDomainConsPageEmpty10}
\pastebutton{ugTypesBasicDomainConsPageFull10}{\hidepaste}
\tab{5}\spadcommand{Complex(Integer) has Field}
\indentrel{3}\begin{verbatim}
   (8)  false
                                         Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesBasicDomainConsPageEmpty10}
\begin{paste}{ugTypesBasicDomainConsPageEmpty10}{ugTypesBasicDomainConsPagePatch10}
\pastebutton{ugTypesBasicDomainConsPageEmpty10}{\showpaste}
\tab{5}\spadcommand{Complex(Integer) has Field}
\end{paste}\end{patch}

\begin{patch}{ugTypesBasicDomainConsPagePatch11}
\begin{paste}{ugTypesBasicDomainConsPageFull11}{ugTypesBasicDomainConsPageEmpty11}
\pastebutton{ugTypesBasicDomainConsPageFull11}{\hidepaste}
\tab{5}\spadcommand{Fraction(Complex(Integer)) has Field}
\indentrel{3}\begin{verbatim}
   (9)  true
                                         Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesBasicDomainConsPageEmpty11}
\begin{paste}{ugTypesBasicDomainConsPageEmpty11}{ugTypesBasicDomainConsPagePatch11}
\pastebutton{ugTypesBasicDomainConsPageEmpty11}{\showpaste}
\tab{5}\spadcommand{Fraction(Complex(Integer)) has Field}
\end{paste}\end{patch}
```

```
\begin{patch}{ugTypesBasicDomainConsPagePatch12}
\begin{paste}{ugTypesBasicDomainConsPageFull12}{ugTypesBasicDomainConsPageEmpty12}
\pastebutton{ugTypesBasicDomainConsPageFull12}{\hidepaste}
\tab{5}\spadcommand{Complex(Fraction(Integer)) has Field}
\indentrel{3}\begin{verbatim}
   (10)  true
                                            Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesBasicDomainConsPageEmpty12}
\begin{paste}{ugTypesBasicDomainConsPageEmpty12}{ugTypesBasicDomainConsPagePatch12}
\pastebutton{ugTypesBasicDomainConsPageEmpty12}{\showpaste}
\tab{5}\spadcommand{Complex(Fraction(Integer)) has Field}
\end{paste}\end{patch}
```

———-

Writing Types and Modes

⇒ "notitle" (ugTypesBasicPage) 7 on page 1614
⇒ "notitle" (ugTypesDeclarePage) 7 on page 1641
⇒ "notitle" (ugTypesConvertPage) 7 on page 1671
⇒ "notitle" (ugTypesPkgCallPage) 7 on page 1686
⇒ "notitle" (ugTypesWritingZeroPage) 7 on page 1632
⇒ "notitle" (ugTypesWritingOnePage) 7 on page 1633
⇒ "notitle" (ugTypesWritingMorePage) 7 on page 1636
⇒ "notitle" (ugTypesWritingModesPage) 7 on page 1637
⇒ "notitle" (ugTypesWritingAbbrPage) 7 on page 1638

— **ug02.ht** —

```
\begin{page}{ugTypesWritingPage}{2.2. Writing Types and Modes}
\beginscroll
%

We have already seen in \texht{the last
section}{\downlink{``The Basic Idea''}{ugTypesBasicPage} in
Section 2.1\ignore{ugTypesBasic}} several examples of
types.  Most of these examples had either no arguments (for example,
\axiomType{Integer}) or one argument (for example,
\axiomType{Polynomial (Integer)}).  In this section we give details
about writing arbitrary types.  We then define
\spadglossSee{modes}{mode} and discuss how to write them.  We conclude
the section with a discussion on constructor abbreviations.
```

```
\xtc{
When might you need to write a type or mode?
You need to do so when you declare variables.
}{
\spadpaste{a : PositiveInteger}
}
\xtc{
You need to do so when you declare functions
(\downlink{``Declarations''}{ugTypesDeclarePage}
in Section 2.3\ignore{ugTypesDeclare}),
}{
\spadpaste{f : Integer -> String}
}
\xtc{
You need to do so when you convert an object from one type to another
(\downlink{``Conversion''}{ugTypesConvertPage}
in Section 2.7\ignore{ugTypesConvert}).
}{
\spadpaste{factor(2 :: Complex(Integer))}
}
\xtc{
}{
\spadpaste{(2 = 3)\$Integer}
}
\xtc{
You need to do so when you give computation target type information
(\downlink{``Package Calling and Target Types''}{ugTypesPkgCallPage}
in Section 2.9\ignore{ugTypesPkgCall}).
}{
\spadpaste{(2 = 3)@Boolean}
}

\beginmenu
    \menudownlink{{2.2.1. Types with No Arguments}}{ugTypesWritingZeroPage}
    \menudownlink{{2.2.2. Types with One Argument}}{ugTypesWritingOnePage}
    \menudownlink{{2.2.3. Types with More Than One Argument}}
{ugTypesWritingMorePage}
    \menudownlink{{2.2.4. Modes}}{ugTypesWritingModesPage}
    \menudownlink{{2.2.5. Abbreviations}}{ugTypesWritingAbbrPage}
\endmenu
\endscroll
\autobuttons
\end{page}

\begin{patch}{ugTypesWritingPagePatch1}
\begin{paste}{ugTypesWritingPageFull1}{ugTypesWritingPageEmpty1}
\pastebutton{ugTypesWritingPageFull1}{\hidepaste}
\tab{5}\spadcommand{a : PositiveInteger}
\indentrel{3}\begin{verbatim}
                                             Type: Void
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesWritingPageEmpty1}
\begin{paste}{ugTypesWritingPageEmpty1}{ugTypesWritingPagePatch1}
\pastebutton{ugTypesWritingPageEmpty1}{\showpaste}
\tab{5}\spadcommand{a : PositiveInteger}
\end{paste}\end{patch}

\begin{patch}{ugTypesWritingPagePatch2}
\begin{paste}{ugTypesWritingPageFull2}{ugTypesWritingPageEmpty2}
\pastebutton{ugTypesWritingPageFull2}{\hidepaste}
\tab{5}\spadcommand{f : Integer -> String}
\indentrel{3}\begin{verbatim}
                                           Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesWritingPageEmpty2}
\begin{paste}{ugTypesWritingPageEmpty2}{ugTypesWritingPagePatch2}
\pastebutton{ugTypesWritingPageEmpty2}{\showpaste}
\tab{5}\spadcommand{f : Integer -> String}
\end{paste}\end{patch}

\begin{patch}{ugTypesWritingPagePatch3}
\begin{paste}{ugTypesWritingPageFull3}{ugTypesWritingPageEmpty3}
\pastebutton{ugTypesWritingPageFull3}{\hidepaste}
\tab{5}\spadcommand{factor(2 :: Complex(Integer))}
\indentrel{3}\begin{verbatim}
                    2
   (3)  - %i (1 + %i)
                        Type: Factored Complex Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesWritingPageEmpty3}
\begin{paste}{ugTypesWritingPageEmpty3}{ugTypesWritingPagePatch3}
\pastebutton{ugTypesWritingPageEmpty3}{\showpaste}
\tab{5}\spadcommand{factor(2 :: Complex(Integer))}
\end{paste}\end{patch}

\begin{patch}{ugTypesWritingPagePatch4}
\begin{paste}{ugTypesWritingPageFull4}{ugTypesWritingPageEmpty4}
\pastebutton{ugTypesWritingPageFull4}{\hidepaste}
\tab{5}\spadcommand{(2 = 3)$Integer}
\indentrel{3}\begin{verbatim}
   (4)  false
                                        Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugTypesWritingPageEmpty4}
\begin{paste}{ugTypesWritingPageEmpty4}{ugTypesWritingPagePatch4}
\pastebutton{ugTypesWritingPageEmpty4}{\showpaste}
\tab{5}\spadcommand{(2 = 3)$Integer}
\end{paste}\end{patch}

\begin{patch}{ugTypesWritingPagePatch5}
\begin{paste}{ugTypesWritingPageFull5}{ugTypesWritingPageEmpty5}
\pastebutton{ugTypesWritingPageFull5}{\hidepaste}
\tab{5}\spadcommand{(2 = 3)@Boolean}
\indentrel{3}\begin{verbatim}
   (5)  false
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesWritingPageEmpty5}
\begin{paste}{ugTypesWritingPageEmpty5}{ugTypesWritingPagePatch5}
\pastebutton{ugTypesWritingPageEmpty5}{\showpaste}
\tab{5}\spadcommand{(2 = 3)@Boolean}
\end{paste}\end{patch}
```

———-

Types with No Arguments

— ug02.ht —

```
\begin{page}{ugTypesWritingZeroPage}{2.2.1. Types with No Arguments}
\beginscroll

A constructor with no arguments can be written either
with or without
trailing opening and closing parentheses (\axiomSyntax{()}).
\texht{
\centerline{{\begin{tabular}{ccc}}}
\centerline{{\axiomType{Boolean()}
is the same as \axiomType{Boolean} & \quad &}}
\centerline{{\axiomType{Integer()}
is the same as \axiomType{Integer} }}
\centerline{{\axiomType{String()}
is the same as \axiomType{String} & \quad &}}
\centerline{{\axiomType{Void()}
is the same as \axiomType{Void} }}
\centerline{{\end{tabular}}}
}{
```

```
\centerline{{\axiomType{Boolean()}
is the same as \axiomType{Boolean} }}
\centerline{{\axiomType{Integer()}
is the same as \axiomType{Integer} }}
\centerline{{\axiomType{String()}
is the same as \axiomType{String} }}
\centerline{{\axiomType{Void()}
is the same as \axiomType{Void} }}
and so on.
}
It is customary to omit the parentheses.

\endscroll
\autobuttons
\end{page}
```

———-

Types with One Argument

⇒ "notitle" (ugTypesPkgCallPage) 7 on page 1686
— **ug02.ht** —

```
\begin{page}{ugTypesWritingOnePage}{2.2.2. Types with One Argument}
\beginscroll

A constructor with one argument can frequently be written with no
parentheses.  Types nest from right to left so that \axiomType{Complex
Fraction Polynomial Integer} is the same as \axiomType{Complex
(Fraction (Polynomial (Integer)))}.  You need to use parentheses to
force the application of a constructor to the correct argument, but
you need not use any more than is necessary to remove ambiguities.

Here are some guidelines for using parentheses (they are possibly
slightly more restrictive than they need to be).
\xtc{
If the argument is an expression like \axiom{2 + 3}
then you must enclose the argument in parentheses.
}{
\spadpaste{e : PrimeField(2 + 3)}
}
%
\xtc{
If the type is to be used with package calling
then you must enclose the argument in parentheses.
}{
\spadpaste{content(2)\$Polynomial(Integer)}
```

```
}
\xtc{
Alternatively, you can write the type without parentheses
then enclose the whole type expression with parentheses.
}{
\spadpaste{content(2)\$(Polynomial Complex Fraction Integer)}
}
\xtc{
If you supply computation target type information
(\downlink{``Package Calling and Target Types''}{ugTypesPkgCallPage}
in Section 2.9\ignore{ugTypesPkgCall})
then you should enclose the argument in parentheses.
}{
\spadpaste{(2/3)@Fraction(Polynomial(Integer))}
}
%
\xtc{
If the type itself has parentheses around it and we are
not in the case of the first example above,
then the parentheses can usually be omitted.
}{
\spadpaste{(2/3)@Fraction(Polynomial Integer)}
}
%
\xtc{
If the type is used in a declaration and the argument is a single-word
type, integer or symbol,
then the parentheses can usually be omitted.
}{
\spadpaste{(d,f,g) : Complex Polynomial Integer}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugTypesWritingOnePagePatch1}
\begin{paste}{ugTypesWritingOnePageFull1}{ugTypesWritingOnePageEmpty1}
\pastebutton{ugTypesWritingOnePageFull1}{\hidepaste}
\tab{5}\spadcommand{e : PrimeField(2 + 3)}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesWritingOnePageEmpty1}
\begin{paste}{ugTypesWritingOnePageEmpty1}{ugTypesWritingOnePagePatch1}
\pastebutton{ugTypesWritingOnePageEmpty1}{\showpaste}
\tab{5}\spadcommand{e : PrimeField(2 + 3)}
\end{paste}\end{patch}
```

```
\begin{patch}{ugTypesWritingOnePagePatch2}
\begin{paste}{ugTypesWritingOnePageFull2}{ugTypesWritingOnePageEmpty2}
\pastebutton{ugTypesWritingOnePageFull2}{\hidepaste}
\tab{5}\spadcommand{content(2)$Polynomial(Integer)}
\indentrel{3}\begin{verbatim}
   (2)  2
                                          Type: Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesWritingOnePageEmpty2}
\begin{paste}{ugTypesWritingOnePageEmpty2}{ugTypesWritingOnePagePatch2}
\pastebutton{ugTypesWritingOnePageEmpty2}{\showpaste}
\tab{5}\spadcommand{content(2)$Polynomial(Integer)}
\end{paste}\end{patch}

\begin{patch}{ugTypesWritingOnePagePatch3}
\begin{paste}{ugTypesWritingOnePageFull3}{ugTypesWritingOnePageEmpty3}
\pastebutton{ugTypesWritingOnePageFull3}{\hidepaste}
\tab{5}\spadcommand{content(2)$(Polynomial Complex Fraction Integer)}
\indentrel{3}\begin{verbatim}
   (3)  2
                         Type: Complex Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesWritingOnePageEmpty3}
\begin{paste}{ugTypesWritingOnePageEmpty3}{ugTypesWritingOnePagePatch3}
\pastebutton{ugTypesWritingOnePageEmpty3}{\showpaste}
\tab{5}\spadcommand{content(2)$(Polynomial Complex Fraction Integer)}
\end{paste}\end{patch}

\begin{patch}{ugTypesWritingOnePagePatch4}
\begin{paste}{ugTypesWritingOnePageFull4}{ugTypesWritingOnePageEmpty4}
\pastebutton{ugTypesWritingOnePageFull4}{\hidepaste}
\tab{5}\spadcommand{(2/3)@Fraction(Polynomial(Integer))}
\indentrel{3}\begin{verbatim}
        2
   (4)
        3
                      Type: Fraction Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesWritingOnePageEmpty4}
\begin{paste}{ugTypesWritingOnePageEmpty4}{ugTypesWritingOnePagePatch4}
\pastebutton{ugTypesWritingOnePageEmpty4}{\showpaste}
\tab{5}\spadcommand{(2/3)@Fraction(Polynomial(Integer))}
\end{paste}\end{patch}
```

```
\begin{patch}{ugTypesWritingOnePagePatch5}
\begin{paste}{ugTypesWritingOnePageFull5}{ugTypesWritingOnePageEmpty5}
\pastebutton{ugTypesWritingOnePageFull5}{\hidepaste}
\tab{5}\spadcommand{(2/3)@Fraction(Polynomial Integer)}
\indentrel{3}\begin{verbatim}
        2
   (5)
        3
                      Type: Fraction Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesWritingOnePageEmpty5}
\begin{paste}{ugTypesWritingOnePageEmpty5}{ugTypesWritingOnePagePatch5}
\pastebutton{ugTypesWritingOnePageEmpty5}{\showpaste}
\tab{5}\spadcommand{(2/3)@Fraction(Polynomial Integer)}
\end{paste}\end{patch}

\begin{patch}{ugTypesWritingOnePagePatch6}
\begin{paste}{ugTypesWritingOnePageFull6}{ugTypesWritingOnePageEmpty6}
\pastebutton{ugTypesWritingOnePageFull6}{\hidepaste}
\tab{5}\spadcommand{(d,f,g) : Complex Polynomial Integer}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesWritingOnePageEmpty6}
\begin{paste}{ugTypesWritingOnePageEmpty6}{ugTypesWritingOnePagePatch6}
\pastebutton{ugTypesWritingOnePageEmpty6}{\showpaste}
\tab{5}\spadcommand{(d,f,g) : Complex Polynomial Integer}
\end{paste}\end{patch}
```

———-

Types with More Than One Argument

— ug02.ht —

```
\begin{page}{ugTypesWritingMorePage}
{2.2.3. Types with More Than One Argument}
\beginscroll

If a constructor
has more than
one argument, you must use parentheses.
```

```
Some examples are
\centerline{{\axiomType{UnivariatePolynomial(x, Float)} }}
\centerline{{\axiomType{MultivariatePolynomial([z,w,r], Complex Float)} }}
\centerline{{\axiomType{SquareMatrix(3, Integer)} }}
\centerline{{\axiomType{FactoredFunctions2(Integer,Fraction Integer)}}}

\endscroll
\autobuttons
\end{page}
```

———-

Modes

⇒ "notitle" (ugTypesDeclarePage) 7 on page 1641
⇒ "notitle" (ugTypesConvertPage) 7 on page 1671

— **ug02.ht** —

```
\begin{page}{ugTypesWritingModesPage}{2.2.4. Modes}
\beginscroll

A \spadgloss{mode} is a type that possibly is a
question mark (\axiomSyntax{?}) or contains one in an argument
position.
For example, the following are all modes.
\texht{
\centerline{{\begin{tabular}{ccc}}}
\centerline{{\axiomType{?} & \quad &}}
\centerline{{\axiomType{Polynomial ?} }}
\centerline{{\axiomType{Matrix Polynomial ?} & \quad &}}
\centerline{{\axiomType{SquareMatrix(3,?)} }}
\centerline{{\axiomType{Integer} & \quad &}}
\centerline{{\axiomType{OneDimensionalArray(Float)}}}
\centerline{{\end{tabular}}}
}{
\centerline{{\axiomType{?} }}
\centerline{{\axiomType{Polynomial ?} }}
\centerline{{\axiomType{Matrix Polynomial ?} }}
\centerline{{\axiomType{SquareMatrix(3,?)} }}
\centerline{{\axiomType{Integer} }}
\centerline{{\axiomType{OneDimensionalArray(Float)}}}
}

As is evident from these examples, a mode is a type with a part that
is not specified (indicated by a question mark).  Only one
\axiomSyntax{?} is allowed per mode and it must appear in the most
```

```
deeply nested argument that is a type. Thus
\nonLibAxiomType{?(Integer)}, \nonLibAxiomType{Matrix(?
(Polynomial))}, \nonLibAxiomType{SquareMatrix(?, Integer)} and
\nonLibAxiomType{SquareMatrix(?, ?)} are all invalid.  The question
mark must take the place of a domain, not data (for example, the
integer that is the dimension of a square matrix).  This rules out,
for example, the two \axiomType{SquareMatrix} expressions.

Modes can be used for declarations
(\downlink{‘‘Declarations’’}{ugTypesDeclarePage}
in Section 2.3\ignore{ugTypesDeclare}) and conversions
(\downlink{‘‘Conversion’’}{ugTypesConvertPage}
in Section 2.7\ignore{ugTypesConvert}).
However, you cannot use a mode for package calling or giving target
type information.

\endscroll
\autobuttons
\end{page}
```

———-

Abbreviations

⇒ “notitle” (ugSysCmdwhatPage) 19 on page 2586

— ug02.ht —

```
\begin{page}{ugTypesWritingAbbrPage}{2.2.5. Abbreviations}
\beginscroll

Every constructor has an abbreviation that you can freely substitute
for the constructor name.  In some cases, the abbreviation is nothing
more than the capitalized version of the constructor name.

\beginImportant
Aside from allowing types to be written more concisely,
abbreviations are used by Axiom to name various system
files for constructors (such as library filenames, test input
files and example files).
Here are some common abbreviations.
\texht{}{\table{
{\axiomType{COMPLEX}    abbreviates \axiomType{Complex}              }
{\axiomType{DFLOAT}     abbreviates \axiomType{DoubleFloat}          }
{\axiomType{EXPR}       abbreviates \axiomType{Expression}           }
{\axiomType{FLOAT}      abbreviates \axiomType{Float}                }
{\axiomType{FRAC}       abbreviates \axiomType{Fraction}             }
{\axiomType{INT}        abbreviates \axiomType{Integer}              }
```

```
{\axiomType{MATRIX}     abbreviates \axiomType{Matrix}               }
{\axiomType{NNI}        abbreviates \axiomType{NonNegativeInteger}   }
{\axiomType{PI}         abbreviates \axiomType{PositiveInteger}      }
{\axiomType{POLY}       abbreviates \axiomType{Polynomial}           }
{\axiomType{STRING}     abbreviates \axiomType{String}               }
{\axiomType{UP}         abbreviates \axiomType{UnivariatePolynomial} }
}}
\endImportant

You can combine both full constructor names and abbreviations
in a type expression.
Here are some types using abbreviations.
\centerline{{\axiomType{POLY INT} is the same as
\axiomType{Polynomial(INT)} }}
\centerline{{\axiomType{POLY(Integer)} is the same
as \axiomType{Polynomial(Integer)} }}
\centerline{{\axiomType{POLY(Integer)} is the same
as \axiomType{Polynomial(INT)} }}
\centerline{{\axiomType{FRAC(COMPLEX(INT))} is the
same as \axiomType{Fraction Complex Integer} }}
\centerline{{\axiomType{FRAC(COMPLEX(INT))} is the
same as \axiomType{FRAC(Complex Integer)} }}

There are several ways of finding the names of constructors and their
abbreviations.  For a specific constructor, use \spadcmd{)abbreviation
query}.  You can also use the \spadcmd{)what} system command to see
the names and abbreviations of constructors.  For more information
about \spadcmd{)what}, see
\downlink{``)what''}{ugSysCmdwhatPage} in Section B.28
\ignore{ugSysCmdwhat}.
\xtc{
\spadcmd{)abbreviation query} can be
abbreviated (no pun intended) to \spadcmd{)abb q}.
}{
\spadpaste{)abb q Integer}
}
\xtc{
The \spadcmd{)abbreviation query} command lists
the constructor name if you give the abbreviation.
Issue \spadcmd{)abb q} if you want to see the names and abbreviations
of all Axiom constructors.
}{
\spadpaste{)abb q DMP}
}
\xtc{
Issue this to see all packages whose names contain the string ``ode''.
}{
\spadpaste{)what packages ode}
}
```

```
\endscroll
\autobuttons
\end{page}

\begin{patch}{ugTypesWritingAbbrPagePatch1}
\begin{paste}{ugTypesWritingAbbrPageFull1}{ugTypesWritingAbbrPageEmpty1}
\pastebutton{ugTypesWritingAbbrPageFull1}{\hidepaste}
\tab{5}\spadcommand{)abb q Integer}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesWritingAbbrPageEmpty1}
\begin{paste}{ugTypesWritingAbbrPageEmpty1}{ugTypesWritingAbbrPagePatch1}
\pastebutton{ugTypesWritingAbbrPageEmpty1}{\showpaste}
\tab{5}\spadcommand{)abb q Integer}
\end{paste}\end{patch}

\begin{patch}{ugTypesWritingAbbrPagePatch2}
\begin{paste}{ugTypesWritingAbbrPageFull2}{ugTypesWritingAbbrPageEmpty2}
\pastebutton{ugTypesWritingAbbrPageFull2}{\hidepaste}
\tab{5}\spadcommand{)abb q DMP}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesWritingAbbrPageEmpty2}
\begin{paste}{ugTypesWritingAbbrPageEmpty2}{ugTypesWritingAbbrPagePatch2}
\pastebutton{ugTypesWritingAbbrPageEmpty2}{\showpaste}
\tab{5}\spadcommand{)abb q DMP}
\end{paste}\end{patch}

\begin{patch}{ugTypesWritingAbbrPagePatch3}
\begin{paste}{ugTypesWritingAbbrPageFull3}{ugTypesWritingAbbrPageEmpty3}
\pastebutton{ugTypesWritingAbbrPageFull3}{\hidepaste}
\tab{5}\spadcommand{)what packages ode}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesWritingAbbrPageEmpty3}
\begin{paste}{ugTypesWritingAbbrPageEmpty3}{ugTypesWritingAbbrPagePatch3}
\pastebutton{ugTypesWritingAbbrPageEmpty3}{\showpaste}
\tab{5}\spadcommand{)what packages ode}
\end{paste}\end{patch}
```

Declarations

⇒ “notitle” (ugLangAssignPage) 9 on page 1745
⇒ “notitle” (ugUserDeclarePage) 10 on page 1838
⇒ “notitle” (ugTypesConvertPage) 7 on page 1671
⇒ “notitle” (ugIntroAssignPage) 6 on page 1497

— **ug02.ht** —

```
\begin{page}{ugTypesDeclarePage}{2.3. Declarations}
\beginscroll
%
A \spadgloss{declaration} is an expression used
to restrict the type of values that can be assigned to variables.
A colon (\axiomSyntax{:}) is always used after a variable or
list of variables to be declared.

\beginImportant
For a single variable, the syntax for declaration is \centerline{{{\it
variableName \axiom{:} typeOrMode}}} For multiple variables, the
syntax is \centerline{{{\tt (\subscriptIt{variableName}{1},
\subscriptIt{variableName}{2}, \ldots \subscriptIt{variableName}{N}):
{\it typeOrMode}}}}
\endImportant

You can always combine a declaration with an assignment.
When you do, it is equivalent to first giving a declaration statement,
then giving an assignment.
For more information on assignment, see
\downlink{``Symbols, Variables, Assignments, and Declarations''}
{ugIntroAssignPage} in Section 1.3.4\ignore{ugIntroAssign} and
\downlink{``Immediate and Delayed Assignments''}{ugLangAssignPage}
in Section 5.1\ignore{ugLangAssign}.
To see how to declare your own functions, see
\downlink{``Declaring the Type of Functions''}{ugUserDeclarePage}
in Section 6.4\ignore{ugUserDeclare}.

\xtc{
This declares one variable to have a type.
}{
\spadpaste{a : Integer \bound{a}}
}
\xtc{
This declares several variables to have a type.
}{
\spadpaste{(b,c) : Integer \bound{b c}}
}
\xtc{
\axiom{a, b} and \axiom{c} can only hold integer values.
```

```
}{
\spadpaste{a := 45 \free{a}}
}
\xtc{
If a value cannot be converted to a declared type,
an error message is displayed.
}{
\spadpaste{b := 4/5 \free{b}}
}
\xtc{
This declares a variable with a mode.
}{
\spadpaste{n : Complex ? \bound{n}}
}
\xtc{
This declares several variables with a mode.
}{
\spadpaste{(p,q,r) : Matrix Polynomial ? \bound{p q r}}
}
\xtc{
This complex object has integer real and imaginary parts.
}{
\spadpaste{n := -36 + 9 * \%i \free{n}}
}
\xtc{
This complex object has fractional symbolic real and imaginary parts.
}{
\spadpaste{n := complex(4/(x + y),y/x) \free{n}}
}
\xtc{
This matrix has entries that are polynomials with integer
coefficients.
}{
\spadpaste{p := [[1,2],[3,4],[5,6]] \free{p}}
}
\xtc{
This matrix has a single entry that is a polynomial with
rational number coefficients.
}{
\spadpaste{q := [[x - 2/3]] \free{q}}
}
\xtc{
This matrix has entries that are polynomials with complex integer
coefficients.
}{
\spadpaste{r := [[1-\%i*x,7*y+4*\%i]] \free{r}}
}
%
\xtc{
Note the difference between this and the next example.
```

```
This is a complex object with polynomial real and imaginary parts.
}{
\spadpaste{f : COMPLEX POLY ? := (x + y*\%i)**2}
}
\xtc{
This is a polynomial with complex integer coefficients.
The objects are convertible from one to the other.
See \downlink{``Conversion''}{ugTypesConvertPage}
in Section 2.7\ignore{ugTypesConvert} for more
information.
}{
\spadpaste{g : POLY COMPLEX ? := (x + y*\%i)**2}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugTypesDeclarePagePatch1}
\begin{paste}{ugTypesDeclarePageFull1}{ugTypesDeclarePageEmpty1}
\pastebutton{ugTypesDeclarePageFull1}{\hidepaste}
\tab{5}\spadcommand{a : Integer\bound{a }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesDeclarePageEmpty1}
\begin{paste}{ugTypesDeclarePageEmpty1}{ugTypesDeclarePagePatch1}
\pastebutton{ugTypesDeclarePageEmpty1}{\showpaste}
\tab{5}\spadcommand{a : Integer\bound{a }}
\end{paste}\end{patch}

\begin{patch}{ugTypesDeclarePagePatch2}
\begin{paste}{ugTypesDeclarePageFull2}{ugTypesDeclarePageEmpty2}
\pastebutton{ugTypesDeclarePageFull2}{\hidepaste}
\tab{5}\spadcommand{(b,c) : Integer\bound{b c }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesDeclarePageEmpty2}
\begin{paste}{ugTypesDeclarePageEmpty2}{ugTypesDeclarePagePatch2}
\pastebutton{ugTypesDeclarePageEmpty2}{\showpaste}
\tab{5}\spadcommand{(b,c) : Integer\bound{b c }}
\end{paste}\end{patch}

\begin{patch}{ugTypesDeclarePagePatch3}
\begin{paste}{ugTypesDeclarePageFull3}{ugTypesDeclarePageEmpty3}
```

```
\pastebutton{ugTypesDeclarePageFull3}{\hidepaste}
\tab{5}\spadcommand{a := 45\free{a }}
\indentrel{3}\begin{verbatim}
   (3)  45
                                          Type: Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesDeclarePageEmpty3}
\begin{paste}{ugTypesDeclarePageEmpty3}{ugTypesDeclarePagePatch3}
\pastebutton{ugTypesDeclarePageEmpty3}{\showpaste}
\tab{5}\spadcommand{a := 45\free{a }}
\end{paste}\end{patch}

\begin{patch}{ugTypesDeclarePagePatch4}
\begin{paste}{ugTypesDeclarePageFull4}{ugTypesDeclarePageEmpty4}
\pastebutton{ugTypesDeclarePageFull4}{\hidepaste}
\tab{5}\spadcommand{b := 4/5\free{b }}
\indentrel{3}\begin{verbatim}
   4

   5
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesDeclarePageEmpty4}
\begin{paste}{ugTypesDeclarePageEmpty4}{ugTypesDeclarePagePatch4}
\pastebutton{ugTypesDeclarePageEmpty4}{\showpaste}
\tab{5}\spadcommand{b := 4/5\free{b }}
\end{paste}\end{patch}

\begin{patch}{ugTypesDeclarePagePatch5}
\begin{paste}{ugTypesDeclarePageFull5}{ugTypesDeclarePageEmpty5}
\pastebutton{ugTypesDeclarePageFull5}{\hidepaste}
\tab{5}\spadcommand{n : Complex ?\bound{n }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesDeclarePageEmpty5}
\begin{paste}{ugTypesDeclarePageEmpty5}{ugTypesDeclarePagePatch5}
\pastebutton{ugTypesDeclarePageEmpty5}{\showpaste}
\tab{5}\spadcommand{n : Complex ?\bound{n }}
\end{paste}\end{patch}

\begin{patch}{ugTypesDeclarePagePatch6}
\begin{paste}{ugTypesDeclarePageFull6}{ugTypesDeclarePageEmpty6}
\pastebutton{ugTypesDeclarePageFull6}{\hidepaste}
\tab{5}\spadcommand{(p,q,r) : Matrix Polynomial ?\bound{p q r }}
```

```
\indentrel{3}\begin{verbatim}
                                                Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesDeclarePageEmpty6}
\begin{paste}{ugTypesDeclarePageEmpty6}{ugTypesDeclarePagePatch6}
\pastebutton{ugTypesDeclarePageEmpty6}{\showpaste}
\tab{5}\spadcommand{(p,q,r) : Matrix Polynomial ?\bound{p q r }}
\end{paste}\end{patch}

\begin{patch}{ugTypesDeclarePagePatch7}
\begin{paste}{ugTypesDeclarePageFull7}{ugTypesDeclarePageEmpty7}
\pastebutton{ugTypesDeclarePageFull7}{\hidepaste}
\tab{5}\spadcommand{n := -36 + 9 * \%i\free{n }}
\indentrel{3}\begin{verbatim}
   (6)  - 36 + 9%i
                                  Type: Complex Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesDeclarePageEmpty7}
\begin{paste}{ugTypesDeclarePageEmpty7}{ugTypesDeclarePagePatch7}
\pastebutton{ugTypesDeclarePageEmpty7}{\showpaste}
\tab{5}\spadcommand{n := -36 + 9 * \%i\free{n }}
\end{paste}\end{patch}

\begin{patch}{ugTypesDeclarePagePatch8}
\begin{paste}{ugTypesDeclarePageFull8}{ugTypesDeclarePageEmpty8}
\pastebutton{ugTypesDeclarePageFull8}{\hidepaste}
\tab{5}\spadcommand{n := complex(4/(x + y),y/x)\free{n }}
\indentrel{3}\begin{verbatim}
          4     y
   (7)   +  %i
        y + x   x
              Type: Complex Fraction Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesDeclarePageEmpty8}
\begin{paste}{ugTypesDeclarePageEmpty8}{ugTypesDeclarePagePatch8}
\pastebutton{ugTypesDeclarePageEmpty8}{\showpaste}
\tab{5}\spadcommand{n := complex(4/(x + y),y/x)\free{n }}
\end{paste}\end{patch}

\begin{patch}{ugTypesDeclarePagePatch9}
\begin{paste}{ugTypesDeclarePageFull9}{ugTypesDeclarePageEmpty9}
\pastebutton{ugTypesDeclarePageFull9}{\hidepaste}
\tab{5}\spadcommand{p := [[1,2],[3,4],[5,6]]\free{p }}
\indentrel{3}\begin{verbatim}
```

```
        1  2

   (8)  3  4

        5  6
                        Type: Matrix Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesDeclarePageEmpty9}
\begin{paste}{ugTypesDeclarePageEmpty9}{ugTypesDeclarePagePatch9}
\pastebutton{ugTypesDeclarePageEmpty9}{\showpaste}
\tab{5}\spadcommand{p := [[1,2],[3,4],[5,6]]\free{p }}
\end{paste}\end{patch}

\begin{patch}{ugTypesDeclarePagePatch10}
\begin{paste}{ugTypesDeclarePageFull10}{ugTypesDeclarePageEmpty10}
\pastebutton{ugTypesDeclarePageFull10}{\hidepaste}
\tab{5}\spadcommand{q := [[x - 2/3]]\free{q }}
\indentrel{3}\begin{verbatim}
            2
   (9)  x -
            3
               Type: Matrix Polynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesDeclarePageEmpty10}
\begin{paste}{ugTypesDeclarePageEmpty10}{ugTypesDeclarePagePatch10}
\pastebutton{ugTypesDeclarePageEmpty10}{\showpaste}
\tab{5}\spadcommand{q := [[x - 2/3]]\free{q }}
\end{paste}\end{patch}

\begin{patch}{ugTypesDeclarePagePatch11}
\begin{paste}{ugTypesDeclarePageFull11}{ugTypesDeclarePageEmpty11}
\pastebutton{ugTypesDeclarePageFull11}{\hidepaste}
\tab{5}\spadcommand{r := [[1-\%i*x,7*y+4*\%i]]\free{r }}
\indentrel{3}\begin{verbatim}
   (10)  [- %i x + 1  7y + 4%i]
                Type: Matrix Polynomial Complex Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesDeclarePageEmpty11}
\begin{paste}{ugTypesDeclarePageEmpty11}{ugTypesDeclarePagePatch11}
\pastebutton{ugTypesDeclarePageEmpty11}{\showpaste}
\tab{5}\spadcommand{r := [[1-\%i*x,7*y+4*\%i]]\free{r }}
\end{paste}\end{patch}

\begin{patch}{ugTypesDeclarePagePatch12}
```

```
\begin{paste}{ugTypesDeclarePageFull12}{ugTypesDeclarePageEmpty12}
\pastebutton{ugTypesDeclarePageFull12}{\hidepaste}
\tab{5}\spadcommand{f : COMPLEX POLY ? := (x + y*\%i)**2}
\indentrel{3}\begin{verbatim}
           2    2
   (11)  - y  + x  + 2x y %i
                       Type: Complex Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesDeclarePageEmpty12}
\begin{paste}{ugTypesDeclarePageEmpty12}{ugTypesDeclarePagePatch12}
\pastebutton{ugTypesDeclarePageEmpty12}{\showpaste}
\tab{5}\spadcommand{f : COMPLEX POLY ? := (x + y*\%i)**2}
\end{paste}\end{patch}

\begin{patch}{ugTypesDeclarePagePatch13}
\begin{paste}{ugTypesDeclarePageFull13}{ugTypesDeclarePageEmpty13}
\pastebutton{ugTypesDeclarePageFull13}{\hidepaste}
\tab{5}\spadcommand{g : POLY COMPLEX ? := (x + y*\%i)**2}
\indentrel{3}\begin{verbatim}
           2              2
   (12)  - y  + 2%i x y + x
                       Type: Polynomial Complex Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesDeclarePageEmpty13}
\begin{paste}{ugTypesDeclarePageEmpty13}{ugTypesDeclarePagePatch13}
\pastebutton{ugTypesDeclarePageEmpty13}{\showpaste}
\tab{5}\spadcommand{g : POLY COMPLEX ? := (x + y*\%i)**2}
\end{paste}\end{patch}
```

———-

Records

— ug02.ht —

```
\begin{page}{ugTypesRecordsPage}{2.4. Records}
\beginscroll
%
A \pspadtype{Record} is an object composed of one or more other objects,
each of which is referenced
with
a \spadgloss{selector}.
Components can all belong to the same type or each can have a different
```

```
type.

\beginImportant
The syntax for writing a \pspadtype{Record} type is \centerline{{{\tt
Record(\subscriptIt{selector}{1}:\subscriptIt{type}{1},
\subscriptIt{selector}{2}:\subscriptIt{type}{2}, \ldots,
\subscriptIt{selector}{N}:\subscriptIt{type}{N})}}} You must be
careful if a selector has the same name as a variable in the
workspace.  If this occurs, precede the selector name by a single
quote.
\endImportant

Record components are implicitly ordered.  All the components of a
record can be set at once by assigning the record a bracketed
\spadgloss{tuple} of values of the proper length (for example,
\axiom{r : Record(a: Integer, b: String) := [1, "two"]}).  To access a
component of a record \axiom{r}, write the name \axiom{r}, followed by
a period, followed by a selector.

%
\xtc{
The object returned by this computation is a record with two components: a
\axiom{quotient} part and a \axiom{remainder} part.
}{
\spadpaste{u := divide(5,2) \bound{u}}
}
%
\xtc{
This is the quotient part.
}{
\spadpaste{u.quotient \free{u}}
}
\xtc{
This is the remainder part.
}{
\spadpaste{u.remainder \free{u}}
}
%
\xtc{
You can use selector expressions on the left-hand side of an assignment
to change destructively the components of a record.
}{
\spadpaste{u.quotient := 8978 \free{u}\bound{u1}}
}
\xtc{
The selected component \axiom{quotient} has the value \axiom{8978},
which is what is returned by the assignment.
Check that the value of \axiom{u} was modified.
}{
\spadpaste{u \free{u}\free{u1}}
```

```
}
\xtc{
Selectors are evaluated.
Thus you can use variables that evaluate to selectors instead of the
selectors themselves.
}{
\spadpaste{s := 'quotient \bound{s}}
}
\xtc{
Be careful!
A selector could have the same name as a variable in the workspace.
If this occurs, precede the selector name by a single quote, as in
\axiom{u.'quotient}.
}{
\spadpaste{divide(5,2).s \free{s}}
}
\xtc{
Here we declare that the value of \axiom{bd}
has two components: a string,
to be accessed via \axiom{name}, and an integer,
to be accessed via \axiom{birthdayMonth}.
}{
\spadpaste{bd : Record(name : String, birthdayMonth : Integer)
\bound{bddec}}
}
\xtc{
You must initially set the value of the entire \pspadtype{Record}
at once.
}{
\spadpaste{bd := ["Judith", 3] \free{bddec}\bound{bd}}
}
\xtc{
Once set, you can change any of the individual components.
}{
\spadpaste{bd.name := "Katie" \free{bd}}
}
\xtc{
Records may be nested and the selector names can be shared at
different levels.
}{
\spadpaste{r : Record(a : Record(b: Integer, c: Integer), b: Integer)
\bound{rdec}}
}
\xtc{
The record \axiom{r} has a \axiom{b} selector at two different levels.
Here is an initial value for \axiom{r}.
}{
\spadpaste{r := [[1,2],3] \bound{r}\free{rdec}}
}
\xtc{
```

```
This extracts the \axiom{b} component from the \axiom{a} component of
\axiom{r}.
}{
\spadpaste{r.a.b \free{r}}
}
\xtc{
This extracts the \axiom{b} component from \axiom{r}.
}{
\spadpaste{r.b \free{r}}
}
%
\xtc{
You can also use spaces or parentheses to refer to \pspadtype{Record}
components.
This is the same as \axiom{r.a}.
}{
\spadpaste{r(a) \free{r}}
}
\xtc{
This is the same as \axiom{r.b}.
}{
\spadpaste{r b \free{r}}
}
\xtc{
This is the same as \axiom{r.b := 10}.
}{
\spadpaste{r(b) := 10 \free{r}\bound{r1}}
}
\xtc{
Look at \axiom{r} to make sure it was modified.
}{
\spadpaste{r \free{r1}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugTypesRecordsPagePatch1}
\begin{paste}{ugTypesRecordsPageFull1}{ugTypesRecordsPageEmpty1}
\pastebutton{ugTypesRecordsPageFull1}{\hidepaste}
\tab{5}\spadcommand{u := divide(5,2)\bound{u }}
\indentrel{3}\begin{verbatim}
   (1)  [quotient= 2,remainder= 1]
     Type: Record(quotient: Integer,remainder: Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesRecordsPageEmpty1}
\begin{paste}{ugTypesRecordsPageEmpty1}{ugTypesRecordsPagePatch1}
```

```
\pastebutton{ugTypesRecordsPageEmpty1}{\showpaste}
\tab{5}\spadcommand{u := divide(5,2)\bound{u }}
\end{paste}\end{patch}

\begin{patch}{ugTypesRecordsPagePatch2}
\begin{paste}{ugTypesRecordsPageFull2}{ugTypesRecordsPageEmpty2}
\pastebutton{ugTypesRecordsPageFull2}{\hidepaste}
\tab{5}\spadcommand{u.quotient\free{u }}
\indentrel{3}\begin{verbatim}
   (2)  2
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesRecordsPageEmpty2}
\begin{paste}{ugTypesRecordsPageEmpty2}{ugTypesRecordsPagePatch2}
\pastebutton{ugTypesRecordsPageEmpty2}{\showpaste}
\tab{5}\spadcommand{u.quotient\free{u }}
\end{paste}\end{patch}

\begin{patch}{ugTypesRecordsPagePatch3}
\begin{paste}{ugTypesRecordsPageFull3}{ugTypesRecordsPageEmpty3}
\pastebutton{ugTypesRecordsPageFull3}{\hidepaste}
\tab{5}\spadcommand{u.remainder\free{u }}
\indentrel{3}\begin{verbatim}
   (3)  1
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesRecordsPageEmpty3}
\begin{paste}{ugTypesRecordsPageEmpty3}{ugTypesRecordsPagePatch3}
\pastebutton{ugTypesRecordsPageEmpty3}{\showpaste}
\tab{5}\spadcommand{u.remainder\free{u }}
\end{paste}\end{patch}

\begin{patch}{ugTypesRecordsPagePatch4}
\begin{paste}{ugTypesRecordsPageFull4}{ugTypesRecordsPageEmpty4}
\pastebutton{ugTypesRecordsPageFull4}{\hidepaste}
\tab{5}\spadcommand{u.quotient := 8978\free{u }\bound{u1 }}
\indentrel{3}\begin{verbatim}
   (4)  8978
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesRecordsPageEmpty4}
\begin{paste}{ugTypesRecordsPageEmpty4}{ugTypesRecordsPagePatch4}
\pastebutton{ugTypesRecordsPageEmpty4}{\showpaste}
\tab{5}\spadcommand{u.quotient := 8978\free{u }\bound{u1 }}
```

```
\end{paste}\end{patch}

\begin{patch}{ugTypesRecordsPagePatch5}
\begin{paste}{ugTypesRecordsPageFull5}{ugTypesRecordsPageEmpty5}
\pastebutton{ugTypesRecordsPageFull5}{\hidepaste}
\tab{5}\spadcommand{u\free{u }\free{u1 }}
\indentrel{3}\begin{verbatim}
   (5)  [quotient= 8978,remainder= 1]
     Type: Record(quotient: Integer,remainder: Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesRecordsPageEmpty5}
\begin{paste}{ugTypesRecordsPageEmpty5}{ugTypesRecordsPagePatch5}
\pastebutton{ugTypesRecordsPageEmpty5}{\showpaste}
\tab{5}\spadcommand{u\free{u }\free{u1 }}
\end{paste}\end{patch}

\begin{patch}{ugTypesRecordsPagePatch6}
\begin{paste}{ugTypesRecordsPageFull6}{ugTypesRecordsPageEmpty6}
\pastebutton{ugTypesRecordsPageFull6}{\hidepaste}
\tab{5}\spadcommand{s := 'quotient\bound{s }}
\indentrel{3}\begin{verbatim}
   (6)  quotient
                                Type: Variable quotient
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesRecordsPageEmpty6}
\begin{paste}{ugTypesRecordsPageEmpty6}{ugTypesRecordsPagePatch6}
\pastebutton{ugTypesRecordsPageEmpty6}{\showpaste}
\tab{5}\spadcommand{s := 'quotient\bound{s }}
\end{paste}\end{patch}

\begin{patch}{ugTypesRecordsPagePatch7}
\begin{paste}{ugTypesRecordsPageFull7}{ugTypesRecordsPageEmpty7}
\pastebutton{ugTypesRecordsPageFull7}{\hidepaste}
\tab{5}\spadcommand{divide(5,2).s\free{s }}
\indentrel{3}\begin{verbatim}
   (7)  2
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesRecordsPageEmpty7}
\begin{paste}{ugTypesRecordsPageEmpty7}{ugTypesRecordsPagePatch7}
\pastebutton{ugTypesRecordsPageEmpty7}{\showpaste}
\tab{5}\spadcommand{divide(5,2).s\free{s }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugTypesRecordsPagePatch8}
\begin{paste}{ugTypesRecordsPageFull8}{ugTypesRecordsPageEmpty8}
\pastebutton{ugTypesRecordsPageFull8}{\hidepaste}
\tab{5}\spadcommand{bd : Record(name : String, birthdayMonth : Integer)\bound{bddec }}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesRecordsPageEmpty8}
\begin{paste}{ugTypesRecordsPageEmpty8}{ugTypesRecordsPagePatch8}
\pastebutton{ugTypesRecordsPageEmpty8}{\showpaste}
\tab{5}\spadcommand{bd : Record(name : String, birthdayMonth : Integer)\bound{bddec }}
\end{paste}\end{patch}

\begin{patch}{ugTypesRecordsPagePatch9}
\begin{paste}{ugTypesRecordsPageFull9}{ugTypesRecordsPageEmpty9}
\pastebutton{ugTypesRecordsPageFull9}{\hidepaste}
\tab{5}\spadcommand{bd := ["Judith", 3]\free{bddec }\bound{bd }}
\indentrel{3}\begin{verbatim}
   (9)  [name= "Judith",birthdayMonth= 3]
      Type: Record(name: String,birthdayMonth: Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesRecordsPageEmpty9}
\begin{paste}{ugTypesRecordsPageEmpty9}{ugTypesRecordsPagePatch9}
\pastebutton{ugTypesRecordsPageEmpty9}{\showpaste}
\tab{5}\spadcommand{bd := ["Judith", 3]\free{bddec }\bound{bd }}
\end{paste}\end{patch}

\begin{patch}{ugTypesRecordsPagePatch10}
\begin{paste}{ugTypesRecordsPageFull10}{ugTypesRecordsPageEmpty10}
\pastebutton{ugTypesRecordsPageFull10}{\hidepaste}
\tab{5}\spadcommand{bd.name := "Katie"\free{bd }}
\indentrel{3}\begin{verbatim}
   (10)  "Katie"
                                          Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesRecordsPageEmpty10}
\begin{paste}{ugTypesRecordsPageEmpty10}{ugTypesRecordsPagePatch10}
\pastebutton{ugTypesRecordsPageEmpty10}{\showpaste}
\tab{5}\spadcommand{bd.name := "Katie"\free{bd }}
\end{paste}\end{patch}

\begin{patch}{ugTypesRecordsPagePatch11}
\begin{paste}{ugTypesRecordsPageFull11}{ugTypesRecordsPageEmpty11}
\pastebutton{ugTypesRecordsPageFull11}{\hidepaste}
```

```
\tab{5}\spadcommand{r : Record(a : Record(b: Integer, c: Integer), b: Integer)\bound{rdec }}
\indentrel{3}\begin{verbatim}
                                              Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesRecordsPageEmpty11}
\begin{paste}{ugTypesRecordsPageEmpty11}{ugTypesRecordsPagePatch11}
\pastebutton{ugTypesRecordsPageEmpty11}{\showpaste}
\tab{5}\spadcommand{r : Record(a : Record(b: Integer, c: Integer), b: Integer)\bound{rdec }}
\end{paste}\end{patch}

\begin{patch}{ugTypesRecordsPagePatch12}
\begin{paste}{ugTypesRecordsPageFull12}{ugTypesRecordsPageEmpty12}
\pastebutton{ugTypesRecordsPageFull12}{\hidepaste}
\tab{5}\spadcommand{r := [[1,2],3]\bound{r }\free{rdec }}
\indentrel{3}\begin{verbatim}
   (12)  [a= [b= 1,c= 2],b= 3]
Type: Record(a: Record(b: Integer,c: Integer),b: Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesRecordsPageEmpty12}
\begin{paste}{ugTypesRecordsPageEmpty12}{ugTypesRecordsPagePatch12}
\pastebutton{ugTypesRecordsPageEmpty12}{\showpaste}
\tab{5}\spadcommand{r := [[1,2],3]\bound{r }\free{rdec }}
\end{paste}\end{patch}

\begin{patch}{ugTypesRecordsPagePatch13}
\begin{paste}{ugTypesRecordsPageFull13}{ugTypesRecordsPageEmpty13}
\pastebutton{ugTypesRecordsPageFull13}{\hidepaste}
\tab{5}\spadcommand{r.a.b\free{r }}
\indentrel{3}\begin{verbatim}
   (13)  1
                                   Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesRecordsPageEmpty13}
\begin{paste}{ugTypesRecordsPageEmpty13}{ugTypesRecordsPagePatch13}
\pastebutton{ugTypesRecordsPageEmpty13}{\showpaste}
\tab{5}\spadcommand{r.a.b\free{r }}
\end{paste}\end{patch}

\begin{patch}{ugTypesRecordsPagePatch14}
\begin{paste}{ugTypesRecordsPageFull14}{ugTypesRecordsPageEmpty14}
\pastebutton{ugTypesRecordsPageFull14}{\hidepaste}
\tab{5}\spadcommand{r.b\free{r }}
\indentrel{3}\begin{verbatim}
   (14)  3
```

```
                                 Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesRecordsPageEmpty14}
\begin{paste}{ugTypesRecordsPageEmpty14}{ugTypesRecordsPagePatch14}
\pastebutton{ugTypesRecordsPageEmpty14}{\showpaste}
\tab{5}\spadcommand{r.b\free{r }}
\end{paste}\end{patch}

\begin{patch}{ugTypesRecordsPagePatch15}
\begin{paste}{ugTypesRecordsPageFull15}{ugTypesRecordsPageEmpty15}
\pastebutton{ugTypesRecordsPageFull15}{\hidepaste}
\tab{5}\spadcommand{r(a)\free{r }}
\indentrel{3}\begin{verbatim}
   (15)  [b= 1,c= 2]
                    Type: Record(b: Integer,c: Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesRecordsPageEmpty15}
\begin{paste}{ugTypesRecordsPageEmpty15}{ugTypesRecordsPagePatch15}
\pastebutton{ugTypesRecordsPageEmpty15}{\showpaste}
\tab{5}\spadcommand{r(a)\free{r }}
\end{paste}\end{patch}

\begin{patch}{ugTypesRecordsPagePatch16}
\begin{paste}{ugTypesRecordsPageFull16}{ugTypesRecordsPageEmpty16}
\pastebutton{ugTypesRecordsPageFull16}{\hidepaste}
\tab{5}\spadcommand{r b\free{r }}
\indentrel{3}\begin{verbatim}
   (16)  3
                                 Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesRecordsPageEmpty16}
\begin{paste}{ugTypesRecordsPageEmpty16}{ugTypesRecordsPagePatch16}
\pastebutton{ugTypesRecordsPageEmpty16}{\showpaste}
\tab{5}\spadcommand{r b\free{r }}
\end{paste}\end{patch}

\begin{patch}{ugTypesRecordsPagePatch17}
\begin{paste}{ugTypesRecordsPageFull17}{ugTypesRecordsPageEmpty17}
\pastebutton{ugTypesRecordsPageFull17}{\hidepaste}
\tab{5}\spadcommand{r(b) := 10\free{r }\bound{r1 }}
\indentrel{3}\begin{verbatim}
   (17)  10
                                 Type: PositiveInteger
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesRecordsPageEmpty17}
\begin{paste}{ugTypesRecordsPageEmpty17}{ugTypesRecordsPagePatch17}
\pastebutton{ugTypesRecordsPageEmpty17}{\showpaste}
\tab{5}\spadcommand{r(b) := 10\free{r }\bound{r1 }}
\end{paste}\end{patch}

\begin{patch}{ugTypesRecordsPagePatch18}
\begin{paste}{ugTypesRecordsPageFull18}{ugTypesRecordsPageEmpty18}
\pastebutton{ugTypesRecordsPageFull18}{\hidepaste}
\tab{5}\spadcommand{r\free{r1 }}
\indentrel{3}\begin{verbatim}
   (18)  [a= [b= 1,c= 2],b= 10]
Type: Record(a: Record(b: Integer,c: Integer),b: Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesRecordsPageEmpty18}
\begin{paste}{ugTypesRecordsPageEmpty18}{ugTypesRecordsPagePatch18}
\pastebutton{ugTypesRecordsPageEmpty18}{\showpaste}
\tab{5}\spadcommand{r\free{r1 }}
\end{paste}\end{patch}
```

———-

Unions

⇒ "notitle" (ugTypesUnionsWOSelPage) 7 on page 1657
⇒ "notitle" (ugTypesUnionsWSelPage) 7 on page 1664

— **ug02.ht** —

```
\begin{page}{ugTypesUnionsPage}{2.5. Unions}
\beginscroll
%
Type \pspadtype{Union} is used for objects that
can be of any of a specific finite set of types.
Two versions of unions are available,
one with selectors (like records) and one without.

\beginmenu
\menudownlink{{2.5.1. Unions Without Selectors}}{ugTypesUnionsWOSelPage}
\menudownlink{{2.5.2. Unions With Selectors}}{ugTypesUnionsWSelPage}
\endmenu
\endscroll
\autobuttons
```

```
\end{page}
```

———-

Unions Without Selectors

— ug02.ht —

```
\begin{page}{ugTypesUnionsWOSelPage}{2.5.1. Unions Without Selectors}
\beginscroll

The declaration \axiom{x : Union(Integer, String, Float)}
states that \axiom{x} can have values that are integers,
strings or ‘‘big’’ floats.
If, for example, the \pspadtype{Union} object is an integer, the object is
said to belong to the \axiomType{Integer} {\it branch}
of the \pspadtype{Union}.\footnote{
Note that we are being a bit careless with the language here.
Technically, the type of \axiom{x} is always
\pspadtype{Union(Integer, String, Float)}.
If it belongs to the \axiomType{Integer} branch, \axiom{x}
may be converted to an object of type \axiomType{Integer}.}

\beginImportant
The syntax for writing a \pspadtype{Union} type without selectors is
\centerline{{{\tt Union(\subscriptIt{type}{1}, \subscriptIt{type}{2},
\ldots, \subscriptIt{type}{N})}}} The types in a union without
selectors must be distinct.
\endImportant

It is possible to create unions like
\pspadtype{Union(Integer, PositiveInteger)} but they are
difficult to work with because of the overlap in the branch
types.
See below for the rules Axiom uses for converting something
into a union object.

The \axiom{case} infix
\spadkey{case}
operator returns a \axiomType{Boolean}
and can be used to determine the branch in which an object lies.

\xtc{
This function displays a message stating in which
branch of the \pspadtype{Union} the object (defined as \axiom{x}
above) lies.
}{
```

```
\begin{spadsrc}[\bound{sayBranch}]
sayBranch(x : Union(Integer,String,Float)) : Void  ==
  output
    x case Integer => "Integer branch"
    x case String  => "String branch"
    "Float branch"
\end{spadsrc}
}
%
\xtc{
This tries \userfun{sayBranch} with an integer.
}{
\spadpaste{sayBranch 1 \free{sayBranch}}
}
\xtc{
This tries \userfun{sayBranch} with a string.
}{
\spadpaste{sayBranch "hello" \free{sayBranch}}
}
\xtc{
This tries \userfun{sayBranch} with a floating-point number.
}{
\spadpaste{sayBranch 2.718281828 \free{sayBranch}}
}
%

There are two things of interest about this particular
example to which we would like to draw your attention.
\indent{4}
\beginitems

\item[1. ] Axiom normally converts a result to the target value before
passing it to the function.  If we left the declaration information
out of this function definition then the \axiom{sayBranch} call would
have been attempted with an \axiomType{Integer} rather than a
\pspadtype{Union}, and an error would have resulted.

\item[2. ] The types in a \pspadtype{Union} are searched in the order given.
So if the type were given as

\noindent
{\small\axiom{sayBranch(x: Union(String,Integer,Float,Any)): Void}}

\noindent
then the result would have been ``String branch'' because there
is a conversion from \axiomType{Integer} to \axiomType{String}.
\enditems
\indent{0}

Sometimes \pspadtype{Union} types can have extremely
```

```
long names.
Axiom therefore abbreviates the names of unions by printing
the type of the branch first within the \pspadtype{Union} and then
eliding the remaining types with an ellipsis (\axiomSyntax{...}).

\xtc{
Here the \axiomType{Integer} branch is displayed first.
Use \axiomSyntax{::} to create a \pspadtype{Union} object from an object.
}{
\spadpaste{78 :: Union(Integer,String)}
}
\xtc{
Here the \axiomType{String} branch is displayed first.
}{
\spadpaste{s := "string" :: Union(Integer,String) \bound{s}}
}
\xtc{
Use \axiom{typeOf} to see the full and actual \pspadtype{Union} type.
\spadkey{typeOf}
}{
\spadpaste{typeOf s}
}
\xtc{
A common operation that returns a union is \axiomFunFrom{exquo}{Integer}
which returns the ``exact quotient'' if the quotient is exact,...
}{
\spadpaste{three := exquo(6,2) \bound{three}}
}
\xtc{
and \axiom{"failed"} if the quotient is not exact.
}{
\spadpaste{exquo(5,2)}
}
\xtc{
A union with a \axiom{"failed"} is frequently used to indicate the failure
or lack of applicability of an object.
As another example, assign an integer a variable \axiom{r} declared to be a
rational number.
}{
\spadpaste{r: FRAC INT := 3 \bound{r}\bound{rdec}}
}
\xtc{
The operation \axiomFunFrom{retractIfCan}{Fraction} tries to retract the
fraction to the underlying domain \axiomType{Integer}.
It produces a union object.
Here it succeeds.
}{
\spadpaste{retractIfCan(r) \free{r}}
}
\xtc{
```

```
Assign it a rational number.
}{
\spadpaste{r := 3/2 \bound{r1}\free{rdec}}
}
\xtc{
Here the retraction fails.
}{
\spadpaste{retractIfCan(r) \free{r1}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugTypesUnionsWOSelPagePatch1}
\begin{paste}{ugTypesUnionsWOSelPageFull1}{ugTypesUnionsWOSelPageEmpty1}
\pastebutton{ugTypesUnionsWOSelPageFull1}{\hidepaste}
\tab{5}\spadcommand{sayBranch(x : Union(Integer,String,Float)) : Void  ==
  output
    x case Integer => "Integer branch"
    x case String  => "String branch"
    "Float branch"
\bound{sayBranch }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesUnionsWOSelPageEmpty1}
\begin{paste}{ugTypesUnionsWOSelPageEmpty1}{ugTypesUnionsWOSelPagePatch1}
\pastebutton{ugTypesUnionsWOSelPageEmpty1}{\showpaste}
\tab{5}\spadcommand{sayBranch(x : Union(Integer,String,Float)) : Void  ==
  output
    x case Integer => "Integer branch"
    x case String  => "String branch"
    "Float branch"
\bound{sayBranch }}
\end{paste}\end{patch}

\begin{patch}{ugTypesUnionsWOSelPagePatch2}
\begin{paste}{ugTypesUnionsWOSelPageFull2}{ugTypesUnionsWOSelPageEmpty2}
\pastebutton{ugTypesUnionsWOSelPageFull2}{\hidepaste}
\tab{5}\spadcommand{sayBranch 1\free{sayBranch }}
\indentrel{3}\begin{verbatim}
   Integer branch
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesUnionsWOSelPageEmpty2}
```

```
\begin{paste}{ugTypesUnionsWOSelPageEmpty2}{ugTypesUnionsWOSelPagePatch2}
\pastebutton{ugTypesUnionsWOSelPageEmpty2}{\showpaste}
\tab{5}\spadcommand{sayBranch 1\free{sayBranch }}
\end{paste}\end{patch}

\begin{patch}{ugTypesUnionsWOSelPagePatch3}
\begin{paste}{ugTypesUnionsWOSelPageFull3}{ugTypesUnionsWOSelPageEmpty3}
\pastebutton{ugTypesUnionsWOSelPageFull3}{\hidepaste}
\tab{5}\spadcommand{sayBranch "hello"\free{sayBranch }}
\indentrel{3}\begin{verbatim}
   String branch
                                          Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesUnionsWOSelPageEmpty3}
\begin{paste}{ugTypesUnionsWOSelPageEmpty3}{ugTypesUnionsWOSelPagePatch3}
\pastebutton{ugTypesUnionsWOSelPageEmpty3}{\showpaste}
\tab{5}\spadcommand{sayBranch "hello"\free{sayBranch }}
\end{paste}\end{patch}

\begin{patch}{ugTypesUnionsWOSelPagePatch4}
\begin{paste}{ugTypesUnionsWOSelPageFull4}{ugTypesUnionsWOSelPageEmpty4}
\pastebutton{ugTypesUnionsWOSelPageFull4}{\hidepaste}
\tab{5}\spadcommand{sayBranch 2.718281828\free{sayBranch }}
\indentrel{3}\begin{verbatim}
   Float branch
                                          Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesUnionsWOSelPageEmpty4}
\begin{paste}{ugTypesUnionsWOSelPageEmpty4}{ugTypesUnionsWOSelPagePatch4}
\pastebutton{ugTypesUnionsWOSelPageEmpty4}{\showpaste}
\tab{5}\spadcommand{sayBranch 2.718281828\free{sayBranch }}
\end{paste}\end{patch}

\begin{patch}{ugTypesUnionsWOSelPagePatch5}
\begin{paste}{ugTypesUnionsWOSelPageFull5}{ugTypesUnionsWOSelPageEmpty5}
\pastebutton{ugTypesUnionsWOSelPageFull5}{\hidepaste}
\tab{5}\spadcommand{78 :: Union(Integer,String)}
\indentrel{3}\begin{verbatim}
   (5)  78
                              Type: Union(Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesUnionsWOSelPageEmpty5}
\begin{paste}{ugTypesUnionsWOSelPageEmpty5}{ugTypesUnionsWOSelPagePatch5}
\pastebutton{ugTypesUnionsWOSelPageEmpty5}{\showpaste}
```

```
\tab{5}\spadcommand{78 :: Union(Integer,String)}
\end{paste}\end{patch}

\begin{patch}{ugTypesUnionsWOSelPagePatch6}
\begin{paste}{ugTypesUnionsWOSelPageFull6}{ugTypesUnionsWOSelPageEmpty6}
\pastebutton{ugTypesUnionsWOSelPageFull6}{\hidepaste}
\tab{5}\spadcommand{s := "string" :: Union(Integer,String)\bound{s }}
\indentrel{3}\begin{verbatim}
   (6)  "string"
                                Type: Union(String,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesUnionsWOSelPageEmpty6}
\begin{paste}{ugTypesUnionsWOSelPageEmpty6}{ugTypesUnionsWOSelPagePatch6}
\pastebutton{ugTypesUnionsWOSelPageEmpty6}{\showpaste}
\tab{5}\spadcommand{s := "string" :: Union(Integer,String)\bound{s }}
\end{paste}\end{patch}

\begin{patch}{ugTypesUnionsWOSelPagePatch7}
\begin{paste}{ugTypesUnionsWOSelPageFull7}{ugTypesUnionsWOSelPageEmpty7}
\pastebutton{ugTypesUnionsWOSelPageFull7}{\hidepaste}
\tab{5}\spadcommand{typeOf s}
\indentrel{3}\begin{verbatim}
   (7)  Union(Integer,String)
                                           Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesUnionsWOSelPageEmpty7}
\begin{paste}{ugTypesUnionsWOSelPageEmpty7}{ugTypesUnionsWOSelPagePatch7}
\pastebutton{ugTypesUnionsWOSelPageEmpty7}{\showpaste}
\tab{5}\spadcommand{typeOf s}
\end{paste}\end{patch}

\begin{patch}{ugTypesUnionsWOSelPagePatch8}
\begin{paste}{ugTypesUnionsWOSelPageFull8}{ugTypesUnionsWOSelPageEmpty8}
\pastebutton{ugTypesUnionsWOSelPageFull8}{\hidepaste}
\tab{5}\spadcommand{three := exquo(6,2)\bound{three }}
\indentrel{3}\begin{verbatim}
   (8)  3
                               Type: Union(Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesUnionsWOSelPageEmpty8}
\begin{paste}{ugTypesUnionsWOSelPageEmpty8}{ugTypesUnionsWOSelPagePatch8}
\pastebutton{ugTypesUnionsWOSelPageEmpty8}{\showpaste}
\tab{5}\spadcommand{three := exquo(6,2)\bound{three }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugTypesUnionsWOSelPagePatch9}
\begin{paste}{ugTypesUnionsWOSelPageFull9}{ugTypesUnionsWOSelPageEmpty9}
\pastebutton{ugTypesUnionsWOSelPageFull9}{\hidepaste}
\tab{5}\spadcommand{exquo(5,2)}
\indentrel{3}\begin{verbatim}
   (9)  "failed"
                              Type: Union("failed",...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesUnionsWOSelPageEmpty9}
\begin{paste}{ugTypesUnionsWOSelPageEmpty9}{ugTypesUnionsWOSelPagePatch9}
\pastebutton{ugTypesUnionsWOSelPageEmpty9}{\showpaste}
\tab{5}\spadcommand{exquo(5,2)}
\end{paste}\end{patch}

\begin{patch}{ugTypesUnionsWOSelPagePatch10}
\begin{paste}{ugTypesUnionsWOSelPageFull10}{ugTypesUnionsWOSelPageEmpty10}
\pastebutton{ugTypesUnionsWOSelPageFull10}{\hidepaste}
\tab{5}\spadcommand{r: FRAC INT := 3\bound{r }\bound{rdec }}
\indentrel{3}\begin{verbatim}
   (10)  3
                                 Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesUnionsWOSelPageEmpty10}
\begin{paste}{ugTypesUnionsWOSelPageEmpty10}{ugTypesUnionsWOSelPagePatch10}
\pastebutton{ugTypesUnionsWOSelPageEmpty10}{\showpaste}
\tab{5}\spadcommand{r: FRAC INT := 3\bound{r }\bound{rdec }}
\end{paste}\end{patch}

\begin{patch}{ugTypesUnionsWOSelPagePatch11}
\begin{paste}{ugTypesUnionsWOSelPageFull11}{ugTypesUnionsWOSelPageEmpty11}
\pastebutton{ugTypesUnionsWOSelPageFull11}{\hidepaste}
\tab{5}\spadcommand{retractIfCan(r)\free{r }}
\indentrel{3}\begin{verbatim}
   (11)  3
                               Type: Union(Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesUnionsWOSelPageEmpty11}
\begin{paste}{ugTypesUnionsWOSelPageEmpty11}{ugTypesUnionsWOSelPagePatch11}
\pastebutton{ugTypesUnionsWOSelPageEmpty11}{\showpaste}
\tab{5}\spadcommand{retractIfCan(r)\free{r }}
\end{paste}\end{patch}

\begin{patch}{ugTypesUnionsWOSelPagePatch12}
```

```
\begin{paste}{ugTypesUnionsWOSelPageFull12}{ugTypesUnionsWOSelPageEmpty12}
\pastebutton{ugTypesUnionsWOSelPageFull12}{\hidepaste}
\tab{5}\spadcommand{r := 3/2\bound{r1 }\free{rdec }}
\indentrel{3}\begin{verbatim}
         3
   (12)
         2
                                  Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesUnionsWOSelPageEmpty12}
\begin{paste}{ugTypesUnionsWOSelPageEmpty12}{ugTypesUnionsWOSelPagePatch12}
\pastebutton{ugTypesUnionsWOSelPageEmpty12}{\showpaste}
\tab{5}\spadcommand{r := 3/2\bound{r1 }\free{rdec }}
\end{paste}\end{patch}

\begin{patch}{ugTypesUnionsWOSelPagePatch13}
\begin{paste}{ugTypesUnionsWOSelPageFull13}{ugTypesUnionsWOSelPageEmpty13}
\pastebutton{ugTypesUnionsWOSelPageFull13}{\hidepaste}
\tab{5}\spadcommand{retractIfCan(r)\free{r1 }}
\indentrel{3}\begin{verbatim}
   (13)  "failed"
                               Type: Union("failed",...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesUnionsWOSelPageEmpty13}
\begin{paste}{ugTypesUnionsWOSelPageEmpty13}{ugTypesUnionsWOSelPagePatch13}
\pastebutton{ugTypesUnionsWOSelPageEmpty13}{\showpaste}
\tab{5}\spadcommand{retractIfCan(r)\free{r1 }}
\end{paste}\end{patch}
```

————-

Unions With Selectors

⇒ "notitle" (ugTypesRecordsPage) 7 on page 1647
⇒ "notitle" (ugTypesUnionsWOSelPage) 7 on page 1657

— **ug02.ht** —

```
\begin{page}{ugTypesUnionsWSelPage}{2.5.2. Unions With Selectors}
\beginscroll

Like records ( \downlink{‘‘Records’’}{ugTypesRecordsPage} in Section
2.4\ignore{ugTypesRecords}), you can write \pspadtype{Union} types
```

```
with selectors.

\beginImportant
The syntax for writing a \pspadtype{Union} type with selectors is
\centerline{{{\tt
Union(\subscriptIt{selector}{1}:\subscriptIt{type}{1},
\subscriptIt{selector}{2}:\subscriptIt{type}{2}, \ldots,
\subscriptIt{selector}{N}:\subscriptIt{type}{N})}}} You must be
careful if a selector has the same name as a variable in the
workspace.  If this occurs, precede the selector name by a single
quote.  It is an error to use a selector that does not correspond to
the branch of the \pspadtype{Union} in which the element actually
lies.
\endImportant

Be sure to understand the difference between records and unions with
selectors.  Records can have more than one component and the selectors
are used to refer to the components.  Unions always have one component
but the type of that one component can vary.  An object of type
\pspadtype{Record(a: Integer, b: Float, c: String)} contains an
integer {\it and} a float {\it and} a string.  An object of type
\pspadtype{Union(a: Integer, b: Float, c: String)} contains an integer
{\it or} a float {\it or} a string.

Here is a version of the \userfun{sayBranch} function (cf.
\downlink{``Unions Without Selectors''}{ugTypesUnionsWOSelPage}
in Section 2.5.1\ignore{ugTypesUnionsWOSel})
that works with a union with selectors.
It displays a message stating in which branch of the \pspadtype{Union} the
object lies.
\begin{verbatim}
sayBranch(x:Union(i:Integer,s:String,f:Float)):Void==
  output
    x case i => "Integer branch"
    x case s  => "String branch"
    "Float branch"
\end{verbatim}
Note that \axiom{case} uses the selector name as its right-hand argument.
\spadkey{case}
If you accidentally use the branch type on the right-hand side of
\axiom{case}, \axiom{false} will be returned.

\xtc{
Declare variable \axiom{u} to have a union type with selectors.
}{
\spadpaste{u : Union(i : Integer, s : String) \bound{undec}}
}
\xtc{
Give an initial value to \axiom{u}.
}{
```

```
\spadpaste{u := "good morning" \bound{u}\free{undec}}
}
\xtc{
Use \axiom{case} to determine in which
branch of a \pspadtype{Union} an object lies.
}{
\spadpaste{u case i \free{u}}
}
\xtc{
}{
\spadpaste{u case s \free{u}}
}
\xtc{
To access the element in a particular branch, use the selector.
}{
\spadpaste{u.s \free{u}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugTypesUnionsWSelPagePatch1}
\begin{paste}{ugTypesUnionsWSelPageFull1}{ugTypesUnionsWSelPageEmpty1}
\pastebutton{ugTypesUnionsWSelPageFull1}{\hidepaste}
\tab{5}\spadcommand{u : Union(i : Integer, s : String)\bound{undec }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesUnionsWSelPageEmpty1}
\begin{paste}{ugTypesUnionsWSelPageEmpty1}{ugTypesUnionsWSelPagePatch1}
\pastebutton{ugTypesUnionsWSelPageEmpty1}{\showpaste}
\tab{5}\spadcommand{u : Union(i : Integer, s : String)\bound{undec }}
\end{paste}\end{patch}

\begin{patch}{ugTypesUnionsWSelPagePatch2}
\begin{paste}{ugTypesUnionsWSelPageFull2}{ugTypesUnionsWSelPageEmpty2}
\pastebutton{ugTypesUnionsWSelPageFull2}{\hidepaste}
\tab{5}\spadcommand{u := "good morning"\bound{u }\free{undec }}
\indentrel{3}\begin{verbatim}
   (2)  "good morning"
                             Type: Union(s: String,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesUnionsWSelPageEmpty2}
\begin{paste}{ugTypesUnionsWSelPageEmpty2}{ugTypesUnionsWSelPagePatch2}
\pastebutton{ugTypesUnionsWSelPageEmpty2}{\showpaste}
```

```
\tab{5}\spadcommand{u := "good morning"\bound{u }\free{undec }}
\end{paste}\end{patch}

\begin{patch}{ugTypesUnionsWSelPagePatch3}
\begin{paste}{ugTypesUnionsWSelPageFull3}{ugTypesUnionsWSelPageEmpty3}
\pastebutton{ugTypesUnionsWSelPageFull3}{\hidepaste}
\tab{5}\spadcommand{u case i\free{u }}
\indentrel{3}\begin{verbatim}
   (3)  false
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesUnionsWSelPageEmpty3}
\begin{paste}{ugTypesUnionsWSelPageEmpty3}{ugTypesUnionsWSelPagePatch3}
\pastebutton{ugTypesUnionsWSelPageEmpty3}{\showpaste}
\tab{5}\spadcommand{u case i\free{u }}
\end{paste}\end{patch}

\begin{patch}{ugTypesUnionsWSelPagePatch4}
\begin{paste}{ugTypesUnionsWSelPageFull4}{ugTypesUnionsWSelPageEmpty4}
\pastebutton{ugTypesUnionsWSelPageFull4}{\hidepaste}
\tab{5}\spadcommand{u case s\free{u }}
\indentrel{3}\begin{verbatim}
   (4)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesUnionsWSelPageEmpty4}
\begin{paste}{ugTypesUnionsWSelPageEmpty4}{ugTypesUnionsWSelPagePatch4}
\pastebutton{ugTypesUnionsWSelPageEmpty4}{\showpaste}
\tab{5}\spadcommand{u case s\free{u }}
\end{paste}\end{patch}

\begin{patch}{ugTypesUnionsWSelPagePatch5}
\begin{paste}{ugTypesUnionsWSelPageFull5}{ugTypesUnionsWSelPageEmpty5}
\pastebutton{ugTypesUnionsWSelPageFull5}{\hidepaste}
\tab{5}\spadcommand{u.s\free{u }}
\indentrel{3}\begin{verbatim}
   (5)  "good morning"
                                           Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesUnionsWSelPageEmpty5}
\begin{paste}{ugTypesUnionsWSelPageEmpty5}{ugTypesUnionsWSelPagePatch5}
\pastebutton{ugTypesUnionsWSelPageEmpty5}{\showpaste}
\tab{5}\spadcommand{u.s\free{u }}
\end{paste}\end{patch}
```

———-

The "Any" Domain

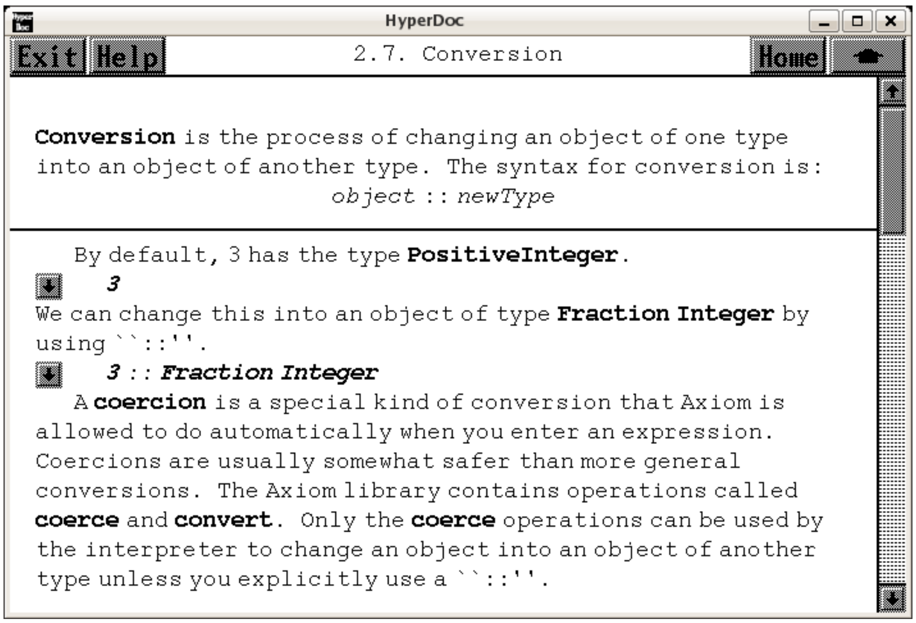
— ug02.ht —

```
\begin{page}{ugTypesAnyNonePage}{2.6. The ‘‘Any’’ Domain}
\beginscroll

With the exception of objects of type \pspadtype{Record}, all Axiom
data structures are homogenous, that is, they hold objects all of the same
type.
If you need to get around this, you can use type \axiomType{Any}.
Using \axiomType{Any}, for example, you can create lists whose
elements are integers, rational numbers, strings, and even other lists.

\xtc{
Declare \axiom{u} to have type \axiomType{Any}.
}{
\spadpaste{u: Any\bound{uany}}
}
\xtc{
Assign a list of mixed type values to \axiom{u}
}{
\spadpaste{u := [1, 7.2, 3/2, x**2, "wally"]\free{uany}\bound{u}}
}
\xtc{
When we ask for the elements, Axiom displays these types.
}{
\spadpaste{u.1 \free{u}}
}
\xtc{
Actually, these objects belong to \axiomType{Any} but Axiom
automatically converts them to their natural types for you.
}{
\spadpaste{u.3 \free{u}}
}
\xtc{
Since type \axiomType{Any} can be anything,
it can only belong to type \axiomType{Type}.
Therefore it cannot be used in algebraic domains.
}{
\spadpaste{v : Matrix(Any)}
}

Perhaps you are wondering how Axiom internally represents
```

```
objects of type \axiomType{Any}.
An object of type \axiomType{Any} consists not only a data part
representing its normal value, but also a type part (a {\it badge}) giving
its type.
For example, the value \axiom{1} of type \axiomType{PositiveInteger} as an
object of type \axiomType{Any} internally looks like
\axiom{[1,\axiomType{PositiveInteger()}]}.

%When should you use \axiomType{Any} instead of a \pspadtype{Union} type?
%Can you plan ahead?
%For a \pspadtype{Union}, you must know in advance exactly which types you
%are
%\index{union!vs. Any@{vs. \protect\nonLibAxiomType{Any}}}
%going to allow.
%For \axiomType{Any}, anything that comes along can be accommodated.

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugTypesAnyNonePagePatch1}
\begin{paste}{ugTypesAnyNonePageFull1}{ugTypesAnyNonePageEmpty1}
\pastebutton{ugTypesAnyNonePageFull1}{\hidepaste}
\tab{5}\spadcommand{u: Any\bound{uany }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesAnyNonePageEmpty1}
\begin{paste}{ugTypesAnyNonePageEmpty1}{ugTypesAnyNonePagePatch1}
\pastebutton{ugTypesAnyNonePageEmpty1}{\showpaste}
\tab{5}\spadcommand{u: Any\bound{uany }}
\end{paste}\end{patch}

\begin{patch}{ugTypesAnyNonePagePatch2}
\begin{paste}{ugTypesAnyNonePageFull2}{ugTypesAnyNonePageEmpty2}
\pastebutton{ugTypesAnyNonePageFull2}{\hidepaste}
\tab{5}\spadcommand{u := [1, 7.2, 3/2, x**2, "wally"]\free{uany }\bound{u }}
\indentrel{3}\begin{verbatim}
               3  2
   (2)  [1,7.2,,x ,"wally"]
               2
                                         Type: List Any
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesAnyNonePageEmpty2}
\begin{paste}{ugTypesAnyNonePageEmpty2}{ugTypesAnyNonePagePatch2}
\pastebutton{ugTypesAnyNonePageEmpty2}{\showpaste}
```

```
\tab{5}\spadcommand{u := [1, 7.2, 3/2, x**2, "wally"]\free{uany }\bound{u }}
\end{paste}\end{patch}

\begin{patch}{ugTypesAnyNonePagePatch3}
\begin{paste}{ugTypesAnyNonePageFull3}{ugTypesAnyNonePageEmpty3}
\pastebutton{ugTypesAnyNonePageFull3}{\hidepaste}
\tab{5}\spadcommand{u.1\free{u }}
\indentrel{3}\begin{verbatim}
   (3)  1
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesAnyNonePageEmpty3}
\begin{paste}{ugTypesAnyNonePageEmpty3}{ugTypesAnyNonePagePatch3}
\pastebutton{ugTypesAnyNonePageEmpty3}{\showpaste}
\tab{5}\spadcommand{u.1\free{u }}
\end{paste}\end{patch}

\begin{patch}{ugTypesAnyNonePagePatch4}
\begin{paste}{ugTypesAnyNonePageFull4}{ugTypesAnyNonePageEmpty4}
\pastebutton{ugTypesAnyNonePageFull4}{\hidepaste}
\tab{5}\spadcommand{u.3\free{u }}
\indentrel{3}\begin{verbatim}
        3
   (4)
        2
                                 Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesAnyNonePageEmpty4}
\begin{paste}{ugTypesAnyNonePageEmpty4}{ugTypesAnyNonePagePatch4}
\pastebutton{ugTypesAnyNonePageEmpty4}{\showpaste}
\tab{5}\spadcommand{u.3\free{u }}
\end{paste}\end{patch}

\begin{patch}{ugTypesAnyNonePagePatch5}
\begin{paste}{ugTypesAnyNonePageFull5}{ugTypesAnyNonePageEmpty5}
\pastebutton{ugTypesAnyNonePageFull5}{\hidepaste}
\tab{5}\spadcommand{v : Matrix(Any)}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesAnyNonePageEmpty5}
\begin{paste}{ugTypesAnyNonePageEmpty5}{ugTypesAnyNonePagePatch5}
\pastebutton{ugTypesAnyNonePageEmpty5}{\showpaste}
\tab{5}\spadcommand{v : Matrix(Any)}
\end{paste}\end{patch}
```

Conversion



⇐ "Complex" (ComplexXmpPage) 3.16 on page 254
⇒ "The Basic Idea" (ugTypesBasicPage) 7 on page 1614

— **ug02.ht** —

```
\begin{page}{ugTypesConvertPage}{2.7. Conversion}
\beginscroll
%
\beginImportant
\spadglossSee{Conversion}{conversion}
is the process of changing an object of one type
into an object of another type.
The syntax for conversion is:
\centerline{{{\it object} {\tt ::} {\it newType}}}
\endImportant

\xtc{
By default, \axiom{3} has the type \axiomType{PositiveInteger}.
}{
\spadpaste{3}
}
```

```
\xtc{
We can change this into an object of type \axiomType{Fraction Integer}
by using \axiomSyntax{::}.
}{
\spadpaste{3 :: Fraction Integer}
}

A \spadgloss{coercion} is a special kind of conversion that Axiom is
allowed to do automatically when you enter an expression.
Coercions are usually somewhat safer than more general conversions.
The Axiom library contains operations called
\axiomFun{coerce} and \axiomFun{convert}.
Only the \axiomFun{coerce} operations can be used by the
interpreter to change an object into an object of another type unless
you explicitly use a \axiomSyntax{::}.

By now you will be quite familiar with what types and modes look like.
It is useful to think of a type or mode as a pattern
for what you want the result to be.
\xtc{
Let's start with a square matrix of polynomials with complex rational
number coefficients.
}{
\spadpaste{m : SquareMatrix(2,POLY COMPLEX FRAC INT) \bound{mdec}}
}
\xtc{
}{
\spadpaste{
m := matrix [[x-3/4*\%i,z*y**2+1/2],[3/7*\%i*y**4 - x,12-\%i*9/5]]
\bound{m}\free{mdec}}
}
\xtc{
We first want to interchange the \axiomType{Complex} and
\axiomType{Fraction} layers.
We do the conversion by doing the interchange in the type expression.
}{
\spadpaste{m1 := m :: SquareMatrix(2,POLY FRAC COMPLEX INT)
\free{m}\bound{m1}}
}
\xtc{
Interchange the \axiomType{Polynomial} and the
\axiomType{Fraction} levels.
}{
\spadpaste{m2 := m1 :: SquareMatrix(2,FRAC POLY COMPLEX INT)
\free{m1}\bound{m2}}
}
\xtc{
Interchange the \axiomType{Polynomial} and the
\axiomType{Complex} levels.
}{
```

```
\spadpaste{m3 := m2 :: SquareMatrix(2,FRAC COMPLEX POLY INT)
\free{m2}\bound{m3}}
}

All the entries have changed types, although in comparing the
last two results only the entry in the lower left corner looks different.
We did all the intermediate steps to show you what Axiom can do.

\xtc{
In fact, we could have combined all these into one conversion.
}{
\spadpaste{m :: SquareMatrix(2,FRAC COMPLEX POLY INT) \free{m}}
}

There are times when Axiom is not be able to do the conversion
in one step.
You may need to break up the transformation into several conversions
in order to get an object of the desired type.

We cannot move either \axiomType{Fraction} or \axiomType{Complex}
above (or to the left of, depending on how you look at it)
\axiomType{SquareMatrix} because each of these levels requires that its
argument type have commutative multiplication, whereas
\axiomType{SquareMatrix} does not.\footnote{\axiomType{Fraction} requires
that its argument belong to the category \axiomType{IntegralDomain} and
\axiomType{Complex} requires that its argument belong to
\axiomType{CommutativeRing}. See
\downlink{``The Basic Idea''}{ugTypesBasicPage}
in Section 2.1\ignore{ugTypesBasic}
for a brief discussion of categories.}
The \axiomType{Integer} level did not move anywhere
because it does not allow any arguments.
We also did not move the \axiomType{SquareMatrix} part anywhere, but
we could have.
\xtc{
Recall that \axiom{m} looks like this.
}{
\spadpaste{m \free{m}}
}
\xtc{
If we want a polynomial with matrix coefficients rather than a matrix
with polynomial entries, we can just do the conversion.
}{
\spadpaste{m :: POLY SquareMatrix(2,COMPLEX FRAC INT) \free{m}}
}
\xtc{
We have not yet used modes for any conversions.
Modes are a great shorthand for indicating the type of the
object you want.
Instead of using the long type expression in the
```

```
last example, we could have simply said this.
}{
\spadpaste{m :: POLY ? \free{m}}
}
\xtc{
We can also indicate more structure if we want the entries
of the matrices to be fractions.
}{
\spadpaste{m :: POLY SquareMatrix(2,FRAC ?) \free{m}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugTypesConvertPagePatch1}
\begin{paste}{ugTypesConvertPageFull1}{ugTypesConvertPageEmpty1}
\pastebutton{ugTypesConvertPageFull1}{\hidepaste}
\tab{5}\spadcommand{3}
\indentrel{3}\begin{verbatim}
   (1)  3
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesConvertPageEmpty1}
\begin{paste}{ugTypesConvertPageEmpty1}{ugTypesConvertPagePatch1}
\pastebutton{ugTypesConvertPageEmpty1}{\showpaste}
\tab{5}\spadcommand{3}
\end{paste}\end{patch}

\begin{patch}{ugTypesConvertPagePatch2}
\begin{paste}{ugTypesConvertPageFull2}{ugTypesConvertPageEmpty2}
\pastebutton{ugTypesConvertPageFull2}{\hidepaste}
\tab{5}\spadcommand{3 :: Fraction Integer}
\indentrel{3}\begin{verbatim}
   (2)  3
                                 Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesConvertPageEmpty2}
\begin{paste}{ugTypesConvertPageEmpty2}{ugTypesConvertPagePatch2}
\pastebutton{ugTypesConvertPageEmpty2}{\showpaste}
\tab{5}\spadcommand{3 :: Fraction Integer}
\end{paste}\end{patch}

\begin{patch}{ugTypesConvertPagePatch3}
\begin{paste}{ugTypesConvertPageFull3}{ugTypesConvertPageEmpty3}
\pastebutton{ugTypesConvertPageFull3}{\hidepaste}
```

```
\tab{5}\spadcommand{m : SquareMatrix(2,POLY COMPLEX FRAC INT)\bound{mdec }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesConvertPageEmpty3}
\begin{paste}{ugTypesConvertPageEmpty3}{ugTypesConvertPagePatch3}
\pastebutton{ugTypesConvertPageEmpty3}{\showpaste}
\tab{5}\spadcommand{m : SquareMatrix(2,POLY COMPLEX FRAC INT)\bound{mdec }}
\end{paste}\end{patch}

\begin{patch}{ugTypesConvertPagePatch4}
\begin{paste}{ugTypesConvertPageFull4}{ugTypesConvertPageEmpty4}
\pastebutton{ugTypesConvertPageFull4}{\hidepaste}
\tab{5}\spadcommand{m := matrix [[x-3/4*\%i,z*y**2+1/2],[3/7*\%i*y**4 - x,12-\%i*9/5]]\bound{m }\free{mdec }
\indentrel{3}\begin{verbatim}
            3        2    1
        x -  %i     y z +
            4             2
   (4)
       3     4          9
        %i y  - x  12 -  %i
       7                5
Type: SquareMatrix(2,Polynomial Complex Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesConvertPageEmpty4}
\begin{paste}{ugTypesConvertPageEmpty4}{ugTypesConvertPagePatch4}
\pastebutton{ugTypesConvertPageEmpty4}{\showpaste}
\tab{5}\spadcommand{m := matrix [[x-3/4*\%i,z*y**2+1/2],[3/7*\%i*y**4 - x,12-\%i*9/5]]\bound{m }\free{mdec }
\end{paste}\end{patch}

\begin{patch}{ugTypesConvertPagePatch5}
\begin{paste}{ugTypesConvertPageFull5}{ugTypesConvertPageEmpty5}
\pastebutton{ugTypesConvertPageFull5}{\hidepaste}
\tab{5}\spadcommand{m1 := m :: SquareMatrix(2,POLY FRAC COMPLEX INT)\free{m }\bound{m1 }}
\indentrel{3}\begin{verbatim}
            3%i     2    1
        x -      y z +
             4           2
   (5)
       3%i  4      60 - 9%i
        y  - x
        7              5
Type: SquareMatrix(2,Polynomial Fraction Complex Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugTypesConvertPageEmpty5}
\begin{paste}{ugTypesConvertPageEmpty5}{ugTypesConvertPagePatch5}
\pastebutton{ugTypesConvertPageEmpty5}{\showpaste}
\tab{5}\spadcommand{m1 := m :: SquareMatrix(2,POLY FRAC COMPLEX INT)\free{m }\bound{m1 }}
\end{paste}\end{patch}

\begin{patch}{ugTypesConvertPagePatch6}
\begin{paste}{ugTypesConvertPageFull6}{ugTypesConvertPageEmpty6}
\pastebutton{ugTypesConvertPageFull6}{\hidepaste}
\tab{5}\spadcommand{m2 := m1 :: SquareMatrix(2,FRAC POLY COMPLEX INT)\free{m1 }\bound{m2 }}
\indentrel{3}\begin{verbatim}
                        2
         4x - 3%i    2y z + 1

             4           2
   (6)
             4
        3%i y  - 7x  60 - 9%i

             7           5
Type: SquareMatrix(2,Fraction Polynomial Complex Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesConvertPageEmpty6}
\begin{paste}{ugTypesConvertPageEmpty6}{ugTypesConvertPagePatch6}
\pastebutton{ugTypesConvertPageEmpty6}{\showpaste}
\tab{5}\spadcommand{m2 := m1 :: SquareMatrix(2,FRAC POLY COMPLEX INT)\free{m1 }\bound{m2 }}
\end{paste}\end{patch}

\begin{patch}{ugTypesConvertPagePatch7}
\begin{paste}{ugTypesConvertPageFull7}{ugTypesConvertPageEmpty7}
\pastebutton{ugTypesConvertPageFull7}{\hidepaste}
\tab{5}\spadcommand{m3 := m2 :: SquareMatrix(2,FRAC COMPLEX POLY INT)\free{m2 }\bound{m3 }}
\indentrel{3}\begin{verbatim}
                        2
          4x - 3%i    2y z + 1

              4           2
   (7)
                 4
        - 7x + 3y %i  60 - 9%i

              7           5
Type: SquareMatrix(2,Fraction Complex Polynomial Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesConvertPageEmpty7}
\begin{paste}{ugTypesConvertPageEmpty7}{ugTypesConvertPagePatch7}
```

```
\pastebutton{ugTypesConvertPageEmpty7}{\showpaste}
\tab{5}\spadcommand{m3 := m2 :: SquareMatrix(2,FRAC COMPLEX POLY INT)\free{m2 }\bound{m3 }}
\end{paste}\end{patch}

\begin{patch}{ugTypesConvertPagePatch8}
\begin{paste}{ugTypesConvertPageFull8}{ugTypesConvertPageEmpty8}
\pastebutton{ugTypesConvertPageFull8}{\hidepaste}
\tab{5}\spadcommand{m :: SquareMatrix(2,FRAC COMPLEX POLY INT)\free{m }}
\indentrel{3}\begin{verbatim}
                        2
          4x - 3%i    2y z + 1

              4           2
   (8)
                 4
        - 7x + 3y %i  60 - 9%i

              7           5
Type: SquareMatrix(2,Fraction Complex Polynomial Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesConvertPageEmpty8}
\begin{paste}{ugTypesConvertPageEmpty8}{ugTypesConvertPagePatch8}
\pastebutton{ugTypesConvertPageEmpty8}{\showpaste}
\tab{5}\spadcommand{m :: SquareMatrix(2,FRAC COMPLEX POLY INT)\free{m }}
\end{paste}\end{patch}

\begin{patch}{ugTypesConvertPagePatch9}
\begin{paste}{ugTypesConvertPageFull9}{ugTypesConvertPageEmpty9}
\pastebutton{ugTypesConvertPageFull9}{\hidepaste}
\tab{5}\spadcommand{m\free{m }}
\indentrel{3}\begin{verbatim}
             3        2    1
         x -  %i     y z +
             4             2
   (9)
        3     4           9
         %i y  - x  12 -  %i
        7                 5
Type: SquareMatrix(2,Polynomial Complex Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesConvertPageEmpty9}
\begin{paste}{ugTypesConvertPageEmpty9}{ugTypesConvertPagePatch9}
\pastebutton{ugTypesConvertPageEmpty9}{\showpaste}
\tab{5}\spadcommand{m\free{m }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugTypesConvertPagePatch10}
\begin{paste}{ugTypesConvertPageFull10}{ugTypesConvertPageEmpty10}
\pastebutton{ugTypesConvertPageFull10}{\hidepaste}
\tab{5}\spadcommand{m :: POLY SquareMatrix(2,COMPLEX FRAC INT)\free{m }}
\indentrel{3}\begin{verbatim}
   (10)
                  0    0
     0  1 2            4    1   0
         y z + 3      y  +       x
     0  0       %i  0     - 1  0
                 7
   +
       3         1
     -  %i
       4         2

                  9
       0     12 -  %i
                  5
Type: Polynomial SquareMatrix(2,Complex Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesConvertPageEmpty10}
\begin{paste}{ugTypesConvertPageEmpty10}{ugTypesConvertPagePatch10}
\pastebutton{ugTypesConvertPageEmpty10}{\showpaste}
\tab{5}\spadcommand{m :: POLY SquareMatrix(2,COMPLEX FRAC INT)\free{m }}
\end{paste}\end{patch}

\begin{patch}{ugTypesConvertPagePatch11}
\begin{paste}{ugTypesConvertPageFull11}{ugTypesConvertPageEmpty11}
\pastebutton{ugTypesConvertPageFull11}{\hidepaste}
\tab{5}\spadcommand{m :: POLY ?\free{m }}
\indentrel{3}\begin{verbatim}
   (11)
                  0    0
     0  1 2            4    1   0
         y z + 3      y  +       x
     0  0       %i  0     - 1  0
                 7
   +
       3         1
     -  %i
       4         2

                  9
       0     12 -  %i
                  5
Type: Polynomial SquareMatrix(2,Complex Fraction Integer)
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesConvertPageEmpty11}
\begin{paste}{ugTypesConvertPageEmpty11}{ugTypesConvertPagePatch11}
\pastebutton{ugTypesConvertPageEmpty11}{\showpaste}
\tab{5}\spadcommand{m :: POLY ?\free{m }}
\end{paste}\end{patch}

\begin{patch}{ugTypesConvertPagePatch12}
\begin{paste}{ugTypesConvertPageFull12}{ugTypesConvertPageEmpty12}
\pastebutton{ugTypesConvertPageFull12}{\hidepaste}
\tab{5}\spadcommand{m :: POLY SquareMatrix(2,FRAC ?)\free{m }}
\indentrel{3}\begin{verbatim}
   (12)
                                          3%i     1
                0   0                 -
   0  1 2          4    1   0       4      2
       y z + 3%i   y  +       x +
   0  0        0     - 1  0          60 - 9%i
                7                      0
                                                   5
Type: Polynomial SquareMatrix(2,Fraction Complex Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesConvertPageEmpty12}
\begin{paste}{ugTypesConvertPageEmpty12}{ugTypesConvertPagePatch12}
\pastebutton{ugTypesConvertPageEmpty12}{\showpaste}
\tab{5}\spadcommand{m :: POLY SquareMatrix(2,FRAC ?)\free{m }}
\end{paste}\end{patch}
```

———-

Subdomains Again

— ug02.ht —

```
\begin{page}{ugTypesSubdomainsPage}{2.8. Subdomains Again}
\beginscroll

A \spadgloss{subdomain} \axiom{S} of a domain \axiom{D} is a domain
consisting of
\indent{4}
\beginitems
\item[1. ] those elements of \axiom{D} that satisfy some
\spadgloss{predicate} (that is, a test that returns \axiom{true} or
\axiom{false}) and
```

```
\item[2. ] a subset of the operations of \axiom{D}.
\enditems
\indent{0}
Every domain is a subdomain of itself, trivially satisfying the
membership test: \axiom{true}.

Currently, there are only two system-defined subdomains in Axiom that
receive substantial use.  \axiomType{PositiveInteger} and
\axiomType{NonNegativeInteger} are subdomains of \axiomType{Integer}.
An element \axiom{x} of \axiomType{NonNegativeInteger} is an integer
that is greater than or equal to zero, that is, satisfies \axiom{x >=
0.}  An element \axiom{x} of \axiomType{PositiveInteger} is a
nonnegative integer that is, in fact, greater than zero, that is,
satisfies \axiom{x > 0.}  Not all operations from \axiomType{Integer}
are available for these subdomains.  For example, negation and
subtraction are not provided since the subdomains are not closed under
those operations.  When you use an integer in an expression, Axiom
assigns to it the type that is the most specific subdomain whose
predicate is satisfied.
\xtc{
This is a positive integer.
}{
\spadpaste{5}
}
\xtc{
This is a nonnegative integer.
}{
\spadpaste{0}
}
\xtc{
This is neither of the above.
}{
\spadpaste{-5}
}
\xtc{
Furthermore, unless you are assigning an integer to a declared variable
or using a conversion, any integer result has as type the most
specific subdomain.
}{
\spadpaste{(-2) - (-3)}
}
\xtc{
}{
\spadpaste{0 :: Integer}
}
\xtc{
}{
\spadpaste{x : NonNegativeInteger := 5}
}
```

```
When necessary, Axiom converts an integer object into one belonging to
a less specific subdomain.  For example, in \axiom{3-2}, the arguments
to \axiomOpFrom{-}{Integer} are both elements of
\axiomType{PositiveInteger}, but this type does not provide a
subtraction operation.  Neither does \axiomType{NonNegativeInteger},
so \axiom{3} and \axiom{2} are viewed as elements of
\axiomType{Integer}, where their difference can be calculated.  The
result is \axiom{1}, which Axiom then automatically assigns the type
\axiomType{PositiveInteger}.

\xtc{
Certain operations are very sensitive to the subdomains to which their
arguments belong.
This is an element of \axiomType{PositiveInteger}.
}{
\spadpaste{2 ** 2}
}
\xtc{
This is an element of \axiomType{Fraction Integer}.
}{
\spadpaste{2 ** (-2)}
}
\xtc{
It makes sense then that this
is a list of elements of \axiomType{PositiveInteger}.
}{
\spadpaste{[10**i for i in 2..5]}
}
What should the type of \axiom{[10**(i-1) for i in 2..5]} be?
On one hand, \axiom{i-1} is always an integer greater than zero
as \axiom{i} ranges from \axiom{2} to \axiom{5} and so \axiom{10**i}
is also always a positive integer.
On the other, \axiom{i-1} is a very simple function of \axiom{i}.
Axiom does not try to analyze every such function over the
index's range of values to determine whether it is always positive
or nowhere negative.
For an arbitrary Axiom function, this analysis is not possible.

\xtc{
So, to be consistent no such analysis is done and we get this.
}{
\spadpaste{[10**(i-1) for i in 2..5]}
}
\xtc{
To get a list of elements of \axiomType{PositiveInteger} instead, you
have two choices.
You can use a conversion.
}{
\spadpaste{[10**((i-1) :: PI) for i in 2..5]}
}
```

```
\xtc{
Or you can use \axiom{pretend}.
\spadkey{pretend}
}{
\spadpaste{[10**((i-1) pretend PI) for i in 2..5]}
}

The operation \axiom{pretend} is used to defeat the Axiom
type system.
The expression \axiom{object pretend D} means ‘‘make a new object
(without copying) of type \axiom{D} from \axiom{object}.’’
If \axiom{object} were an integer and you told Axiom
to pretend it was a list, you would probably see a message about a
fatal error being caught and memory possibly being damaged.
Lists do not have the same internal representation as integers!

You use \axiom{pretend} at your peril.

\xtc{
Use \axiom{pretend} with great care!
Axiom trusts you that the value is of the specified type.
}{
\spadpaste{(2/3) pretend Complex Integer}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugTypesSubdomainsPagePatch1}
\begin{paste}{ugTypesSubdomainsPageFull1}{ugTypesSubdomainsPageEmpty1}
\pastebutton{ugTypesSubdomainsPageFull1}{\hidepaste}
\tab{5}\spadcommand{5}
\indentrel{3}\begin{verbatim}
   (1)  5
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesSubdomainsPageEmpty1}
\begin{paste}{ugTypesSubdomainsPageEmpty1}{ugTypesSubdomainsPagePatch1}
\pastebutton{ugTypesSubdomainsPageEmpty1}{\showpaste}
\tab{5}\spadcommand{5}
\end{paste}\end{patch}

\begin{patch}{ugTypesSubdomainsPagePatch2}
\begin{paste}{ugTypesSubdomainsPageFull2}{ugTypesSubdomainsPageEmpty2}
\pastebutton{ugTypesSubdomainsPageFull2}{\hidepaste}
\tab{5}\spadcommand{0}
\indentrel{3}\begin{verbatim}
```

```
   (2)  0
                                   Type: NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesSubdomainsPageEmpty2}
\begin{paste}{ugTypesSubdomainsPageEmpty2}{ugTypesSubdomainsPagePatch2}
\pastebutton{ugTypesSubdomainsPageEmpty2}{\showpaste}
\tab{5}\spadcommand{0}
\end{paste}\end{patch}

\begin{patch}{ugTypesSubdomainsPagePatch3}
\begin{paste}{ugTypesSubdomainsPageFull3}{ugTypesSubdomainsPageEmpty3}
\pastebutton{ugTypesSubdomainsPageFull3}{\hidepaste}
\tab{5}\spadcommand{-5}
\indentrel{3}\begin{verbatim}
   (3)  - 5
                                              Type: Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesSubdomainsPageEmpty3}
\begin{paste}{ugTypesSubdomainsPageEmpty3}{ugTypesSubdomainsPagePatch3}
\pastebutton{ugTypesSubdomainsPageEmpty3}{\showpaste}
\tab{5}\spadcommand{-5}
\end{paste}\end{patch}

\begin{patch}{ugTypesSubdomainsPagePatch4}
\begin{paste}{ugTypesSubdomainsPageFull4}{ugTypesSubdomainsPageEmpty4}
\pastebutton{ugTypesSubdomainsPageFull4}{\hidepaste}
\tab{5}\spadcommand{(-2) - (-3)}
\indentrel{3}\begin{verbatim}
   (4)  1
                                      Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesSubdomainsPageEmpty4}
\begin{paste}{ugTypesSubdomainsPageEmpty4}{ugTypesSubdomainsPagePatch4}
\pastebutton{ugTypesSubdomainsPageEmpty4}{\showpaste}
\tab{5}\spadcommand{(-2) - (-3)}
\end{paste}\end{patch}

\begin{patch}{ugTypesSubdomainsPagePatch5}
\begin{paste}{ugTypesSubdomainsPageFull5}{ugTypesSubdomainsPageEmpty5}
\pastebutton{ugTypesSubdomainsPageFull5}{\hidepaste}
\tab{5}\spadcommand{0 :: Integer}
\indentrel{3}\begin{verbatim}
   (5)  0
                                              Type: Integer
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesSubdomainsPageEmpty5}
\begin{paste}{ugTypesSubdomainsPageEmpty5}{ugTypesSubdomainsPagePatch5}
\pastebutton{ugTypesSubdomainsPageEmpty5}{\showpaste}
\tab{5}\spadcommand{0 :: Integer}
\end{paste}\end{patch}

\begin{patch}{ugTypesSubdomainsPagePatch6}
\begin{paste}{ugTypesSubdomainsPageFull6}{ugTypesSubdomainsPageEmpty6}
\pastebutton{ugTypesSubdomainsPageFull6}{\hidepaste}
\tab{5}\spadcommand{x : NonNegativeInteger := 5}
\indentrel{3}\begin{verbatim}
   (6)  5
                               Type: NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesSubdomainsPageEmpty6}
\begin{paste}{ugTypesSubdomainsPageEmpty6}{ugTypesSubdomainsPagePatch6}
\pastebutton{ugTypesSubdomainsPageEmpty6}{\showpaste}
\tab{5}\spadcommand{x : NonNegativeInteger := 5}
\end{paste}\end{patch}

\begin{patch}{ugTypesSubdomainsPagePatch7}
\begin{paste}{ugTypesSubdomainsPageFull7}{ugTypesSubdomainsPageEmpty7}
\pastebutton{ugTypesSubdomainsPageFull7}{\hidepaste}
\tab{5}\spadcommand{2 ** 2}
\indentrel{3}\begin{verbatim}
   (7)  4
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesSubdomainsPageEmpty7}
\begin{paste}{ugTypesSubdomainsPageEmpty7}{ugTypesSubdomainsPagePatch7}
\pastebutton{ugTypesSubdomainsPageEmpty7}{\showpaste}
\tab{5}\spadcommand{2 ** 2}
\end{paste}\end{patch}

\begin{patch}{ugTypesSubdomainsPagePatch8}
\begin{paste}{ugTypesSubdomainsPageFull8}{ugTypesSubdomainsPageEmpty8}
\pastebutton{ugTypesSubdomainsPageFull8}{\hidepaste}
\tab{5}\spadcommand{2 ** (-2)}
\indentrel{3}\begin{verbatim}
        1
   (8)
        4
                                 Type: Fraction Integer
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesSubdomainsPageEmpty8}
\begin{paste}{ugTypesSubdomainsPageEmpty8}{ugTypesSubdomainsPagePatch8}
\pastebutton{ugTypesSubdomainsPageEmpty8}{\showpaste}
\tab{5}\spadcommand{2 ** (-2)}
\end{paste}\end{patch}

\begin{patch}{ugTypesSubdomainsPagePatch9}
\begin{paste}{ugTypesSubdomainsPageFull9}{ugTypesSubdomainsPageEmpty9}
\pastebutton{ugTypesSubdomainsPageFull9}{\hidepaste}
\tab{5}\spadcommand{[10**i for i in 2..5]}
\indentrel{3}\begin{verbatim}
   (9)  [100,1000,10000,100000]
                            Type: List PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesSubdomainsPageEmpty9}
\begin{paste}{ugTypesSubdomainsPageEmpty9}{ugTypesSubdomainsPagePatch9}
\pastebutton{ugTypesSubdomainsPageEmpty9}{\showpaste}
\tab{5}\spadcommand{[10**i for i in 2..5]}
\end{paste}\end{patch}

\begin{patch}{ugTypesSubdomainsPagePatch10}
\begin{paste}{ugTypesSubdomainsPageFull10}{ugTypesSubdomainsPageEmpty10}
\pastebutton{ugTypesSubdomainsPageFull10}{\hidepaste}
\tab{5}\spadcommand{[10**(i-1) for i in 2..5]}
\indentrel{3}\begin{verbatim}
   (10)  [10,100,1000,10000]
                           Type: List Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesSubdomainsPageEmpty10}
\begin{paste}{ugTypesSubdomainsPageEmpty10}{ugTypesSubdomainsPagePatch10}
\pastebutton{ugTypesSubdomainsPageEmpty10}{\showpaste}
\tab{5}\spadcommand{[10**(i-1) for i in 2..5]}
\end{paste}\end{patch}

\begin{patch}{ugTypesSubdomainsPagePatch11}
\begin{paste}{ugTypesSubdomainsPageFull11}{ugTypesSubdomainsPageEmpty11}
\pastebutton{ugTypesSubdomainsPageFull11}{\hidepaste}
\tab{5}\spadcommand{[10**((i-1) :: PI) for i in 2..5]}
\indentrel{3}\begin{verbatim}
   (11)  [10,100,1000,10000]
                            Type: List PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugTypesSubdomainsPageEmpty11}
\begin{paste}{ugTypesSubdomainsPageEmpty11}{ugTypesSubdomainsPagePatch11}
\pastebutton{ugTypesSubdomainsPageEmpty11}{\showpaste}
\tab{5}\spadcommand{[10**((i-1) :: PI) for i in 2..5]}
\end{paste}\end{patch}

\begin{patch}{ugTypesSubdomainsPagePatch12}
\begin{paste}{ugTypesSubdomainsPageFull12}{ugTypesSubdomainsPageEmpty12}
\pastebutton{ugTypesSubdomainsPageFull12}{\hidepaste}
\tab{5}\spadcommand{[10**((i-1) pretend PI) for i in 2..5]}
\indentrel{3}\begin{verbatim}
   (12)  [10,100,1000,10000]
                            Type: List PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesSubdomainsPageEmpty12}
\begin{paste}{ugTypesSubdomainsPageEmpty12}{ugTypesSubdomainsPagePatch12}
\pastebutton{ugTypesSubdomainsPageEmpty12}{\showpaste}
\tab{5}\spadcommand{[10**((i-1) pretend PI) for i in 2..5]}
\end{paste}\end{patch}

\begin{patch}{ugTypesSubdomainsPagePatch13}
\begin{paste}{ugTypesSubdomainsPageFull13}{ugTypesSubdomainsPageEmpty13}
\pastebutton{ugTypesSubdomainsPageFull13}{\hidepaste}
\tab{5}\spadcommand{(2/3) pretend Complex Integer}
\indentrel{3}\begin{verbatim}
   (13)  2 + 3%i
                                 Type: Complex Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesSubdomainsPageEmpty13}
\begin{paste}{ugTypesSubdomainsPageEmpty13}{ugTypesSubdomainsPagePatch13}
\pastebutton{ugTypesSubdomainsPageEmpty13}{\showpaste}
\tab{5}\spadcommand{(2/3) pretend Complex Integer}
\end{paste}\end{patch}
```

———-

Package Calling and Target Types

⇒ "notitle" (ugTypesDeclarePage) 7 on page 1641
⇒ "notitle" (ugUserUsePage) 10 on page 1854

— **ug02.ht** —

```
\begin{page}{ugTypesPkgCallPage}{2.9. Package Calling and Target Types}
\beginscroll

Axiom works hard to figure out what you mean by an
expression without your having to qualify it with type
information.
Nevertheless, there are times when you need to help it along by
providing hints (or even orders!) to get Axiom to do what
you want.

We saw in \downlink{``Declarations''}{ugTypesDeclarePage}
in Section 2.3\ignore{ugTypesDeclare} that
declarations using types
and modes control the type of the results produced.
For example, we can either produce a complex object with
polynomial real and imaginary parts or a polynomial with complex
integer coefficients, depending on the declaration.

\spadglossSee{Package calling}{package call} is how you tell
Axiom to use a particular function from a particular part of
the library.

\xtc{
Use the \axiomOpFrom{/}{Fraction} from \axiomType{Fraction Integer}
to create a fraction of two integers.
}{
\spadpaste{2/3}
}
\xtc{
If we wanted a floating point number, we can say ``use the
\axiomOpFrom{/}{Float} in \axiomType{Float}.''
}{
\spadpaste{(2/3)\$Float}
}
\xtc{
Perhaps we actually wanted a fraction of complex integers.
}{
\spadpaste{(2/3)\$Fraction(Complex Integer)}
}

In each case, Axiom used the indicated operations, sometimes
first needing to convert the two integers into objects of an
appropriate type.
In these examples, \axiomOpFrom{/}{Fraction} is written as an
infix operator.

\beginImportant
To use package calling with an infix operator, use the
following syntax:
\centerline{{{\tt ( \subscriptIt{arg}{1} {\it op}
```

```
\subscriptIt{arg}{1} )\${\it type} }}}
\endImportant

We used, for example, \axiom{(2/3)\$Float}.  The expression \axiom{2 +
3 + 4} is equivalent to \axiom{(2+3) + 4.}  Therefore in the
expression \axiom{(2 + 3 + 4)\$Float} the second \axiomOp{+} comes
from the \axiomType{Float} domain.  Can you guess whether the first
\axiomOp{+} comes from \axiomType{Integer} or
\axiomType{Float}?\footnote{\axiomType{Float}, because the package
call causes Axiom to convert \axiom{(2 + 3)} and \axiom{4} to type
\axiomType{Float}.  Before the sum is converted, it is given a target
type (see below) of \axiomType{Float} by Axiom and then evaluated.
The target type causes the \axiomOp{+} from \axiomType{Float} to be
used.}

\beginImportant
For an operator written before its arguments, you must use
parentheses around the arguments (even if there is only one),
and follow the closing parenthesis by a \axiomSyntax{\$}
and then the type.
\centerline{{{\tt {\it fun} ( \subscriptIt{arg}{1},
\subscriptIt{arg}{1}, \ldots, \subscriptIt{arg}{N} )\${\it type}}}}
\endImportant

For example, to call the ``minimum'' function from \axiomType{DoubleFloat}
on two integers, you could write \axiom{min(4,89)\$DoubleFloat}.
Another use of package calling is to tell Axiom to use a library
function rather than a function you defined.
We discuss this in
\downlink{``How Axiom Determines What Function to Use''}{ugUserUsePage}
in Section 6.9\ignore{ugUserUse}.

Sometimes rather than specifying where an operation comes from, you just
want to say what type the result should be.
We say that you provide
a
\spadglossSee{target type}{target} for the expression.
Instead of using a \axiomSyntax{\$}, use a \axiomSyntax{@} to specify
the requested target type.
Otherwise, the syntax is the same.
Note that giving a target type is not the same as explicitly doing a
conversion.
The first says ``try to pick operations so that the result has
such-and-such a type.''
The second says ``compute the result and then convert to an object of
such-and-such a type.''

\xtc{
Sometimes it makes sense, as in this expression,
to say ``choose the operations in this expression so that
```

```
the final result is a \axiomType{Float}.''
}{
\spadpaste{(2/3)@Float}
}

Here we used \axiomSyntax{@} to say that the target type of the
left-hand side was \axiomType{Float}.
In this simple case, there was no real difference
between using \axiomSyntax{\$} and \axiomSyntax{@}.
You can see the difference if you try the following.
\xtc{
This says to try to choose \axiomOp{+} so that the result is
a string.
Axiom cannot do this.
}{
\spadpaste{(2 + 3)@String}
}
\xtc{
This says to get the \axiomOp{+} from \axiomType{String} and apply
it to the two integers.
Axiom also cannot do this because there is no \axiomOp{+}
exported by \axiomType{String}.
}{
\spadpaste{(2 + 3)\$String}
}
(By the way, the operation \axiomFunFrom{concat}{String} or juxtaposition
is used to concatenate two strings.)

When we have more than one operation in an expression, the
difference is even more evident.
The following two expressions show that Axiom uses the
target type to create different objects.
The \axiomOp{+}, \axiomOp{*} and \axiomOp{**} operations are all
chosen so that an object of the correct final type is created.

\xtc{
This says that the operations should be chosen so
that the result is a \axiomType{Complex} object.
}{
\spadpaste{((x + y * \%i)**2)@(Complex Polynomial Integer)}
}
\xtc{
This says that the operations should be chosen so
that the result is a \axiomType{Polynomial} object.
}{
\spadpaste{((x + y * \%i)**2)@(Polynomial Complex Integer)}
}
\xtc{
What do you think might happen if we left off all
target type and package call information in this last example?
```

```
}{
\spadpaste{(x + y * \%i)**2 \bound{prevC}}
}
\xtc{
We can convert it to \axiomType{Complex} as an afterthought.
But this is more work than just saying making what we want in the first
place.
}{
\spadpaste{\% :: Complex ? \free{prevC}}
}

Finally, another use of package calling is to qualify fully an
operation that is passed as an argument to a function.

\xtc{
Start with a small matrix of integers.
}{
\spadpaste{h := matrix [[8,6],[-4,9]] \bound{h}}
}
%
\xtc{
We want to produce a new matrix that has for entries the multiplicative
inverses of the entries of \axiom{h}.
One way to do this is by calling
\axiomFunFrom{map}{MatrixCategoryFunctions2} with the
\axiomFunFrom{inv}{Fraction} function from \axiomType{Fraction (Integer)}.
}{
\spadpaste{map(inv\$Fraction(Integer),h) \free{h}}
}
\xtc{
We could have been a bit less verbose and used abbreviations.
}{
\spadpaste{map(inv\$FRAC(INT),h) \free{h}\bound{h1}}
}
%
\xtc{
As it turns out, Axiom is smart enough to know what we mean
anyway.
We can just say this.
}{
\spadpaste{map(inv,h) \free{h}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugTypesPkgCallPagePatch1}
\begin{paste}{ugTypesPkgCallPageFull1}{ugTypesPkgCallPageEmpty1}
\pastebutton{ugTypesPkgCallPageFull1}{\hidepaste}
```

```
\tab{5}\spadcommand{2/3}
\indentrel{3}\begin{verbatim}
        2
   (1)
        3
                                 Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesPkgCallPageEmpty1}
\begin{paste}{ugTypesPkgCallPageEmpty1}{ugTypesPkgCallPagePatch1}
\pastebutton{ugTypesPkgCallPageEmpty1}{\showpaste}
\tab{5}\spadcommand{2/3}
\end{paste}\end{patch}

\begin{patch}{ugTypesPkgCallPagePatch2}
\begin{paste}{ugTypesPkgCallPageFull2}{ugTypesPkgCallPageEmpty2}
\pastebutton{ugTypesPkgCallPageFull2}{\hidepaste}
\tab{5}\spadcommand{(2/3)$Float}
\indentrel{3}\begin{verbatim}
   (2)  0.6666666666 6666666667
                                            Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesPkgCallPageEmpty2}
\begin{paste}{ugTypesPkgCallPageEmpty2}{ugTypesPkgCallPagePatch2}
\pastebutton{ugTypesPkgCallPageEmpty2}{\showpaste}
\tab{5}\spadcommand{(2/3)$Float}
\end{paste}\end{patch}

\begin{patch}{ugTypesPkgCallPagePatch3}
\begin{paste}{ugTypesPkgCallPageFull3}{ugTypesPkgCallPageEmpty3}
\pastebutton{ugTypesPkgCallPageFull3}{\hidepaste}
\tab{5}\spadcommand{(2/3)$Fraction(Complex Integer)}
\indentrel{3}\begin{verbatim}
        2
   (3)
        3
                         Type: Fraction Complex Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesPkgCallPageEmpty3}
\begin{paste}{ugTypesPkgCallPageEmpty3}{ugTypesPkgCallPagePatch3}
\pastebutton{ugTypesPkgCallPageEmpty3}{\showpaste}
\tab{5}\spadcommand{(2/3)$Fraction(Complex Integer)}
\end{paste}\end{patch}

\begin{patch}{ugTypesPkgCallPagePatch4}
```

```
\begin{paste}{ugTypesPkgCallPageFull4}{ugTypesPkgCallPageEmpty4}
\pastebutton{ugTypesPkgCallPageFull4}{\hidepaste}
\tab{5}\spadcommand{(2/3)@Float}
\indentrel{3}\begin{verbatim}
   (4)  0.6666666666 6666666667
                                             Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesPkgCallPageEmpty4}
\begin{paste}{ugTypesPkgCallPageEmpty4}{ugTypesPkgCallPagePatch4}
\pastebutton{ugTypesPkgCallPageEmpty4}{\showpaste}
\tab{5}\spadcommand{(2/3)@Float}
\end{paste}\end{patch}

\begin{patch}{ugTypesPkgCallPagePatch5}
\begin{paste}{ugTypesPkgCallPageFull5}{ugTypesPkgCallPageEmpty5}
\pastebutton{ugTypesPkgCallPageFull5}{\hidepaste}
\tab{5}\spadcommand{(2 + 3)@String}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesPkgCallPageEmpty5}
\begin{paste}{ugTypesPkgCallPageEmpty5}{ugTypesPkgCallPagePatch5}
\pastebutton{ugTypesPkgCallPageEmpty5}{\showpaste}
\tab{5}\spadcommand{(2 + 3)@String}
\end{paste}\end{patch}

\begin{patch}{ugTypesPkgCallPagePatch6}
\begin{paste}{ugTypesPkgCallPageFull6}{ugTypesPkgCallPageEmpty6}
\pastebutton{ugTypesPkgCallPageFull6}{\hidepaste}
\tab{5}\spadcommand{(2 + 3)$String}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesPkgCallPageEmpty6}
\begin{paste}{ugTypesPkgCallPageEmpty6}{ugTypesPkgCallPagePatch6}
\pastebutton{ugTypesPkgCallPageEmpty6}{\showpaste}
\tab{5}\spadcommand{(2 + 3)$String}
\end{paste}\end{patch}

\begin{patch}{ugTypesPkgCallPagePatch7}
\begin{paste}{ugTypesPkgCallPageFull7}{ugTypesPkgCallPageEmpty7}
\pastebutton{ugTypesPkgCallPageFull7}{\hidepaste}
\tab{5}\spadcommand{((x + y * \%i)**2)@(Complex Polynomial Integer)}
\indentrel{3}\begin{verbatim}
           2    2
   (5)  - y  + x  + 2x y %i
```

```
                       Type: Complex Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesPkgCallPageEmpty7}
\begin{paste}{ugTypesPkgCallPageEmpty7}{ugTypesPkgCallPagePatch7}
\pastebutton{ugTypesPkgCallPageEmpty7}{\showpaste}
\tab{5}\spadcommand{((x + y * \%i)**2)@(Complex Polynomial Integer)}
\end{paste}\end{patch}

\begin{patch}{ugTypesPkgCallPagePatch8}
\begin{paste}{ugTypesPkgCallPageFull8}{ugTypesPkgCallPageEmpty8}
\pastebutton{ugTypesPkgCallPageFull8}{\hidepaste}
\tab{5}\spadcommand{((x + y * \%i)**2)@(Polynomial Complex Integer)}
\indentrel{3}\begin{verbatim}
           2              2
   (6)  - y  + 2%i x y + x
                       Type: Polynomial Complex Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesPkgCallPageEmpty8}
\begin{paste}{ugTypesPkgCallPageEmpty8}{ugTypesPkgCallPagePatch8}
\pastebutton{ugTypesPkgCallPageEmpty8}{\showpaste}
\tab{5}\spadcommand{((x + y * \%i)**2)@(Polynomial Complex Integer)}
\end{paste}\end{patch}

\begin{patch}{ugTypesPkgCallPagePatch9}
\begin{paste}{ugTypesPkgCallPageFull9}{ugTypesPkgCallPageEmpty9}
\pastebutton{ugTypesPkgCallPageFull9}{\hidepaste}
\tab{5}\spadcommand{(x + y * \%i)**2\bound{prevC }}
\indentrel{3}\begin{verbatim}
           2              2
   (7)  - y  + 2%i x y + x
                       Type: Polynomial Complex Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesPkgCallPageEmpty9}
\begin{paste}{ugTypesPkgCallPageEmpty9}{ugTypesPkgCallPagePatch9}
\pastebutton{ugTypesPkgCallPageEmpty9}{\showpaste}
\tab{5}\spadcommand{(x + y * \%i)**2\bound{prevC }}
\end{paste}\end{patch}

\begin{patch}{ugTypesPkgCallPagePatch10}
\begin{paste}{ugTypesPkgCallPageFull10}{ugTypesPkgCallPageEmpty10}
\pastebutton{ugTypesPkgCallPageFull10}{\hidepaste}
\tab{5}\spadcommand{\% :: Complex ?\free{prevC }}
\indentrel{3}\begin{verbatim}
           2    2
```

```
   (8)  - y  + x  + 2x y %i
                       Type: Complex Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesPkgCallPageEmpty10}
\begin{paste}{ugTypesPkgCallPageEmpty10}{ugTypesPkgCallPagePatch10}
\pastebutton{ugTypesPkgCallPageEmpty10}{\showpaste}
\tab{5}\spadcommand{\% :: Complex ?\free{prevC }}
\end{paste}\end{patch}

\begin{patch}{ugTypesPkgCallPagePatch11}
\begin{paste}{ugTypesPkgCallPageFull11}{ugTypesPkgCallPageEmpty11}
\pastebutton{ugTypesPkgCallPageFull11}{\hidepaste}
\tab{5}\spadcommand{h := matrix [[8,6],[-4,9]]\bound{h }}
\indentrel{3}\begin{verbatim}
         8   6
   (9)
        - 4  9
                                   Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesPkgCallPageEmpty11}
\begin{paste}{ugTypesPkgCallPageEmpty11}{ugTypesPkgCallPagePatch11}
\pastebutton{ugTypesPkgCallPageEmpty11}{\showpaste}
\tab{5}\spadcommand{h := matrix [[8,6],[-4,9]]\bound{h }}
\end{paste}\end{patch}

\begin{patch}{ugTypesPkgCallPagePatch12}
\begin{paste}{ugTypesPkgCallPageFull12}{ugTypesPkgCallPageEmpty12}
\pastebutton{ugTypesPkgCallPageFull12}{\hidepaste}
\tab{5}\spadcommand{map(inv$Fraction(Integer),h)\free{h }}
\indentrel{3}\begin{verbatim}
          1   1

          8   6
   (10)
           1  1
         -
           4  9
                          Type: Matrix Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesPkgCallPageEmpty12}
\begin{paste}{ugTypesPkgCallPageEmpty12}{ugTypesPkgCallPagePatch12}
\pastebutton{ugTypesPkgCallPageEmpty12}{\showpaste}
\tab{5}\spadcommand{map(inv$Fraction(Integer),h)\free{h }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugTypesPkgCallPagePatch13}
\begin{paste}{ugTypesPkgCallPageFull13}{ugTypesPkgCallPageEmpty13}
\pastebutton{ugTypesPkgCallPageFull13}{\hidepaste}
\tab{5}\spadcommand{map(inv$FRAC(INT),h)\free{h }\bound{h1 }}
\indentrel{3}\begin{verbatim}
         1   1

         8   6
   (11)
          1  1
        -
          4  9
                          Type: Matrix Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesPkgCallPageEmpty13}
\begin{paste}{ugTypesPkgCallPageEmpty13}{ugTypesPkgCallPagePatch13}
\pastebutton{ugTypesPkgCallPageEmpty13}{\showpaste}
\tab{5}\spadcommand{map(inv$FRAC(INT),h)\free{h }\bound{h1 }}
\end{paste}\end{patch}

\begin{patch}{ugTypesPkgCallPagePatch14}
\begin{paste}{ugTypesPkgCallPageFull14}{ugTypesPkgCallPageEmpty14}
\pastebutton{ugTypesPkgCallPageFull14}{\hidepaste}
\tab{5}\spadcommand{map(inv,h)\free{h }}
\indentrel{3}\begin{verbatim}
         1   1

         8   6
   (12)
          1  1
        -
          4  9
                          Type: Matrix Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesPkgCallPageEmpty14}
\begin{paste}{ugTypesPkgCallPageEmpty14}{ugTypesPkgCallPagePatch14}
\pastebutton{ugTypesPkgCallPageEmpty14}{\showpaste}
\tab{5}\spadcommand{map(inv,h)\free{h }}
\end{paste}\end{patch}
```

Resolving Types

— ug02.ht —

```
\begin{page}{ugTypesResolvePage}{2.10. Resolving Types}
\beginscroll

In this section we briefly describe an internal process by which
Axiom determines a type to which two objects of possibly
different types can be converted.
We do this to give you further insight into how Axiom takes
your input, analyzes it, and produces a result.

What happens when you enter \axiom{x + 1} to Axiom?
Let's look at what you get from the two terms of this expression.

\xtc{
This is a symbolic object whose type indicates the name.
}{
\spadpaste{x}
}
\xtc{
This is a positive integer.
}{
\spadpaste{1}
}

There are no operations in \axiomType{PositiveInteger} that add
positive integers to objects of type \axiomType{Variable(x)} nor
are there any in \axiomType{Variable(x)}.
Before it can add the two parts, Axiom must come up with
a common type to which both \axiom{x} and \axiom{1} can be
converted.
We say that Axiom must {\it resolve} the two types
into a common type.
In this example, the common type is \axiomType{Polynomial(Integer)}.

\xtc{
Once this is determined, both parts are converted into polynomials,
and the addition operation from \axiomType{Polynomial(Integer)} is used
to get the answer.
}{
\spadpaste{x + 1}
}
\xtc{
Axiom can always resolve two types: if nothing resembling
the original types can be found, then \axiomType{Any} is be used.
This is fine and useful in some cases.
}{
```

```
\spadpaste{["string",3.14159]}
}
\xtc{
In other cases objects of type \axiomType{Any} can't be used
by the operations you specified.
}{
\spadpaste{"string" + 3.14159}
}
Although this example was contrived, your expressions may need
to be qualified slightly to help Axiom resolve the
types involved.
You may need to declare a few variables, do some package calling,
provide some target type information or do some explicit
conversions.

We suggest that you just enter the expression you want evaluated and
see what Axiom does.
We think you will be impressed with its ability to ``do what I
mean.''
If Axiom is still being obtuse, give it some hints.
As you work with Axiom, you will learn where it needs a
little help to analyze quickly and perform your computations.

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugTypesResolvePagePatch1}
\begin{paste}{ugTypesResolvePageFull1}{ugTypesResolvePageEmpty1}
\pastebutton{ugTypesResolvePageFull1}{\hidepaste}
\tab{5}\spadcommand{x}
\indentrel{3}\begin{verbatim}
   (1)  x
                                       Type: Variable x
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesResolvePageEmpty1}
\begin{paste}{ugTypesResolvePageEmpty1}{ugTypesResolvePagePatch1}
\pastebutton{ugTypesResolvePageEmpty1}{\showpaste}
\tab{5}\spadcommand{x}
\end{paste}\end{patch}

\begin{patch}{ugTypesResolvePagePatch2}
\begin{paste}{ugTypesResolvePageFull2}{ugTypesResolvePageEmpty2}
\pastebutton{ugTypesResolvePageFull2}{\hidepaste}
\tab{5}\spadcommand{1}
\indentrel{3}\begin{verbatim}
   (2)  1
                                  Type: PositiveInteger
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesResolvePageEmpty2}
\begin{paste}{ugTypesResolvePageEmpty2}{ugTypesResolvePagePatch2}
\pastebutton{ugTypesResolvePageEmpty2}{\showpaste}
\tab{5}\spadcommand{1}
\end{paste}\end{patch}

\begin{patch}{ugTypesResolvePagePatch3}
\begin{paste}{ugTypesResolvePageFull3}{ugTypesResolvePageEmpty3}
\pastebutton{ugTypesResolvePageFull3}{\hidepaste}
\tab{5}\spadcommand{x + 1}
\indentrel{3}\begin{verbatim}
   (3)  x + 1
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesResolvePageEmpty3}
\begin{paste}{ugTypesResolvePageEmpty3}{ugTypesResolvePagePatch3}
\pastebutton{ugTypesResolvePageEmpty3}{\showpaste}
\tab{5}\spadcommand{x + 1}
\end{paste}\end{patch}

\begin{patch}{ugTypesResolvePagePatch4}
\begin{paste}{ugTypesResolvePageFull4}{ugTypesResolvePageEmpty4}
\pastebutton{ugTypesResolvePageFull4}{\hidepaste}
\tab{5}\spadcommand{["string",3.14159]}
\indentrel{3}\begin{verbatim}
   (4)  ["string",3.14159]
                                         Type: List Any
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesResolvePageEmpty4}
\begin{paste}{ugTypesResolvePageEmpty4}{ugTypesResolvePagePatch4}
\pastebutton{ugTypesResolvePageEmpty4}{\showpaste}
\tab{5}\spadcommand{["string",3.14159]}
\end{paste}\end{patch}

\begin{patch}{ugTypesResolvePagePatch5}
\begin{paste}{ugTypesResolvePageFull5}{ugTypesResolvePageEmpty5}
\pastebutton{ugTypesResolvePageFull5}{\hidepaste}
\tab{5}\spadcommand{"string" + 3.14159}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesResolvePageEmpty5}
```

```
\begin{paste}{ugTypesResolvePageEmpty5}{ugTypesResolvePagePatch5}
\pastebutton{ugTypesResolvePageEmpty5}{\showpaste}
\tab{5}\spadcommand{"string" + 3.14159}
\end{paste}\end{patch}
```

———-

Exposing Domains and Packages

⇒ "notitle" (ugTypesPkgCallPage) 7 on page 1686
⇒ "notitle" (ugUserTrianglePage) 10 on page 1939
⇒ "notitle" (ugSysCmdframePage) 19 on page 2556

— **ug02.ht** —

```
\begin{page}{ugTypesExposePage}{2.11. Exposing Domains and Packages}
\beginscroll

In this section we discuss how Axiom makes some operations
available to you while hiding others that are meant to be used by
developers or only in rare cases.
If you are a new user of Axiom, it is likely that everything
you need is available by default and you may want
to skip over this section on first reading.

Every
domain and package in the Axiom library
is
either
\spadglossSee{exposed}{expose} (meaning that you can use its
operations without doing
anything special) or it is {\it hidden} (meaning you have to either
package call
(see \downlink{``Package Calling and Target Types''}{ugTypesPkgCallPage}
in Section 2.9\ignore{ugTypesPkgCall})
the operations it contains or explicitly expose it to use the
operations).
The initial exposure status for a constructor is set in the
file {\bf exposed.lsp} (see the {\it Installer's Note}
for Axiom
if you need to know the location of this file).
Constructors are collected together in
{\it exposure groups}.
Categories are all in the exposure group ``categories'' and the
bulk of the basic set of packages and domains that are exposed
are in the exposure group ``basic.''
Here is an abbreviated sample of the file (without the Lisp parentheses):
```

```
\begin{verbatim}
basic
        AlgebraicNumber                         AN
        AlgebraGivenByStructuralConstants       ALGSC
        Any                                     ANY
        AnyFunctions1                           ANY1
        BinaryExpansion                         BINARY
        Boolean                                 BOOLEAN
        CardinalNumber                          CARD
        CartesianTensor                         CARTEN
        Character                               CHAR
        CharacterClass                          CCLASS
        CliffordAlgebra                         CLIF
        Color                                   COLOR
        Complex                                 COMPLEX
        ContinuedFraction                       CONTFRAC
        DecimalExpansion                        DECIMAL
        ...
\end{verbatim}
\begin{verbatim}
categories
        AbelianGroup                            ABELGRP
        AbelianMonoid                           ABELMON
        AbelianMonoidRing                       AMR
        AbelianSemiGroup                        ABELSG
        Aggregate                               AGG
        Algebra                                 ALGEBRA
        AlgebraicallyClosedField                ACF
        AlgebraicallyClosedFunctionSpace        ACFS
        ArcHyperbolicFunctionCategory           AHYP
        ...
\end{verbatim}
For each constructor in a group, the full name and the abbreviation
is given.
There are other groups in {\bf exposed.lsp} but initially only the
constructors in exposure groups ``basic'' ``categories'' ``naglink''
and ``anna'' are exposed.

As an interactive user of Axiom, you do not need to modify this file.
Instead, use \spadcmd{)set expose} to expose, hide or query the
exposure status of an individual constructor or exposure group.  The
reason for having exposure groups is to be able to expose or hide
multiple constructors with a single command.  For example, you might
group together into exposure group ``quantum'' a number of domains and
packages useful for quantum mechanical computations.  These probably
should not be available to every user, but you want an easy way to
make the whole collection visible to Axiom when it is looking for
operations to apply.

If you wanted to hide all the basic constructors available by default,
```

you would issue \spadcmd{)set expose drop group basic}. If, however, you discover that you have hidden all the basic constructors, you should issue \spadcmd{)set expose add group basic} to restore your default environment.

It is more likely that you would want to expose or hide individual constructors. In \downlink{``A Famous Triangle''}{ugUserTrianglePage} in Section 6.19\ignore{ugUserTriangle} we use several operations from \axiomType{OutputForm}, a domain usually hidden. To avoid package calling every operation from \axiomType{OutputForm}, we expose the domain and let Axiom conclude that those operations should be used. Use \spadcmd{)set expose add constructor} and \spadcmd{)set expose drop constructor} to expose and hide a constructor, respectively. You should use the constructor name, not the abbreviation. The \spadcmd{)set expose} command guides you through these options.

If you expose a previously hidden constructor, Axiom exhibits new behavior (that was your intention) though you might not expect the results that you get. \axiomType{OutputForm} is, in fact, one of the worst offenders in this regard. This domain is meant to be used by other domains for creating a structure that Axiom knows how to display. It has functions like \axiomOpFrom{+}{OutputForm} that form output representations rather than do mathematical calculations. Because of the order in which Axiom looks at constructors when it is deciding what operation to apply, \axiomType{OutputForm} might be used instead of what you expect. \xtc{ This is a polynomial. }{ \spadpaste{x + x} } \xtc{ Expose \axiomType{OutputForm}. }{ \spadpaste{)set expose add constructor OutputForm \bound{setexposeadd}} } \xtc{ This is what we get when \axiomType{OutputForm} is automatically available. }{ \spadpaste{x + x \free{setexposeadd}} } \xtc{ Hide \axiomType{OutputForm} so we don't run into problems with any later examples! }{ \spadpaste{)set expose drop constructor OutputForm \bound{setexposedrop}} }

Finally, exposure is done on a frame-by-frame basis. A \spadgloss{frame} (see \downlink{``)frame''}{ugSysCmdframePage} in Section B.11 \ignore{ugSysCmdframe}) is one of possibly several logical Axiom workspaces within a physical one, each having its own environment (for example, variables and function definitions). If you have several Axiom workspace windows on your screen, they are all different frames, automatically created for you by Hyperdoc. Frames can be manually created, made active and destroyed by the \spadcmd{)frame} system command. They do not share exposure information, so you need to use \spadcmd{)set expose} in each one to add or drop constructors from view.

\endscroll

```
\autobuttons
\end{page}

\begin{patch}{ugTypesExposePagePatch1}
\begin{paste}{ugTypesExposePageFull1}{ugTypesExposePageEmpty1}
\pastebutton{ugTypesExposePageFull1}{\hidepaste}
\tab{5}\spadcommand{x + x}
\indentrel{3}\begin{verbatim}
   (1)  2x
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesExposePageEmpty1}
\begin{paste}{ugTypesExposePageEmpty1}{ugTypesExposePagePatch1}
\pastebutton{ugTypesExposePageEmpty1}{\showpaste}
\tab{5}\spadcommand{x + x}
\end{paste}\end{patch}

\begin{patch}{ugTypesExposePagePatch2}
\begin{paste}{ugTypesExposePageFull2}{ugTypesExposePageEmpty2}
\pastebutton{ugTypesExposePageFull2}{\hidepaste}
\tab{5}\spadcommand{)set expose add constructor OutputForm\bound{setexposeadd }}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesExposePageEmpty2}
\begin{paste}{ugTypesExposePageEmpty2}{ugTypesExposePagePatch2}
\pastebutton{ugTypesExposePageEmpty2}{\showpaste}
\tab{5}\spadcommand{)set expose add constructor OutputForm\bound{setexposeadd }}
\end{paste}\end{patch}

\begin{patch}{ugTypesExposePagePatch3}
\begin{paste}{ugTypesExposePageFull3}{ugTypesExposePageEmpty3}
\pastebutton{ugTypesExposePageFull3}{\hidepaste}
\tab{5}\spadcommand{x + x\free{setexposeadd }}
\indentrel{3}\begin{verbatim}
   (2)  x + x
                                       Type: OutputForm
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesExposePageEmpty3}
\begin{paste}{ugTypesExposePageEmpty3}{ugTypesExposePagePatch3}
\pastebutton{ugTypesExposePageEmpty3}{\showpaste}
\tab{5}\spadcommand{x + x\free{setexposeadd }}
\end{paste}\end{patch}

\begin{patch}{ugTypesExposePagePatch4}
```

```
\begin{paste}{ugTypesExposePageFull4}{ugTypesExposePageEmpty4}
\pastebutton{ugTypesExposePageFull4}{\hidepaste}
\tab{5}\spadcommand{)set expose drop constructor OutputForm\bound{setexposedrop }}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugTypesExposePageEmpty4}
\begin{paste}{ugTypesExposePageEmpty4}{ugTypesExposePagePatch4}
\pastebutton{ugTypesExposePageEmpty4}{\showpaste}
\tab{5}\spadcommand{)set expose drop constructor OutputForm\bound{setexposedrop }}
\end{paste}\end{patch}
```

———-

Commands for Snooping

⇒ "notitle" (ugBrowsePage) 17 on page 2471
⇒ "notitle" (ComplexXmpPage) 3.16 on page 254
⇒ "notitle" (ugUserDeclarePage) 10 on page 1838

— **ug02.ht** —

```
\begin{page}{ugAvailSnoopPage}{2.12. Commands for Snooping}
\beginscroll

To conclude this chapter, we introduce you to some system commands
that you can use for getting more information about domains,
packages, categories, and operations.
The most powerful Axiom facility for getting information about
constructors and operations is the \Browse{} component of Hyperdoc.
This is discussed in \downlink{``Browse''}{ugBrowsePage}
in Chapter 14\ignore{ugBrowse}.

Use the \spadsys{)what} system command to see lists of system objects
whose name contain a particular substring (uppercase or lowercase is
not significant).

\xtc{
Issue this to see a list of all operations with
``{\tt complex}'' in their names.
}{
\spadpaste{)what operation complex}
}
\xtc{
If you want to see all domains with ``{\tt matrix}'' in their names, issue
this.
```

```
}{
\spadpaste{)what domain matrix}
}
\xtc{
Similarly, if you wish to see all packages whose names contain
‘‘{\tt gauss}’’, enter this.
}{
\spadpaste{)what package gauss}
}
\xtc{
This command shows all
the operations that \axiomType{Any} provides.
Wherever \axiomSyntax{\$} appears, it means ‘‘\axiomType{Any}’’.
}{
\spadpaste{)show Any}
}
\xtc{
This displays all operations with the name \axiomFun{complex}.
}{
\spadpaste{)display operation complex}
}
Let’s analyze this output.
\xtc{
First we find out what some of the abbreviations mean.
}{
\spadpaste{)abbreviation query COMPCAT}
}
\xtc{
}{
\spadpaste{)abbreviation query COMRING}
}

So if \axiom{D1} is a commutative ring (such as the integers or
floats) and \axiom{D} belongs to \axiomType{ComplexCategory
D1}, then there is an operation called \axiomFun{complex} that
takes two elements of \axiom{D1} and creates an element of
\axiom{D}.
The primary example of a constructor implementing domains
belonging to \axiomType{ComplexCategory} is \axiomType{Complex}.
See \downlink{‘Complex’}{ComplexXmpPage}\ignore{Complex}
for more information on that and see
\downlink{‘‘Declaring the Type of Functions’’}{ugUserDeclarePage} in
Section 6.4\ignore{ugUserDeclare}
for more information on function types.
\endscroll
\autobuttons
\end{page}

\begin{patch}{ugAvailSnoopPagePatch1}
\begin{paste}{ugAvailSnoopPageFull1}{ugAvailSnoopPageEmpty1}
```

```
\pastebutton{ugAvailSnoopPageFull1}{\hidepaste}
\tab{5}\spadcommand{)what operation complex}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugAvailSnoopPageEmpty1}
\begin{paste}{ugAvailSnoopPageEmpty1}{ugAvailSnoopPagePatch1}
\pastebutton{ugAvailSnoopPageEmpty1}{\showpaste}
\tab{5}\spadcommand{)what operation complex}
\end{paste}\end{patch}

\begin{patch}{ugAvailSnoopPagePatch2}
\begin{paste}{ugAvailSnoopPageFull2}{ugAvailSnoopPageEmpty2}
\pastebutton{ugAvailSnoopPageFull2}{\hidepaste}
\tab{5}\spadcommand{)what domain matrix}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugAvailSnoopPageEmpty2}
\begin{paste}{ugAvailSnoopPageEmpty2}{ugAvailSnoopPagePatch2}
\pastebutton{ugAvailSnoopPageEmpty2}{\showpaste}
\tab{5}\spadcommand{)what domain matrix}
\end{paste}\end{patch}

\begin{patch}{ugAvailSnoopPagePatch3}
\begin{paste}{ugAvailSnoopPageFull3}{ugAvailSnoopPageEmpty3}
\pastebutton{ugAvailSnoopPageFull3}{\hidepaste}
\tab{5}\spadcommand{)what package gauss}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugAvailSnoopPageEmpty3}
\begin{paste}{ugAvailSnoopPageEmpty3}{ugAvailSnoopPagePatch3}
\pastebutton{ugAvailSnoopPageEmpty3}{\showpaste}
\tab{5}\spadcommand{)what package gauss}
\end{paste}\end{patch}

\begin{patch}{ugAvailSnoopPagePatch4}
\begin{paste}{ugAvailSnoopPageFull4}{ugAvailSnoopPageEmpty4}
\pastebutton{ugAvailSnoopPageFull4}{\hidepaste}
\tab{5}\spadcommand{)show Any}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugAvailSnoopPageEmpty4}
\begin{paste}{ugAvailSnoopPageEmpty4}{ugAvailSnoopPagePatch4}
```

```
\pastebutton{ugAvailSnoopPageEmpty4}{\showpaste}
\tab{5}\spadcommand{)show Any}
\end{paste}\end{patch}

\begin{patch}{ugAvailSnoopPagePatch5}
\begin{paste}{ugAvailSnoopPageFull5}{ugAvailSnoopPageEmpty5}
\pastebutton{ugAvailSnoopPageFull5}{\hidepaste}
\tab{5}\spadcommand{)display operation complex}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugAvailSnoopPageEmpty5}
\begin{paste}{ugAvailSnoopPageEmpty5}{ugAvailSnoopPagePatch5}
\pastebutton{ugAvailSnoopPageEmpty5}{\showpaste}
\tab{5}\spadcommand{)display operation complex}
\end{paste}\end{patch}

\begin{patch}{ugAvailSnoopPagePatch6}
\begin{paste}{ugAvailSnoopPageFull6}{ugAvailSnoopPageEmpty6}
\pastebutton{ugAvailSnoopPageFull6}{\hidepaste}
\tab{5}\spadcommand{)abbreviation query COMPCAT}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugAvailSnoopPageEmpty6}
\begin{paste}{ugAvailSnoopPageEmpty6}{ugAvailSnoopPagePatch6}
\pastebutton{ugAvailSnoopPageEmpty6}{\showpaste}
\tab{5}\spadcommand{)abbreviation query COMPCAT}
\end{paste}\end{patch}

\begin{patch}{ugAvailSnoopPagePatch7}
\begin{paste}{ugAvailSnoopPageFull7}{ugAvailSnoopPageEmpty7}
\pastebutton{ugAvailSnoopPageFull7}{\hidepaste}
\tab{5}\spadcommand{)abbreviation query COMRING}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugAvailSnoopPageEmpty7}
\begin{paste}{ugAvailSnoopPageEmpty7}{ugAvailSnoopPagePatch7}
\pastebutton{ugAvailSnoopPageEmpty7}{\showpaste}
\tab{5}\spadcommand{)abbreviation query COMRING}
\end{paste}\end{patch}
```

———-

Chapter 8

Users Guide Chapter 3 (ug03.ht)

Using Hyperdoc

⇒ "notitle" (YouTriedIt) 3.78 on page 1018
⇒ "notitle" (ugHyperHeadingsPage) 8 on page 1708
⇒ "notitle" (ugHyperKeysPage) 8 on page 1709
⇒ "notitle" (ugHyperScrollPage) 8 on page 1710
⇒ "notitle" (ugHyperInputPage) 8 on page 1711
⇒ "notitle" (ugHyperButtonsPage) 8 on page 1713
⇒ "notitle" (ugHyperSearchPage) 8 on page 1714
⇒ "notitle" (ugHyperExamplePage) 8 on page 1716
⇒ "notitle" (ugHyperResourcesPage) 8 on page 1717

— **ug03.ht** —

```
\begin{page}{ugHyperPage}{3. Using Hyperdoc}
\beginscroll


Hyperdoc is the gateway to Axiom.
It's both an on-line tutorial and an on-line reference manual.
It also enables you to use Axiom simply by using the mouse and
filling in templates.
Hyperdoc is available to you if you are running Axiom under the
X Window System.

Pages usually have active areas, marked in
\texht{{\bf this font} (bold face).}{\downlink{this font.}{YouTriedIt}}
As you move the mouse pointer to an active area, the pointer changes from
a filled dot to an open circle.
```

```
The active areas are usually linked to other pages.
When you click on an active area, you move to the linked page.
\texht{}{Try clicking \downlink{here}{YouTriedIt} now.}

We suggest that you learn more about other features of
Hyperdoc by clicking on an active area in the menu below.

\beginmenu
    \menudownlink{{3.1. Headings}}{ugHyperHeadingsPage}
    \menudownlink{{3.2. Key Definitions}}{ugHyperKeysPage}
    \menudownlink{{3.3. Scroll Bars}}{ugHyperScrollPage}
    \menudownlink{{3.4. Input Areas}}{ugHyperInputPage}
    \menudownlink{{3.5. Radio Buttons and Toggles}}{ugHyperButtonsPage}
    \menudownlink{{3.6. Search Strings}}{ugHyperSearchPage}
    \menudownlink{{3.7. Example Pages}}{ugHyperExamplePage}
    \menudownlink{{3.8. X Window Resources for Hyperdoc}}
                  {ugHyperResourcesPage}
\endmenu
\endscroll
\autobuttons
\end{page}
```

———-

Headings

— ug03.ht —

```
\begin{page}{ugHyperHeadingsPage}{3.1. Headings}
\beginscroll
%
Most pages have a standard set of buttons at the top of the page.
This is what they mean:

\indent{0}
\beginitems
\item[\StdHelpButton{}] Click on this to get help.
The button only appears if there is specific help for the page you are
viewing.
You can get {\it general} help for Hyperdoc by clicking the help
button on the home page.

\item[\UpButton{}] Click here to go back one page.
By clicking on this button repeatedly, you can go back several pages and
then take off in a new direction.

\item[\ReturnButton{}] Go back to the home page, that is,
```

```
the page on which you started.
Use Hyperdoc to explore, to make forays into new topics.
Don't worry about how to get back.
Hyperdoc remembers where you came from.
Just click on this button to return.

\item[\StdExitButton{}] From the root window (the one that is
displayed when you start the system) this button leaves the Hyperdoc
program, and it must be restarted if you want to use it again.  From
any other Hyperdoc window, it just makes that one window go away.  You
{\it must} use this button to get rid of a window.  If you use the
window manager ``Close'' button, then all of Hyperdoc goes away.
\enditems
\indent{0}
%
The buttons are not displayed if they are not applicable to the page
you are viewing.
For example, there is no \ReturnButton{} button on the top-level menu.

\endscroll
\autobuttons
\end{page}
```

———-

Key Definitions

⇒ "notitle" (ugHyperScrollPage) 8 on page 1710
⇒ "notitle" (ugHyperInputPage) 8 on page 1711

— **ug03.ht** —

```
\begin{page}{ugHyperKeysPage}{3.2. Key Definitions}
\beginscroll

The following keyboard definitions are in effect throughout
Hyperdoc.
See \downlink{``Scroll Bars''}{ugHyperScrollPage}
in Section 3.3\ignore{ugHyperScroll} and
\downlink{``Input Areas''}{ugHyperInputPage}
in Section 3.4\ignore{ugHyperInput}
for some contextual key definitions.
%
\indent{0}
\beginitems
\item[F1] Display the main help page.
\item[F3] Same as \StdExitButton{}, makes the window go away if you are
```

```
not at the top-level window or quits the Hyperdoc facility if you are
at the top-level.
\item[F5] Rereads the Hyperdoc database, if necessary (for system
developers).
\item[F9] Displays this information about key definitions.
\item[F12] Same as {\bf F3}.
\item[Up Arrow] Scroll up one line.
\item[Down Arrow] Scroll down one line.
\item[Page Up] Scroll up one page.
\item[Page Down] Scroll down one page.
\enditems
\indent{0}

\endscroll
\autobuttons
\end{page}
```

———-

Scroll Bars

⇒ “notitle” (ugHyperInputPage) 8 on page 1711

— **ug03.ht** —

```
\begin{page}{ugHyperScrollPage}{3.3. Scroll Bars}
\beginscroll

Whenever there is too much text to fit on a page, a {\it scroll
bar} automatically appears along the right side.

With a scroll bar, your page becomes an aperture, that is, a
window into a larger amount of text than can be displayed at one
time.
The scroll bar lets you move up and down in the text to see
different parts.
It also shows where the aperture is relative to the whole text.
The aperture is indicated by a strip on the scroll bar.

Move the cursor with the mouse to the ``down-arrow'' at the bottom
of the scroll bar and click.
See that the aperture moves down one line.
Do it several times.
Each time you click, the aperture moves down one line.
Move the mouse to the ``up-arrow'' at the top of the scroll bar and
click.
The aperture moves up one line each time you click.
```

```
Next move the mouse to any position along the middle of the scroll bar
and click.  Hyperdoc attempts to move the top of the aperture to this
point in the text.

You cannot make the aperture go off the bottom edge.  When the
aperture is about half the size of text, the lowest you can move the
aperture is halfway down.

To move up or down one screen at a time, use the
\texht{\fbox{\bf PageUp}}{{\bf PageUp}} and
\texht{\fbox{\bf PageDown}}{{\bf PageDown}} keys on your keyboard.
They move the visible part of the region up and down one page each
time you press them.

If the Hyperdoc page does not contain an input area
(see \downlink{``Input Areas''}{ugHyperInputPage}
in Section 3.4\ignore{ugHyperInput}),
you can also use the
\texht{\fbox{\bf Home}}{{\bf Home}} and
\texht{\fbox{$\uparrow$}}{up} and
\texht{\fbox{$\downarrow$}}{down} arrow keys to navigate.
When you press the \texht{\fbox{\bf Home}}{{\bf Home}} key,
the screen is positioned at the very top of the page.
Use the \texht{\fbox{$\uparrow$}}{up} and
\texht{\fbox{$\downarrow$}}{down} arrow keys to move the screen up
and down one line at a time, respectively.

\endscroll
\autobuttons
\end{page}
```

———-

Input Areas

— ug03.ht —

```
\begin{page}{ugHyperInputPage}{3.4. Input Areas}
\beginscroll
%
Input areas are boxes where you can put data.
Here is one:
\centerline{\inputstring{one}{40}{some text}}
\newline
As you can see, the input area has some initial text {\it some text}
followed by an underscore cursor (the character {\it _}).
```

```
To enter characters, first
move your mouse cursor to somewhere within the Hyperdoc page.
Characters that you type are inserted in front of the underscore.
This means that when you type characters at your keyboard, they
go into this first input area.

The input area grows to accommodate as many characters as you type.
Use the \texht{\fbox{\bf Backspace}}{{\bf Backspace}}
key to erase characters to the left.
To modify what you type, use the right-arrow \texht{\fbox{$\rightarrow$}}{}
and left-arrow keys \texht{\fbox{$\leftarrow$}}{} and the
keys \texht{\fbox{\bf Insert}}{{\bf Insert}},
\texht{\fbox{\bf Delete}}{{\bf Delete}},
\texht{\fbox{\bf Home}}{{\bf Home}} and
\texht{\fbox{\bf End}}{{\bf End}}.
These keys are found immediately on the right of the standard IBM keyboard.

If you press the
\texht{\fbox{\bf Home}}{{\bf Home}}
key, the cursor moves to the beginning of the line and if you press the
\texht{\fbox{\bf End}}{{\bf End}}
key, the cursor moves to the end of the line.
Pressing
\texht{\fbox{\bf Ctrl}--\fbox{\bf End}}{{\bf Ctrl-End}}
deletes all the text from the cursor to the end of the line.

A page may have more than one input area.
Only one input area has an underscore cursor.
When you first see apage, the top-most input area contains the
cursor.
To type information into another input area,
use the \texht{\fbox{\bf Enter}}{{\bf Enter}} or
\texht{\fbox{\bf Tab}}{{\bf Tab}} key to move
from one input area to another.
To move in the reverse order, use
\texht{\fbox{\bf Shift}--\fbox{\bf Tab}}{{\bf Shift-Tab}}.

You can also move from one input area to another using your mouse.
Notice that each input area is active. Click on one of the areas.
As you can see, the underscore cursor moves to that window.

\endscroll
\autobuttons
\end{page}
```

Radio Buttons and Toggles

— ug03.ht —

```
\begin{page}{ugHyperButtonsPage}{3.5. Radio Buttons and Toggles}
\beginscroll
%
Some pages have {\it radio buttons} and {\it toggles}.
Radio buttons are a group of buttons like those on car radios: you can
select only one at a time.
\radioboxes{sample}{\htbmfile{pick}}{\htbmfile{unpick}}
Here are three radio buttons:
\centerline{
{\em\radiobox[1]{rone}{sample}\space{}\ First one}\space{3}
{\em\radiobox[0]{rtwo}{sample}\space{}\ Second one}\space{3}
{\em\radiobox[0]{rthree}{sample}\space{}\ Third one}
}
\newline
Once you have selected a button, it appears to be inverted and
contains a checkmark.
To change the selection, move the cursor with the mouse to a
different radio button and click.
\texht{}{Try it now.}

A toggle is an independent button that displays some on/off
state.
When ``on'', the button appears to be inverted and
contains a checkmark.
When ``off'', the button is raised.
%
Unlike radio buttons, you can set a group of them any way you like.
Here are three:
\centerline{
{\em\inputbox[1]{one}{\htbmfile{pick}}{\htbmfile{unpick}}\space{}
\ First one}
\space{3}
{\em\inputbox[0]{two}{\htbmfile{pick}}{\htbmfile{unpick}}\space{}
\ Second one}
\space{3}
{\em\inputbox[1]{three}{\htbmfile{pick}}{\htbmfile{unpick}}\space{}
\ Third one}
}
\newline
To change toggle the selection, move the cursor with the mouse
to the button and click.

\endscroll
\autobuttons
\end{page}
```

Search Strings

⇒ “notitle” (ugLogicalSearchesPage) 8 on page 1715

— **ug03.ht** —

```
\begin{page}{ugHyperSearchPage}{3.6. Search Strings}
\beginscroll
%
A {\it search string} is used for searching some database.
To learn about search strings, we suggest that
you bring up the Hyperdoc glossary.
To do this from the top-level page of Hyperdoc:
\indent{4}
\beginitems
\item[1. ] Click on \windowlink{Reference}{TopReferencePage},
bringing up the Axiom Reference page.
\item[2. ] Click on \windowlink{Glossary}{GlossaryPage},
bringing up the glossary.
\texht{}{(You can also just click on the word ‘‘Glossary’’ in the
last sentence.)}
\enditems
\indent{0}
Once you get the window containing the glossary, move it so that
it and this window are both visible.

The glossary has an input area at its bottom.
We review the various kinds of search strings
you can enter to search the glossary.

The simplest search string is a word, for example, {\tt operation}.
A word only matches an entry having exactly that spelling.
Enter the word {\tt operation} into the input area above then click on
{\bf Search}.
As you can see, {\tt operation} matches only one entry, namely with {\tt
operation} itself.

Normally matching is insensitive to whether the alphabetic characters
of your search string are in uppercase or lowercase.  Thus {\tt
operation} and {\tt OperAtion} both have the same effect.  %If you
prefer that matching be case-sensitive, issue the command
%\spadsys{set HHyperName mixedCase} command to the interpreter.

You will very often want to use the wildcard \spadSyntax{*} in your
search string so as to match multiple entries in the list.  The search
```

```
key \spadSyntax{*} matches every entry in the list.  You can also use
\spadSyntax{*} anywhere within a search string to match an arbitrary
substring.  Try {\tt cat*} for example: enter {\tt cat*} into the
input area and click on {\bf Search}.  This matches several entries.

You use any number of wildcards in a search string as long as they are
not adjacent.  Try search strings such as {\tt *dom*}.  As you see,
this search string matches {\tt domain}, {\tt domain constructor},
{\tt subdomain}, and so on.

\beginmenu
    \menudownlink{{3.6.1. Logical Searches}}{ugLogicalSearchesPage}
\endmenu
\endscroll
\autobuttons
\end{page}
```

———-

Logical Searches

— ug03.ht —

```
\begin{page}{ugLogicalSearchesPage}{3.6.1. Logical Searches}
\beginscroll

For more complicated searches, you can use
\spadSyntax{and}, \spadSyntax{or}, and \spadSyntax{not}
with basic search strings;
write logical expressions using these three operators just as
in the Axiom language.
For example, {\tt domain or package}  matches the two
entries {\tt domain} and {\tt package}.
Similarly, {\tt dom* and *con*} matches {\tt domain constructor}
and others.
Also {\tt not *a*} matches every entry that does not contain
the letter {\tt a} somewhere.

Use parentheses for grouping.
For example, {\tt dom* and (not *con*)}
matches {\tt domain} but not {\tt domain constructor}.

There is no limit to how complex your logical expression can be.
For example,
\centerline{{{\tt a* or b* or c* or d* or e* and (not *a*)}}}
is a valid expression.
```

```
\endscroll
\autobuttons
\end{page}
```

————-

Example Pages

— ug03.ht —

```
\begin{page}{ugHyperExamplePage}{3.7. Example Pages}
\beginscroll
%
Many pages have Axiom example commands.
%
Here are two:
\spadpaste{a:= x**2 + 1 \bound{a}}
\spadpaste{(a - 2)**2 \free{a}}
%
Each command has an active ‘‘button’’ along the left margin.
When you click on this button, the output for the command is
‘‘pasted-in.’’
Click again on the button and you see that the pasted-in output
disappears.

Maybe you would like to run an example?
To do so, just click on any part of its text!
When you do, the example line is copied into a new interactive
Axiom buffer for this Hyperdoc page.

Sometimes one example line cannot be run before you run an earlier one.
Don’t worry---Hyperdoc automatically runs all the necessary
lines in the right order!
For instance, the second example line above refers to \spad{a} which is
assigned in the first example line.
What happens if you first click on the second example line?
Axiom first issues the first line (to assign \spad{a}), then the
second (to do the computation using \spad{a}).

The new interactive Axiom buffer disappears when you leave
Hyperdoc.
If you want to get rid of it beforehand,
use the {\bf Cancel} button of the X Window manager
or issue the Axiom system command \spadsys{)close.}

\endscroll
\autobuttons
```

```
\end{page}

\begin{patch}{ugHyperExamplePagePatch1}
\begin{paste}{ugHyperExamplePageFull1}{ugHyperExamplePageEmpty1}
\pastebutton{ugHyperExamplePageFull1}{\hidepaste}
\tab{5}\spadcommand{a:= x**2 + 1\bound{a }}
\indentrel{3}\begin{verbatim}
         2
   (1)  x  + 1
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugHyperExamplePageEmpty1}
\begin{paste}{ugHyperExamplePageEmpty1}{ugHyperExamplePagePatch1}
\pastebutton{ugHyperExamplePageEmpty1}{\showpaste}
\tab{5}\spadcommand{a:= x**2 + 1\bound{a }}
\end{paste}\end{patch}

\begin{patch}{ugHyperExamplePagePatch2}
\begin{paste}{ugHyperExamplePageFull2}{ugHyperExamplePageEmpty2}
\pastebutton{ugHyperExamplePageFull2}{\hidepaste}
\tab{5}\spadcommand{(a - 2)**2\free{a }}
\indentrel{3}\begin{verbatim}
         4     2
   (2)  x  - 2x  + 1
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugHyperExamplePageEmpty2}
\begin{paste}{ugHyperExamplePageEmpty2}{ugHyperExamplePagePatch2}
\pastebutton{ugHyperExamplePageEmpty2}{\showpaste}
\tab{5}\spadcommand{(a - 2)**2\free{a }}
\end{paste}\end{patch}
```

———-

X Window Resources for Hyperdoc

— **ug03.ht** —

```
\begin{page}{ugHyperResourcesPage}{3.8. X Window Resources for Hyperdoc}
\beginscroll
%
You can control the appearance of Hyperdoc while running under Version 11
of the X Window System by placing the following resources
```

```
in the file {\bf .Xdefaults} in your home directory.
In what follows, {\it font} is any valid X11 font name
(for example, {\tt Rom14}) and {\it color} is any valid X11 color
specification (for example, {\tt NavyBlue}).
For more information about fonts and colors, refer to the
X Window documentation for your system.
\indent{0}
\beginitems
\item[{\tt Axiom.hyperdoc.RmFont:} {\it font}] \ \newline
This is the standard text font.  \xdefault{Rom14}
\item[{\tt Axiom.hyperdoc.RmColor:} {\it color}] \ \newline
This is the standard text color.  \xdefault{black}
\item[{\tt Axiom.hyperdoc.ActiveFont:} {\it font}] \ \newline
This is the font used for Hyperdoc link buttons.  \xdefault{Bld14}
\item[{\tt Axiom.hyperdoc.ActiveColor:} {\it color}] \ \newline
This is the color used for Hyperdoc link buttons.  \xdefault{black}
\item[{\tt Axiom.hyperdoc.AxiomFont:} {\it font}] \ \newline
This is the font used for active Axiom commands.\footnote{
This was called {\tt Axiom.hyperdoc.SpadFont} in early versions
of Axiom.}
\xdefault{Bld14}
\item[{\tt Axiom.hyperdoc.AxiomColor:} {\it color}] \ \newline
This is the color used for active Axiom commands.\footnote{
This was called {\tt Axiom.hyperdoc.SpadColor} in early versions
of Axiom.}
\xdefault{black}
\item[{\tt Axiom.hyperdoc.BoldFont:} {\it font}] \ \newline
This is the font used for bold face.  \xdefault{Bld14}
\item[{\tt Axiom.hyperdoc.BoldColor:} {\it color}] \ \newline
This is the color used for bold face.  \xdefault{black}
\item[{\tt Axiom.hyperdoc.TtFont:} {\it font}] \ \newline
This is the font used for Axiom output in Hyperdoc.
This font must be fixed-width.  \xdefault{Rom14}
\item[{\tt Axiom.hyperdoc.TtColor:} {\it color}] \ \newline
This is the color used for Axiom output in Hyperdoc.
\xdefault{black}
\item[{\tt Axiom.hyperdoc.EmphasizeFont:} {\it font}] \ \newline
This is the font used for italics.  \xdefault{Itl14}
\item[{\tt Axiom.hyperdoc.EmphasizeColor:} {\it color}] \ \newline
This is the color used for italics.  \xdefault{black}
\item[{\tt Axiom.hyperdoc.InputBackground:} {\it color}] \ \newline
This is the color used as the background for input areas.
\xdefault{black}
\item[{\tt Axiom.hyperdoc.InputForeground:} {\it color}] \ \newline
This is the color used as the foreground for input areas.
\xdefault{white}
\item[{\tt Axiom.hyperdoc.BorderColor:} {\it color}] \ \newline
This is the color used for drawing border lines.
\xdefault{black}
\item[{\tt Axiom.hyperdoc.Background:} {\it color}] \ \newline
```

```
This is the color used for the background of all windows.
\xdefault{white}
\enditems
\indent{0}
\endscroll
\autobuttons
\end{page}
```

Chapter 9

Users Guide Chapter 4 (ug04.ht)

Input Files and Output Styles

— **ug04.ht** —

```
\begin{page}{ugInOutPage}{4. Input Files and Output Styles}
\beginscroll

In this chapter we discuss how to collect Axiom statements
and commands into files and then read the contents into the
workspace.
We also show how to display the results of your computations in
several different styles including \texht{\TeX}{TeX}, FORTRAN and
monospace two-dimensional format.\footnote{\texht{\TeX}{TeX} is a
trademark of the American Mathematical Society.}

The printed version of this book uses the Axiom
\texht{\TeX}{TeX} output formatter.
When we demonstrate a particular output style, we will need to
turn \texht{\TeX}{TeX} formatting off and the output style on so
that the correct output is shown in the text.
```

```
\beginmenu
    \menudownlink{{4.1. Input Files}}{ugInOutInPage}
    \menudownlink{{4.2. The axiom.input File}}{ugInOutSpadprofPage}
    \menudownlink{{4.3. Common Features of Using Output Formats}}
{ugInOutOutPage}
    \menudownlink{{4.4. Monospace Two-Dimensional Mathematical Format}}
{ugInOutAlgebraPage}
    \menudownlink{{4.5. TeX Format}}{ugInOutTeXPage}
    \menudownlink{{4.6. IBM Script Formula Format}}{ugInOutScriptPage}
    \menudownlink{{4.7. FORTRAN Format}}{ugInOutFortranPage}
    \menudownlink{{4.8. HTML Format}}{ugInOutHTMLPage}
    \menudownlink{{4.9. MathML Format}}{ugInOutMathMLPage}
\endmenu
\endscroll
\autobuttons
\end{page}
```

———-

Input Files

⇒ “notitle” (ugLangBlocksPage) 9 on page 1753

— **ug04.ht** —

```
\begin{page}{ugInOutInPage}{4.1. Input Files}
\beginscroll
%
In this section we explain what an {\it input file} is and
why you would want to know about it.
We discuss where Axiom looks for input files and how you can
direct it to look elsewhere.
We also show how to read the contents of an input file into the
\spadgloss{workspace} and how to use the \spadgloss{history}
facility to generate an input file from the statements you have
entered directly into the workspace.

An {\it input} file contains Axiom expressions and system
commands.
Anything that you can enter directly to Axiom can be put
into an input file.
This is how you save input functions and expressions that you wish
to read into Axiom more than one time.

To read an input file into Axiom, use the \spadcmd{)read}
system command.
For example, you can read a file in a particular directory by issuing
\begin{verbatim}
```

```
)read /spad/src/input/matrix.input
\end{verbatim}
The ``{\bf .input}'' is optional; this also works:
\begin{verbatim}
)read /spad/src/input/matrix
\end{verbatim}
What happens if you just enter
\spadcmd{)read matrix.input} or even \spadcmd{)read matrix}?
Axiom looks in your current working directory for input files
that are not qualified by a directory name.
Typically, this directory is the directory from which you invoked
Axiom.
To change the current working directory, use the \spadcmd{)cd} system
command. The command \spadsys{)cd} by itself shows the current
working
directory.
To change it to
the \spadsys{src/input} subdirectory for user ``babar'',
issue
\begin{verbatim}
)cd /u/babar/src/input
\end{verbatim}
Axiom looks first in this directory for an input file.
If it is not found, it looks in the system's directories, assuming
you meant some input file that was provided with Axiom.

\beginImportant
If you have the Axiom history facility turned on (which it is
by default), you can save all the lines you have entered into the
workspace by entering
\begin{verbatim}
)history )write
\end{verbatim}

Axiom tells you what input file to edit to see your
statements.
The file is in your home directory or in the directory you
specified with \spadsys{)cd}.
\endImportant

In \downlink{``Blocks''}{ugLangBlocksPage} in Section 5.2
\ignore{ugLangBlocks}
we discuss using indentation in input files to group statements
into {\it blocks.}

\endscroll
\autobuttons
\end{page}
```

———-

The .axiom.input File

— ug04.ht —

```
\begin{page}{ugInOutSpadprofPage}{4.2. The .axiom.input File}
\beginscroll

When Axiom starts up, it tries to read the input file
{\bf .axiom.input} from your home
directory.
It there is no {\bf axiom.input} in your home directory, it reads the copy
located in its own {\bf src/input} directory.
The file usually contains
system commands to personalize your Axiom environment.
In the remainder of this section we mention a few things
that users frequently place in their
{\bf axiom.input} files.

In order to have FORTRAN output always produced from your
computations, place the system command
\spadcmd{)set output fortran on}
in {\bf axiom.input}.
If you do not want to be prompted for confirmation when you issue
the \spadcmd{)quit} system command, place
\spadcmd{)set quit unprotected}
in {\bf axiom.input}.
If you then decide that you do want to be prompted, issue
\spadcmd{)set quit protected}.
This is the default setting
so that new users do not leave Axiom
inadvertently.\footnote{The
system command \spadsys{)pquit} always prompts you for
confirmation.}

To see the other system variables you can set, issue \spadsys{)set}
or use the Hyperdoc {\bf Settings} facility to view and change
Axiom system variables.

\endscroll
\autobuttons
\end{page}
```

———-

Common Features of Using Output Formats

— **ug04.ht** —

```
\begin{page}{ugInOutOutPage}{4.3. Common Features of Using Output Formats}
\beginscroll

In this section we discuss how to start and stop the display
of the different output formats and how to send the output to the
screen or to a file.
To fix ideas, we use FORTRAN output format for most of the
examples.

You can use the \spadcmd{)set output}
system
command to
toggle or redirect the different kinds of output.
The name of the kind of output follows ``output'' in the command.
The names are

\indent{0}
\beginitems
\item[fortran]  for FORTRAN output.
\item[algebra]  for monospace two-dimensional mathematical output.
\item[tex]      for \texht{\TeX}{TeX} output.
\item[html]     for HTML output.
\item[mathml]   for MathML output.
\item[script]   for IBM Script Formula Format output.
\enditems
\indent{0}

For example, issue \spadsys{)set output fortran on} to turn on
FORTRAN format and
issue \spadsys{)set output fortran off} to turn it off.
By default, {\tt algebra} is {\tt on} and all others are {\tt off}.
When output is started, it is sent to the screen.
To send the output to a file, give the file name without
directory or extension.
Axiom appends a file extension depending on the kind of
output being produced.
\xtc{
Issue this to redirect FORTRAN output to, for example, the file
{\bf linalg.sfort}.
}{
\spadpaste{)set output fortran linalg}
}
\noOutputXtc{
You must {\it also} turn on the creation of FORTRAN output.
The above just says where it goes if it is created.
```

```
}{
\spadpaste{)set output fortran on}
}
In what directory is this output placed?
It goes into the directory from which you started Axiom,
or if you have used the \spadsys{)cd} system command, the one
that you specified with \spadsys{)cd}.
You should use \spadcmd{)cd} before you send the output to the file.

\noOutputXtc{
You can always direct output back to the screen by issuing this.
}{
\spadpaste{)set output fortran console}
}
\noOutputXtc{
Let's make sure FORTRAN formatting is off so that nothing we
do from now on produces FORTRAN output.
}{
\spadpaste{)set output fortran off}
}
\noOutputXtc{
We also delete the demonstrated output file we created.
}{
\spadpaste{)system rm linalg.sfort}
}

You can abbreviate the words ``\spad{on},'' ``\spad{off}'' and
``\spad{console}'' to the minimal number
of characters needed to distinguish them.
Because of this, you cannot send output to files called
{\bf on.sfort, off.sfort, of.sfort,
console.sfort, consol.sfort} and so on.

The width of the output on the page is set by
\spadcmd{)set output length}
for all formats except FORTRAN.
Use \spadcmd{)set fortran fortlength} to
change the FORTRAN line length from its default value of \spad{72}.

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugInOutOutPagePatch1}
\begin{paste}{ugInOutOutPageFull1}{ugInOutOutPageEmpty1}
\pastebutton{ugInOutOutPageFull1}{\hidepaste}
\tab{5}\spadcommand{)set output fortran linalg}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugInOutOutPageEmpty1}
\begin{paste}{ugInOutOutPageEmpty1}{ugInOutOutPagePatch1}
\pastebutton{ugInOutOutPageEmpty1}{\showpaste}
\tab{5}\spadcommand{)set output fortran linalg}
\end{paste}\end{patch}

\begin{patch}{ugInOutOutPagePatch2}
\begin{paste}{ugInOutOutPageFull2}{ugInOutOutPageEmpty2}
\pastebutton{ugInOutOutPageFull2}{\hidepaste}
\tab{5}\spadcommand{)set output fortran on}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugInOutOutPageEmpty2}
\begin{paste}{ugInOutOutPageEmpty2}{ugInOutOutPagePatch2}
\pastebutton{ugInOutOutPageEmpty2}{\showpaste}
\tab{5}\spadcommand{)set output fortran on}
\end{paste}\end{patch}

\begin{patch}{ugInOutOutPagePatch3}
\begin{paste}{ugInOutOutPageFull3}{ugInOutOutPageEmpty3}
\pastebutton{ugInOutOutPageFull3}{\hidepaste}
\tab{5}\spadcommand{)set output fortran console}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugInOutOutPageEmpty3}
\begin{paste}{ugInOutOutPageEmpty3}{ugInOutOutPagePatch3}
\pastebutton{ugInOutOutPageEmpty3}{\showpaste}
\tab{5}\spadcommand{)set output fortran console}
\end{paste}\end{patch}

\begin{patch}{ugInOutOutPagePatch4}
\begin{paste}{ugInOutOutPageFull4}{ugInOutOutPageEmpty4}
\pastebutton{ugInOutOutPageFull4}{\hidepaste}
\tab{5}\spadcommand{)set output fortran off}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugInOutOutPageEmpty4}
\begin{paste}{ugInOutOutPageEmpty4}{ugInOutOutPagePatch4}
\pastebutton{ugInOutOutPageEmpty4}{\showpaste}
\tab{5}\spadcommand{)set output fortran off}
\end{paste}\end{patch}

\begin{patch}{ugInOutOutPagePatch5}
```

```
\begin{paste}{ugInOutOutPageFull5}{ugInOutOutPageEmpty5}
\pastebutton{ugInOutOutPageFull5}{\hidepaste}
\tab{5}\spadcommand{)system rm linalg.sfort}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugInOutOutPageEmpty5}
\begin{paste}{ugInOutOutPageEmpty5}{ugInOutOutPagePatch5}
\pastebutton{ugInOutOutPageEmpty5}{\showpaste}
\tab{5}\spadcommand{)system rm linalg.sfort}
\end{paste}\end{patch}
```

———-

Monospace 2D Mathematical Format

— ug04.ht —

```
\begin{page}{ugInOutAlgebraPage}
{4.4. Monospace 2D Mathematical Format}
\beginscroll

This is the default output format for Axiom.
It is usually on when you start the system.

\texht{\vskip 4pc}{}
\noOutputXtc{
If it is not, issue this.
}{
\spadpaste{)set output algebra on \bound{algon}}
}
\noOutputXtc{
Since the printed version of this book
(as opposed to the Hyperdoc version)
shows output produced by the
\texht{\TeX}{TeX}{} output formatter,
let us temporarily turn off
\texht{\TeX}{TeX}{} output.
}{
\spadpaste{)set output tex off \bound{texoff}}
}
\xtc{
Here is an example of what it looks like.
}{
\spadpaste{matrix [[i*x**i + j*\%i*y**j for i in 1..2] for j in 3..4]
\free{algon texoff}}
```

```
}
\noOutputXtc{
Issue this to turn off this kind of formatting.
}{
\spadpaste{)set output algebra off}
}
\noOutputXtc{
Turn \texht{\TeX}{TeX}{} output on again.
}{
\spadpaste{)set output tex on}
}

The characters used for the matrix brackets above are rather ugly.
You get this character set when you issue
\spadcmd{)set output characters plain}.
This character set should be used when you are running on a machine
that does not support the IBM extended ASCII character set.
If you are running on an IBM workstation, for example, issue
\spadcmd{)set output characters default}
to get better looking output.

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugInOutAlgebraPagePatch1}
\begin{paste}{ugInOutAlgebraPageFull1}{ugInOutAlgebraPageEmpty1}
\pastebutton{ugInOutAlgebraPageFull1}{\hidepaste}
\tab{5}\spadcommand{)set output algebra on\bound{algon }}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugInOutAlgebraPageEmpty1}
\begin{paste}{ugInOutAlgebraPageEmpty1}{ugInOutAlgebraPagePatch1}
\pastebutton{ugInOutAlgebraPageEmpty1}{\showpaste}
\tab{5}\spadcommand{)set output algebra on\bound{algon }}
\end{paste}\end{patch}

\begin{patch}{ugInOutAlgebraPagePatch2}
\begin{paste}{ugInOutAlgebraPageFull2}{ugInOutAlgebraPageEmpty2}
\pastebutton{ugInOutAlgebraPageFull2}{\hidepaste}
\tab{5}\spadcommand{)set output tex off\bound{texoff }}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugInOutAlgebraPageEmpty2}
\begin{paste}{ugInOutAlgebraPageEmpty2}{ugInOutAlgebraPagePatch2}
\pastebutton{ugInOutAlgebraPageEmpty2}{\showpaste}
```

```
\tab{5}\spadcommand{)set output tex off\bound{texoff }}
\end{paste}\end{patch}

\begin{patch}{ugInOutAlgebraPagePatch3}
\begin{paste}{ugInOutAlgebraPageFull3}{ugInOutAlgebraPageEmpty3}
\pastebutton{ugInOutAlgebraPageFull3}{\hidepaste}
\tab{5}\spadcommand{matrix [[i*x**i + j*\%i*y**j for i in 1..2] for j in 3..4]\free{algon texoff
\indentrel{3}\begin{verbatim}
             3           3     2
        3%i y  + x  3%i y  + 2x
   (1)
             4           4     2
        4%i y  + x  4%i y  + 2x
                Type: Matrix Polynomial Complex Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugInOutAlgebraPageEmpty3}
\begin{paste}{ugInOutAlgebraPageEmpty3}{ugInOutAlgebraPagePatch3}
\pastebutton{ugInOutAlgebraPageEmpty3}{\showpaste}
\tab{5}\spadcommand{matrix [[i*x**i + j*\%i*y**j for i in 1..2] for j in 3..4]\free{algon texoff
\end{paste}\end{patch}

\begin{patch}{ugInOutAlgebraPagePatch4}
\begin{paste}{ugInOutAlgebraPageFull4}{ugInOutAlgebraPageEmpty4}
\pastebutton{ugInOutAlgebraPageFull4}{\hidepaste}
\tab{5}\spadcommand{)set output algebra off}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugInOutAlgebraPageEmpty4}
\begin{paste}{ugInOutAlgebraPageEmpty4}{ugInOutAlgebraPagePatch4}
\pastebutton{ugInOutAlgebraPageEmpty4}{\showpaste}
\tab{5}\spadcommand{)set output algebra off}
\end{paste}\end{patch}

\begin{patch}{ugInOutAlgebraPagePatch5}
\begin{paste}{ugInOutAlgebraPageFull5}{ugInOutAlgebraPageEmpty5}
\pastebutton{ugInOutAlgebraPageFull5}{\hidepaste}
\tab{5}\spadcommand{)set output tex on}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugInOutAlgebraPageEmpty5}
\begin{paste}{ugInOutAlgebraPageEmpty5}{ugInOutAlgebraPagePatch5}
\pastebutton{ugInOutAlgebraPageEmpty5}{\showpaste}
\tab{5}\spadcommand{)set output tex on}
\end{paste}\end{patch}
```

TeX Format

— ug04.ht —

```
\begin{page}{ugInOutTeXPage}{4.5. TeX Format}
\beginscroll

Axiom can produce \texht{\TeX}{TeX}{} output for your
expressions.
The output is produced using macros from the
\texht{\LaTeX}{LaTeX} document preparation system by
Leslie Lamport.\footnote{See Leslie Lamport, {\it LaTeX: A Document
Preparation System,} Reading, Massachusetts: Addison-Wesley
Publishing Company, Inc., 1986.}
The printed version of this book was produced using this formatter.

\noOutputXtc{
To turn on \texht{\TeX}{TeX}{} output formatting, issue this.
}{
\spadpaste{)set output tex on \bound{texon}}
}
Here is an example of its output.
\begin{verbatim}
matrix [[i*x**i + j*\%i*y**j for i in 1..2] for j in 3..4]

\[
\left[
\begin{array}{cc}
\displaystyle {{3 \  \%i \  {y \sp 3}}+x}&
\displaystyle {{3 \  \%i \  {y \sp 3}}+{2 \  {x \sp 2}}} \\
\displaystyle {{4 \  \%i \  {y \sp 4}}+x}&
\displaystyle {{4 \  \%i \  {y \sp 4}}+{2 \  {x \sp 2}}}
\end{array}
\right]
\leqno(3)
\]
%Axiom STEP NUMBER: 3
\end{verbatim}
To turn \texht{\TeX}{TeX}{} output formatting off, issue \spadsys{)set
output tex off}.
The \texht{\LaTeX}{LaTeX} macros in the output generated by Axiom
are all standard except for the following definitions:
\begin{verbatim}
\def\csch{\mathop{\rm csch}\nolimits}
```

```
\def\erf{\mathop{\rm erf}\nolimits}

\def\zag#1#2{
  {{\hfill \left. {#1} \right|}
   \over
   {\left| {#2} \right. \hfill}
  }
}
\end{verbatim}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugInOutTeXPagePatch1}
\begin{paste}{ugInOutTeXPageFull1}{ugInOutTeXPageEmpty1}
\pastebutton{ugInOutTeXPageFull1}{\hidepaste}
\tab{5}\spadcommand{)set output tex on\bound{texon }}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugInOutTeXPageEmpty1}
\begin{paste}{ugInOutTeXPageEmpty1}{ugInOutTeXPagePatch1}
\pastebutton{ugInOutTeXPageEmpty1}{\showpaste}
\tab{5}\spadcommand{)set output tex on\bound{texon }}
\end{paste}\end{patch}
```

———-

IBM Script Formula Format

— ug04.ht —

```
\begin{page}{ugInOutScriptPage}{4.6. IBM Script Formula Format}
\beginscroll

Axiom can
produce IBM Script Formula Format output for your
expressions.

\texht{\vskip 2pc}{}
\noOutputXtc{
To turn IBM Script Formula Format on, issue this.
}{
\spadpaste{)set output script on}
```

```
}
Here is an example of its output.
\begin{verbatim}
matrix [[i*x**i + j*%i*y**j for i in 1..2] for j in 3..4]

.eq set blank @
:df.
<left lb < < < <3 @@ %i @@ <y sup 3>>+x> here < <3 @@ %i @@
<y sup 3>>+<2 @@ <x sup 2>>>> habove < < <4 @@ %i @@
<y sup 4>>+x> here < <4 @@ %i @@ <y sup 4>>+<2 @@
<x up 2>>>>> right rb>
:edf.
\end{verbatim}
\noOutputXtc{
To turn IBM Script Formula Format output formatting off, issue this.
}{
\spadpaste{)set output script off}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugInOutScriptPagePatch1}
\begin{paste}{ugInOutScriptPageFull1}{ugInOutScriptPageEmpty1}
\pastebutton{ugInOutScriptPageFull1}{\hidepaste}
\tab{5}\spadcommand{)set output script on}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugInOutScriptPageEmpty1}
\begin{paste}{ugInOutScriptPageEmpty1}{ugInOutScriptPagePatch1}
\pastebutton{ugInOutScriptPageEmpty1}{\showpaste}
\tab{5}\spadcommand{)set output script on}
\end{paste}\end{patch}

\begin{patch}{ugInOutScriptPagePatch2}
\begin{paste}{ugInOutScriptPageFull2}{ugInOutScriptPageEmpty2}
\pastebutton{ugInOutScriptPageFull2}{\hidepaste}
\tab{5}\spadcommand{)set output script off}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugInOutScriptPageEmpty2}
\begin{paste}{ugInOutScriptPageEmpty2}{ugInOutScriptPagePatch2}
\pastebutton{ugInOutScriptPageEmpty2}{\showpaste}
\tab{5}\spadcommand{)set output script off}
\end{paste}\end{patch}
```

———-

FORTRAN Format

— ug04.ht —

```
\begin{page}{ugInOutFortranPage}{4.7. FORTRAN Format}
\beginscroll

In addition to turning FORTRAN output on and off and stating where the
output should be placed, there are many options that control the
appearance of the generated code.  In this section we describe some of
the basic options.  Issue \spadcmd{)set fortran} to see a full list
with their current settings.

The output FORTRAN expression usually begins in column 7.  If the
expression needs more than one line, the ampersand character
\spadSyntax{\&} is used in column 6.  Since some versions of FORTRAN
have restrictions on the number of lines per statement, Axiom breaks
long expressions into segments with a maximum of 1320 characters (20
lines of 66 characters) per segment.  If you want to change this, say,
to 660 characters, issue the system command \spadcmd{)set fortran
explength 660}.  You can turn off the line breaking by issuing
\spadcmd{)set fortran segment off}.  Various code optimization levels
are available.  % \noOutputXtc{ FORTRAN output is produced after you
issue this.  }{ \spadpaste{)set output fortran on \bound{forton}} }
\noOutputXtc{ For the initial examples, we set the optimization level
to 0, which is the lowest level.  }{ \spadpaste{)set fortran optlevel
0 \bound{opt0}\free{forton}} } \noOutputXtc{ The output is usually in
columns 7 through 72, although fewer columns are used in the following
examples so that the output fits nicely on the page.  }{
\spadpaste{)set fortran fortlength 60} } \xtc{ By default, the output
goes to the screen and is displayed before the standard Axiom
two-dimensional output.  In this example, an assignment to the
variable \spad{R1} was generated because this is the result of step 1.
}{ \spadpaste{(x+y)**3 \free{opt0}} }

\xtc{ Here is an example that illustrates the line breaking.  }
{ \spadpaste{(x+y+z)**3 \free{opt0}}
}

Note in the above examples that integers are generally converted to
floating point numbers, except in exponents.
This is the default behavior but can be turned off by issuing
\spadcmd{)set fortran ints2floats off}.
The rules governing when the conversion is done are:
```

```
\indent{4}
\beginitems
\item[1. ] If an integer is an exponent, convert it to a floating point
number if it is greater than 32767 in absolute value, otherwise leave it
as an integer.
\item[2. ] Convert all other integers in an expression to floating
point numbers.
\enditems
\indent{0}
These rules only govern integers in expressions.
Numbers generated by Axiom for \spad{DIMENSION} statements are also
integers.

To set the type of generated FORTRAN data,
use one of the following:
\begin{verbatim}
)set fortran defaulttype REAL
)set fortran defaulttype INTEGER
)set fortran defaulttype COMPLEX
)set fortran defaulttype LOGICAL
)set fortran defaulttype CHARACTER
\end{verbatim}

\xtc{
When temporaries are created, they are given a default type of
{\tt REAL.}
Also, the {\tt REAL} versions of functions are used by default.
}{
\spadpaste{sin(x) \free{opt1}}
}
\noOutputXtc{
At optimization level 1, Axiom removes common subexpressions.
}{
\spadpaste{)set fortran optlevel 1 \bound{opt1}\free{forton}}
}
\xtc{
}{
\spadpaste{(x+y+z)**3 \free{opt1}}
}
\noOutputXtc{
This changes the precision to {\tt DOUBLE}.
Substitute \spad{single} for \spad{double}
to return to single precision.
}{
\spadpaste{)set fortran precision double \free{opt1}\bound{double1}}
}
\xtc{
Complex constants display the precision.
}{
\spadpaste{2.3 + 5.6*\%i  \free{double1}}
```

```
}
\xtc{
The function names that Axiom generates depend on the chosen
precision.
}{
\spadpaste{sin \%e  \free{double1}}
}
\noOutputXtc{
Reset the precision to \spad{single} and look at these two
examples again.
}{
\spadpaste{)set fortran precision single \free{opt1}\bound{single1}}
}
\xtc{
}{
\spadpaste{2.3 + 5.6*\%i  \free{single1}}
}
\xtc{
}{
\spadpaste{sin \%e  \free{single1}}
}
\xtc{
Expressions that look like lists, streams, sets or matrices cause
array code to be generated.
}{
\spadpaste{[x+1,y+1,z+1] \free{opt1}}
}
\xtc{
A temporary variable is generated to be the name of the array.
This may have to be changed in your particular application.
}{
\spadpaste{set[2,3,4,3,5] \free{opt1}}
}
\xtc{
By default, the starting index for generated FORTRAN arrays is \spad{0}.
}{
\spadpaste{matrix [[2.3,9.7],[0.0,18.778]] \free{opt1}}
}
\noOutputXtc{
To change the starting index for generated FORTRAN arrays to be \spad{1},
issue this.
This value can only be \spad{0} or \spad{1}.
}{
\spadpaste{)set fortran startindex 1 \free{opt1}\bound{start1}}
}
\xtc{
Look at the code generated for the matrix again.
}{
\spadpaste{matrix [[2.3,9.7],[0.0,18.778]] \free{start1}}
}
```

```
\endscroll
\autobuttons
\end{page}

\begin{patch}{ugInOutFortranPagePatch1}
\begin{paste}{ugInOutFortranPageFull1}{ugInOutFortranPageEmpty1}
\pastebutton{ugInOutFortranPageFull1}{\hidepaste}
\tab{5}\spadcommand{)set output fortran on\bound{forton }}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugInOutFortranPageEmpty1}
\begin{paste}{ugInOutFortranPageEmpty1}{ugInOutFortranPagePatch1}
\pastebutton{ugInOutFortranPageEmpty1}{\showpaste}
\tab{5}\spadcommand{)set output fortran on\bound{forton }}
\end{paste}\end{patch}

\begin{patch}{ugInOutFortranPagePatch2}
\begin{paste}{ugInOutFortranPageFull2}{ugInOutFortranPageEmpty2}
\pastebutton{ugInOutFortranPageFull2}{\hidepaste}
\tab{5}\spadcommand{)set fortran optlevel 0\bound{opt0 }\free{forton }}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugInOutFortranPageEmpty2}
\begin{paste}{ugInOutFortranPageEmpty2}{ugInOutFortranPagePatch2}
\pastebutton{ugInOutFortranPageEmpty2}{\showpaste}
\tab{5}\spadcommand{)set fortran optlevel 0\bound{opt0 }\free{forton }}
\end{paste}\end{patch}

\begin{patch}{ugInOutFortranPagePatch3}
\begin{paste}{ugInOutFortranPageFull3}{ugInOutFortranPageEmpty3}
\pastebutton{ugInOutFortranPageFull3}{\hidepaste}
\tab{5}\spadcommand{)set fortran fortlength 60}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugInOutFortranPageEmpty3}
\begin{paste}{ugInOutFortranPageEmpty3}{ugInOutFortranPagePatch3}
\pastebutton{ugInOutFortranPageEmpty3}{\showpaste}
\tab{5}\spadcommand{)set fortran fortlength 60}
\end{paste}\end{patch}

\begin{patch}{ugInOutFortranPagePatch4}
\begin{paste}{ugInOutFortranPageFull4}{ugInOutFortranPageEmpty4}
\pastebutton{ugInOutFortranPageFull4}{\hidepaste}
\tab{5}\spadcommand{(x+y)**3\free{opt0 }}
```

```
\indentrel{3}\begin{verbatim}
         3       2     2     3
   (1)  y  + 3x y  + 3x y + x
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugInOutFortranPageEmpty4}
\begin{paste}{ugInOutFortranPageEmpty4}{ugInOutFortranPagePatch4}
\pastebutton{ugInOutFortranPageEmpty4}{\showpaste}
\tab{5}\spadcommand{(x+y)**3\free{opt0 }}
\end{paste}\end{patch}

\begin{patch}{ugInOutFortranPagePatch5}
\begin{paste}{ugInOutFortranPageFull5}{ugInOutFortranPageEmpty5}
\pastebutton{ugInOutFortranPageFull5}{\hidepaste}
\tab{5}\spadcommand{(x+y+z)**3\free{opt0 }}
\indentrel{3}\begin{verbatim}
   (2)
      3            2      2            2      3       2
     z  + (3y + 3x)z  + (3y  + 6x y + 3x )z + y  + 3x y
   +
       2     3
     3x y + x
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugInOutFortranPageEmpty5}
\begin{paste}{ugInOutFortranPageEmpty5}{ugInOutFortranPagePatch5}
\pastebutton{ugInOutFortranPageEmpty5}{\showpaste}
\tab{5}\spadcommand{(x+y+z)**3\free{opt0 }}
\end{paste}\end{patch}

\begin{patch}{ugInOutFortranPagePatch6}
\begin{paste}{ugInOutFortranPageFull6}{ugInOutFortranPageEmpty6}
\pastebutton{ugInOutFortranPageFull6}{\hidepaste}
\tab{5}\spadcommand{sin(x)\free{opt1 }}
\indentrel{3}\begin{verbatim}
   (3)  sin(x)
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugInOutFortranPageEmpty6}
\begin{paste}{ugInOutFortranPageEmpty6}{ugInOutFortranPagePatch6}
\pastebutton{ugInOutFortranPageEmpty6}{\showpaste}
\tab{5}\spadcommand{sin(x)\free{opt1 }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugInOutFortranPagePatch7}
\begin{paste}{ugInOutFortranPageFull7}{ugInOutFortranPageEmpty7}
\pastebutton{ugInOutFortranPageFull7}{\hidepaste}
\tab{5}\spadcommand{)set fortran optlevel 1\bound{opt1 }\free{forton }}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugInOutFortranPageEmpty7}
\begin{paste}{ugInOutFortranPageEmpty7}{ugInOutFortranPagePatch7}
\pastebutton{ugInOutFortranPageEmpty7}{\showpaste}
\tab{5}\spadcommand{)set fortran optlevel 1\bound{opt1 }\free{forton }}
\end{paste}\end{patch}

\begin{patch}{ugInOutFortranPagePatch8}
\begin{paste}{ugInOutFortranPageFull8}{ugInOutFortranPageEmpty8}
\pastebutton{ugInOutFortranPageFull8}{\hidepaste}
\tab{5}\spadcommand{(x+y+z)**3\free{opt1 }}
\indentrel{3}\begin{verbatim}
   (4)
      3             2      2            2      3       2
     z  + (3y + 3x)z  + (3y  + 6x y + 3x )z + y  + 3x y
   +
       2     3
     3x y + x
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugInOutFortranPageEmpty8}
\begin{paste}{ugInOutFortranPageEmpty8}{ugInOutFortranPagePatch8}
\pastebutton{ugInOutFortranPageEmpty8}{\showpaste}
\tab{5}\spadcommand{(x+y+z)**3\free{opt1 }}
\end{paste}\end{patch}

\begin{patch}{ugInOutFortranPagePatch9}
\begin{paste}{ugInOutFortranPageFull9}{ugInOutFortranPageEmpty9}
\pastebutton{ugInOutFortranPageFull9}{\hidepaste}
\tab{5}\spadcommand{)set fortran precision double\free{opt1 }\bound{double1 }}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugInOutFortranPageEmpty9}
\begin{paste}{ugInOutFortranPageEmpty9}{ugInOutFortranPagePatch9}
\pastebutton{ugInOutFortranPageEmpty9}{\showpaste}
\tab{5}\spadcommand{)set fortran precision double\free{opt1 }\bound{double1 }}
\end{paste}\end{patch}

\begin{patch}{ugInOutFortranPagePatch10}
```

```
\begin{paste}{ugInOutFortranPageFull10}{ugInOutFortranPageEmpty10}
\pastebutton{ugInOutFortranPageFull10}{\hidepaste}
\tab{5}\spadcommand{2.3 + 5.6*\%i\free{double1 }}
\indentrel{3}\begin{verbatim}
   (5)  2.3 + 5.6 %i
                                    Type: Complex Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugInOutFortranPageEmpty10}
\begin{paste}{ugInOutFortranPageEmpty10}{ugInOutFortranPagePatch10}
\pastebutton{ugInOutFortranPageEmpty10}{\showpaste}
\tab{5}\spadcommand{2.3 + 5.6*\%i\free{double1 }}
\end{paste}\end{patch}

\begin{patch}{ugInOutFortranPagePatch11}
\begin{paste}{ugInOutFortranPageFull11}{ugInOutFortranPageEmpty11}
\pastebutton{ugInOutFortranPageFull11}{\hidepaste}
\tab{5}\spadcommand{sin \%e\free{double1 }}
\indentrel{3}\begin{verbatim}
   (6)  sin(%e)
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugInOutFortranPageEmpty11}
\begin{paste}{ugInOutFortranPageEmpty11}{ugInOutFortranPagePatch11}
\pastebutton{ugInOutFortranPageEmpty11}{\showpaste}
\tab{5}\spadcommand{sin \%e\free{double1 }}
\end{paste}\end{patch}

\begin{patch}{ugInOutFortranPagePatch12}
\begin{paste}{ugInOutFortranPageFull12}{ugInOutFortranPageEmpty12}
\pastebutton{ugInOutFortranPageFull12}{\hidepaste}
\tab{5}\spadcommand{)set fortran precision single\free{opt1 }\bound{single1 }}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugInOutFortranPageEmpty12}
\begin{paste}{ugInOutFortranPageEmpty12}{ugInOutFortranPagePatch12}
\pastebutton{ugInOutFortranPageEmpty12}{\showpaste}
\tab{5}\spadcommand{)set fortran precision single\free{opt1 }\bound{single1 }}
\end{paste}\end{patch}

\begin{patch}{ugInOutFortranPagePatch13}
\begin{paste}{ugInOutFortranPageFull13}{ugInOutFortranPageEmpty13}
\pastebutton{ugInOutFortranPageFull13}{\hidepaste}
\tab{5}\spadcommand{2.3 + 5.6*\%i\free{single1 }}
\indentrel{3}\begin{verbatim}
```

```
   (7)  2.3 + 5.6 %i
                                    Type: Complex Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugInOutFortranPageEmpty13}
\begin{paste}{ugInOutFortranPageEmpty13}{ugInOutFortranPagePatch13}
\pastebutton{ugInOutFortranPageEmpty13}{\showpaste}
\tab{5}\spadcommand{2.3 + 5.6*\%i\free{single1 }}
\end{paste}\end{patch}

\begin{patch}{ugInOutFortranPagePatch14}
\begin{paste}{ugInOutFortranPageFull14}{ugInOutFortranPageEmpty14}
\pastebutton{ugInOutFortranPageFull14}{\hidepaste}
\tab{5}\spadcommand{sin \%e\free{single1 }}
\indentrel{3}\begin{verbatim}
   (8)  sin(%e)
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugInOutFortranPageEmpty14}
\begin{paste}{ugInOutFortranPageEmpty14}{ugInOutFortranPagePatch14}
\pastebutton{ugInOutFortranPageEmpty14}{\showpaste}
\tab{5}\spadcommand{sin \%e\free{single1 }}
\end{paste}\end{patch}

\begin{patch}{ugInOutFortranPagePatch15}
\begin{paste}{ugInOutFortranPageFull15}{ugInOutFortranPageEmpty15}
\pastebutton{ugInOutFortranPageFull15}{\hidepaste}
\tab{5}\spadcommand{[x+1,y+1,z+1]\free{opt1 }}
\indentrel{3}\begin{verbatim}
   (9)  [x + 1,y + 1,z + 1]
                          Type: List Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugInOutFortranPageEmpty15}
\begin{paste}{ugInOutFortranPageEmpty15}{ugInOutFortranPagePatch15}
\pastebutton{ugInOutFortranPageEmpty15}{\showpaste}
\tab{5}\spadcommand{[x+1,y+1,z+1]\free{opt1 }}
\end{paste}\end{patch}

\begin{patch}{ugInOutFortranPagePatch16}
\begin{paste}{ugInOutFortranPageFull16}{ugInOutFortranPageEmpty16}
\pastebutton{ugInOutFortranPageFull16}{\hidepaste}
\tab{5}\spadcommand{set[2,3,4,3,5]\free{opt1 }}
\indentrel{3}\begin{verbatim}
   (10)  {2,3,4,5}
                              Type: Set PositiveInteger
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugInOutFortranPageEmpty16}
\begin{paste}{ugInOutFortranPageEmpty16}{ugInOutFortranPagePatch16}
\pastebutton{ugInOutFortranPageEmpty16}{\showpaste}
\tab{5}\spadcommand{set[2,3,4,3,5]\free{opt1 }}
\end{paste}\end{patch}

\begin{patch}{ugInOutFortranPagePatch17}
\begin{paste}{ugInOutFortranPageFull17}{ugInOutFortranPageEmpty17}
\pastebutton{ugInOutFortranPageFull17}{\hidepaste}
\tab{5}\spadcommand{matrix [[2.3,9.7],[0.0,18.778]]\free{opt1 }}
\indentrel{3}\begin{verbatim}
         2.3   9.7
   (11)
         0.0  18.778
                                        Type: Matrix Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugInOutFortranPageEmpty17}
\begin{paste}{ugInOutFortranPageEmpty17}{ugInOutFortranPagePatch17}
\pastebutton{ugInOutFortranPageEmpty17}{\showpaste}
\tab{5}\spadcommand{matrix [[2.3,9.7],[0.0,18.778]]\free{opt1 }}
\end{paste}\end{patch}

\begin{patch}{ugInOutFortranPagePatch18}
\begin{paste}{ugInOutFortranPageFull18}{ugInOutFortranPageEmpty18}
\pastebutton{ugInOutFortranPageFull18}{\hidepaste}
\tab{5}\spadcommand{)set fortran startindex 1\free{opt1 }\bound{start1 }}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugInOutFortranPageEmpty18}
\begin{paste}{ugInOutFortranPageEmpty18}{ugInOutFortranPagePatch18}
\pastebutton{ugInOutFortranPageEmpty18}{\showpaste}
\tab{5}\spadcommand{)set fortran startindex 1\free{opt1 }\bound{start1 }}
\end{paste}\end{patch}

\begin{patch}{ugInOutFortranPagePatch19}
\begin{paste}{ugInOutFortranPageFull19}{ugInOutFortranPageEmpty19}
\pastebutton{ugInOutFortranPageFull19}{\hidepaste}
\tab{5}\spadcommand{matrix [[2.3,9.7],[0.0,18.778]]\free{start1 }}
\indentrel{3}\begin{verbatim}
         2.3   9.7
   (12)
         0.0  18.778
                                        Type: Matrix Float
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugInOutFortranPageEmpty19}
\begin{paste}{ugInOutFortranPageEmpty19}{ugInOutFortranPagePatch19}
\pastebutton{ugInOutFortranPageEmpty19}{\showpaste}
\tab{5}\spadcommand{matrix [[2.3,9.7],[0.0,18.778]]\free{start1 }}
\end{paste}\end{patch}
```

———-

HTML Format

— ug04.ht —

```
\begin{page}{ugInOutHTMLPage}{4.8. HTML Format}
\beginscroll

Axiom can produce HTML output for your expressions.
The output is produced by the HTMLFormat domain.

To turn on HTML output formatting, issue this.

)set output html on

To turn HTML output formatting off, issue

)set output html off

\endscroll
\autobuttons
\end{page}

\pagetitle{ugInOutMathMLPage}{ug04.ht}{MathML Format}
\begin{chunk}{ug04.ht}
\begin{page}{ugInOutMathMLPage}{4.9. MathML Format}
\beginscroll

Axiom can produce MathML output for your expressions.
The output is produced by the MathMLFormat domain.

To turn on MathML output formatting, issue this.

)set output mathml on

To turn MathML output formatting off, issue
```

```
)set output mathml off

\endscroll
\autobuttons
\end{page}

\chapter{Users Guide Chapter 5 (ug05.ht)}
\pagetitle{ugLangPage}{ug05.ht}
{Introduction to the Axiom Interactive Language}
\pagepic{ps/v71uglangpage.eps}{uglangpage}{0.50}
\pagefrom{Reference}{TopReferencePage}
\pageto{Immediate and Delayed Assignments}
{ugLangAssignPage}
\pageto{Blocks}{ugLangBlocksPage}
\pageto{if-then-else}{ugLangIfPage}
\pageto{Loops}{ugLangLoopsPage}
\pageto{Creating Lists and Streams with Iterators}{ugLangItsPage}
\pageto{An Example: Streams of Primes}{ugLangStreamsPrimesPage}
\begin{chunk}{ug05.ht}
\begin{page}{ugLangPage}{5. Introduction to the Axiom Interactive Language}
\beginscroll

In this chapter we look at some of the basic components of the
Axiom language that you can use interactively.
We show how to create a \spadgloss{block} of expressions,
how to form loops and list iterations, how to modify the sequential
evaluation of a block and how to use {\tt if-then-else} to
evaluate parts of your program conditionally.
We suggest you first read the boxed material in each section and then
proceed to a more thorough reading of the chapter.

\beginmenu
    \menudownlink{{5.1. Immediate and Delayed Assignments}}
              {ugLangAssignPage}
    \menudownlink{{5.2. Blocks}}{ugLangBlocksPage}
    \menudownlink{{5.3. if-then-else}}{ugLangIfPage}
    \menudownlink{{5.4. Loops}}{ugLangLoopsPage}
    \menudownlink{{5.5. Creating Lists and Streams with Iterators}}
{ugLangItsPage}
    \menudownlink{{5.6. An Example: Streams of Primes}}
{ugLangStreamsPrimesPage}
\endmenu
\endscroll
\autobuttons
\end{page}
```

Immediate and Delayed Assignments

⇒ "notitle" (ugUserDelayPage) 10 on page 1851

— ug05.ht —

```
\begin{page}{ugLangAssignPage}{5.1. Immediate and Delayed Assignments}
\beginscroll

A \spadgloss{variable} in Axiom refers to a value.
A variable has a name beginning with an uppercase or lowercase alphabetic
character, \axiomSyntax{\%}, or \axiomSyntax{!}.
Successive characters (if any) can be any of the above, digits, or
\axiomSyntax{?}.
Case is distinguished.
The following are all examples of valid, distinct variable names:
\begin{verbatim}
a             tooBig?    a1B2c3%!?
A             %j         numberOfPoints
beta6         %J         numberofpoints
\end{verbatim}

The \axiomSyntax{:=} operator is the immediate \spadgloss{assignment}
operator.
Use it to associate a value with a variable.

\beginImportant
The syntax for immediate assignment for a single variable is
\centerline{{{\it variable} \axiom{:=} {\it expression}}}
The value returned by an immediate assignment is the value of
{\it expression}.
\endImportant

\xtc{
The right-hand side of the expression is evaluated,
yielding \axiom{1}.  This value is then assigned to \axiom{a}.
}{
\spadpaste{a := 1 \bound{a}}
}
\xtc{
The right-hand side of the expression is evaluated,
yielding \axiom{1}.  This value is then assigned to \axiom{b}.
Thus \axiom{a} and \axiom{b} both have the value \axiom{1} after
the sequence of assignments.
}{
\spadpaste{b := a \free{a}\bound{b}}
}
\xtc{
What is the value of \axiom{b} if \axiom{a} is
assigned the value \axiom{2}?
```

```
}{
\spadpaste{a := 2 \bound{a2}}
}
\xtc{
As you see, the value of \axiom{b} is left unchanged.
}{
\spadpaste{b \free{b}}
}
This is what we mean when we say this kind of assignment is
{\it immediate};
\axiom{b} has no dependency on \axiom{a} after the initial assignment.
This is the usual notion of assignment found in programming
languages such as C,
PASCAL
and FORTRAN.

Axiom provides delayed assignment with \axiomSyntax{==}.
This implements a
delayed evaluation of the right-hand side and dependency
checking.

\beginImportant
The syntax for delayed assignment is
\centerline{{{\it variable} \axiom{==} {\it expression}}}
The value returned by a delayed assignment is \void{}.
\endImportant

\xtc{
Using \axiom{a} and \axiom{b} as above, these are the corresponding
delayed assignments.
}{
\spadpaste{a == 1 \bound{ad}}
}
\xtc{
}{
\spadpaste{b == a \free{ad}\bound{bd}}
}
\xtc{
The right-hand side of each delayed assignment
is left unevaluated until the
variables on the left-hand sides are evaluated.
Therefore this evaluation and \ldots
}{
\spadpaste{a \free{ad}}
}
\xtc{
this evaluation seem the same as before.
}{
\spadpaste{b \free{bd}}
}
```

```
\xtc{
If we change \axiom{a} to \axiom{2}
}{
\spadpaste{a == 2 \bound{ad2}}
}
\xtc{
then
\axiom{a} evaluates to \axiom{2}, as expected, but
}{
\spadpaste{a \free{ad2}}
}
\xtc{
the value of \axiom{b} reflects the change to \axiom{a}.
}{
\spadpaste{b \free{bd ad2}}
}

It is possible to set several variables at the same time
by using
a \spadgloss{tuple} of variables and a tuple of expressions.\footnote{A
\spadgloss{tuple} is a collection of things separated by commas, often
surrounded by parentheses.}

\beginImportant
The syntax for multiple immediate assignments is
\centerline{{{\tt ( \subscriptIt{var}{1}, \subscriptIt{var}{2},
\ldots, \subscriptIt{var}{N} ) := ( \subscriptIt{expr}{1},
\subscriptIt{expr}{2}, \ldots, \subscriptIt{expr}{N} ) }}}
The value returned by an immediate assignment is the value of
\subscriptIt{expr}{N}.
\endImportant

\xtc{
This sets \axiom{x} to \axiom{1} and \axiom{y} to \axiom{2}.
}{
\spadpaste{(x,y) := (1,2) \bound{x}\bound{y}}
}
Multiple immediate assigments are parallel in the sense that the
expressions on the right are all evaluated before any assignments
on the left are made.
However, the order of evaluation of these expressions is undefined.
\xtc{
You can use multiple immediate assignment to swap the
values held by variables.
}{
\spadpaste{(x,y) := (y,x) \free{x y}\bound{swap}}
}
\xtc{
\axiom{x} has the previous value of \axiom{y}.
}{
```

```
\spadpaste{x \free{swap}}
}
\xtc{
\axiom{y} has the previous value of \axiom{x}.
}{
\spadpaste{y \free{swap}}
}

There is no syntactic form for multiple delayed assignments.
See the discussion in
\downlink{‘‘Delayed Assignments vs. Functions with No Arguments’’}
{ugUserDelayPage}
in Section 6.8\ignore{ugUserDelay}
about how Axiom differentiates between delayed assignments and
user functions of no arguments.

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugLangAssignPagePatch1}
\begin{paste}{ugLangAssignPageFull1}{ugLangAssignPageEmpty1}
\pastebutton{ugLangAssignPageFull1}{\hidepaste}
\tab{5}\spadcommand{a := 1\bound{a }}
\indentrel{3}\begin{verbatim}
   (1)  1
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangAssignPageEmpty1}
\begin{paste}{ugLangAssignPageEmpty1}{ugLangAssignPagePatch1}
\pastebutton{ugLangAssignPageEmpty1}{\showpaste}
\tab{5}\spadcommand{a := 1\bound{a }}
\end{paste}\end{patch}

\begin{patch}{ugLangAssignPagePatch2}
\begin{paste}{ugLangAssignPageFull2}{ugLangAssignPageEmpty2}
\pastebutton{ugLangAssignPageFull2}{\hidepaste}
\tab{5}\spadcommand{b := a\free{a }\bound{b }}
\indentrel{3}\begin{verbatim}
   (2)  1
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangAssignPageEmpty2}
\begin{paste}{ugLangAssignPageEmpty2}{ugLangAssignPagePatch2}
\pastebutton{ugLangAssignPageEmpty2}{\showpaste}
\tab{5}\spadcommand{b := a\free{a }\bound{b }}
```

```
\end{paste}\end{patch}

\begin{patch}{ugLangAssignPagePatch3}
\begin{paste}{ugLangAssignPageFull3}{ugLangAssignPageEmpty3}
\pastebutton{ugLangAssignPageFull3}{\hidepaste}
\tab{5}\spadcommand{a := 2\bound{a2 }}
\indentrel{3}\begin{verbatim}
   (3)  2
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangAssignPageEmpty3}
\begin{paste}{ugLangAssignPageEmpty3}{ugLangAssignPagePatch3}
\pastebutton{ugLangAssignPageEmpty3}{\showpaste}
\tab{5}\spadcommand{a := 2\bound{a2 }}
\end{paste}\end{patch}

\begin{patch}{ugLangAssignPagePatch4}
\begin{paste}{ugLangAssignPageFull4}{ugLangAssignPageEmpty4}
\pastebutton{ugLangAssignPageFull4}{\hidepaste}
\tab{5}\spadcommand{b\free{b }}
\indentrel{3}\begin{verbatim}
   (4)  1
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangAssignPageEmpty4}
\begin{paste}{ugLangAssignPageEmpty4}{ugLangAssignPagePatch4}
\pastebutton{ugLangAssignPageEmpty4}{\showpaste}
\tab{5}\spadcommand{b\free{b }}
\end{paste}\end{patch}

\begin{patch}{ugLangAssignPagePatch5}
\begin{paste}{ugLangAssignPageFull5}{ugLangAssignPageEmpty5}
\pastebutton{ugLangAssignPageFull5}{\hidepaste}
\tab{5}\spadcommand{a == 1\bound{ad }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangAssignPageEmpty5}
\begin{paste}{ugLangAssignPageEmpty5}{ugLangAssignPagePatch5}
\pastebutton{ugLangAssignPageEmpty5}{\showpaste}
\tab{5}\spadcommand{a == 1\bound{ad }}
\end{paste}\end{patch}

\begin{patch}{ugLangAssignPagePatch6}
```

```
\begin{paste}{ugLangAssignPageFull6}{ugLangAssignPageEmpty6}
\pastebutton{ugLangAssignPageFull6}{\hidepaste}
\tab{5}\spadcommand{b == a\free{ad }\bound{bd }}
\indentrel{3}\begin{verbatim}
                                                Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangAssignPageEmpty6}
\begin{paste}{ugLangAssignPageEmpty6}{ugLangAssignPagePatch6}
\pastebutton{ugLangAssignPageEmpty6}{\showpaste}
\tab{5}\spadcommand{b == a\free{ad }\bound{bd }}
\end{paste}\end{patch}

\begin{patch}{ugLangAssignPagePatch7}
\begin{paste}{ugLangAssignPageFull7}{ugLangAssignPageEmpty7}
\pastebutton{ugLangAssignPageFull7}{\hidepaste}
\tab{5}\spadcommand{a\free{ad }}
\indentrel{3}\begin{verbatim}
   (7)  1
                                     Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangAssignPageEmpty7}
\begin{paste}{ugLangAssignPageEmpty7}{ugLangAssignPagePatch7}
\pastebutton{ugLangAssignPageEmpty7}{\showpaste}
\tab{5}\spadcommand{a\free{ad }}
\end{paste}\end{patch}

\begin{patch}{ugLangAssignPagePatch8}
\begin{paste}{ugLangAssignPageFull8}{ugLangAssignPageEmpty8}
\pastebutton{ugLangAssignPageFull8}{\hidepaste}
\tab{5}\spadcommand{b\free{bd }}
\indentrel{3}\begin{verbatim}
   (8)  1
                                     Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangAssignPageEmpty8}
\begin{paste}{ugLangAssignPageEmpty8}{ugLangAssignPagePatch8}
\pastebutton{ugLangAssignPageEmpty8}{\showpaste}
\tab{5}\spadcommand{b\free{bd }}
\end{paste}\end{patch}

\begin{patch}{ugLangAssignPagePatch9}
\begin{paste}{ugLangAssignPageFull9}{ugLangAssignPageEmpty9}
\pastebutton{ugLangAssignPageFull9}{\hidepaste}
\tab{5}\spadcommand{a == 2\bound{ad2 }}
```

```
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangAssignPageEmpty9}
\begin{paste}{ugLangAssignPageEmpty9}{ugLangAssignPagePatch9}
\pastebutton{ugLangAssignPageEmpty9}{\showpaste}
\tab{5}\spadcommand{a == 2\bound{ad2 }}
\end{paste}\end{patch}

\begin{patch}{ugLangAssignPagePatch10}
\begin{paste}{ugLangAssignPageFull10}{ugLangAssignPageEmpty10}
\pastebutton{ugLangAssignPageFull10}{\hidepaste}
\tab{5}\spadcommand{a\free{ad2 }}
\indentrel{3}\begin{verbatim}
   (10)  2
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangAssignPageEmpty10}
\begin{paste}{ugLangAssignPageEmpty10}{ugLangAssignPagePatch10}
\pastebutton{ugLangAssignPageEmpty10}{\showpaste}
\tab{5}\spadcommand{a\free{ad2 }}
\end{paste}\end{patch}

\begin{patch}{ugLangAssignPagePatch11}
\begin{paste}{ugLangAssignPageFull11}{ugLangAssignPageEmpty11}
\pastebutton{ugLangAssignPageFull11}{\hidepaste}
\tab{5}\spadcommand{b\free{bd ad2 }}
\indentrel{3}\begin{verbatim}
   (11)  2
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangAssignPageEmpty11}
\begin{paste}{ugLangAssignPageEmpty11}{ugLangAssignPagePatch11}
\pastebutton{ugLangAssignPageEmpty11}{\showpaste}
\tab{5}\spadcommand{b\free{bd ad2 }}
\end{paste}\end{patch}

\begin{patch}{ugLangAssignPagePatch12}
\begin{paste}{ugLangAssignPageFull12}{ugLangAssignPageEmpty12}
\pastebutton{ugLangAssignPageFull12}{\hidepaste}
\tab{5}\spadcommand{(x,y) := (1,2)\bound{x }\bound{y }}
\indentrel{3}\begin{verbatim}
   (12)  2
                                  Type: PositiveInteger
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangAssignPageEmpty12}
\begin{paste}{ugLangAssignPageEmpty12}{ugLangAssignPagePatch12}
\pastebutton{ugLangAssignPageEmpty12}{\showpaste}
\tab{5}\spadcommand{(x,y) := (1,2)\bound{x }\bound{y }}
\end{paste}\end{patch}

\begin{patch}{ugLangAssignPagePatch13}
\begin{paste}{ugLangAssignPageFull13}{ugLangAssignPageEmpty13}
\pastebutton{ugLangAssignPageFull13}{\hidepaste}
\tab{5}\spadcommand{(x,y) := (y,x)\free{x y }\bound{swap }}
\indentrel{3}\begin{verbatim}
   (13)  1
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangAssignPageEmpty13}
\begin{paste}{ugLangAssignPageEmpty13}{ugLangAssignPagePatch13}
\pastebutton{ugLangAssignPageEmpty13}{\showpaste}
\tab{5}\spadcommand{(x,y) := (y,x)\free{x y }\bound{swap }}
\end{paste}\end{patch}

\begin{patch}{ugLangAssignPagePatch14}
\begin{paste}{ugLangAssignPageFull14}{ugLangAssignPageEmpty14}
\pastebutton{ugLangAssignPageFull14}{\hidepaste}
\tab{5}\spadcommand{x\free{swap }}
\indentrel{3}\begin{verbatim}
   (14)  2
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangAssignPageEmpty14}
\begin{paste}{ugLangAssignPageEmpty14}{ugLangAssignPagePatch14}
\pastebutton{ugLangAssignPageEmpty14}{\showpaste}
\tab{5}\spadcommand{x\free{swap }}
\end{paste}\end{patch}

\begin{patch}{ugLangAssignPagePatch15}
\begin{paste}{ugLangAssignPageFull15}{ugLangAssignPageEmpty15}
\pastebutton{ugLangAssignPageFull15}{\hidepaste}
\tab{5}\spadcommand{y\free{swap }}
\indentrel{3}\begin{verbatim}
   (15)  1
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugLangAssignPageEmpty15}
\begin{paste}{ugLangAssignPageEmpty15}{ugLangAssignPagePatch15}
\pastebutton{ugLangAssignPageEmpty15}{\showpaste}
\tab{5}\spadcommand{y\free{swap }}
\end{paste}\end{patch}
```

———-

Blocks

⇒ “notitle” (ugLangIfPage) 9 on page 1761

— **ug05.ht** —

```
\begin{page}{ugLangBlocksPage}{5.2. Blocks}
\beginscroll

%%
%% We should handle tabs in pile correctly but so far we do not.
%%

A \spadgloss{block} is a sequence of expressions evaluated
in the order that they appear, except as modified by control expressions
such as \axiom{break},
\spadkey{break}
\axiom{return},
\spadkey{return}
\axiom{iterate} and
\spadkey{iterate}
\axiom{if-then-else} constructions.
The value of a block is the value of the expression last evaluated
in the block.

To leave a block early, use \axiomSyntax{=>}.
For example, \axiom{i < 0 => x}.
The expression before the \axiomSyntax{=>} must evaluate to
\axiom{true} or \axiom{false}.
The expression following the \axiomSyntax{=>} is the return value
for the block.

A block can be constructed in two ways:
\indent{4}
\beginitems
\item[1. ] the expressions can be separated by semicolons
and the resulting expression surrounded by parentheses, and
\item[2. ] the expressions can be written on succeeding lines with
each line indented the same number of spaces (which must be greater
```

```
than zero).
A block entered in this form is
called a \spadgloss{pile}.
\enditems
\indent{0}
Only the first form is available if you are entering expressions
directly to Axiom.
Both forms are available in {\bf .input} files.

\beginImportant
The syntax for a simple block of expressions entered interactively is
\centerline{{{\tt ( \subscriptIt{expression}{1};
\subscriptIt{expression}{2}; \ldots; \subscriptIt{expression}{N} )}}}
The value returned by a block is the value of an
\axiomSyntax{=>} expression, or \subscriptIt{expression}{N}
if no \axiomSyntax{=>} is encountered.
\endImportant

In {\bf .input} files, blocks can also be written using
\spadglossSee{piles}{pile}.
The examples throughout this book are assumed to come from
{\bf .input} files.

\xtc{
In this example, we assign a rational number to \axiom{a} using a block
consisting of three expressions.
This block is written as a pile.
Each expression in the pile has the same indentation, in this case two
spaces to the right of the first line.
}{
\begin{spadsrc}
a :=
  i := gcd(234,672)
  i := 3*i**5 - i + 1
  1 / i
\end{spadsrc}
}
\xtc{
Here is the same block written on one line.
This is how you are required to enter it at the input prompt.
}{
\spadpaste{a := (i := gcd(234,672); i := 3*i**5 - i + 1; 1 / i)}
}
\xtc{
Blocks can be used to put several expressions on one line.
The value returned is that of the last expression.
}{
\spadpaste{(a := 1; b := 2; c := 3; [a,b,c]) \bound{a b c}}
}
```

```
Axiom gives you two ways of writing a block and the
preferred way in an {\bf .input} file is to use a pile.
Roughly speaking, a pile is
a block whose constituent expressions are indented the same amount.
You begin a pile by starting a new line for the first expression,
indenting it to the right of the previous line.
You then enter the second expression on a new line, vertically aligning
it with the first line. And so on.
If you need to enter an inner pile, further indent its lines to the right
of the outer pile.
Axiom knows where a pile ends.
It ends when a subsequent line is indented to the left of the pile or
the end of the file.

\xtc{
Blocks can be used to perform several steps before an assignment
(immediate or delayed) is made.
}{
\begin{spadsrc}[\free{a b}]
d :=
   c := a**2 + b**2
   sqrt(c * 1.3)
\end{spadsrc}
}
\xtc{
Blocks can be used in the arguments to functions.
(Here \axiom{h} is assigned \axiom{2.1 + 3.5}.)
}{
\begin{spadsrc}[\bound{h}]
h := 2.1 +
   1.0
   3.5
\end{spadsrc}
}
\xtc{
Here the second argument to \axiomFun{eval} is \axiom{x = z}, where
the value of \axiom{z} is computed in the first line of the block
starting on the second line.
}{
\begin{spadsrc}
eval(x**2 - x*y**2,
     z := %pi/2.0 - exp(4.1)
     x = z
   )
\end{spadsrc}
}
\xtc{
Blocks can be used in the clauses of \axiom{if-then-else}
expressions (see \downlink{``if-then-else''}{ugLangIfPage}
in Section 5.3\ignore{ugLangIf}).
```

```
}{
\spadpaste{if h > 3.1 then 1.0 else (z := cos(h); max(z,0.5)) \free{h}}
}
\xtc{
This is the pile version of the last block.
}{
\begin{spadsrc}[\free{h}]
if h > 3.1 then
    1.0
  else
    z := cos(h)
    max(z,0.5)
\end{spadsrc}
}
\xtc{
Blocks can be nested.
}{
\spadpaste{
a := (b := factorial(12); c := (d := eulerPhi(22); factorial(d));b+c)}
}
\xtc{
This is the pile version of the last block.
}{
\begin{spadsrc}
a :=
  b := factorial(12)
  c :=
    d := eulerPhi(22)
    factorial(d)
  b+c
\end{spadsrc}
}

\xtc{
Since \axiom{c + d} does equal \axiom{3628855}, \axiom{a} has the value
of \axiom{c} and the last line is never evaluated.
}{
\begin{spadsrc}
a :=
  c := factorial 10
  d := fibonacci 10
  c + d = 3628855 => c
  d
\end{spadsrc}
}

\endscroll
\autobuttons
\end{page}
```

```
\begin{patch}{ugLangBlocksPagePatch1}
\begin{paste}{ugLangBlocksPageFull1}{ugLangBlocksPageEmpty1}
\pastebutton{ugLangBlocksPageFull1}{\hidepaste}
\tab{5}\spadcommand{a :=
  i := gcd(234,672)
  i := 3*i**5 - i + 1
  1 / i
}
\indentrel{3}\begin{verbatim}
          1
   (1)
        23323
                                 Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangBlocksPageEmpty1}
\begin{paste}{ugLangBlocksPageEmpty1}{ugLangBlocksPagePatch1}
\pastebutton{ugLangBlocksPageEmpty1}{\showpaste}
\tab{5}\spadcommand{a :=
  i := gcd(234,672)
  i := 3*i**5 - i + 1
  1 / i
}
\end{paste}\end{patch}

\begin{patch}{ugLangBlocksPagePatch2}
\begin{paste}{ugLangBlocksPageFull2}{ugLangBlocksPageEmpty2}
\pastebutton{ugLangBlocksPageFull2}{\hidepaste}
\tab{5}\spadcommand{a := (i := gcd(234,672); i := 3*i**5 - i + 1; 1 / i)}
\indentrel{3}\begin{verbatim}
          1
   (2)
        23323
                                 Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangBlocksPageEmpty2}
\begin{paste}{ugLangBlocksPageEmpty2}{ugLangBlocksPagePatch2}
\pastebutton{ugLangBlocksPageEmpty2}{\showpaste}
\tab{5}\spadcommand{a := (i := gcd(234,672); i := 3*i**5 - i + 1; 1 / i)}
\end{paste}\end{patch}

\begin{patch}{ugLangBlocksPagePatch3}
\begin{paste}{ugLangBlocksPageFull3}{ugLangBlocksPageEmpty3}
\pastebutton{ugLangBlocksPageFull3}{\hidepaste}
\tab{5}\spadcommand{(a := 1; b := 2; c := 3; [a,b,c])\bound{a b c }}
\indentrel{3}\begin{verbatim}
   (3)  [1,2,3]
```

```
                              Type: List PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangBlocksPageEmpty3}
\begin{paste}{ugLangBlocksPageEmpty3}{ugLangBlocksPagePatch3}
\pastebutton{ugLangBlocksPageEmpty3}{\showpaste}
\tab{5}\spadcommand{(a := 1; b := 2; c := 3; [a,b,c])\bound{a b c }}
\end{paste}\end{patch}

\begin{patch}{ugLangBlocksPagePatch4}
\begin{paste}{ugLangBlocksPageFull4}{ugLangBlocksPageEmpty4}
\pastebutton{ugLangBlocksPageFull4}{\hidepaste}
\tab{5}\spadcommand{d :=
   c := a**2 + b**2
   sqrt(c * 1.3)
\free{a b }}
\indentrel{3}\begin{verbatim}
   (4)  2.5495097567 96392415
                                             Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangBlocksPageEmpty4}
\begin{paste}{ugLangBlocksPageEmpty4}{ugLangBlocksPagePatch4}
\pastebutton{ugLangBlocksPageEmpty4}{\showpaste}
\tab{5}\spadcommand{d :=
   c := a**2 + b**2
   sqrt(c * 1.3)
\free{a b }}
\end{paste}\end{patch}

\begin{patch}{ugLangBlocksPagePatch5}
\begin{paste}{ugLangBlocksPageFull5}{ugLangBlocksPageEmpty5}
\pastebutton{ugLangBlocksPageFull5}{\hidepaste}
\tab{5}\spadcommand{h := 2.1 +
   1.0
   3.5
\bound{h }}
\indentrel{3}\begin{verbatim}
   (5)  5.6
                                             Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangBlocksPageEmpty5}
\begin{paste}{ugLangBlocksPageEmpty5}{ugLangBlocksPagePatch5}
\pastebutton{ugLangBlocksPageEmpty5}{\showpaste}
\tab{5}\spadcommand{h := 2.1 +
   1.0
```

```
   3.5
\bound{h }}
\end{paste}\end{patch}

\begin{patch}{ugLangBlocksPagePatch6}
\begin{paste}{ugLangBlocksPageFull6}{ugLangBlocksPageEmpty6}
\pastebutton{ugLangBlocksPageFull6}{\hidepaste}
\tab{5}\spadcommand{eval(x**2 - x*y**2,
     z := \%pi/2.0 - exp(4.1)
     x = z
   )
}
\indentrel{3}\begin{verbatim}
   (6)
                           2
   58.7694912705 67072878 y  + 3453.8531042012 59382
                                  Type: Polynomial Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangBlocksPageEmpty6}
\begin{paste}{ugLangBlocksPageEmpty6}{ugLangBlocksPagePatch6}
\pastebutton{ugLangBlocksPageEmpty6}{\showpaste}
\tab{5}\spadcommand{eval(x**2 - x*y**2,
     z := \%pi/2.0 - exp(4.1)
     x = z
   )
}
\end{paste}\end{patch}

\begin{patch}{ugLangBlocksPagePatch7}
\begin{paste}{ugLangBlocksPageFull7}{ugLangBlocksPageEmpty7}
\pastebutton{ugLangBlocksPageFull7}{\hidepaste}
\tab{5}\spadcommand{if h > 3.1 then 1.0 else (z := cos(h); max(z,0.5))\free{h }}
\indentrel{3}\begin{verbatim}
   (7)  1.0
                                             Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangBlocksPageEmpty7}
\begin{paste}{ugLangBlocksPageEmpty7}{ugLangBlocksPagePatch7}
\pastebutton{ugLangBlocksPageEmpty7}{\showpaste}
\tab{5}\spadcommand{if h > 3.1 then 1.0 else (z := cos(h); max(z,0.5))\free{h }}
\end{paste}\end{patch}

\begin{patch}{ugLangBlocksPagePatch8}
\begin{paste}{ugLangBlocksPageFull8}{ugLangBlocksPageEmpty8}
\pastebutton{ugLangBlocksPageFull8}{\hidepaste}
\tab{5}\spadcommand{if h > 3.1 then
```

```
    1.0
  else
    z := cos(h)
    max(z,0.5)
\free{h }}
\indentrel{3}\begin{verbatim}
   (8)  1.0
                                            Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangBlocksPageEmpty8}
\begin{paste}{ugLangBlocksPageEmpty8}{ugLangBlocksPagePatch8}
\pastebutton{ugLangBlocksPageEmpty8}{\showpaste}
\tab{5}\spadcommand{if h > 3.1 then
    1.0
  else
    z := cos(h)
    max(z,0.5)
\free{h }}
\end{paste}\end{patch}

\begin{patch}{ugLangBlocksPagePatch9}
\begin{paste}{ugLangBlocksPageFull9}{ugLangBlocksPageEmpty9}
\pastebutton{ugLangBlocksPageFull9}{\hidepaste}
\tab{5}\spadcommand{a := (b := factorial(12); c := (d := eulerPhi(22); factorial(d));b+c)}
\indentrel{3}\begin{verbatim}
   (9)  482630400
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangBlocksPageEmpty9}
\begin{paste}{ugLangBlocksPageEmpty9}{ugLangBlocksPagePatch9}
\pastebutton{ugLangBlocksPageEmpty9}{\showpaste}
\tab{5}\spadcommand{a := (b := factorial(12); c := (d := eulerPhi(22); factorial(d));b+c)}
\end{paste}\end{patch}

\begin{patch}{ugLangBlocksPagePatch10}
\begin{paste}{ugLangBlocksPageFull10}{ugLangBlocksPageEmpty10}
\pastebutton{ugLangBlocksPageFull10}{\hidepaste}
\tab{5}\spadcommand{a :=
  b := factorial(12)
  c :=
    d := eulerPhi(22)
    factorial(d)
  b+c
}
\indentrel{3}\begin{verbatim}
   (10)  482630400
```

```
                                    Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangBlocksPageEmpty10}
\begin{paste}{ugLangBlocksPageEmpty10}{ugLangBlocksPagePatch10}
\pastebutton{ugLangBlocksPageEmpty10}{\showpaste}
\tab{5}\spadcommand{a :=
  b := factorial(12)
  c :=
    d := eulerPhi(22)
    factorial(d)
  b+c
}
\end{paste}\end{patch}

\begin{patch}{ugLangBlocksPagePatch11}
\begin{paste}{ugLangBlocksPageFull11}{ugLangBlocksPageEmpty11}
\pastebutton{ugLangBlocksPageFull11}{\hidepaste}
\tab{5}\spadcommand{a :=
  c := factorial 10
  d := fibonacci 10
  c + d = 3628855 => c
  d
}
\indentrel{3}\begin{verbatim}
   (11)  3628800
                                    Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangBlocksPageEmpty11}
\begin{paste}{ugLangBlocksPageEmpty11}{ugLangBlocksPagePatch11}
\pastebutton{ugLangBlocksPageEmpty11}{\showpaste}
\tab{5}\spadcommand{a :=
  c := factorial 10
  d := fibonacci 10
  c + d = 3628855 => c
  d
}
\end{paste}\end{patch}
```

if-then-else

⇒ "notitle" (ugTypesResolvePage) 7 on page 1696

⇒ “notitle” (ugTypesPkgCallPage) 7 on page 1686

— ug05.ht —

```
\begin{page}{ugLangIfPage}{5.3. if-then-else}
\beginscroll

Like many other programming languages, Axiom uses the three
keywords \spadkey{if} \axiom{if, then} \spadkey{then} and \axiom{else}
\spadkey{else} to form
conditional expressions.
The \axiom{else} part of the conditional is optional.
The expression between the \axiom{if} and \axiom{then} keywords
is a
\spadgloss{predicate}: an expression that evaluates to or is
convertible to either {\tt true} or {\tt false}, that is,
a \axiomType{Boolean}.

\beginImportant The syntax for conditional expressions is
\centerline{{{\tt if {\it predicate} then \subscriptIt{expression}{1}
else \subscriptIt{expression}{2}}}} where the \axiom{else}
\subscriptIt{\it expression}{2} part is optional.  The value returned
from a conditional expression is \subscriptIt{\it expression}{1} if
the predicate evaluates to \axiom{true} and \subscriptIt{\it
expression}{2} otherwise.  If no \axiom{else} clause is given, the
value is always \void{}.  \endImportant

An \axiom{if-then-else} expression always returns a value.
If the
\axiom{else} clause is missing then the entire expression returns
\void{}.
If both clauses are present, the type of the value returned by \axiom{if}
is obtained by resolving the types of the values of the two clauses.
See \downlink{``Resolving Types''}{ugTypesResolvePage}
in Section 2.10\ignore{ugTypesResolve}
for more information.

The predicate must evaluate to, or be convertible to, an object of type
\axiomType{Boolean}: {\tt true} or {\tt false}.
By default, the equal sign \spadopFrom{=}{Equation} creates
an equation.
\xtc{
This is an equation.
In particular, it is an object of type
\axiomType{Equation Polynomial Integer}.
}{
\spadpaste{x + 1 = y}
}
However, for predicates in \axiom{if} expressions, Axiom
places a default target type of \axiomType{Boolean} on the
predicate and equality testing is performed.
```

```
Thus you need not qualify the \axiomSyntax{=} in any way.
In other contexts you may need to tell Axiom that you want
to test for equality rather than create an equation.
In those cases, use \axiomSyntax{@} and a target type of
\axiomType{Boolean}.
See \downlink{``Package Calling and Target Types''}
{ugTypesPkgCallPage} in Section 2.9
\ignore{ugTypesPkgCall} for more information.

The compound symbol meaning ``not equal'' in Axiom is
``\texht{$\sim =$}{\axiom{~=}}''.
This can be used directly without a package call or a target
specification. The expression
\axiom{a} \texht{$\sim =$}{\axiom{~=}} \axiom{b} is directly
translated into \axiom{not (a = b)}.

Many other functions have return values of type \axiomType{Boolean}.
These include \axiom{<}, \axiom{<=}, \axiom{>},
\axiom{>=}, \texht{$\sim =$}{\axiom{~=}} and \axiom{member?}.
By convention, operations with names ending in \axiomSyntax{?}
return \axiomType{Boolean} values.

The usual rules for piles are suspended for conditional expressions.
In {\bf .input} files, the \axiom{then} and
\axiom{else} keywords can begin in the same column as the corresponding
\axiom{if} but may also appear to the right.
Each of the following styles of writing \axiom{if-then-else}
expressions is acceptable:
\begin{verbatim}
if i>0 then output("positive") else output("nonpositive")

if i > 0 then output("positive")
  else output("nonpositive")

if i > 0 then output("positive")
else output("nonpositive")

if i > 0
then output("positive")
else output("nonpositive")

if i > 0
  then output("positive")
  else output("nonpositive")
\end{verbatim}

A block can follow the \axiom{then} or \axiom{else} keywords.
In the following two assignments to \axiom{a},
the \axiom{then} and \axiom{else}
clauses each are followed by two-line piles.
```

```
The value returned in each is the value of the second line.

\begin{verbatim}
a :=
  if i > 0 then
    j := sin(i * pi())
    exp(j + 1/j)
  else
    j := cos(i * 0.5 * pi())
    log(abs(j)**5 + 1)

a :=
  if i > 0
    then
      j := sin(i * pi())
      exp(j + 1/j)
    else
      j := cos(i * 0.5 * pi())
      log(abs(j)**5 + 1)
\end{verbatim}
These are both equivalent to the following:
\begin{verbatim}
a :=
  if i > 0 then (j := sin(i * pi()); exp(j + 1/j))
  else (j := cos(i * 0.5 * pi()); log(abs(j)**5 + 1))
\end{verbatim}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugLangIfPagePatch1}
\begin{paste}{ugLangIfPageFull1}{ugLangIfPageEmpty1}
\pastebutton{ugLangIfPageFull1}{\hidepaste}
\tab{5}\spadcommand{x + 1 = y}
\indentrel{3}\begin{verbatim}
   (1)  x + 1= y
                      Type: Equation Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangIfPageEmpty1}
\begin{paste}{ugLangIfPageEmpty1}{ugLangIfPagePatch1}
\pastebutton{ugLangIfPageEmpty1}{\showpaste}
\tab{5}\spadcommand{x + 1 = y}
\end{paste}\end{patch}
```

Loops

⇒ “notitle” (ugLangLoopsCompIntPage) 9 on page 1766
⇒ “notitle” (ugLangLoopsReturnPage) 9 on page 1767
⇒ “notitle” (ugLangLoopsBreakPage) 9 on page 1770
⇒ “notitle” (ugLangLoopsBreakVsPage) 9 on page 1773
⇒ “notitle” (ugLangLoopsBreakMorePage) 9 on page 1774
⇒ “notitle” (ugLangLoopsIteratePage) 9 on page 1782
⇒ “notitle” (ugLangLoopsWhilePage) 9 on page 1784
⇒ “notitle” (ugLangLoopsForInPage) 9 on page 1790
⇒ “notitle” (ugLangLoopsForInNMPage) 9 on page 1791
⇒ “notitle” (ugLangLoopsForInNMSPage) 9 on page 1795
⇒ “notitle” (ugLangLoopsForInNPage) 9 on page 1797
⇒ “notitle” (ugLangLoopsForInXLPage) 9 on page 1798
⇒ “notitle” (ugLangLoopsForInPredPage) 9 on page 1800
⇒ “notitle” (ugLangLoopsParPage) 9 on page 1802

— **ug05.ht** —

```
\begin{page}{ugLangLoopsPage}{5.4. Loops}
\beginscroll

A \spadgloss{loop} is an expression that contains another expression,
called the {\it loop body}, which is to be evaluated zero or more
times.
All loops contain the \axiom{repeat} keyword and return \void{}.
Loops can contain inner loops to any depth.

\beginImportant
The most basic loop is of the form
\centerline{{\axiom{repeat} {\it loopBody}}}
Unless {\it loopBody} contains a \axiom{break} or \axiom{return}
expression, the loop repeats forever.
The value returned by the loop is \void{}.
\endImportant

\beginmenu
    \menudownlink{{5.4.1. Compiling vs. Interpreting Loops}}
{ugLangLoopsCompIntPage}
    \menudownlink{{5.4.2. return in Loops}}{ugLangLoopsReturnPage}
    \menudownlink{{5.4.3. break in Loops}}{ugLangLoopsBreakPage}
    \menudownlink{{5.4.4. break vs. {\tt =>} in Loop Bodies}}
{ugLangLoopsBreakVsPage}
    \menudownlink{{5.4.5. More Examples of break}}
{ugLangLoopsBreakMorePage}
    \menudownlink{{5.4.6. iterate in Loops}}{ugLangLoopsIteratePage}
    \menudownlink{{5.4.7. while Loops}}{ugLangLoopsWhilePage}
    \menudownlink{{5.4.8. for Loops}}{ugLangLoopsForInPage}
```

```
    \menudownlink{{5.4.9. for i in n..m repeat}}
{ugLangLoopsForInNMPage}
    \menudownlink{{5.4.10. for i in n..m by s repeat}}
{ugLangLoopsForInNMSPage}
    \menudownlink{{5.4.11. for i in n.. repeat}}{ugLangLoopsForInNPage}
    \menudownlink{{5.4.12. for x in l repeat}}{ugLangLoopsForInXLPage}
    \menudownlink{{5.4.13. ‘‘Such that’’ Predicates}}
{ugLangLoopsForInPredPage}
    \menudownlink{{5.4.14. Parallel Iteration}}{ugLangLoopsParPage}
\endmenu
\endscroll
\autobuttons
\end{page}
```

———-

Compiling vs. Interpreting Loops

⇒ “notitle” (ugUserCompIntPage) 10 on page 1858

— **ug05.ht** —

```
\begin{page}{ugLangLoopsCompIntPage}
{5.4.1. Compiling vs. Interpreting Loops}
\beginscroll

Axiom tries to determine completely the type of every
object in a loop and then to translate the loop body to LISP or even to
machine code.
This translation is called \spadglossSee{compilation}{compiler}.

If Axiom decides that it cannot compile the loop, it issues a
message stating the problem and then the following message:
%
\centerline{
{{\bf We will attempt to step through and interpret the code.}}}
%
It is still possible that Axiom can evaluate the loop but in
\spadgloss{interpret-code mode}.
See \downlink{‘‘Compiling vs. Interpreting’’}{ugUserCompIntPage}
in Section 6.10\ignore{ugUserCompInt} where
this is discussed in terms
of compiling versus interpreting functions.

\endscroll
\autobuttons
\end{page}
```

———-

return in Loops

⇒ “notitle” (ugUserBlocksPage) 10 on page 1898

— ug05.ht —

```
\begin{page}{ugLangLoopsReturnPage}{5.4.2. return in Loops}
\beginscroll

A \axiom{return} expression is used to exit a function with
a particular value.
In particular, if a \axiom{return} is in a loop within the
\spadkey{return}
function, the loop is terminated whenever the \axiom{return}
is evaluated.
%> This is a bug! The compiler should never accept allow
%> Void to be the return type of a function when it has to use
%> resolve to determine it.
\xtc{
Suppose we start with this.
}{
\begin{spadsrc}[\bound{f}]
f() ==
  i := 1
  repeat
    if factorial(i) > 1000 then return i
    i := i + 1
\end{spadsrc}
}
\xtc{
When \axiom{factorial(i)} is big enough, control passes from
inside the loop all the way outside the function, returning the
value of \axiom{i} (or so we think).
}{
\spadpaste{f() \free{f}}
}

What went wrong?
Isn't it obvious that this function should return an integer?
Well, Axiom makes no attempt to analyze the structure of a
loop to determine if it always returns a value because, in
general, this is impossible.
So Axiom has this simple rule: the type of the function is
determined by the type of its body, in this case a block.
The normal value of a block is the value of its last expression,
in this case, a loop.
And the value of every loop is \void{}!
```

```
So the return type of \userfun{f} is \axiomType{Void}.

There are two ways to fix this.
The best way is for you to tell Axiom what the return type
of \axiom{f} is.
You do this by giving \axiom{f} a declaration \axiom{f: () ->
Integer} prior to calling for its value.
This tells Axiom: ``trust me---an integer is returned.''
We'll explain more about this in the next chapter.
Another clumsy way is to add a dummy expression as follows.

\xtc{
Since we want an integer, let's stick in a dummy final expression that is
an integer and will never be evaluated.
}{
\begin{spadsrc}[\bound{f1}]
f() ==
  i := 1
  repeat
    if factorial(i) > 1000 then return i
    i := i + 1
  0
\end{spadsrc}
}
\xtc{
When we try \userfun{f} again we get what we wanted.
See
\downlink{``Functions Defined with Blocks''}{ugUserBlocksPage}
in Section 6.15\ignore{ugUserBlocks}
for more information.
}{
\spadpaste{f() \free{f1}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugLangLoopsReturnPagePatch1}
\begin{paste}{ugLangLoopsReturnPageFull1}{ugLangLoopsReturnPageEmpty1}
\pastebutton{ugLangLoopsReturnPageFull1}{\hidepaste}
\tab{5}\spadcommand{f() ==
  i := 1
  repeat
    if factorial(i) > 1000 then return i
    i := i + 1
\bound{f }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsReturnPageEmpty1}
\begin{paste}{ugLangLoopsReturnPageEmpty1}{ugLangLoopsReturnPagePatch1}
\pastebutton{ugLangLoopsReturnPageEmpty1}{\showpaste}
\tab{5}\spadcommand{f() ==
  i := 1
  repeat
    if factorial(i) > 1000 then return i
    i := i + 1
\bound{f }}
\end{paste}\end{patch}

\begin{patch}{ugLangLoopsReturnPagePatch2}
\begin{paste}{ugLangLoopsReturnPageFull2}{ugLangLoopsReturnPageEmpty2}
\pastebutton{ugLangLoopsReturnPageFull2}{\hidepaste}
\tab{5}\spadcommand{f()\free{f }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsReturnPageEmpty2}
\begin{paste}{ugLangLoopsReturnPageEmpty2}{ugLangLoopsReturnPagePatch2}
\pastebutton{ugLangLoopsReturnPageEmpty2}{\showpaste}
\tab{5}\spadcommand{f()\free{f }}
\end{paste}\end{patch}

\begin{patch}{ugLangLoopsReturnPagePatch3}
\begin{paste}{ugLangLoopsReturnPageFull3}{ugLangLoopsReturnPageEmpty3}
\pastebutton{ugLangLoopsReturnPageFull3}{\hidepaste}
\tab{5}\spadcommand{f() ==
  i := 1
  repeat
    if factorial(i) > 1000 then return i
    i := i + 1
  0
\bound{f1 }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsReturnPageEmpty3}
\begin{paste}{ugLangLoopsReturnPageEmpty3}{ugLangLoopsReturnPagePatch3}
\pastebutton{ugLangLoopsReturnPageEmpty3}{\showpaste}
\tab{5}\spadcommand{f() ==
  i := 1
  repeat
    if factorial(i) > 1000 then return i
```

```
    i := i + 1
  0
\bound{f1 }}
\end{paste}\end{patch}

\begin{patch}{ugLangLoopsReturnPagePatch4}
\begin{paste}{ugLangLoopsReturnPageFull4}{ugLangLoopsReturnPageEmpty4}
\pastebutton{ugLangLoopsReturnPageFull4}{\hidepaste}
\tab{5}\spadcommand{f()\free{f1 }}
\indentrel{3}\begin{verbatim}
   (4)  7
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsReturnPageEmpty4}
\begin{paste}{ugLangLoopsReturnPageEmpty4}{ugLangLoopsReturnPagePatch4}
\pastebutton{ugLangLoopsReturnPageEmpty4}{\showpaste}
\tab{5}\spadcommand{f()\free{f1 }}
\end{paste}\end{patch}
```

———-

break in Loops

⇒ “notitle” (ugLangLoopsReturnPage) 9 on page 1767

— **ug05.ht** —

```
\begin{page}{ugLangLoopsBreakPage}{5.4.3. break in Loops}
\beginscroll

The \axiom{break} keyword is often more useful
\spadkey{break}
in terminating
a loop.
%>  and more in keeping with the ideas of structured programming.
A \axiom{break} causes control to transfer to the expression
immediately following the loop.
As loops always return \void{},
you cannot return a value with \axiom{break}.
That is, \axiom{break} takes no argument.

\xtc{
This example is a modification of the last example in
\texthtt{the previous section}{
\downlink{``return in Loops''}{ugLangLoopsReturnPage}
in Section 5.4.2\ignore{ugLangLoopsReturn}}.
```

```
Instead of using \axiom{return}, we'll use \axiom{break}.
}{
\begin{spadsrc}[\bound{f1}]
f() ==
  i := 1
  repeat
    if factorial(i) > 1000 then break
    i := i + 1
  i
\end{spadsrc}
}
\xtc{
The loop terminates when \axiom{factorial(i)} gets big enough,
the last line of the function evaluates to the corresponding ``good''
value of \axiom{i}, and the function terminates, returning that value.
}{
\spadpaste{f() \free{f1}}
}
\xtc{
You can only use \axiom{break} to terminate the evaluation of one loop.
Let's consider a loop within a loop, that is, a loop with a nested loop.
First, we initialize two counter variables.
}{
\spadpaste{(i,j) := (1, 1) \bound{i}\bound{j}}
}
\xtc{
Nested loops must have multiple \axiom{break}
expressions at the appropriate nesting level.
How would you rewrite this so \axiom{(i + j) > 10} is only evaluated once?
}{
\begin{spadsrc}[\free{i j}]
repeat
  repeat
    if (i + j) > 10 then break
    j := j + 1
  if (i + j) > 10 then break
  i := i + 1
\end{spadsrc}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugLangLoopsBreakPagePatch1}
\begin{paste}{ugLangLoopsBreakPageFull1}{ugLangLoopsBreakPageEmpty1}
\pastebutton{ugLangLoopsBreakPageFull1}{\hidepaste}
\tab{5}\spadcommand{f() ==
  i := 1
  repeat
```

```
    if factorial(i) > 1000 then break
    i := i + 1
  i
\bound{f1 }}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsBreakPageEmpty1}
\begin{paste}{ugLangLoopsBreakPageEmpty1}{ugLangLoopsBreakPagePatch1}
\pastebutton{ugLangLoopsBreakPageEmpty1}{\showpaste}
\tab{5}\spadcommand{f() ==
  i := 1
  repeat
    if factorial(i) > 1000 then break
    i := i + 1
  i
\bound{f1 }}
\end{paste}\end{patch}

\begin{patch}{ugLangLoopsBreakPagePatch2}
\begin{paste}{ugLangLoopsBreakPageFull2}{ugLangLoopsBreakPageEmpty2}
\pastebutton{ugLangLoopsBreakPageFull2}{\hidepaste}
\tab{5}\spadcommand{f()\free{f1 }}
\indentrel{3}\begin{verbatim}
   (2)  7
                                 Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsBreakPageEmpty2}
\begin{paste}{ugLangLoopsBreakPageEmpty2}{ugLangLoopsBreakPagePatch2}
\pastebutton{ugLangLoopsBreakPageEmpty2}{\showpaste}
\tab{5}\spadcommand{f()\free{f1 }}
\end{paste}\end{patch}

\begin{patch}{ugLangLoopsBreakPagePatch3}
\begin{paste}{ugLangLoopsBreakPageFull3}{ugLangLoopsBreakPageEmpty3}
\pastebutton{ugLangLoopsBreakPageFull3}{\hidepaste}
\tab{5}\spadcommand{(i,j) := (1, 1)\bound{i }\bound{j }}
\indentrel{3}\begin{verbatim}
   (3)  1
                                 Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsBreakPageEmpty3}
\begin{paste}{ugLangLoopsBreakPageEmpty3}{ugLangLoopsBreakPagePatch3}
\pastebutton{ugLangLoopsBreakPageEmpty3}{\showpaste}
```

```
\tab{5}\spadcommand{(i,j) := (1, 1)\bound{i }\bound{j }}
\end{paste}\end{patch}

\begin{patch}{ugLangLoopsBreakPagePatch4}
\begin{paste}{ugLangLoopsBreakPageFull4}{ugLangLoopsBreakPageEmpty4}
\pastebutton{ugLangLoopsBreakPageFull4}{\hidepaste}
\tab{5}\spadcommand{repeat
  repeat
    if (i + j) > 10 then break
    j := j + 1
  if (i + j) > 10 then break
  i := i + 1
\free{i j }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsBreakPageEmpty4}
\begin{paste}{ugLangLoopsBreakPageEmpty4}{ugLangLoopsBreakPagePatch4}
\pastebutton{ugLangLoopsBreakPageEmpty4}{\showpaste}
\tab{5}\spadcommand{repeat
  repeat
    if (i + j) > 10 then break
    j := j + 1
  if (i + j) > 10 then break
  i := i + 1
\free{i j }}
\end{paste}\end{patch}
```

———-

break vs. => in Loop Bodies

— **ug05.ht** —

```
\begin{page}{ugLangLoopsBreakVsPage}
{5.4.4. break vs. {\tt =>} in Loop Bodies}
\beginscroll

Compare the following two loops:

\begin{verbatim}
i := 1                            i := 1
repeat                            repeat
  i := i + 1                        i := i + 1
  i > 3 => i                        if i > 3 then break
```

```
  output(i)                          output(i)
\end{verbatim}

In the example on the left, the values
\mathOrSpad{2} and \mathOrSpad{3} for \axiom{i} are displayed
but then the \axiomSyntax{=>} does not allow control to reach the call to
\axiomFunFrom{output}{OutputForm} again.
The loop will not terminate
until you run out of space or interrupt the execution.
The variable \axiom{i} will continue to be incremented because
the \axiomSyntax{=>} only means to leave the {\it block,} not the loop.

In the example on the right,
upon reaching \mathOrSpad{4}, the \axiom{break} will be
executed, and both the block and the loop will terminate.
This is one of the reasons why both \axiomSyntax{=>} and \axiom{break} are
provided.
Using a \axiom{while} clause (see below) with the \axiomSyntax{=>}
\spadkey{while}
lets you simulate the action of \axiom{break}.

\endscroll
\autobuttons
\end{page}
```

———-

More Examples of break

⇒ “notitle” (ugLangLoopsForInPage) 9 on page 1790

— **ug05.ht** —

```
\begin{page}{ugLangLoopsBreakMorePage}{5.4.5. More Examples of break}
\beginscroll

Here we give four examples of \axiom{repeat} loops that
terminate when a value exceeds a given bound.

\texht{\vskip 1pc}{}
\xtc{
First, initialize \axiom{i} as the loop counter.
}{
\spadpaste{i := 0 \bound{i}}
}
\xtc{
Here is the first loop.
When the square of \axiom{i} exceeds \axiom{100}, the loop terminates.
```

```
}{
\begin{spadsrc}[\free{i}\bound{i1}]
repeat
  i := i + 1
  if i**2 > 100 then break
\end{spadsrc}
}
\xtc{
Upon completion, \axiom{i} should have the value \axiom{11}.
}{
\spadpaste{i \free{i1}}
}
%
%
\xtc{
Do the same thing except use \axiomSyntax{=>} instead
an \axiom{if-then} expression.
}{
\spadpaste{i := 0 \bound{i2}}
}
\xtc{
}{
\begin{spadsrc}[\free{i2}\bound{i3}]
repeat
  i := i + 1
  i**2 > 100 => break
\end{spadsrc}
}
\xtc{
}{
\spadpaste{i \free{i3}}
}
%
%
\xtc{
As a third example, we use a simple loop to compute \axiom{n!}.
}{
\spadpaste{(n, i, f) := (100, 1, 1) \bound{n}\bound{i4}\bound{f}}
}
\xtc{
Use \axiom{i} as the iteration variable and \axiom{f}
to compute the factorial.
}{
\begin{spadsrc}[\bound{f1}\bound{i5}\free{f i4 n}]
repeat
  if i > n then break
  f := f * i
  i := i + 1
\end{spadsrc}
}
```

```
\xtc{
Look at the value of \axiom{f}.
}{
\spadpaste{f \free{f1}}
}
%
%
\xtc{
Finally, we show an example of nested loops.
First define a four by four matrix.
}{
\spadpaste{m :=
matrix [[21,37,53,14], [8,-24,22,-16], [2,10,15,14],
[26,33,55,-13]] \bound{m2}}
}
\xtc{
Next, set row counter \axiom{r} and column counter \axiom{c} to
\mathOrSpad{1}.
Note: if we were writing a function, these would all be local
variables rather than global workspace variables.
}{
\spadpaste{(r, c) := (1, 1) \bound{r}\bound{c}}
}
\xtc{
Also, let \axiom{lastrow} and
\axiom{lastcol} be the final row and column index.
}{
\spadpaste{(lastrow, lastcol) := (nrows(m), ncols(m))
\bound{lastrow}\bound{lastcol}\free{m2}}
}
%
\xtc{
Scan the rows looking for the first negative element.
We remark that you can reformulate this example in a better, more
concise form by using a \axiom{for} clause with \axiom{repeat}.
See
\downlink{``for Loops''}{ugLangLoopsForInPage}
in Section 5.4.8\ignore{ugLangLoopsForIn}
for more information.
}{
\begin{spadsrc}[\free{m2 r c lastrow lastcol}]
repeat
  if r > lastrow then break
  c := 1
  repeat
    if c > lastcol then break
    if elt(m,r,c) < 0 then
      output [r, c, elt(m,r,c)]
      r := lastrow
      break     -- don't look any further
```

```
    c := c + 1
  r := r + 1
\end{spadsrc}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugLangLoopsBreakMorePagePatch1}
\begin{paste}{ugLangLoopsBreakMorePageFull1}{ugLangLoopsBreakMorePageEmpty1}
\pastebutton{ugLangLoopsBreakMorePageFull1}{\hidepaste}
\tab{5}\spadcommand{i := 0\bound{i }}
\indentrel{3}\begin{verbatim}
   (1)  0
                              Type: NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsBreakMorePageEmpty1}
\begin{paste}{ugLangLoopsBreakMorePageEmpty1}{ugLangLoopsBreakMorePagePatch1}
\pastebutton{ugLangLoopsBreakMorePageEmpty1}{\showpaste}
\tab{5}\spadcommand{i := 0\bound{i }}
\end{paste}\end{patch}

\begin{patch}{ugLangLoopsBreakMorePagePatch2}
\begin{paste}{ugLangLoopsBreakMorePageFull2}{ugLangLoopsBreakMorePageEmpty2}
\pastebutton{ugLangLoopsBreakMorePageFull2}{\hidepaste}
\tab{5}\spadcommand{repeat
  i := i + 1
  if i**2 > 100 then break
\free{i }\bound{i1 }}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsBreakMorePageEmpty2}
\begin{paste}{ugLangLoopsBreakMorePageEmpty2}{ugLangLoopsBreakMorePagePatch2}
\pastebutton{ugLangLoopsBreakMorePageEmpty2}{\showpaste}
\tab{5}\spadcommand{repeat
  i := i + 1
  if i**2 > 100 then break
\free{i }\bound{i1 }}
\end{paste}\end{patch}

\begin{patch}{ugLangLoopsBreakMorePagePatch3}
\begin{paste}{ugLangLoopsBreakMorePageFull3}{ugLangLoopsBreakMorePageEmpty3}
\pastebutton{ugLangLoopsBreakMorePageFull3}{\hidepaste}
\tab{5}\spadcommand{i\free{i1 }}
```

```
\indentrel{3}\begin{verbatim}
   (3)  11
                               Type: NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsBreakMorePageEmpty3}
\begin{paste}{ugLangLoopsBreakMorePageEmpty3}{ugLangLoopsBreakMorePagePatch3}
\pastebutton{ugLangLoopsBreakMorePageEmpty3}{\showpaste}
\tab{5}\spadcommand{i\free{i1 }}
\end{paste}\end{patch}

\begin{patch}{ugLangLoopsBreakMorePagePatch4}
\begin{paste}{ugLangLoopsBreakMorePageFull4}{ugLangLoopsBreakMorePageEmpty4}
\pastebutton{ugLangLoopsBreakMorePageFull4}{\hidepaste}
\tab{5}\spadcommand{i := 0\bound{i2 }}
\indentrel{3}\begin{verbatim}
   (4)  0
                               Type: NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsBreakMorePageEmpty4}
\begin{paste}{ugLangLoopsBreakMorePageEmpty4}{ugLangLoopsBreakMorePagePatch4}
\pastebutton{ugLangLoopsBreakMorePageEmpty4}{\showpaste}
\tab{5}\spadcommand{i := 0\bound{i2 }}
\end{paste}\end{patch}

\begin{patch}{ugLangLoopsBreakMorePagePatch5}
\begin{paste}{ugLangLoopsBreakMorePageFull5}{ugLangLoopsBreakMorePageEmpty5}
\pastebutton{ugLangLoopsBreakMorePageFull5}{\hidepaste}
\tab{5}\spadcommand{repeat
  i := i + 1
  i**2 > 100 => break
\free{i2 }\bound{i3 }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsBreakMorePageEmpty5}
\begin{paste}{ugLangLoopsBreakMorePageEmpty5}{ugLangLoopsBreakMorePagePatch5}
\pastebutton{ugLangLoopsBreakMorePageEmpty5}{\showpaste}
\tab{5}\spadcommand{repeat
  i := i + 1
  i**2 > 100 => break
\free{i2 }\bound{i3 }}
\end{paste}\end{patch}

\begin{patch}{ugLangLoopsBreakMorePagePatch6}
```

```
\begin{paste}{ugLangLoopsBreakMorePageFull6}{ugLangLoopsBreakMorePageEmpty6}
\pastebutton{ugLangLoopsBreakMorePageFull6}{\hidepaste}
\tab{5}\spadcommand{i\free{i3 }}
\indentrel{3}\begin{verbatim}
   (6)  11
                              Type: NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsBreakMorePageEmpty6}
\begin{paste}{ugLangLoopsBreakMorePageEmpty6}{ugLangLoopsBreakMorePagePatch6}
\pastebutton{ugLangLoopsBreakMorePageEmpty6}{\showpaste}
\tab{5}\spadcommand{i\free{i3 }}
\end{paste}\end{patch}

\begin{patch}{ugLangLoopsBreakMorePagePatch7}
\begin{paste}{ugLangLoopsBreakMorePageFull7}{ugLangLoopsBreakMorePageEmpty7}
\pastebutton{ugLangLoopsBreakMorePageFull7}{\hidepaste}
\tab{5}\spadcommand{(n, i, f) := (100, 1, 1)\bound{n }\bound{i4 }\bound{f }}
\indentrel{3}\begin{verbatim}
   (7)  1
                                 Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsBreakMorePageEmpty7}
\begin{paste}{ugLangLoopsBreakMorePageEmpty7}{ugLangLoopsBreakMorePagePatch7}
\pastebutton{ugLangLoopsBreakMorePageEmpty7}{\showpaste}
\tab{5}\spadcommand{(n, i, f) := (100, 1, 1)\bound{n }\bound{i4 }\bound{f }}
\end{paste}\end{patch}

\begin{patch}{ugLangLoopsBreakMorePagePatch8}
\begin{paste}{ugLangLoopsBreakMorePageFull8}{ugLangLoopsBreakMorePageEmpty8}
\pastebutton{ugLangLoopsBreakMorePageFull8}{\hidepaste}
\tab{5}\spadcommand{repeat
  if i > n then break
  f := f * i
  i := i + 1
\bound{f1 }\bound{i5 }\free{f i4 n }}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsBreakMorePageEmpty8}
\begin{paste}{ugLangLoopsBreakMorePageEmpty8}{ugLangLoopsBreakMorePagePatch8}
\pastebutton{ugLangLoopsBreakMorePageEmpty8}{\showpaste}
\tab{5}\spadcommand{repeat
  if i > n then break
  f := f * i
```

```
  i := i + 1
\bound{f1 }\bound{i5 }\free{f i4 n }}
\end{paste}\end{patch}

\begin{patch}{ugLangLoopsBreakMorePagePatch9}
\begin{paste}{ugLangLoopsBreakMorePageFull9}{ugLangLoopsBreakMorePageEmpty9}
\pastebutton{ugLangLoopsBreakMorePageFull9}{\hidepaste}
\tab{5}\spadcommand{f\free{f1 }}
\indentrel{3}\begin{verbatim}
   (9)
  933262154439441526816992388562667004907159682643816214_
   68592963895217599993229915608941463976156518286253697_
   920827223758251185210916864000000000000000000000000
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsBreakMorePageEmpty9}
\begin{paste}{ugLangLoopsBreakMorePageEmpty9}{ugLangLoopsBreakMorePagePatch9}
\pastebutton{ugLangLoopsBreakMorePageEmpty9}{\showpaste}
\tab{5}\spadcommand{f\free{f1 }}
\end{paste}\end{patch}

\begin{patch}{ugLangLoopsBreakMorePagePatch10}
\begin{paste}{ugLangLoopsBreakMorePageFull10}{ugLangLoopsBreakMorePageEmpty10}
\pastebutton{ugLangLoopsBreakMorePageFull10}{\hidepaste}
\tab{5}\spadcommand{m := matrix [[21,37,53,14], [8,-24,22,-16], [2,10,15,14], [26,33,55,-13]]\bo
\indentrel{3}\begin{verbatim}
        21   37   53   14

        8   - 24  22  - 16
   (10)
        2    10   15   14

        26   33   55  - 13
                                   Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsBreakMorePageEmpty10}
\begin{paste}{ugLangLoopsBreakMorePageEmpty10}{ugLangLoopsBreakMorePagePatch10}
\pastebutton{ugLangLoopsBreakMorePageEmpty10}{\showpaste}
\tab{5}\spadcommand{m := matrix [[21,37,53,14], [8,-24,22,-16], [2,10,15,14], [26,33,55,-13]]\bo
\end{paste}\end{patch}

\begin{patch}{ugLangLoopsBreakMorePagePatch11}
\begin{paste}{ugLangLoopsBreakMorePageFull11}{ugLangLoopsBreakMorePageEmpty11}
\pastebutton{ugLangLoopsBreakMorePageFull11}{\hidepaste}
\tab{5}\spadcommand{(r, c) := (1, 1)\bound{r }\bound{c }}
\indentrel{3}\begin{verbatim}
```

```
   (11)  1
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsBreakMorePageEmpty11}
\begin{paste}{ugLangLoopsBreakMorePageEmpty11}{ugLangLoopsBreakMorePagePatch11}
\pastebutton{ugLangLoopsBreakMorePageEmpty11}{\showpaste}
\tab{5}\spadcommand{(r, c) := (1, 1)\bound{r }\bound{c }}
\end{paste}\end{patch}

\begin{patch}{ugLangLoopsBreakMorePagePatch12}
\begin{paste}{ugLangLoopsBreakMorePageFull12}{ugLangLoopsBreakMorePageEmpty12}
\pastebutton{ugLangLoopsBreakMorePageFull12}{\hidepaste}
\tab{5}\spadcommand{(lastrow, lastcol) := (nrows(m), ncols(m))\bound{lastrow }\bound{lastcol }\free{m2 }}
\indentrel{3}\begin{verbatim}
   (12)  4
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsBreakMorePageEmpty12}
\begin{paste}{ugLangLoopsBreakMorePageEmpty12}{ugLangLoopsBreakMorePagePatch12}
\pastebutton{ugLangLoopsBreakMorePageEmpty12}{\showpaste}
\tab{5}\spadcommand{(lastrow, lastcol) := (nrows(m), ncols(m))\bound{lastrow }\bound{lastcol }\free{m2 }}
\end{paste}\end{patch}

\begin{patch}{ugLangLoopsBreakMorePagePatch13}
\begin{paste}{ugLangLoopsBreakMorePageFull13}{ugLangLoopsBreakMorePageEmpty13}
\pastebutton{ugLangLoopsBreakMorePageFull13}{\hidepaste}
\tab{5}\spadcommand{repeat
  if r > lastrow then break
  c := 1
  repeat
    if c > lastcol then break
    if elt(m,r,c) < 0 then
      output [r, c, elt(m,r,c)]
      r := lastrow
      break     -- don't look any further
    c := c + 1
  r := r + 1
\free{m2 r c lastrow lastcol }}
\indentrel{3}\begin{verbatim}
   [2,2,- 24]
                                              Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsBreakMorePageEmpty13}
\begin{paste}{ugLangLoopsBreakMorePageEmpty13}{ugLangLoopsBreakMorePagePatch13}
```

```
\pastebutton{ugLangLoopsBreakMorePageEmpty13}{\showpaste}
\tab{5}\spadcommand{repeat
  if r > lastrow then break
  c := 1
  repeat
    if c > lastcol then break
    if elt(m,r,c) < 0 then
      output [r, c, elt(m,r,c)]
      r := lastrow
      break     -- don't look any further
    c := c + 1
  r := r + 1
\free{m2 r c lastrow lastcol }}
\end{paste}\end{patch}
```

———-

iterate in Loops

— ug05.ht —

```
\begin{page}{ugLangLoopsIteratePage}{5.4.6. iterate in Loops}
\beginscroll

Axiom provides an \axiom{iterate} expression that
\spadkey{iterate}
skips over the remainder of a loop body and starts the next loop iteration.
\xtc{
We first initialize a counter.
}{
\spadpaste{i := 0 \bound{i}}
}
\xtc{
Display the even integers from \axiom{2} to \axiom{5}.
}{
\begin{spadsrc}[\free{i}]
repeat
  i := i + 1
  if i > 5 then break
  if odd?(i) then iterate
  output(i)
\end{spadsrc}
}

\endscroll
\autobuttons
\end{page}
```

```
\begin{patch}{ugLangLoopsIteratePagePatch1}
\begin{paste}{ugLangLoopsIteratePageFull1}{ugLangLoopsIteratePageEmpty1}
\pastebutton{ugLangLoopsIteratePageFull1}{\hidepaste}
\tab{5}\spadcommand{i := 0\bound{i }}
\indentrel{3}\begin{verbatim}
   (1)  0
                               Type: NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsIteratePageEmpty1}
\begin{paste}{ugLangLoopsIteratePageEmpty1}{ugLangLoopsIteratePagePatch1}
\pastebutton{ugLangLoopsIteratePageEmpty1}{\showpaste}
\tab{5}\spadcommand{i := 0\bound{i }}
\end{paste}\end{patch}

\begin{patch}{ugLangLoopsIteratePagePatch2}
\begin{paste}{ugLangLoopsIteratePageFull2}{ugLangLoopsIteratePageEmpty2}
\pastebutton{ugLangLoopsIteratePageFull2}{\hidepaste}
\tab{5}\spadcommand{repeat
  i := i + 1
  if i > 5 then break
  if odd?(i) then iterate
  output(i)
\free{i }}
\indentrel{3}\begin{verbatim}
   2
   4
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsIteratePageEmpty2}
\begin{paste}{ugLangLoopsIteratePageEmpty2}{ugLangLoopsIteratePagePatch2}
\pastebutton{ugLangLoopsIteratePageEmpty2}{\showpaste}
\tab{5}\spadcommand{repeat
  i := i + 1
  if i > 5 then break
  if odd?(i) then iterate
  output(i)
\free{i }}
\end{paste}\end{patch}
```

while Loops

— ug05.ht —

```
\begin{page}{ugLangLoopsWhilePage}{5.4.7. while Loops}
\beginscroll

The \axiom{repeat} in a loop can be modified by adding one or
more \axiom{while} clauses.
\spadkey{while}
Each clause contains a \spadgloss{predicate}
immediately following the \axiom{while} keyword.
The predicate is tested {\it before}
the evaluation of the body of the loop.
The loop body is evaluated whenever the predicates in a \axiom{while}
clause are all \axiom{true}.

\beginImportant
The syntax for a simple loop using \axiom{while} is
\centerline{{\axiom{while} {\it predicate} \axiom{repeat} {\it loopBody}}}
The {\it predicate} is evaluated before {\it loopBody} is evaluated.
A \axiom{while} loop terminates immediately when {\it predicate}
evaluates to \axiom{false} or when a \axiom{break} or \axiom{return}
expression is evaluated in {\it loopBody}.
The value returned by the loop is \void{}.
\endImportant

\xtc{
Here is a simple example of using \axiom{while} in a loop.
We first initialize the counter.
}{
\spadpaste{i := 1 \bound{i}}
}
\xtc{
The steps involved in computing this example are
(1) set \axiom{i} to \axiom{1}, (2) test the condition \axiom{i < 1} and
determine that it is not true, and (3) do not evaluate the
loop body and therefore do not display \axiom{"hello"}.
}{
\begin{spadsrc}[\free{i}]
while i < 1 repeat
  output "hello"
  i := i + 1
\end{spadsrc}
}
\xtc{
If you have multiple predicates to be tested use the
logical \axiom{and} operation to separate them.
Axiom evaluates these predicates from left to right.
```

```
}{
\spadpaste{(x, y) := (1, 1) \bound{x}\bound{y}}
}
\xtc{
}{
\begin{spadsrc}[\free{x y}]
while x < 4 and y < 10 repeat
  output [x,y]
  x := x + 1
  y := y + 2
\end{spadsrc}
}
\xtc{
A \axiom{break} expression can be included in a loop body to terminate
a loop even if the predicate in any \axiom{while} clauses are not
\axiom{false}.
}{
\spadpaste{(x, y) := (1, 1) \bound{x1}\bound{y1}}
}
\xtc{
This loop has multiple \axiom{while} clauses and the loop terminates
before any one of their conditions evaluates to \axiom{false}.
}{
\begin{spadsrc}[\free{x1 y1}]
while x < 4 while y < 10 repeat
  if x + y > 7 then break
  output [x,y]
  x := x + 1
  y := y + 2
\end{spadsrc}
}
\xtc{
Here's a different version of the nested loops that looked
for the first negative element in a matrix.
}{
\spadpaste{m := matrix [[21,37,53,14], [8,-24,22,-16], [2,10,15,14],
[26,33,55,-13]] \bound{m2}}
}
\xtc{
Initialized the row index to \axiom{1} and
get the number of rows and columns.
If we were writing a function, these would all be
local variables.
}{
\spadpaste{r := 1 \bound{r}}
}
\xtc{
}{
\spadpaste{(lastrow, lastcol) := (nrows(m), ncols(m))
\bound{lastrow}\bound{lastcol}\free{m2}}
```

```
}
%
\xtc{
Scan the rows looking for the first negative element.
}{
\begin{spadsrc}[\free{m2 r lastrow lastcol}]
while r <= lastrow repeat
  c := 1  -- index of first column
  while c <= lastcol repeat
    if elt(m,r,c) < 0 then
      output [r, c, elt(m,r,c)]
      r := lastrow
      break     -- don't look any further
    c := c + 1
  r := r + 1
\end{spadsrc}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugLangLoopsWhilePagePatch1}
\begin{paste}{ugLangLoopsWhilePageFull1}{ugLangLoopsWhilePageEmpty1}
\pastebutton{ugLangLoopsWhilePageFull1}{\hidepaste}
\tab{5}\spadcommand{i := 1\bound{i }}
\indentrel{3}\begin{verbatim}
   (1)  1
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsWhilePageEmpty1}
\begin{paste}{ugLangLoopsWhilePageEmpty1}{ugLangLoopsWhilePagePatch1}
\pastebutton{ugLangLoopsWhilePageEmpty1}{\showpaste}
\tab{5}\spadcommand{i := 1\bound{i }}
\end{paste}\end{patch}

\begin{patch}{ugLangLoopsWhilePagePatch2}
\begin{paste}{ugLangLoopsWhilePageFull2}{ugLangLoopsWhilePageEmpty2}
\pastebutton{ugLangLoopsWhilePageFull2}{\hidepaste}
\tab{5}\spadcommand{while i < 1 repeat
  output "hello"
  i := i + 1
\free{i }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugLangLoopsWhilePageEmpty2}
\begin{paste}{ugLangLoopsWhilePageEmpty2}{ugLangLoopsWhilePagePatch2}
\pastebutton{ugLangLoopsWhilePageEmpty2}{\showpaste}
\tab{5}\spadcommand{while i < 1 repeat
  output "hello"
  i := i + 1
\free{i }}
\end{paste}\end{patch}

\begin{patch}{ugLangLoopsWhilePagePatch3}
\begin{paste}{ugLangLoopsWhilePageFull3}{ugLangLoopsWhilePageEmpty3}
\pastebutton{ugLangLoopsWhilePageFull3}{\hidepaste}
\tab{5}\spadcommand{(x, y) := (1, 1)\bound{x }\bound{y }}
\indentrel{3}\begin{verbatim}
   (3)  1
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsWhilePageEmpty3}
\begin{paste}{ugLangLoopsWhilePageEmpty3}{ugLangLoopsWhilePagePatch3}
\pastebutton{ugLangLoopsWhilePageEmpty3}{\showpaste}
\tab{5}\spadcommand{(x, y) := (1, 1)\bound{x }\bound{y }}
\end{paste}\end{patch}

\begin{patch}{ugLangLoopsWhilePagePatch4}
\begin{paste}{ugLangLoopsWhilePageFull4}{ugLangLoopsWhilePageEmpty4}
\pastebutton{ugLangLoopsWhilePageFull4}{\hidepaste}
\tab{5}\spadcommand{while x < 4 and y < 10 repeat
  output [x,y]
  x := x + 1
  y := y + 2
\free{x y }}
\indentrel{3}\begin{verbatim}
   [1,1]
   [2,3]
   [3,5]
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsWhilePageEmpty4}
\begin{paste}{ugLangLoopsWhilePageEmpty4}{ugLangLoopsWhilePagePatch4}
\pastebutton{ugLangLoopsWhilePageEmpty4}{\showpaste}
\tab{5}\spadcommand{while x < 4 and y < 10 repeat
  output [x,y]
  x := x + 1
  y := y + 2
\free{x y }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugLangLoopsWhilePagePatch5}
\begin{paste}{ugLangLoopsWhilePageFull5}{ugLangLoopsWhilePageEmpty5}
\pastebutton{ugLangLoopsWhilePageFull5}{\hidepaste}
\tab{5}\spadcommand{(x, y) := (1, 1)\bound{x1 }\bound{y1 }}
\indentrel{3}\begin{verbatim}
   (5)  1
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsWhilePageEmpty5}
\begin{paste}{ugLangLoopsWhilePageEmpty5}{ugLangLoopsWhilePagePatch5}
\pastebutton{ugLangLoopsWhilePageEmpty5}{\showpaste}
\tab{5}\spadcommand{(x, y) := (1, 1)\bound{x1 }\bound{y1 }}
\end{paste}\end{patch}

\begin{patch}{ugLangLoopsWhilePagePatch6}
\begin{paste}{ugLangLoopsWhilePageFull6}{ugLangLoopsWhilePageEmpty6}
\pastebutton{ugLangLoopsWhilePageFull6}{\hidepaste}
\tab{5}\spadcommand{while x < 4 while y < 10 repeat
  if x + y > 7 then break
  output [x,y]
  x := x + 1
  y := y + 2
\free{x1 y1 }}
\indentrel{3}\begin{verbatim}
   [1,1]
   [2,3]
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsWhilePageEmpty6}
\begin{paste}{ugLangLoopsWhilePageEmpty6}{ugLangLoopsWhilePagePatch6}
\pastebutton{ugLangLoopsWhilePageEmpty6}{\showpaste}
\tab{5}\spadcommand{while x < 4 while y < 10 repeat
  if x + y > 7 then break
  output [x,y]
  x := x + 1
  y := y + 2
\free{x1 y1 }}
\end{paste}\end{patch}

\begin{patch}{ugLangLoopsWhilePagePatch7}
\begin{paste}{ugLangLoopsWhilePageFull7}{ugLangLoopsWhilePageEmpty7}
\pastebutton{ugLangLoopsWhilePageFull7}{\hidepaste}
\tab{5}\spadcommand{m := matrix [[21,37,53,14], [8,-24,22,-16], [2,10,15,14], [26,33,55,-13]]\bo
\indentrel{3}\begin{verbatim}
        21   37   53   14
```

```
        8   - 24  22  - 16
   (7)
        2    10   15   14

        26   33   55  - 13
                                  Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsWhilePageEmpty7}
\begin{paste}{ugLangLoopsWhilePageEmpty7}{ugLangLoopsWhilePagePatch7}
\pastebutton{ugLangLoopsWhilePageEmpty7}{\showpaste}
\tab{5}\spadcommand{m := matrix [[21,37,53,14], [8,-24,22,-16], [2,10,15,14], [26,33,55,-13]]\bound{m2 }}
\end{paste}\end{patch}

\begin{patch}{ugLangLoopsWhilePagePatch8}
\begin{paste}{ugLangLoopsWhilePageFull8}{ugLangLoopsWhilePageEmpty8}
\pastebutton{ugLangLoopsWhilePageFull8}{\hidepaste}
\tab{5}\spadcommand{r := 1\bound{r }}
\indentrel{3}\begin{verbatim}
   (8)  1
                                 Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsWhilePageEmpty8}
\begin{paste}{ugLangLoopsWhilePageEmpty8}{ugLangLoopsWhilePagePatch8}
\pastebutton{ugLangLoopsWhilePageEmpty8}{\showpaste}
\tab{5}\spadcommand{r := 1\bound{r }}
\end{paste}\end{patch}

\begin{patch}{ugLangLoopsWhilePagePatch9}
\begin{paste}{ugLangLoopsWhilePageFull9}{ugLangLoopsWhilePageEmpty9}
\pastebutton{ugLangLoopsWhilePageFull9}{\hidepaste}
\tab{5}\spadcommand{(lastrow, lastcol) := (nrows(m), ncols(m))\bound{lastrow }\bound{lastcol }\free{m2 }}
\indentrel{3}\begin{verbatim}
   (9)  4
                                 Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsWhilePageEmpty9}
\begin{paste}{ugLangLoopsWhilePageEmpty9}{ugLangLoopsWhilePagePatch9}
\pastebutton{ugLangLoopsWhilePageEmpty9}{\showpaste}
\tab{5}\spadcommand{(lastrow, lastcol) := (nrows(m), ncols(m))\bound{lastrow }\bound{lastcol }\free{m2 }}
\end{paste}\end{patch}

\begin{patch}{ugLangLoopsWhilePagePatch10}
\begin{paste}{ugLangLoopsWhilePageFull10}{ugLangLoopsWhilePageEmpty10}
```

```
\pastebutton{ugLangLoopsWhilePageFull10}{\hidepaste}
\tab{5}\spadcommand{while r <= lastrow repeat
  c := 1  -- index of first column
  while c <= lastcol repeat
    if elt(m,r,c) < 0 then
      output [r, c, elt(m,r,c)]
      r := lastrow
      break     -- don't look any further
    c := c + 1
  r := r + 1
\free{m2 r lastrow lastcol }}
\indentrel{3}\begin{verbatim}
   [2,2,- 24]
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsWhilePageEmpty10}
\begin{paste}{ugLangLoopsWhilePageEmpty10}{ugLangLoopsWhilePagePatch10}
\pastebutton{ugLangLoopsWhilePageEmpty10}{\showpaste}
\tab{5}\spadcommand{while r <= lastrow repeat
  c := 1  -- index of first column
  while c <= lastcol repeat
    if elt(m,r,c) < 0 then
      output [r, c, elt(m,r,c)]
      r := lastrow
      break     -- don't look any further
    c := c + 1
  r := r + 1
\free{m2 r lastrow lastcol }}
\end{paste}\end{patch}
```

———-

for Loops

— ug05.ht —

```
\begin{page}{ugLangLoopsForInPage}{5.4.8. for Loops}
\beginscroll

Axiom provides the \axiom{for}
\spadkey{for}
and \axiom{in}
\spadkey{in}
keywords in \axiom{repeat} loops,
allowing you to iterate across all
```

```
elements of a list, or to have a variable take on integral values
from a lower bound to an upper bound.
We shall refer to these modifying clauses of \axiom{repeat} loops as
\axiom{for} clauses.
These clauses can be present in addition to \axiom{while} clauses.
As with all other types of \axiom{repeat} loops, \axiom{break} can
\spadkey{break}
be used to prematurely terminate the evaluation of the loop.

\beginImportant
The syntax for a simple loop using \axiom{for} is
\centerline{{\axiom{for} {\it iterator} \axiom{repeat} {\it loopBody}}}
The {\it iterator} has several forms.
Each form has an end test which is evaluated
before {\it loopBody} is evaluated.
A \axiom{for} loop terminates immediately when the end test
succeeds (evaluates to \axiom{true}) or when a \axiom{break} or
\axiom{return} expression is evaluated in {\it loopBody}.
The value returned by the loop is \void{}.
\endImportant

\endscroll
\autobuttons
\end{page}
```

———-

for i in n..m repeat

⇒ “notitle” (SegmentXmpPage) 3.95 on page 1218

— **ug05.ht** —

```
\begin{page}{ugLangLoopsForInNMPage}{5.4.9. for i in n..m repeat}
\beginscroll

If \axiom{for}
\spadkey{for}
is followed by a variable name, the \axiom{in}
\spadkey{in}
keyword and then an integer segment of the form \axiom{n..m},
the end test for this loop is the predicate \axiom{i > m}.
The body of the loop is evaluated \axiom{m-n+1} times if this
number is greater than 0.
If this number is less than or equal to 0, the loop body is not evaluated
at all.

The variable \axiom{i} has the value
```

```
\axiom{n, n+1, ..., m} for successive iterations
of the loop body.
The loop variable is a \spadgloss{local variable}
within the loop body: its value is not available outside the loop body
and its value and type within the loop body completely mask any outer
definition of a variable with the same name.

%
\xtc{
This loop prints the values of
\texht{${10}^3$, ${11}^3$, and $12^3$}{\axiom{10**3, 11**3, and 12**3}}:
}{
\spadpaste{for i in 10..12 repeat output(i**3)}
}
%
\xtc{
Here is a sample list.
}{
\spadpaste{a := [1,2,3] \bound{a}}
}
\xtc{
Iterate across this list,
using \axiomSyntax{.} to access the elements of a list and
the \axiomFun{\#} operation to count its elements.
}{
\spadpaste{for i in 1..\#a repeat output(a.i) \free{a}}
}
%
This type of iteration is applicable to anything that uses \axiomSyntax{.}.
You can also use it with functions that use indices to extract elements.
%
\xtc{
Define \axiom{m} to be a matrix.
}{
\spadpaste{m := matrix [[1,2],[4,3],[9,0]] \bound{m}}
}
\xtc{
Display the rows of \axiom{m}.
}{
\spadpaste{for i in 1..nrows(m) repeat output row(m,i) \free{m}}
}
%
You can use \axiom{iterate} with \axiom{for}-loops.
\spadkey{iterate}
\xtc{
Display the even integers in a segment.
}{
\begin{spadsrc}
for i in 1..5 repeat
  if odd?(i) then iterate
```

```
  output(i)
\end{spadsrc}
}

See \downlink{‘Segment’}{SegmentXmpPage}\ignore{Segment}
for more information about segments.

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugLangLoopsForInNMPagePatch1}
\begin{paste}{ugLangLoopsForInNMPageFull1}{ugLangLoopsForInNMPageEmpty1}
\pastebutton{ugLangLoopsForInNMPageFull1}{\hidepaste}
\tab{5}\spadcommand{for i in 10..12 repeat output(i**3)}
\indentrel{3}\begin{verbatim}
   1000
   1331
   1728
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsForInNMPageEmpty1}
\begin{paste}{ugLangLoopsForInNMPageEmpty1}{ugLangLoopsForInNMPagePatch1}
\pastebutton{ugLangLoopsForInNMPageEmpty1}{\showpaste}
\tab{5}\spadcommand{for i in 10..12 repeat output(i**3)}
\end{paste}\end{patch}

\begin{patch}{ugLangLoopsForInNMPagePatch2}
\begin{paste}{ugLangLoopsForInNMPageFull2}{ugLangLoopsForInNMPageEmpty2}
\pastebutton{ugLangLoopsForInNMPageFull2}{\hidepaste}
\tab{5}\spadcommand{a := [1,2,3]\bound{a }}
\indentrel{3}\begin{verbatim}
   (2)  [1,2,3]
                            Type: List PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsForInNMPageEmpty2}
\begin{paste}{ugLangLoopsForInNMPageEmpty2}{ugLangLoopsForInNMPagePatch2}
\pastebutton{ugLangLoopsForInNMPageEmpty2}{\showpaste}
\tab{5}\spadcommand{a := [1,2,3]\bound{a }}
\end{paste}\end{patch}

\begin{patch}{ugLangLoopsForInNMPagePatch3}
\begin{paste}{ugLangLoopsForInNMPageFull3}{ugLangLoopsForInNMPageEmpty3}
\pastebutton{ugLangLoopsForInNMPageFull3}{\hidepaste}
\tab{5}\spadcommand{for i in 1..\#a repeat output(a.i)\free{a }}
\indentrel{3}\begin{verbatim}
```

```
   1
   2
   3
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsForInNMPageEmpty3}
\begin{paste}{ugLangLoopsForInNMPageEmpty3}{ugLangLoopsForInNMPagePatch3}
\pastebutton{ugLangLoopsForInNMPageEmpty3}{\showpaste}
\tab{5}\spadcommand{for i in 1..\#a repeat output(a.i)\free{a }}
\end{paste}\end{patch}

\begin{patch}{ugLangLoopsForInNMPagePatch4}
\begin{paste}{ugLangLoopsForInNMPageFull4}{ugLangLoopsForInNMPageEmpty4}
\pastebutton{ugLangLoopsForInNMPageFull4}{\hidepaste}
\tab{5}\spadcommand{m := matrix [[1,2],[4,3],[9,0]]\bound{m }}
\indentrel{3}\begin{verbatim}
        1  2

   (4)  4  3

        9  0
                                   Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsForInNMPageEmpty4}
\begin{paste}{ugLangLoopsForInNMPageEmpty4}{ugLangLoopsForInNMPagePatch4}
\pastebutton{ugLangLoopsForInNMPageEmpty4}{\showpaste}
\tab{5}\spadcommand{m := matrix [[1,2],[4,3],[9,0]]\bound{m }}
\end{paste}\end{patch}

\begin{patch}{ugLangLoopsForInNMPagePatch5}
\begin{paste}{ugLangLoopsForInNMPageFull5}{ugLangLoopsForInNMPageEmpty5}
\pastebutton{ugLangLoopsForInNMPageFull5}{\hidepaste}
\tab{5}\spadcommand{for i in 1..nrows(m) repeat output row(m,i)\free{m }}
\indentrel{3}\begin{verbatim}
   [1,2]
   [4,3]
   [9,0]
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsForInNMPageEmpty5}
\begin{paste}{ugLangLoopsForInNMPageEmpty5}{ugLangLoopsForInNMPagePatch5}
\pastebutton{ugLangLoopsForInNMPageEmpty5}{\showpaste}
\tab{5}\spadcommand{for i in 1..nrows(m) repeat output row(m,i)\free{m }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugLangLoopsForInNMPagePatch6}
\begin{paste}{ugLangLoopsForInNMPageFull6}{ugLangLoopsForInNMPageEmpty6}
\pastebutton{ugLangLoopsForInNMPageFull6}{\hidepaste}
\tab{5}\spadcommand{for i in 1..5 repeat
  if odd?(i) then iterate
  output(i)
}
\indentrel{3}\begin{verbatim}
   2
   4
                                          Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsForInNMPageEmpty6}
\begin{paste}{ugLangLoopsForInNMPageEmpty6}{ugLangLoopsForInNMPagePatch6}
\pastebutton{ugLangLoopsForInNMPageEmpty6}{\showpaste}
\tab{5}\spadcommand{for i in 1..5 repeat
  if odd?(i) then iterate
  output(i)
}
\end{paste}\end{patch}
```

———-

for i in n..m by s repeat

— ug05.ht —

```
\begin{page}{ugLangLoopsForInNMSPage}{5.4.10. for i in n..m by s repeat}
\beginscroll

By default, the difference between values taken on by a variable in loops
such as \axiom{for i in n..m repeat ...} is \mathOrSpad{1}.
It is possible to supply another, possibly negative, step value by using
the \axiom{by}
\spadkey{by}
keyword along with \axiom{for} and \axiom{in}.
Like the upper and lower bounds, the step value following the
\axiom{by} keyword must be an integer.
Note that the loop
\axiom{for i in 1..2 by 0 repeat output(i)}
will not terminate by itself, as the step value does not change the index
from its initial value of \mathOrSpad{1}.

\xtc{
```

```
This expression displays the odd integers between two bounds.
}{
\spadpaste{for i in 1..5 by 2 repeat output(i)}
}
\xtc{
Use this to display the numbers in reverse order.
}{
\spadpaste{for i in 5..1 by -2 repeat output(i)}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugLangLoopsForInNMSPagePatch1}
\begin{paste}{ugLangLoopsForInNMSPageFull1}{ugLangLoopsForInNMSPageEmpty1}
\pastebutton{ugLangLoopsForInNMSPageFull1}{\hidepaste}
\tab{5}\spadcommand{for i in 1..5 by 2 repeat output(i)}
\indentrel{3}\begin{verbatim}
   1
   3
   5
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsForInNMSPageEmpty1}
\begin{paste}{ugLangLoopsForInNMSPageEmpty1}{ugLangLoopsForInNMSPagePatch1}
\pastebutton{ugLangLoopsForInNMSPageEmpty1}{\showpaste}
\tab{5}\spadcommand{for i in 1..5 by 2 repeat output(i)}
\end{paste}\end{patch}

\begin{patch}{ugLangLoopsForInNMSPagePatch2}
\begin{paste}{ugLangLoopsForInNMSPageFull2}{ugLangLoopsForInNMSPageEmpty2}
\pastebutton{ugLangLoopsForInNMSPageFull2}{\hidepaste}
\tab{5}\spadcommand{for i in 5..1 by -2 repeat output(i)}
\indentrel{3}\begin{verbatim}
   5
   3
   1
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsForInNMSPageEmpty2}
\begin{paste}{ugLangLoopsForInNMSPageEmpty2}{ugLangLoopsForInNMSPagePatch2}
\pastebutton{ugLangLoopsForInNMSPageEmpty2}{\showpaste}
\tab{5}\spadcommand{for i in 5..1 by -2 repeat output(i)}
\end{paste}\end{patch}
```

for i in n.. repeat

— ug05.ht —

```
\begin{page}{ugLangLoopsForInNPage}{5.4.11. for i in n.. repeat}
\beginscroll

If the value after the \axiomSyntax{..}
is omitted, the loop has no end test.
A potentially infinite loop is thus created.
The variable is given the successive values \axiom{n, n+1, n+2, ...}
and the loop is terminated only if a \axiom{break} or \axiom{return}
expression is evaluated in the loop body.
However you may also add some other modifying clause on the
\axiom{repeat} (for example, a \axiom{while} clause) to stop the loop.

\xtc{
This loop displays the integers greater than or equal to \axiom{15}
and less than the first prime greater than \axiom{15}.
}{
\spadpaste{for i in 15.. while not prime?(i) repeat output(i)}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugLangLoopsForInNPagePatch1}
\begin{paste}{ugLangLoopsForInNPageFull1}{ugLangLoopsForInNPageEmpty1}
\pastebutton{ugLangLoopsForInNPageFull1}{\hidepaste}
\tab{5}\spadcommand{for i in 15.. while not prime?(i) repeat output(i)}
\indentrel{3}\begin{verbatim}
   15
   16
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsForInNPageEmpty1}
\begin{paste}{ugLangLoopsForInNPageEmpty1}{ugLangLoopsForInNPagePatch1}
\pastebutton{ugLangLoopsForInNPageEmpty1}{\showpaste}
\tab{5}\spadcommand{for i in 15.. while not prime?(i) repeat output(i)}
\end{paste}\end{patch}
```

———

for x in l repeat

— **ug05.ht** —

```
\begin{page}{ugLangLoopsForInXLPage}{5.4.12. for x in l repeat}
\beginscroll

Another variant of the \axiom{for} loop has the form:
\centerline{{{\it \axiom{for} x \axiom{in} list \axiom{repeat} loopBody}}}
This form is used when you want to iterate directly over the
elements of a list.
In this form of the \axiom{for} loop, the variable
\axiom{x} takes on the value of each successive element in \axiom{l}.
The end test is most simply stated in English: ``are there no more
\axiom{x} in \axiom{l}?''

\xtc{
If \axiom{l} is this list,
}{
\spadpaste{l := [0,-5,3] \bound{l}}
}
\xtc{
display all elements of \axiom{l}, one per line.
}{
\spadpaste{for x in l repeat output(x) \free{l}}
}

Since the list constructing expression \axiom{expand [n..m]} creates the
list \axiom{[n, n+1, ..., m]}\footnote{This list is empty if \axiom{n >
m}.}, you might be tempted to think that the loops
\begin{verbatim}
for i in n..m repeat output(i)
\end{verbatim}
and
\begin{verbatim}
for x in expand [n..m] repeat output(x)
\end{verbatim}
are equivalent.
The second form first creates the list
\axiom{expand [n..m]} (no matter how large it might be) and
then does the iteration.
The first form potentially runs in much less space, as the index variable
\axiom{i} is simply incremented once per loop and the list is not actually
created.
Using the first form is much more efficient.
%
\xtc{
Of course, sometimes you really want to iterate across a specific list.
This displays each of the factors of \axiom{2400000}.
```

```
}{
\spadpaste{for f in factors(factor(2400000)) repeat output(f)}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugLangLoopsForInXLPagePatch1}
\begin{paste}{ugLangLoopsForInXLPageFull1}{ugLangLoopsForInXLPageEmpty1}
\pastebutton{ugLangLoopsForInXLPageFull1}{\hidepaste}
\tab{5}\spadcommand{l := [0,-5,3]\bound{l }}
\indentrel{3}\begin{verbatim}
   (1)  [0,- 5,3]
                                     Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsForInXLPageEmpty1}
\begin{paste}{ugLangLoopsForInXLPageEmpty1}{ugLangLoopsForInXLPagePatch1}
\pastebutton{ugLangLoopsForInXLPageEmpty1}{\showpaste}
\tab{5}\spadcommand{l := [0,-5,3]\bound{l }}
\end{paste}\end{patch}

\begin{patch}{ugLangLoopsForInXLPagePatch2}
\begin{paste}{ugLangLoopsForInXLPageFull2}{ugLangLoopsForInXLPageEmpty2}
\pastebutton{ugLangLoopsForInXLPageFull2}{\hidepaste}
\tab{5}\spadcommand{for x in l repeat output(x)\free{l }}
\indentrel{3}\begin{verbatim}
   0
   - 5
   3
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsForInXLPageEmpty2}
\begin{paste}{ugLangLoopsForInXLPageEmpty2}{ugLangLoopsForInXLPagePatch2}
\pastebutton{ugLangLoopsForInXLPageEmpty2}{\showpaste}
\tab{5}\spadcommand{for x in l repeat output(x)\free{l }}
\end{paste}\end{patch}

\begin{patch}{ugLangLoopsForInXLPagePatch3}
\begin{paste}{ugLangLoopsForInXLPageFull3}{ugLangLoopsForInXLPageEmpty3}
\pastebutton{ugLangLoopsForInXLPageFull3}{\hidepaste}
\tab{5}\spadcommand{for f in factors(factor(2400000)) repeat output(f)}
\indentrel{3}\begin{verbatim}
   [factor= 2,exponent= 8]
   [factor= 3,exponent= 1]
   [factor= 5,exponent= 5]
```

```
                                          Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsForInXLPageEmpty3}
\begin{paste}{ugLangLoopsForInXLPageEmpty3}{ugLangLoopsForInXLPagePatch3}
\pastebutton{ugLangLoopsForInXLPageEmpty3}{\showpaste}
\tab{5}\spadcommand{for f in factors(factor(2400000)) repeat output(f)}
\end{paste}\end{patch}
```

———-

"Such that" Predicates

— ug05.ht —

```
\begin{page}{ugLangLoopsForInPredPage}{5.4.13. ``Such that'' Predicates}
\beginscroll

A \axiom{for} loop can be followed by a \axiomSyntax{|} and then a
predicate.
The predicate qualifies the use of the values from the iterator following
the \axiom{for}.
Think of the vertical bar
\axiomSyntax{|} as the phrase ``such that.''
\xtc{
This loop expression
prints out the integers \axiom{n} in the given segment
such that \axiom{n} is odd.
}{
\spadpaste{for n in 0..4 | odd? n repeat output n}
}

\beginImportant
A \axiom{for} loop can also be written
\centerline{{\axiom{for} {\it iterator} \axiom{|} {\it predicate}
\axiom{repeat} {\it loopBody}}}
which is equivalent to:
\centerline{{\axiom{for} {\it iterator} \axiom{repeat if}}}
\centerline{{{\it predicate} \axiom{then} {\it loopBody} \axiom{else}
\axiom{iterate}}}
\endImportant

The predicate need not refer only to the variable in the \axiom{for} clause:
any variable in an outer scope can be part of the predicate.
\xtc{
In this example, the predicate on the inner \axiom{for} loop uses
```

```
\axiom{i} from the outer loop and the \axiom{j} from the \axiom{for}
clause that it directly modifies.
}{
\begin{spadsrc}
for i in 1..50 repeat
  for j in 1..50 | factorial(i+j) < 25 repeat
    output [i,j]
\end{spadsrc}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugLangLoopsForInPredPagePatch1}
\begin{paste}{ugLangLoopsForInPredPageFull1}{ugLangLoopsForInPredPageEmpty1}
\pastebutton{ugLangLoopsForInPredPageFull1}{\hidepaste}
\tab{5}\spadcommand{for n in 0..4 | odd? n repeat output n}
\indentrel{3}\begin{verbatim}
   1
   3
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsForInPredPageEmpty1}
\begin{paste}{ugLangLoopsForInPredPageEmpty1}{ugLangLoopsForInPredPagePatch1}
\pastebutton{ugLangLoopsForInPredPageEmpty1}{\showpaste}
\tab{5}\spadcommand{for n in 0..4 | odd? n repeat output n}
\end{paste}\end{patch}

\begin{patch}{ugLangLoopsForInPredPagePatch2}
\begin{paste}{ugLangLoopsForInPredPageFull2}{ugLangLoopsForInPredPageEmpty2}
\pastebutton{ugLangLoopsForInPredPageFull2}{\hidepaste}
\tab{5}\spadcommand{for i in 1..50 repeat
  for j in 1..50 | factorial(i+j) < 25 repeat
    output [i,j]
}
\indentrel{3}\begin{verbatim}
   [1,1]
   [1,2]
   [1,3]
   [2,1]
   [2,2]
   [3,1]
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsForInPredPageEmpty2}
```

```
\begin{paste}{ugLangLoopsForInPredPageEmpty2}{ugLangLoopsForInPredPagePatch2}
\pastebutton{ugLangLoopsForInPredPageEmpty2}{\showpaste}
\tab{5}\spadcommand{for i in 1..50 repeat
  for j in 1..50 | factorial(i+j) < 25 repeat
    output [i,j]
}
\end{paste}\end{patch}
```

———–

Parallel Iteration

⇒ “notitle” (ugLangLoopsForInPredPage) 9 on page 1800

— **ug05.ht** —

```
\begin{page}{ugLangLoopsParPage}{5.4.14. Parallel Iteration}
\beginscroll

The last example of
\texht{the previous section}{
\downlink{``Such that Predicates''}{ugLangLoopsForInPredPage}
in Section 5.4.13\ignore{ugLangLoopsForInPred}}
gives an example of
\spadgloss{nested iteration}: a loop is contained
in another loop.
Sometimes you want to iterate across two lists in parallel, or perhaps
you want to traverse a list while incrementing a variable.

\beginImportant
The general syntax of a repeat loop is
\centerline{{{\tt \subscriptIt{iterator}{1} \subscriptIt{iterator}{2}
\ldots \subscriptIt{iterator}{N} repeat {\it loopBody}}}}
where each {\it iterator} is either a \axiom{for} or a \axiom{while} clause.
The loop terminates immediately when the end test of any {\it iterator}
succeeds or when a \axiom{break} or \axiom{return} expression is evaluated
in {\it loopBody}.
The value returned by the loop is \void{}.
\endImportant

\xtc{
Here we write a loop to iterate across
two lists, computing the sum of the pairwise product
of elements. Here is the first list.
}{
\spadpaste{l := [1,3,5,7] \bound{l}}
}
\xtc{
```

```
And the second.
}{
\spadpaste{m := [100,200] \bound{m}}
}
\xtc{
The initial value of the sum counter.
}{
\spadpaste{sum := 0 \bound{sum}}
}
\xtc{
The last two elements of \axiom{l} are not used in the calculation
because \axiom{m} has two fewer elements than \axiom{l}.
}{
\begin{spadsrc}[\bound{doit}\free{sum l m}]
for x in l for y in m repeat
    sum := sum + x*y
\end{spadsrc}
}
\xtc{
Display the ``dot product.''
}{
\spadpaste{sum \free{doit}}
}

\xtc{
Next, we write a loop to compute the sum of the products of the loop
elements with
their positions in the loop.
}{
\spadpaste{l := [2,3,5,7,11,13,17,19,23,29,31,37] \bound{l1}}
}
\xtc{
The initial sum.
}{
\spadpaste{sum := 0 \bound{sum1}}
}
\xtc{
Here looping stops when the list \axiom{l} is exhausted, even though
the \axiom{for i in 0..} specifies no terminating condition.
}{
\spadpaste{
for i in 0.. for x in l repeat sum := i * x \bound{doit1}\free{sum1 l1}}
}
\xtc{
Display this weighted sum.
}{
\spadpaste{sum \free{doit1}}
}

When \axiomSyntax{|} is used to qualify any of the \axiom{for} clauses in a
```

```
parallel iteration, the variables in the predicates can be from an outer
scope or from a \axiom{for} clause in or to the left of a modified clause.

This is correct:
\begin{verbatim}
for i in 1..10 repeat
  for j in 200..300 | odd? (i+j) repeat
    output [i,j]
\end{verbatim}
This is not correct since the variable \axiom{j} has not been
defined outside the inner loop.
\begin{verbatim}
for i in 1..10 | odd? (i+j) repeat  -- wrong, j not defined
  for j in 200..300 repeat
    output [i,j]
\end{verbatim}

%>\head{subsection}{Mixing Loop Modifiers}{ugLangLoopsMix}

\xtc{
This example shows that it is possible to mix several of the
forms of \axiom{repeat} modifying clauses on a loop.
}{
\begin{spadsrc}
for i in 1..10
    for j in 151..160 | odd? j
      while i + j < 160 repeat
        output [i,j]
\end{spadsrc}
}
%
Here are useful rules for composing loop expressions:
\indent{4}
\beginitems
\item[1. ] \axiom{while} predicates can only refer to variables that
are global (or in an outer scope)
or that are defined in \axiom{for} clauses to the left of the
predicate.
\item[2. ] A ``such that'' predicate (something following \axiomSyntax{|})
must directly follow a \axiom{for} clause and can only refer to
variables that are global (or in an outer scope)
or defined in the modified \axiom{for} clause
or any \axiom{for} clause to the left.
\enditems
\indent{0}

\endscroll
\autobuttons
\end{page}
```

```
\begin{patch}{ugLangLoopsParPagePatch1}
\begin{paste}{ugLangLoopsParPageFull1}{ugLangLoopsParPageEmpty1}
\pastebutton{ugLangLoopsParPageFull1}{\hidepaste}
\tab{5}\spadcommand{l := [1,3,5,7]\bound{l }}
\indentrel{3}\begin{verbatim}
   (1)  [1,3,5,7]
                             Type: List PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsParPageEmpty1}
\begin{paste}{ugLangLoopsParPageEmpty1}{ugLangLoopsParPagePatch1}
\pastebutton{ugLangLoopsParPageEmpty1}{\showpaste}
\tab{5}\spadcommand{l := [1,3,5,7]\bound{l }}
\end{paste}\end{patch}

\begin{patch}{ugLangLoopsParPagePatch2}
\begin{paste}{ugLangLoopsParPageFull2}{ugLangLoopsParPageEmpty2}
\pastebutton{ugLangLoopsParPageFull2}{\hidepaste}
\tab{5}\spadcommand{m := [100,200]\bound{m }}
\indentrel{3}\begin{verbatim}
   (2)  [100,200]
                             Type: List PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsParPageEmpty2}
\begin{paste}{ugLangLoopsParPageEmpty2}{ugLangLoopsParPagePatch2}
\pastebutton{ugLangLoopsParPageEmpty2}{\showpaste}
\tab{5}\spadcommand{m := [100,200]\bound{m }}
\end{paste}\end{patch}

\begin{patch}{ugLangLoopsParPagePatch3}
\begin{paste}{ugLangLoopsParPageFull3}{ugLangLoopsParPageEmpty3}
\pastebutton{ugLangLoopsParPageFull3}{\hidepaste}
\tab{5}\spadcommand{sum := 0\bound{sum }}
\indentrel{3}\begin{verbatim}
   (3)  0
                               Type: NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsParPageEmpty3}
\begin{paste}{ugLangLoopsParPageEmpty3}{ugLangLoopsParPagePatch3}
\pastebutton{ugLangLoopsParPageEmpty3}{\showpaste}
\tab{5}\spadcommand{sum := 0\bound{sum }}
\end{paste}\end{patch}

\begin{patch}{ugLangLoopsParPagePatch4}
\begin{paste}{ugLangLoopsParPageFull4}{ugLangLoopsParPageEmpty4}
```

```
\pastebutton{ugLangLoopsParPageFull4}{\hidepaste}
\tab{5}\spadcommand{for x in l for y in m repeat
    sum := sum + x*y
\bound{doit }\free{sum l m }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsParPageEmpty4}
\begin{paste}{ugLangLoopsParPageEmpty4}{ugLangLoopsParPagePatch4}
\pastebutton{ugLangLoopsParPageEmpty4}{\showpaste}
\tab{5}\spadcommand{for x in l for y in m repeat
    sum := sum + x*y
\bound{doit }\free{sum l m }}
\end{paste}\end{patch}

\begin{patch}{ugLangLoopsParPagePatch5}
\begin{paste}{ugLangLoopsParPageFull5}{ugLangLoopsParPageEmpty5}
\pastebutton{ugLangLoopsParPageFull5}{\hidepaste}
\tab{5}\spadcommand{sum\free{doit }}
\indentrel{3}\begin{verbatim}
   (5)  700
                               Type: NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsParPageEmpty5}
\begin{paste}{ugLangLoopsParPageEmpty5}{ugLangLoopsParPagePatch5}
\pastebutton{ugLangLoopsParPageEmpty5}{\showpaste}
\tab{5}\spadcommand{sum\free{doit }}
\end{paste}\end{patch}

\begin{patch}{ugLangLoopsParPagePatch6}
\begin{paste}{ugLangLoopsParPageFull6}{ugLangLoopsParPageEmpty6}
\pastebutton{ugLangLoopsParPageFull6}{\hidepaste}
\tab{5}\spadcommand{l := [2,3,5,7,11,13,17,19,23,29,31,37]\bound{l1 }}
\indentrel{3}\begin{verbatim}
   (6)  [2,3,5,7,11,13,17,19,23,29,31,37]
                             Type: List PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsParPageEmpty6}
\begin{paste}{ugLangLoopsParPageEmpty6}{ugLangLoopsParPagePatch6}
\pastebutton{ugLangLoopsParPageEmpty6}{\showpaste}
\tab{5}\spadcommand{l := [2,3,5,7,11,13,17,19,23,29,31,37]\bound{l1 }}
\end{paste}\end{patch}

\begin{patch}{ugLangLoopsParPagePatch7}
```

```
\begin{paste}{ugLangLoopsParPageFull7}{ugLangLoopsParPageEmpty7}
\pastebutton{ugLangLoopsParPageFull7}{\hidepaste}
\tab{5}\spadcommand{sum := 0\bound{sum1 }}
\indentrel{3}\begin{verbatim}
   (7)  0
                              Type: NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsParPageEmpty7}
\begin{paste}{ugLangLoopsParPageEmpty7}{ugLangLoopsParPagePatch7}
\pastebutton{ugLangLoopsParPageEmpty7}{\showpaste}
\tab{5}\spadcommand{sum := 0\bound{sum1 }}
\end{paste}\end{patch}

\begin{patch}{ugLangLoopsParPagePatch8}
\begin{paste}{ugLangLoopsParPageFull8}{ugLangLoopsParPageEmpty8}
\pastebutton{ugLangLoopsParPageFull8}{\hidepaste}
\tab{5}\spadcommand{for i in 0.. for x in l repeat sum := i * x\bound{doit1 }\free{sum1 l1 }}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsParPageEmpty8}
\begin{paste}{ugLangLoopsParPageEmpty8}{ugLangLoopsParPagePatch8}
\pastebutton{ugLangLoopsParPageEmpty8}{\showpaste}
\tab{5}\spadcommand{for i in 0.. for x in l repeat sum := i * x\bound{doit1 }\free{sum1 l1 }}
\end{paste}\end{patch}

\begin{patch}{ugLangLoopsParPagePatch9}
\begin{paste}{ugLangLoopsParPageFull9}{ugLangLoopsParPageEmpty9}
\pastebutton{ugLangLoopsParPageFull9}{\hidepaste}
\tab{5}\spadcommand{sum\free{doit1 }}
\indentrel{3}\begin{verbatim}
   (9)  407
                              Type: NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsParPageEmpty9}
\begin{paste}{ugLangLoopsParPageEmpty9}{ugLangLoopsParPagePatch9}
\pastebutton{ugLangLoopsParPageEmpty9}{\showpaste}
\tab{5}\spadcommand{sum\free{doit1 }}
\end{paste}\end{patch}

\begin{patch}{ugLangLoopsParPagePatch10}
\begin{paste}{ugLangLoopsParPageFull10}{ugLangLoopsParPageEmpty10}
\pastebutton{ugLangLoopsParPageFull10}{\hidepaste}
\tab{5}\spadcommand{for i in 1..10
```

```
    for j in 151..160 | odd? j
      while i + j < 160 repeat
        output [i,j]
}
\indentrel{3}\begin{verbatim}
   [1,151]
   [3,153]
                                              Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangLoopsParPageEmpty10}
\begin{paste}{ugLangLoopsParPageEmpty10}{ugLangLoopsParPagePatch10}
\pastebutton{ugLangLoopsParPageEmpty10}{\showpaste}
\tab{5}\spadcommand{for i in 1..10
    for j in 151..160 | odd? j
      while i + j < 160 repeat
        output [i,j]
}
\end{paste}\end{patch}
```

——————

Creating Lists and Streams with Iterators

⇒ "notitle" (ugLangLoopsPage) 9 on page 1765
⇒ "notitle" (ListXmpPage) 3.64 on page 866
⇒ "notitle" (StreamXmpPage) 3.102 on page 1263

— **ug05.ht** —

```
\begin{page}{ugLangItsPage}{5.5. Creating Lists and Streams with Iterators}
\beginscroll

All of what we did for loops in
\downlink{``Loops''}{ugLangLoopsPage}
in Section 5.4\ignore{ugLangLoops}
can be transformed into expressions that create lists
and streams.
The \axiom{repeat,} \axiom{break} or \axiom{iterate} words are not used but
all the other ideas carry over.
Before we give you the general rule, here are some examples which
give you the idea.

\xtc{
This creates a simple list of the integers from \axiom{1} to \axiom{10}.
}{
```

```
\spadpaste{list := [i for i in 1..10] \bound{list}}
}
\xtc{
Create a stream of the integers greater than or equal to \axiom{1}.
}{
\spadpaste{stream := [i for i in 1..] \bound{stream}}
}
\xtc{
This is a list of the prime integers between \axiom{1} and \axiom{10},
inclusive.
}{
\spadpaste{[i for i in 1..10 | prime? i]}
}
\xtc{
This is a stream of the prime integers greater than or equal to \axiom{1}.
}{
\spadpaste{[i for i in 1..   | prime? i]}
}
\xtc{
This is a list of the integers between \axiom{1} and \axiom{10},
inclusive, whose squares are less than \axiom{700}.
}{
\spadpaste{[i for i in 1..10 while i*i < 700]}
}
\xtc{
This is a stream of the integers greater than or equal to \axiom{1}
whose squares are less than \axiom{700}.
}{
\spadpaste{[i for i in 1..   while i*i < 700]}
}

Got the idea?
Here is the general rule.

\beginImportant
The general syntax of a collection is
\centerline{{{\tt [ {\it collectExpression} \subscriptIt{iterator}{1}
\subscriptIt{iterator}{2}  \ldots  \subscriptIt{iterator}{N} ]}}}
where each \subscriptIt{iterator}{i} is either a \axiom{for} or a
\axiom{while} clause.
The loop terminates immediately when the end test of any
\subscriptIt{iterator}{i} succeeds or when a \axiom{return} expression is
evaluated in {\it collectExpression}.
The value returned by the collection is either a list or a stream of
elements, one for each iteration of the {\it collectExpression}.
\endImportant

Be careful when you use \axiom{while}
to create a stream.
By default, Axiom tries to compute and display the first ten elements
```

```
of a stream.
If the \axiom{while} condition is not satisfied quickly, Axiom
can spend a long (possibly infinite) time trying to compute
the elements.
Use \spadcmd{)set streams calculate} to change the default
to something else.
This also affects the number of terms computed and displayed for power
series.
For the purposes of this book, we have used this system
command to display fewer than ten terms.
\xtc{
Use nested iterators to create lists of
lists which can then be given as an argument to \axiomFun{matrix}.
}{
\spadpaste{matrix [[x**i+j for i in 1..3] for j in 10..12]}
}
\xtc{
You can also create lists of streams, streams of lists and
streams of streams.
Here is a stream of streams.
}{
\spadpaste{[[i/j for i in j+1..] for j in 1..]}
}
\xtc{
You can use parallel iteration across lists and streams to create
new lists.
}{
\spadpaste{[i/j for i in 3.. by 10 for j in 2..]}
}
\xtc{
Iteration stops if the end of a list or stream is reached.
}{
\spadpaste{[i**j for i in 1..7 for j in 2.. ]}
}
%\xtc{
%or a while condition fails.
%}{
%\spadcommand{[i**j for i in 1..  for j in 2.. while i + j < 5 ]}
%}
\xtc{
As with loops, you can combine these modifiers to make very
complicated conditions.
}{
\spadpaste{[[[i,j] for i in 10..15 | prime? i]
for j in 17..22 | j = squareFreePart j]}
}

See \downlink{`List'}{ListXmpPage}\ignore{List} and
\downlink{`Stream'}{StreamXmpPage}\ignore{Stream}
for more information on creating and
```

manipulating lists and streams, respectively.

```
\endscroll
\autobuttons
\end{page}

\begin{patch}{ugLangItsPagePatch1}
\begin{paste}{ugLangItsPageFull1}{ugLangItsPageEmpty1}
\pastebutton{ugLangItsPageFull1}{\hidepaste}
\tab{5}\spadcommand{list := [i for i in 1..10]\bound{list }}
\indentrel{3}\begin{verbatim}
   (1)  [1,2,3,4,5,6,7,8,9,10]
                             Type: List PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangItsPageEmpty1}
\begin{paste}{ugLangItsPageEmpty1}{ugLangItsPagePatch1}
\pastebutton{ugLangItsPageEmpty1}{\showpaste}
\tab{5}\spadcommand{list := [i for i in 1..10]\bound{list }}
\end{paste}\end{patch}

\begin{patch}{ugLangItsPagePatch2}
\begin{paste}{ugLangItsPageFull2}{ugLangItsPageEmpty2}
\pastebutton{ugLangItsPageFull2}{\hidepaste}
\tab{5}\spadcommand{stream := [i for i in 1..]\bound{stream }}
\indentrel{3}\begin{verbatim}
   (2)  [1,2,3,4,5,6,7,8,9,10,...]
                           Type: Stream PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangItsPageEmpty2}
\begin{paste}{ugLangItsPageEmpty2}{ugLangItsPagePatch2}
\pastebutton{ugLangItsPageEmpty2}{\showpaste}
\tab{5}\spadcommand{stream := [i for i in 1..]\bound{stream }}
\end{paste}\end{patch}

\begin{patch}{ugLangItsPagePatch3}
\begin{paste}{ugLangItsPageFull3}{ugLangItsPageEmpty3}
\pastebutton{ugLangItsPageFull3}{\hidepaste}
\tab{5}\spadcommand{[i for i in 1..10 | prime? i]}
\indentrel{3}\begin{verbatim}
   (3)  [2,3,5,7]
                             Type: List PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangItsPageEmpty3}
\begin{paste}{ugLangItsPageEmpty3}{ugLangItsPagePatch3}
```

```
\pastebutton{ugLangItsPageEmpty3}{\showpaste}
\tab{5}\spadcommand{[i for i in 1..10 | prime? i]}
\end{paste}\end{patch}

\begin{patch}{ugLangItsPagePatch4}
\begin{paste}{ugLangItsPageFull4}{ugLangItsPageEmpty4}
\pastebutton{ugLangItsPageFull4}{\hidepaste}
\tab{5}\spadcommand{[i for i in 1.. | prime? i]}
\indentrel{3}\begin{verbatim}
   (4)  [2,3,5,7,11,13,17,19,23,29,...]
                           Type: Stream PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangItsPageEmpty4}
\begin{paste}{ugLangItsPageEmpty4}{ugLangItsPagePatch4}
\pastebutton{ugLangItsPageEmpty4}{\showpaste}
\tab{5}\spadcommand{[i for i in 1.. | prime? i]}
\end{paste}\end{patch}

\begin{patch}{ugLangItsPagePatch5}
\begin{paste}{ugLangItsPageFull5}{ugLangItsPageEmpty5}
\pastebutton{ugLangItsPageFull5}{\hidepaste}
\tab{5}\spadcommand{[i for i in 1..10 while i*i < 700]}
\indentrel{3}\begin{verbatim}
   (5)  [1,2,3,4,5,6,7,8,9,10]
                             Type: List PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangItsPageEmpty5}
\begin{paste}{ugLangItsPageEmpty5}{ugLangItsPagePatch5}
\pastebutton{ugLangItsPageEmpty5}{\showpaste}
\tab{5}\spadcommand{[i for i in 1..10 while i*i < 700]}
\end{paste}\end{patch}

\begin{patch}{ugLangItsPagePatch6}
\begin{paste}{ugLangItsPageFull6}{ugLangItsPageEmpty6}
\pastebutton{ugLangItsPageFull6}{\hidepaste}
\tab{5}\spadcommand{[i for i in 1.. while i*i < 700]}
\indentrel{3}\begin{verbatim}
   (6)  [1,2,3,4,5,6,7,8,9,10,...]
                           Type: Stream PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangItsPageEmpty6}
\begin{paste}{ugLangItsPageEmpty6}{ugLangItsPagePatch6}
\pastebutton{ugLangItsPageEmpty6}{\showpaste}
\tab{5}\spadcommand{[i for i in 1.. while i*i < 700]}
```

```
\end{paste}\end{patch}

\begin{patch}{ugLangItsPagePatch7}
\begin{paste}{ugLangItsPageFull7}{ugLangItsPageEmpty7}
\pastebutton{ugLangItsPageFull7}{\hidepaste}
\tab{5}\spadcommand{matrix [[x**i+j for i in 1..3] for j in 10..12]}
\indentrel{3}\begin{verbatim}
                2       3
        x + 10  x  + 10  x  + 10

   (7)          2       3
        x + 11  x  + 11  x  + 11

                2       3
        x + 12  x  + 12  x  + 12
                        Type: Matrix Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangItsPageEmpty7}
\begin{paste}{ugLangItsPageEmpty7}{ugLangItsPagePatch7}
\pastebutton{ugLangItsPageEmpty7}{\showpaste}
\tab{5}\spadcommand{matrix [[x**i+j for i in 1..3] for j in 10..12]}
\end{paste}\end{patch}

\begin{patch}{ugLangItsPagePatch8}
\begin{paste}{ugLangItsPageFull8}{ugLangItsPageEmpty8}
\pastebutton{ugLangItsPageFull8}{\hidepaste}
\tab{5}\spadcommand{[[i/j for i in j+1..] for j in 1..]}
\indentrel{3}\begin{verbatim}
   (8)
   [[2,3,4,5,6,7,8,9,10,11,...],
     3   5   7   9   11
    [-,2,-,3,-,4,--,5,--,6,...],
     2   2   2   2    2
     4 5   7 8   10 11   13
    [-,-,2,-,-,3,--,--,4,--,...],
     3 3   3 3    3  3    3
     5 3 7   9 5 11   13 7
    [-,-,-,2,-,-,--,3,--,-,...],
     4 2 4   4 2  4    4 2
     6 7 8 9   11 12 13 14
    [-,-,-,-,2,--,--,--,--,3,...],
     5 5 5 5    5  5  5  5
     7 4 3 5 11   13 7 5 8
    [-,-,-,-,-,2,--,--,-,-,-,...],
     6 3 2 3  6    6 3 2 3
     8 9 10 11 12 13   15 16 17
    [-,-,--,--,--,--,2,--,--,--,...],
     7 7  7  7  7  7    7  7  7
```

```
     9 5 11 3 13 7 15   17 9
    [,,,,,,,2,,,...],
     8 4  8 2  8 4  8    8 4
     10 11 4 13 14 5 16 17   19
    [,,,,,,,,2,,...],
      9  9 3  9  9 3  9  9    9
     11 6 13 7 3 8 17 9 19
    [,,,,,,,,,2,...], ...]
     10 5 10 5 2 5 10 5 10
                  Type: Stream Stream Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangItsPageEmpty8}
\begin{paste}{ugLangItsPageEmpty8}{ugLangItsPagePatch8}
\pastebutton{ugLangItsPageEmpty8}{\showpaste}
\tab{5}\spadcommand{[[i/j for i in j+1..] for j in 1..]}
\end{paste}\end{patch}

\begin{patch}{ugLangItsPagePatch9}
\begin{paste}{ugLangItsPageFull9}{ugLangItsPageEmpty9}
\pastebutton{ugLangItsPageFull9}{\hidepaste}
\tab{5}\spadcommand{[i/j for i in 3.. by 10 for j in 2..]}
\indentrel{3}\begin{verbatim}
        3 13 23 33 43 53 63 73 83 93
   (9)  [,,,,,,,,,,,...]
        2  3  4  5  6  7  8  9 10 11
                         Type: Stream Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangItsPageEmpty9}
\begin{paste}{ugLangItsPageEmpty9}{ugLangItsPagePatch9}
\pastebutton{ugLangItsPageEmpty9}{\showpaste}
\tab{5}\spadcommand{[i/j for i in 3.. by 10 for j in 2..]}
\end{paste}\end{patch}

\begin{patch}{ugLangItsPagePatch10}
\begin{paste}{ugLangItsPageFull10}{ugLangItsPageEmpty10}
\pastebutton{ugLangItsPageFull10}{\hidepaste}
\tab{5}\spadcommand{[i**j for i in 1..7 for j in 2.. ]}
\indentrel{3}\begin{verbatim}
   (10)  [1,8,81,1024,15625,279936,5764801]
                                  Type: Stream Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangItsPageEmpty10}
\begin{paste}{ugLangItsPageEmpty10}{ugLangItsPagePatch10}
\pastebutton{ugLangItsPageEmpty10}{\showpaste}
```

```
\tab{5}\spadcommand{[i**j for i in 1..7 for j in 2.. ]}
\end{paste}\end{patch}

\begin{patch}{ugLangItsPagePatch11}
\begin{paste}{ugLangItsPageFull11}{ugLangItsPageEmpty11}
\pastebutton{ugLangItsPageFull11}{\hidepaste}
\tab{5}\spadcommand{[[[i,j] for i in 10..15 | prime? i] for j in 17..22 | j = squareFreePart j]}
\indentrel{3}\begin{verbatim}
   (11)
   [[[11,17],[13,17]], [[11,19],[13,19]],
    [[11,21],[13,21]], [[11,22],[13,22]]]
                   Type: List List List PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangItsPageEmpty11}
\begin{paste}{ugLangItsPageEmpty11}{ugLangItsPagePatch11}
\pastebutton{ugLangItsPageEmpty11}{\showpaste}
\tab{5}\spadcommand{[[[i,j] for i in 10..15 | prime? i] for j in 17..22 | j = squareFreePart j]}
\end{paste}\end{patch}
```

An Example: Streams of Primes

— ug05.ht —

```
\begin{page}{ugLangStreamsPrimesPage}{5.6. An Example: Streams of Primes}
\beginscroll

We conclude this chapter with an example of the creation and
manipulation of infinite streams of prime integers.  This might be
useful for experiments with numbers or other applications where you
are using sequences of primes over and over again.  As for all
streams, the stream of primes is only computed as far out as you need.
Once computed, however, all the primes up to that point are saved for
future reference.

Two useful operations provided by the Axiom library are
\axiomFunFrom{prime?}{IntegerPrimesPackage} and
\axiomFunFrom{nextPrime}{IntegerPrimesPackage}.
A straight-forward way to create a stream of
prime numbers is to start with the stream of positive integers
\axiom{[2,..]} and filter out those that are prime.
\xtc{
Create a stream of primes.
}{
```

```
\spadpaste{primes : Stream Integer := [i for i in 2.. | prime? i]}
}
A more elegant way, however, is to use the \axiomFunFrom{generate}{Stream}
operation from \axiomType{Stream}.
Given an initial value \axiom{a} and a function \axiom{f},
\axiomFunFrom{generate}{Stream}
constructs the stream \axiom{[a, f(a), f(f(a)), ...]}.
This function gives you the quickest method of getting the stream of primes.
\xtc{
This is how you use
\axiomFunFrom{generate}{Stream} to
generate an infinite stream of primes.
}{
\spadpaste{primes := generate(nextPrime,2)}
}
\xtc{
Once the stream is generated, you might only be interested in
primes starting at a particular value.
}{
\spadpaste{smallPrimes := [p for p in primes | p > 1000]
\bound{smallPrimes}}
}
\xtc{
Here are the first 11 primes greater than 1000.
}{
\spadpaste{[p for p in smallPrimes for i in 1..11] \free{smallPrimes}}
}
\xtc{
Here is a stream of primes between 1000 and 1200.
}{
\spadpaste{[p for p in smallPrimes while p < 1200] \free{smallPrimes}}
}
\xtc{
To get these expanded into a finite stream,
you call \axiomFunFrom{complete}{Stream} on the stream.
}{
\spadpaste{complete \%}
}
\xtc{
Twin primes are consecutive odd number pairs which are prime.
Here is the stream of twin primes.
}{
\spadpaste{twinPrimes := [[p,p+2] for p in primes | prime?(p + 2)]}
}
\xtc{
Since we already have the primes computed we can
avoid the call to \axiomFunFrom{prime?}{IntegerPrimesPackage}
by using a double iteration.
This time we'll just generate a stream of the first of the twin primes.
}{
```

```
\spadpaste{
firstOfTwins:= [p for p in primes for q in rest primes | q=p+2]}
}

Let's try to compute the infinite stream of triplet primes,
the set of primes \axiom{p} such that \axiom{[p,p+2,p+4]}
are primes. For example, \axiom{[3,5,7]} is a triple prime.
We could do this by a triple \axiom{for} iteration.
A more economical way is to use \userfun{firstOfTwins}.
This time however, put a semicolon at the end of the line.

\xtc{Create the stream of firstTriplets.
Put a semicolon at the end so that no
elements are computed.
}{
\spadpaste{firstTriplets :=
[p for p in firstOfTwins for q in rest firstOfTwins | q = p+2];}
}

What happened?  As you know, by default Axiom displays the first ten
elements of a stream when you first display it.  And, therefore, it
needs to compute them!  If you want {\it no} elements computed, just
terminate the expression by a semicolon (\axiomSyntax{;}).\footnote{
Why does this happen? The semi-colon prevents the display of the
result of evaluating the expression.  Since no stream elements are
needed for display (or anything else, so far), none are computed.
}

\xtc{
Compute the first triplet prime.
}{
\spadpaste{firstTriplets.1}
}

If you want to compute another, just ask for it.
But wait a second!
Given three consecutive odd integers, one of them must be divisible
by 3. Thus there is only one triplet prime.
But suppose that you did not know this and wanted to know what was the
tenth triplet prime.
\begin{verbatim}
firstTriples.10
\end{verbatim}
To compute the tenth triplet prime, Axiom first must compute the second,
the third, and so on.
But since there isn't even a second triplet prime, Axiom will
compute forever.
Nonetheless, this effort can produce a useful result.
After waiting a bit, hit
\texht{\fbox{\bf Ctrl}--\fbox{\bf c}}{{\bf Ctrl-c}}.
```

```
The system responds as follows.
\begin{verbatim}
   >> System error:
   Console interrupt.
   You are being returned to the top level of
   the interpreter.
\end{verbatim}
Let's say that you want to know how many primes have been computed.
Issue
\begin{verbatim}
numberOfComputedEntries primes
\end{verbatim}
and, for this discussion, let's say that the result is \axiom{2045.}
\xtc{
How big is the \eth{\axiom{2045}} prime?
}{
\spadpaste{primes.2045}
}

What you have learned is that there are no triplet primes between 5
and 17837.  Although this result is well known (some might even say
trivial), there are many experiments you could make where the result
is not known.  What you see here is a paradigm for testing of
hypotheses.  Here our hypothesis could have been: ``there is more than
one triplet prime.''  We have tested this hypothesis for 17837 cases.
With streams, you can let your machine run, interrupt it to see how
far it has progressed, then start it up and let it continue from where
it left off.

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugLangStreamsPrimesPagePatch1}
\begin{paste}{ugLangStreamsPrimesPageFull1}{ugLangStreamsPrimesPageEmpty1}
\pastebutton{ugLangStreamsPrimesPageFull1}{\hidepaste}
\tab{5}\spadcommand{primes : Stream Integer := [i for i in 2.. | prime? i]}
\indentrel{3}\begin{verbatim}
   (1)  [2,3,5,7,11,13,17,19,23,29,...]
                                   Type: Stream Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangStreamsPrimesPageEmpty1}
\begin{paste}{ugLangStreamsPrimesPageEmpty1}{ugLangStreamsPrimesPagePatch1}
\pastebutton{ugLangStreamsPrimesPageEmpty1}{\showpaste}
\tab{5}\spadcommand{primes : Stream Integer := [i for i in 2.. | prime? i]}
\end{paste}\end{patch}

\begin{patch}{ugLangStreamsPrimesPagePatch2}
```

```
\begin{paste}{ugLangStreamsPrimesPageFull2}{ugLangStreamsPrimesPageEmpty2}
\pastebutton{ugLangStreamsPrimesPageFull2}{\hidepaste}
\tab{5}\spadcommand{primes := generate(nextPrime,2)}
\indentrel{3}\begin{verbatim}
   (2)  [2,3,5,7,11,13,17,19,23,29,...]
                                Type: Stream Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangStreamsPrimesPageEmpty2}
\begin{paste}{ugLangStreamsPrimesPageEmpty2}{ugLangStreamsPrimesPagePatch2}
\pastebutton{ugLangStreamsPrimesPageEmpty2}{\showpaste}
\tab{5}\spadcommand{primes := generate(nextPrime,2)}
\end{paste}\end{patch}

\begin{patch}{ugLangStreamsPrimesPagePatch3}
\begin{paste}{ugLangStreamsPrimesPageFull3}{ugLangStreamsPrimesPageEmpty3}
\pastebutton{ugLangStreamsPrimesPageFull3}{\hidepaste}
\tab{5}\spadcommand{smallPrimes := [p for p in primes | p > 1000]\bound{smallPrimes }}
\indentrel{3}\begin{verbatim}
   (3)
   [1009,1013,1019,1021,1031,1033,1039,1049,1051,1061,...]
                                Type: Stream Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangStreamsPrimesPageEmpty3}
\begin{paste}{ugLangStreamsPrimesPageEmpty3}{ugLangStreamsPrimesPagePatch3}
\pastebutton{ugLangStreamsPrimesPageEmpty3}{\showpaste}
\tab{5}\spadcommand{smallPrimes := [p for p in primes | p > 1000]\bound{smallPrimes }}
\end{paste}\end{patch}

\begin{patch}{ugLangStreamsPrimesPagePatch4}
\begin{paste}{ugLangStreamsPrimesPageFull4}{ugLangStreamsPrimesPageEmpty4}
\pastebutton{ugLangStreamsPrimesPageFull4}{\hidepaste}
\tab{5}\spadcommand{[p for p in smallPrimes for i in 1..11]\free{smallPrimes }}
\indentrel{3}\begin{verbatim}
   (4)
   [1009,1013,1019,1021,1031,1033,1039,1049,1051,1061,...]
                                Type: Stream Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangStreamsPrimesPageEmpty4}
\begin{paste}{ugLangStreamsPrimesPageEmpty4}{ugLangStreamsPrimesPagePatch4}
\pastebutton{ugLangStreamsPrimesPageEmpty4}{\showpaste}
\tab{5}\spadcommand{[p for p in smallPrimes for i in 1..11]\free{smallPrimes }}
\end{paste}\end{patch}

\begin{patch}{ugLangStreamsPrimesPagePatch5}
```

```
\begin{paste}{ugLangStreamsPrimesPageFull5}{ugLangStreamsPrimesPageEmpty5}
\pastebutton{ugLangStreamsPrimesPageFull5}{\hidepaste}
\tab{5}\spadcommand{[p for p in smallPrimes while p < 1200]\free{smallPrimes }}
\indentrel{3}\begin{verbatim}
   (5)
   [1009,1013,1019,1021,1031,1033,1039,1049,1051,1061,...]
                                  Type: Stream Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangStreamsPrimesPageEmpty5}
\begin{paste}{ugLangStreamsPrimesPageEmpty5}{ugLangStreamsPrimesPagePatch5}
\pastebutton{ugLangStreamsPrimesPageEmpty5}{\showpaste}
\tab{5}\spadcommand{[p for p in smallPrimes while p < 1200]\free{smallPrimes }}
\end{paste}\end{patch}

\begin{patch}{ugLangStreamsPrimesPagePatch6}
\begin{paste}{ugLangStreamsPrimesPageFull6}{ugLangStreamsPrimesPageEmpty6}
\pastebutton{ugLangStreamsPrimesPageFull6}{\hidepaste}
\tab{5}\spadcommand{complete \%}
\indentrel{3}\begin{verbatim}
   (6)
   [1009,1013,1019,1021,1031,1033,1039,1049,1051,1061,...]
                                  Type: Stream Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangStreamsPrimesPageEmpty6}
\begin{paste}{ugLangStreamsPrimesPageEmpty6}{ugLangStreamsPrimesPagePatch6}
\pastebutton{ugLangStreamsPrimesPageEmpty6}{\showpaste}
\tab{5}\spadcommand{complete \%}
\end{paste}\end{patch}

\begin{patch}{ugLangStreamsPrimesPagePatch7}
\begin{paste}{ugLangStreamsPrimesPageFull7}{ugLangStreamsPrimesPageEmpty7}
\pastebutton{ugLangStreamsPrimesPageFull7}{\hidepaste}
\tab{5}\spadcommand{twinPrimes := [[p,p+2] for p in primes | prime?(p + 2)]}
\indentrel{3}\begin{verbatim}
   (7)
   [[3,5], [5,7], [11,13], [17,19], [29,31], [41,43],
    [59,61], [71,73], [101,103], [107,109], ...]
                             Type: Stream List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangStreamsPrimesPageEmpty7}
\begin{paste}{ugLangStreamsPrimesPageEmpty7}{ugLangStreamsPrimesPagePatch7}
\pastebutton{ugLangStreamsPrimesPageEmpty7}{\showpaste}
\tab{5}\spadcommand{twinPrimes := [[p,p+2] for p in primes | prime?(p + 2)]}
\end{paste}\end{patch}
```

```
\begin{patch}{ugLangStreamsPrimesPagePatch8}
\begin{paste}{ugLangStreamsPrimesPageFull8}{ugLangStreamsPrimesPageEmpty8}
\pastebutton{ugLangStreamsPrimesPageFull8}{\hidepaste}
\tab{5}\spadcommand{firstOfTwins:= [p for p in primes for q in rest primes | q=p+2]}
\indentrel{3}\begin{verbatim}
   (8)  [3,5,11,17,29,41,59,71,101,107,...]
                                   Type: Stream Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangStreamsPrimesPageEmpty8}
\begin{paste}{ugLangStreamsPrimesPageEmpty8}{ugLangStreamsPrimesPagePatch8}
\pastebutton{ugLangStreamsPrimesPageEmpty8}{\showpaste}
\tab{5}\spadcommand{firstOfTwins:= [p for p in primes for q in rest primes | q=p+2]}
\end{paste}\end{patch}

\begin{patch}{ugLangStreamsPrimesPagePatch9}
\begin{paste}{ugLangStreamsPrimesPageFull9}{ugLangStreamsPrimesPageEmpty9}
\pastebutton{ugLangStreamsPrimesPageFull9}{\hidepaste}
\tab{5}\spadcommand{firstTriplets := [p for p in firstOfTwins for q in rest firstOfTwins | q = p+2];}
\indentrel{3}\begin{verbatim}
                                   Type: Stream Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangStreamsPrimesPageEmpty9}
\begin{paste}{ugLangStreamsPrimesPageEmpty9}{ugLangStreamsPrimesPagePatch9}
\pastebutton{ugLangStreamsPrimesPageEmpty9}{\showpaste}
\tab{5}\spadcommand{firstTriplets := [p for p in firstOfTwins for q in rest firstOfTwins | q = p+2];}
\end{paste}\end{patch}

\begin{patch}{ugLangStreamsPrimesPagePatch10}
\begin{paste}{ugLangStreamsPrimesPageFull10}{ugLangStreamsPrimesPageEmpty10}
\pastebutton{ugLangStreamsPrimesPageFull10}{\hidepaste}
\tab{5}\spadcommand{firstTriplets.1}
\indentrel{3}\begin{verbatim}
   (10)  3
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangStreamsPrimesPageEmpty10}
\begin{paste}{ugLangStreamsPrimesPageEmpty10}{ugLangStreamsPrimesPagePatch10}
\pastebutton{ugLangStreamsPrimesPageEmpty10}{\showpaste}
\tab{5}\spadcommand{firstTriplets.1}
\end{paste}\end{patch}

\begin{patch}{ugLangStreamsPrimesPagePatch11}
\begin{paste}{ugLangStreamsPrimesPageFull11}{ugLangStreamsPrimesPageEmpty11}
```

```
\pastebutton{ugLangStreamsPrimesPageFull11}{\hidepaste}
\tab{5}\spadcommand{primes.2045}
\indentrel{3}\begin{verbatim}
   (11)  17837
                                      Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugLangStreamsPrimesPageEmpty11}
\begin{paste}{ugLangStreamsPrimesPageEmpty11}{ugLangStreamsPrimesPagePatch11}
\pastebutton{ugLangStreamsPrimesPageEmpty11}{\showpaste}
\tab{5}\spadcommand{primes.2045}
\end{paste}\end{patch}
```

Chapter 10

Users Guide Chapter 6 (ug06.ht)

User-Defined Functions, Macros and Rules

— **ug06.ht** —

```
\begin{page}{ugUserPage}{6. User-Defined Functions, Macros and Rules}
```

```
\beginscroll

In this chapter we show you how to write functions and macros,
and we explain how Axiom looks for and applies them.
We show some simple one-line examples of functions, together
with larger ones that are defined piece-by-piece or through the use of
piles.

\beginmenu
    \menudownlink{{6.1. Functions vs. Macros}}{ugUserFunMacPage}
    \menudownlink{{6.2. Macros}}{ugUserMacrosPage}
    \menudownlink{{6.3. Introduction to Functions}}{ugUserIntroPage}
    \menudownlink{{6.4. Declaring the Type of Functions}}
{ugUserDeclarePage}
    \menudownlink{{6.5. One-Line Functions}}{ugUserOnePage}
    \menudownlink{{6.6. Declared vs. Undeclared Functions}}
{ugUserDecUndecPage}
    \menudownlink{{6.7. Functions vs. Operations}}{ugUserDecOpersPage}
    \menudownlink{
{6.8. Delayed Assignments vs. Functions with No Arguments}}
{ugUserDelayPage}
    \menudownlink{{6.9. How Axiom Determines What Function to Use}}
{ugUserUsePage}
    \menudownlink{{6.10. Compiling vs. Interpreting}}
{ugUserCompIntPage}
    \menudownlink{{6.11. Piece-Wise Function Definitions}}
{ugUserPiecePage}
    \menudownlink{{6.12. Caching Previously Computed Results}}
{ugUserCachePage}
    \menudownlink{{6.13. Recurrence Relations}}{ugUserRecurPage}
    \menudownlink{{6.14. Making Functions from Objects}}
{ugUserMakePage}
    \menudownlink{{6.15. Functions Defined with Blocks}}
{ugUserBlocksPage}
    \menudownlink{{6.16. Free and Local Variables}}{ugUserFreeLocalPage}
    \menudownlink{{6.17. Anonymous Functions}}{ugUserAnonPage}
    \menudownlink{{6.18. Example: A Database}}{ugUserDatabasePage}
    \menudownlink{{6.19. Example: A Famous Triangle}}
{ugUserTrianglePage}
    \menudownlink{{6.20. Example: Testing for Palindromes}}
{ugUserPalPage}
    \menudownlink{{6.21. Rules and Pattern Matching}}{ugUserRulesPage}
\endmenu
\endscroll
\autobuttons
\end{page}
```

———

Functions vs. Macros

⇒ "notitle" (ugUserAnonPage) 10 on page 1921
⇒ "notitle" (ExitXmpPage) 3.28 on page 398
⇒ "notitle" (VoidXmpPage) 3.115 on page 1357

— **ug06.ht** —

```
\begin{page}{ugUserFunMacPage}{6.1. Functions vs. Macros}
\beginscroll

A function is a program to perform some
computation.
Most functions have names so that it is easy to refer to them.
A simple example of a function is one named
\axiomFunFrom{abs}{Integer} which
computes the absolute value of an integer.
%
\xtc{
This is a use of the ``absolute value'' library function for integers.
}{
\spadpaste{abs(-8)}
}
\xtc{
This is an unnamed function that does the same thing, using the
``maps-to'' syntax \axiomSyntax{+->} that we discuss in
\downlink{``Anonymous Functions''}{ugUserAnonPage}
in Section 6.17\ignore{ugUserAnon}.
}{
\spadpaste{(x +-> if x < 0 then -x else x)(-8)}
}
%
Functions can be used alone or serve as the building blocks for larger
programs.
Usually they return a value that you might want to use in the next stage
of a computation, but not always (for example, see
\downlink{`Exit'}{ExitXmpPage}\ignore{Exit} and
\downlink{`Void'}{VoidXmpPage}\ignore{Void}).
They may also read data from your keyboard, move information from one
place to another, or format and display results on your screen.

In Axiom, as in mathematics, functions are usually
\spadglossSee{parameterized}{parameterized form}.  Each time you {\it
call} (some people say \spadgloss{apply} or
\spadglossSee{invoke}{invocation}) a function, you give values to the
parameters (variables).  Such a value is called an
\spadgloss{argument} of the function.  Axiom uses the arguments for
the computation.  In this way you get different results depending on
what you ``feed'' the function.
```

```
Functions can have local variables or refer to global variables in the
workspace.
Axiom can often \spadglossSee{compile}{compiler} functions so that
they execute very efficiently.
Functions can be passed as arguments to other functions.

Macros are textual substitutions.  They are used to clarify the
meaning of constants or expressions and to be templates for frequently
used expressions.  Macros can be parameterized but they are not
objects that can be passed as arguments to functions.  In effect,
macros are extensions to the Axiom expression parser.

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugUserFunMacPagePatch1}
\begin{paste}{ugUserFunMacPageFull1}{ugUserFunMacPageEmpty1}
\pastebutton{ugUserFunMacPageFull1}{\hidepaste}
\tab{5}\spadcommand{abs(-8)}
\indentrel{3}\begin{verbatim}
   (1)  8
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserFunMacPageEmpty1}
\begin{paste}{ugUserFunMacPageEmpty1}{ugUserFunMacPagePatch1}
\pastebutton{ugUserFunMacPageEmpty1}{\showpaste}
\tab{5}\spadcommand{abs(-8)}
\end{paste}\end{patch}

\begin{patch}{ugUserFunMacPagePatch2}
\begin{paste}{ugUserFunMacPageFull2}{ugUserFunMacPageEmpty2}
\pastebutton{ugUserFunMacPageFull2}{\hidepaste}
\tab{5}\spadcommand{(x +-> if x < 0 then -x else x)(-8)}
\indentrel{3}\begin{verbatim}
   (2)  8
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserFunMacPageEmpty2}
\begin{paste}{ugUserFunMacPageEmpty2}{ugUserFunMacPagePatch2}
\pastebutton{ugUserFunMacPageEmpty2}{\showpaste}
\tab{5}\spadcommand{(x +-> if x < 0 then -x else x)(-8)}
\end{paste}\end{patch}
```

———-

Macros

— **ug06.ht** —

```
\begin{page}{ugUserMacrosPage}{6.2. Macros}
\beginscroll

A \spadgloss{macro} provides general textual substitution of
an Axiom expression for a name.
You can think of a macro as being a generalized abbreviation.
You can only have one macro in your workspace with
a given name, no matter how many arguments it has.

\beginImportant
The two general forms for macros are
\centerline{{{\tt macro} {\it name} {\tt ==} {\it body} }}
\centerline{{{\tt macro} {\it name(arg1,...)} {\tt ==} {\it body}}}
where the body of the macro can be any Axiom expression.
\endImportant

%
\xtc{
For example, suppose you decided that you
like to use \axiom{df} for \axiomFun{D}.
You define the macro \axiom{df} like this.
}{
\spadpaste{macro df == D \bound{df}}
}
\xtc{
Whenever you type \axiom{df}, the system expands it to
\axiomFun{D}.
}{
\spadpaste{df(x**2 + x + 1,x) \free{df}}
}
\xtc{
Macros can be parameterized and so can be used for many different
kinds of objects.
}{
\spadpaste{macro ff(x) == x**2 + 1 \bound{ff}}
}
\xtc{
Apply it to a number, a symbol, or an expression.
}{
\spadpaste{ff z \free{ff}}
}
\xtc{
```

```
Macros can also be nested, but you get an error message if you
run out of space because of an infinite nesting loop.
}{
\spadpaste{macro gg(x) == ff(2*x - 2/3) \bound{gg}\free{ff}}
}
\xtc{
This new macro is fine as it does not produce a loop.
}{
\spadpaste{gg(1/w) \free{gg}}
}
%
\xtc{
This, however, loops since \axiom{gg} is
defined in terms of \axiom{ff}.
}{
\spadpaste{macro ff(x) == gg(-x) \free{gg}}
}
\xtc{
The body of a macro can be a block.
}{
\spadpaste{macro next == (past := present; present := future;
future := past + present) \bound{next}}
}
\xtc{
Before entering \axiom{next}, we need
values for \axiom{present} and \axiom{future}.
}{
\spadpaste{present : Integer := 0 \bound{present}}
}
\xtc{
}{
\spadpaste{future : Integer := 1 \bound{future}}
}
\xtc{
Repeatedly evaluating \axiom{next} produces the next Fibonacci number.
}{
\spadpaste{next \free{future}\free{present}}
}
\xtc{
And the next one.
}{
\spadpaste{next \free{future}\free{present}}
}
\xtc{
Here is the infinite stream of the rest of the Fibonacci numbers.
}{
\spadpaste{[next for i in 1..] \free{future}\free{present}}
}
\xtc{
Bundle all the above lines into a single macro.
```

```
}{
\begin{spadsrc}[\bound{fibstr}]
macro fibStream ==
  present : Integer := 1
  future : Integer := 1
  [next for i in 1..] where
    macro next ==
      past := present
      present := future
      future := past + present
\end{spadsrc}
}
\xtc{
Use \axiomFunFrom{concat}{Stream} to start with the first two
Fibonacci numbers.
}{
\spadpaste{concat([0,1],fibStream) \free{fibstr}}
}
\xtc{
An easier way to compute these numbers is to
use the library operation \axiomFun{fibonacci}.
}{
\spadpaste{[fibonacci i for i in 1..]}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugUserMacrosPagePatch1}
\begin{paste}{ugUserMacrosPageFull1}{ugUserMacrosPageEmpty1}
\pastebutton{ugUserMacrosPageFull1}{\hidepaste}
\tab{5}\spadcommand{macro df == D\bound{df }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserMacrosPageEmpty1}
\begin{paste}{ugUserMacrosPageEmpty1}{ugUserMacrosPagePatch1}
\pastebutton{ugUserMacrosPageEmpty1}{\showpaste}
\tab{5}\spadcommand{macro df == D\bound{df }}
\end{paste}\end{patch}

\begin{patch}{ugUserMacrosPagePatch2}
\begin{paste}{ugUserMacrosPageFull2}{ugUserMacrosPageEmpty2}
\pastebutton{ugUserMacrosPageFull2}{\hidepaste}
\tab{5}\spadcommand{df(x**2 + x + 1,x)\free{df }}
\indentrel{3}\begin{verbatim}
   (2)  2x + 1
```

```
                              Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserMacrosPageEmpty2}
\begin{paste}{ugUserMacrosPageEmpty2}{ugUserMacrosPagePatch2}
\pastebutton{ugUserMacrosPageEmpty2}{\showpaste}
\tab{5}\spadcommand{df(x**2 + x + 1,x)\free{df }}
\end{paste}\end{patch}

\begin{patch}{ugUserMacrosPagePatch3}
\begin{paste}{ugUserMacrosPageFull3}{ugUserMacrosPageEmpty3}
\pastebutton{ugUserMacrosPageFull3}{\hidepaste}
\tab{5}\spadcommand{macro ff(x) == x**2 + 1\bound{ff }}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserMacrosPageEmpty3}
\begin{paste}{ugUserMacrosPageEmpty3}{ugUserMacrosPagePatch3}
\pastebutton{ugUserMacrosPageEmpty3}{\showpaste}
\tab{5}\spadcommand{macro ff(x) == x**2 + 1\bound{ff }}
\end{paste}\end{patch}

\begin{patch}{ugUserMacrosPagePatch4}
\begin{paste}{ugUserMacrosPageFull4}{ugUserMacrosPageEmpty4}
\pastebutton{ugUserMacrosPageFull4}{\hidepaste}
\tab{5}\spadcommand{ff z\free{ff }}
\indentrel{3}\begin{verbatim}
         2
   (4)  z  + 1
                              Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserMacrosPageEmpty4}
\begin{paste}{ugUserMacrosPageEmpty4}{ugUserMacrosPagePatch4}
\pastebutton{ugUserMacrosPageEmpty4}{\showpaste}
\tab{5}\spadcommand{ff z\free{ff }}
\end{paste}\end{patch}

\begin{patch}{ugUserMacrosPagePatch5}
\begin{paste}{ugUserMacrosPageFull5}{ugUserMacrosPageEmpty5}
\pastebutton{ugUserMacrosPageFull5}{\hidepaste}
\tab{5}\spadcommand{macro gg(x) == ff(2*x - 2/3)\bound{gg }\free{ff }}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugUserMacrosPageEmpty5}
\begin{paste}{ugUserMacrosPageEmpty5}{ugUserMacrosPagePatch5}
\pastebutton{ugUserMacrosPageEmpty5}{\showpaste}
\tab{5}\spadcommand{macro gg(x) == ff(2*x - 2/3)\bound{gg }\free{ff }}
\end{paste}\end{patch}

\begin{patch}{ugUserMacrosPagePatch6}
\begin{paste}{ugUserMacrosPageFull6}{ugUserMacrosPageEmpty6}
\pastebutton{ugUserMacrosPageFull6}{\hidepaste}
\tab{5}\spadcommand{gg(1/w)\free{gg }}
\indentrel{3}\begin{verbatim}
           2
        13w  - 24w + 36
   (6)
                2
              9w
                      Type: Fraction Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserMacrosPageEmpty6}
\begin{paste}{ugUserMacrosPageEmpty6}{ugUserMacrosPagePatch6}
\pastebutton{ugUserMacrosPageEmpty6}{\showpaste}
\tab{5}\spadcommand{gg(1/w)\free{gg }}
\end{paste}\end{patch}

\begin{patch}{ugUserMacrosPagePatch7}
\begin{paste}{ugUserMacrosPageFull7}{ugUserMacrosPageEmpty7}
\pastebutton{ugUserMacrosPageFull7}{\hidepaste}
\tab{5}\spadcommand{macro ff(x) == gg(-x)\free{gg }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserMacrosPageEmpty7}
\begin{paste}{ugUserMacrosPageEmpty7}{ugUserMacrosPagePatch7}
\pastebutton{ugUserMacrosPageEmpty7}{\showpaste}
\tab{5}\spadcommand{macro ff(x) == gg(-x)\free{gg }}
\end{paste}\end{patch}

\begin{patch}{ugUserMacrosPagePatch8}
\begin{paste}{ugUserMacrosPageFull8}{ugUserMacrosPageEmpty8}
\pastebutton{ugUserMacrosPageFull8}{\hidepaste}
\tab{5}\spadcommand{macro next == (past := present; present := future; future := past + present)\bound{next
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugUserMacrosPageEmpty8}
\begin{paste}{ugUserMacrosPageEmpty8}{ugUserMacrosPagePatch8}
\pastebutton{ugUserMacrosPageEmpty8}{\showpaste}
\tab{5}\spadcommand{macro next == (past := present; present := future; future := past + present)
\end{paste}\end{patch}

\begin{patch}{ugUserMacrosPagePatch9}
\begin{paste}{ugUserMacrosPageFull9}{ugUserMacrosPageEmpty9}
\pastebutton{ugUserMacrosPageFull9}{\hidepaste}
\tab{5}\spadcommand{present : Integer := 0\bound{present }}
\indentrel{3}\begin{verbatim}
   (9)  0
                                          Type: Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserMacrosPageEmpty9}
\begin{paste}{ugUserMacrosPageEmpty9}{ugUserMacrosPagePatch9}
\pastebutton{ugUserMacrosPageEmpty9}{\showpaste}
\tab{5}\spadcommand{present : Integer := 0\bound{present }}
\end{paste}\end{patch}

\begin{patch}{ugUserMacrosPagePatch10}
\begin{paste}{ugUserMacrosPageFull10}{ugUserMacrosPageEmpty10}
\pastebutton{ugUserMacrosPageFull10}{\hidepaste}
\tab{5}\spadcommand{future : Integer := 1\bound{future }}
\indentrel{3}\begin{verbatim}
   (10)  1
                                          Type: Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserMacrosPageEmpty10}
\begin{paste}{ugUserMacrosPageEmpty10}{ugUserMacrosPagePatch10}
\pastebutton{ugUserMacrosPageEmpty10}{\showpaste}
\tab{5}\spadcommand{future : Integer := 1\bound{future }}
\end{paste}\end{patch}

\begin{patch}{ugUserMacrosPagePatch11}
\begin{paste}{ugUserMacrosPageFull11}{ugUserMacrosPageEmpty11}
\pastebutton{ugUserMacrosPageFull11}{\hidepaste}
\tab{5}\spadcommand{next\free{future }\free{present }}
\indentrel{3}\begin{verbatim}
   (11)  1
                                          Type: Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserMacrosPageEmpty11}
```

```
\begin{paste}{ugUserMacrosPageEmpty11}{ugUserMacrosPagePatch11}
\pastebutton{ugUserMacrosPageEmpty11}{\showpaste}
\tab{5}\spadcommand{next\free{future }\free{present }}
\end{paste}\end{patch}

\begin{patch}{ugUserMacrosPagePatch12}
\begin{paste}{ugUserMacrosPageFull12}{ugUserMacrosPageEmpty12}
\pastebutton{ugUserMacrosPageFull12}{\hidepaste}
\tab{5}\spadcommand{next\free{future }\free{present }}
\indentrel{3}\begin{verbatim}
   (12)  2
                                          Type: Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserMacrosPageEmpty12}
\begin{paste}{ugUserMacrosPageEmpty12}{ugUserMacrosPagePatch12}
\pastebutton{ugUserMacrosPageEmpty12}{\showpaste}
\tab{5}\spadcommand{next\free{future }\free{present }}
\end{paste}\end{patch}

\begin{patch}{ugUserMacrosPagePatch13}
\begin{paste}{ugUserMacrosPageFull13}{ugUserMacrosPageEmpty13}
\pastebutton{ugUserMacrosPageFull13}{\hidepaste}
\tab{5}\spadcommand{[next for i in 1..]\free{future }\free{present }}
\indentrel{3}\begin{verbatim}
   (13)  [3,5,8,13,21,34,55,89,144,233,...]
                                   Type: Stream Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserMacrosPageEmpty13}
\begin{paste}{ugUserMacrosPageEmpty13}{ugUserMacrosPagePatch13}
\pastebutton{ugUserMacrosPageEmpty13}{\showpaste}
\tab{5}\spadcommand{[next for i in 1..]\free{future }\free{present }}
\end{paste}\end{patch}

\begin{patch}{ugUserMacrosPagePatch14}
\begin{paste}{ugUserMacrosPageFull14}{ugUserMacrosPageEmpty14}
\pastebutton{ugUserMacrosPageFull14}{\hidepaste}
\tab{5}\spadcommand{macro fibStream ==
  present : Integer := 1
  future : Integer := 1
  [next for i in 1..] where
    macro next ==
      past := present
      present := future
      future := past + present
\bound{fibstr }}
\indentrel{3}\begin{verbatim}
```

```
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserMacrosPageEmpty14}
\begin{paste}{ugUserMacrosPageEmpty14}{ugUserMacrosPagePatch14}
\pastebutton{ugUserMacrosPageEmpty14}{\showpaste}
\tab{5}\spadcommand{macro fibStream ==
  present : Integer := 1
  future : Integer := 1
  [next for i in 1..] where
    macro next ==
      past := present
      present := future
      future := past + present
\bound{fibstr }}
\end{paste}\end{patch}

\begin{patch}{ugUserMacrosPagePatch15}
\begin{paste}{ugUserMacrosPageFull15}{ugUserMacrosPageEmpty15}
\pastebutton{ugUserMacrosPageFull15}{\hidepaste}
\tab{5}\spadcommand{concat([0,1],fibStream)\free{fibstr }}
\indentrel{3}\begin{verbatim}
   (15)  [0,1,2,3,5,8,13,21,34,55,...]
                                   Type: Stream Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserMacrosPageEmpty15}
\begin{paste}{ugUserMacrosPageEmpty15}{ugUserMacrosPagePatch15}
\pastebutton{ugUserMacrosPageEmpty15}{\showpaste}
\tab{5}\spadcommand{concat([0,1],fibStream)\free{fibstr }}
\end{paste}\end{patch}

\begin{patch}{ugUserMacrosPagePatch16}
\begin{paste}{ugUserMacrosPageFull16}{ugUserMacrosPageEmpty16}
\pastebutton{ugUserMacrosPageFull16}{\hidepaste}
\tab{5}\spadcommand{[fibonacci i for i in 1..]}
\indentrel{3}\begin{verbatim}
   (16)  [1,1,2,3,5,8,13,21,34,55,...]
                                   Type: Stream Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserMacrosPageEmpty16}
\begin{paste}{ugUserMacrosPageEmpty16}{ugUserMacrosPagePatch16}
\pastebutton{ugUserMacrosPageEmpty16}{\showpaste}
\tab{5}\spadcommand{[fibonacci i for i in 1..]}
\end{paste}\end{patch}
```

———-

Introduction to Functions

⇒ “notitle” (ugTypesPkgCallPage) 7 on page 1686
⇒ “notitle” (ugUserAnonPage) 10 on page 1921

— **ug06.ht** —

```
\begin{page}{ugUserIntroPage}{6.3. Introduction to Functions}
\beginscroll

Each name in your workspace can refer to a single object.
This may be any kind of object including a function.
You can use interactively any function from the library or any that you
define in the workspace.
In the library the same name can have very many functions, but you
can have only one function with a given name, although it can have any
number of arguments that you choose.

If you define a function in the workspace that has the same name and number
of arguments as one in the library, then your definition takes precedence.
In fact, to get the library function you must
\spadglossSee{package-call}{package call} it (see
\downlink{‘‘Package Calling and Target Types’’}{ugTypesPkgCallPage}
in Section 2.9\ignore{ugTypesPkgCall}).

To use a function in Axiom, you apply it to its arguments.
Most functions are applied by entering the name of the function followed by
its argument or arguments.
\xtc{
}{
\spadpaste{factor(12)}
}
%
\xtc{
Some functions like
\axiomOp{+} have {\it infix} \spadgloss{operators} as names.
}{
\spadpaste{3 + 4}
}
\xtc{
The function \axiomOp{+} has two arguments.
When you give it more than two arguments,
Axiom groups the arguments to the left.
This expression is equivalent to \axiom{(1 + 2) + 7}.
}{
\spadpaste{1 + 2 + 7}
}
```

```
All operations, including infix operators, can be written in prefix form,
that is, with the operation name followed by the arguments
in parentheses.
For example, \axiom{2 + 3} can alternatively be written as \axiom{+(2,3)}.
But \axiom{+(2,3,4)} is an error since \axiomOp{+}
takes only two arguments.

Prefix operations are generally applied before the infix operation.
Thus \axiom{factorial 3 + 1} means \axiom{factorial(3) + 1} producing
\axiom{7}, and
\axiom{- 2 + 5} means \axiom{(-2) + 5} producing \axiom{3}.
An example of a prefix operator is prefix \axiomOp{-}.
For example, \axiom{- 2 + 5} converts to \axiom{(- 2) + 5} producing
the value \axiom{3}.
Any prefix function taking two arguments can be written in
an infix manner by putting an
ampersand (\axiomSyntax{\&}) before the name.
Thus \axiom{D(2*x,x)} can be written as
\axiom{2*x \&D x} returning \axiom{2}.

Every function in Axiom is identified by
a \spadgloss{name} and \spadgloss{type}.\footnote{An exception is
an ``anonymous function''
discussed in
\downlink{``Anonymous Functions''}{ugUserAnonPage}
in Section 6.17\ignore{ugUserAnon}.}
The type of a function is always a mapping of the form
\spadsig{Source}{Target}
where \axiom{Source} and \axiom{Target} are types.
To enter a type from the keyboard, enter the arrow by using
a hyphen \axiomSyntax{-} followed by a greater-than sign
\axiomSyntax{>}, e.g. {\tt Integer -> Integer}.

Let's go back to \axiomOp{+}.
There are many \axiomOp{+} functions in the
Axiom library: one for integers, one for floats, another for
rational numbers, and so on.
These \axiomOp{+} functions have different types and thus are
different functions.
You've seen examples of this \spadgloss{overloading}
before---using the same name for different functions.
Overloading is the rule rather than the exception.
You can add two integers, two polynomials, two matrices or
two power series.
These are all done with the same function name
but with different functions.

\endscroll
\autobuttons
```

```
\end{page}

\begin{patch}{ugUserIntroPagePatch1}
\begin{paste}{ugUserIntroPageFull1}{ugUserIntroPageEmpty1}
\pastebutton{ugUserIntroPageFull1}{\hidepaste}
\tab{5}\spadcommand{factor(12)}
\indentrel{3}\begin{verbatim}
         2
   (1)  2 3
                                  Type: Factored Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserIntroPageEmpty1}
\begin{paste}{ugUserIntroPageEmpty1}{ugUserIntroPagePatch1}
\pastebutton{ugUserIntroPageEmpty1}{\showpaste}
\tab{5}\spadcommand{factor(12)}
\end{paste}\end{patch}

\begin{patch}{ugUserIntroPagePatch2}
\begin{paste}{ugUserIntroPageFull2}{ugUserIntroPageEmpty2}
\pastebutton{ugUserIntroPageFull2}{\hidepaste}
\tab{5}\spadcommand{3 + 4}
\indentrel{3}\begin{verbatim}
   (2)  7
                                   Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserIntroPageEmpty2}
\begin{paste}{ugUserIntroPageEmpty2}{ugUserIntroPagePatch2}
\pastebutton{ugUserIntroPageEmpty2}{\showpaste}
\tab{5}\spadcommand{3 + 4}
\end{paste}\end{patch}

\begin{patch}{ugUserIntroPagePatch3}
\begin{paste}{ugUserIntroPageFull3}{ugUserIntroPageEmpty3}
\pastebutton{ugUserIntroPageFull3}{\hidepaste}
\tab{5}\spadcommand{1 + 2 + 7}
\indentrel{3}\begin{verbatim}
   (3)  10
                                   Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserIntroPageEmpty3}
\begin{paste}{ugUserIntroPageEmpty3}{ugUserIntroPagePatch3}
\pastebutton{ugUserIntroPageEmpty3}{\showpaste}
\tab{5}\spadcommand{1 + 2 + 7}
\end{paste}\end{patch}
```

———-

Declaring the Type of Functions

⇒ "notitle" (ugTypesDeclarePage) 7 on page 1641
— **ug06.ht** —

```
\begin{page}{ugUserDeclarePage}{6.4. Declaring the Type of Functions}
\beginscroll

In \downlink{‘‘Declarations’’}{ugTypesDeclarePage}
in Section 2.3\ignore{ugTypesDeclare} we
discussed how to declare a variable
to restrict the kind of values that can be assigned to it.
In this section we show how to declare a variable that refers to
function objects.

\beginImportant
A function is an object of type
\centerline{{\spadsig{Source}{Type}}}
where \axiom{Source} and \axiom{Target} can be any type.
A common type for \axiom{Source} is
\axiomType{Tuple}(\subscriptIt{T}{1}, \ldots, \subscriptIt{T}{n}),
usually written
(\subscriptIt{T}{1}, \ldots, \subscriptIt{T}{n}),
to indicate a function of \axiom{n} arguments.
\endImportant

\xtc{
If \axiom{g} takes an \axiomType{Integer}, a \axiomType{Float} and
another \axiomType{Integer}, and returns a
\axiomType{String}, the declaration is written this way.
}{
\spadpaste{g: (Integer,Float,Integer) -> String}
}
\xtc{
The types need not be written fully; using abbreviations, the above
declaration is:
}{
\spadpaste{g: (INT,FLOAT,INT) -> STRING}
}
\xtc{
It is possible for a function to take no arguments.
If \axiom{ h} takes no arguments
but returns a \axiomType{Polynomial} \axiomType{Integer}, any
of the following declarations is acceptable.
```

```
}{
\spadpaste{h: () -> POLY INT}
}
\xtc{
}{
\spadpaste{h: () -> Polynomial INT}
}
\xtc{
}{
\spadpaste{h: () -> POLY Integer}
}


\beginImportant
Functions can also be declared when they are being defined.
The syntax for combined declaration/definition is:
\centerline{{\frenchspacing{\tt {\it functionName}(
\subscriptIt{parm}{1}: \subscriptIt{parmType}{1}, \ldots,
\subscriptIt{parm}{N}: \subscriptIt{parmType}{N}):
{\it functionReturnType}}}}
\endImportant

The following definition fragments show how this can be done for
the functions \axiom{g} and \axiom{h} above.
\begin{verbatim}
g(arg1: INT, arg2: FLOAT, arg3: INT): STRING == ...

h(): POLY INT == ...
\end{verbatim}

A current restriction on function declarations is that they must
involve fully specified types (that is, cannot include modes involving
explicit or implicit \axiomSyntax{?}).
For more information on declaring things in general, see
\downlink{``Declarations''}{ugTypesDeclarePage}
in Section 2.3\ignore{ugTypesDeclare}.

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugUserDeclarePagePatch1}
\begin{paste}{ugUserDeclarePageFull1}{ugUserDeclarePageEmpty1}
\pastebutton{ugUserDeclarePageFull1}{\hidepaste}
\tab{5}\spadcommand{g: (Integer,Float,Integer) -> String}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugUserDeclarePageEmpty1}
\begin{paste}{ugUserDeclarePageEmpty1}{ugUserDeclarePagePatch1}
\pastebutton{ugUserDeclarePageEmpty1}{\showpaste}
\tab{5}\spadcommand{g: (Integer,Float,Integer) -> String}
\end{paste}\end{patch}

\begin{patch}{ugUserDeclarePagePatch2}
\begin{paste}{ugUserDeclarePageFull2}{ugUserDeclarePageEmpty2}
\pastebutton{ugUserDeclarePageFull2}{\hidepaste}
\tab{5}\spadcommand{g: (INT,FLOAT,INT) -> STRING}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserDeclarePageEmpty2}
\begin{paste}{ugUserDeclarePageEmpty2}{ugUserDeclarePagePatch2}
\pastebutton{ugUserDeclarePageEmpty2}{\showpaste}
\tab{5}\spadcommand{g: (INT,FLOAT,INT) -> STRING}
\end{paste}\end{patch}

\begin{patch}{ugUserDeclarePagePatch3}
\begin{paste}{ugUserDeclarePageFull3}{ugUserDeclarePageEmpty3}
\pastebutton{ugUserDeclarePageFull3}{\hidepaste}
\tab{5}\spadcommand{h: () -> POLY INT}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserDeclarePageEmpty3}
\begin{paste}{ugUserDeclarePageEmpty3}{ugUserDeclarePagePatch3}
\pastebutton{ugUserDeclarePageEmpty3}{\showpaste}
\tab{5}\spadcommand{h: () -> POLY INT}
\end{paste}\end{patch}

\begin{patch}{ugUserDeclarePagePatch4}
\begin{paste}{ugUserDeclarePageFull4}{ugUserDeclarePageEmpty4}
\pastebutton{ugUserDeclarePageFull4}{\hidepaste}
\tab{5}\spadcommand{h: () -> Polynomial INT}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserDeclarePageEmpty4}
\begin{paste}{ugUserDeclarePageEmpty4}{ugUserDeclarePagePatch4}
\pastebutton{ugUserDeclarePageEmpty4}{\showpaste}
\tab{5}\spadcommand{h: () -> Polynomial INT}
\end{paste}\end{patch}
```

```
\begin{patch}{ugUserDeclarePagePatch5}
\begin{paste}{ugUserDeclarePageFull5}{ugUserDeclarePageEmpty5}
\pastebutton{ugUserDeclarePageFull5}{\hidepaste}
\tab{5}\spadcommand{h: () -> POLY Integer}
\indentrel{3}\begin{verbatim}
                                          Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserDeclarePageEmpty5}
\begin{paste}{ugUserDeclarePageEmpty5}{ugUserDeclarePagePatch5}
\pastebutton{ugUserDeclarePageEmpty5}{\showpaste}
\tab{5}\spadcommand{h: () -> POLY Integer}
\end{paste}\end{patch}
```

———-

One-Line Functions

— ug06.ht —

```
\begin{page}{ugUserOnePage}{6.5. One-Line Functions}
\beginscroll

As you use Axiom, you will find that you will write many short functions
to codify sequences of operations that you often perform.
In this section we write some simple one-line functions.

\xtc{
This is a simple recursive factorial function for positive integers.
}{
\spadpaste{fac n == if n < 3 then n else n * fac(n-1) \bound{fac}}
}
\xtc{
}{
\spadpaste{fac 10 \free{fac}}
}
%>> Thankfully, the $ is no longer needed in the next example.
\xtc{
This function computes \axiom{1 + 1/2 + 1/3 + ... + 1/n}.
}{
\spadpaste{s n == reduce(+,[1/i for i in 1..n]) \bound{s}}
}
\xtc{
}{
\spadpaste{s 50 \free{s}}
```

```
}
\xtc{
This function computes a Mersenne number, several of which are prime.
}{
\spadpaste{mersenne i == 2**i - 1 \bound{mersenne}}
}
\xtc{
If you type \axiom{mersenne}, Axiom shows you
the function definition.
}{
\spadpaste{mersenne \free{mersenne}}
}
\xtc{
Generate a stream of Mersenne numbers.
}{
\spadpaste{[mersenne i for i in 1..] \free{mersenne}}
}
\xtc{
Create a stream of those values of \axiom{i} such that
\axiom{mersenne(i)} is prime.
}{
\spadpaste{mersenneIndex := [n for n in 1.. | prime?(mersenne(n))]
\bound{mersenneIndex}\free{mersenne}}
}
\xtc{
Finally, write a function that returns the \eth{\axiom{n}} Mersenne
prime.
}{
\spadpaste{mersennePrime n == mersenne mersenneIndex(n)
\free{mersenne mersenneIndex}\bound{mersennePrime}}
}
\xtc{
}{
\spadpaste{mersennePrime 5 \free{mersennePrime}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugUserOnePagePatch1}
\begin{paste}{ugUserOnePageFull1}{ugUserOnePageEmpty1}
\pastebutton{ugUserOnePageFull1}{\hidepaste}
\tab{5}\spadcommand{fac n == if n < 3 then n else n * fac(n-1)\bound{fac }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserOnePageEmpty1}
```

```
\begin{paste}{ugUserOnePageEmpty1}{ugUserOnePagePatch1}
\pastebutton{ugUserOnePageEmpty1}{\showpaste}
\tab{5}\spadcommand{fac n == if n < 3 then n else n * fac(n-1)\bound{fac }}
\end{paste}\end{patch}

\begin{patch}{ugUserOnePagePatch2}
\begin{paste}{ugUserOnePageFull2}{ugUserOnePageEmpty2}
\pastebutton{ugUserOnePageFull2}{\hidepaste}
\tab{5}\spadcommand{fac 10\free{fac }}
\indentrel{3}\begin{verbatim}
   (2)  3628800
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserOnePageEmpty2}
\begin{paste}{ugUserOnePageEmpty2}{ugUserOnePagePatch2}
\pastebutton{ugUserOnePageEmpty2}{\showpaste}
\tab{5}\spadcommand{fac 10\free{fac }}
\end{paste}\end{patch}

\begin{patch}{ugUserOnePagePatch3}
\begin{paste}{ugUserOnePageFull3}{ugUserOnePageEmpty3}
\pastebutton{ugUserOnePageFull3}{\hidepaste}
\tab{5}\spadcommand{s n == reduce(+,[1/i for i in 1..n])\bound{s }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserOnePageEmpty3}
\begin{paste}{ugUserOnePageEmpty3}{ugUserOnePagePatch3}
\pastebutton{ugUserOnePageEmpty3}{\showpaste}
\tab{5}\spadcommand{s n == reduce(+,[1/i for i in 1..n])\bound{s }}
\end{paste}\end{patch}

\begin{patch}{ugUserOnePagePatch4}
\begin{paste}{ugUserOnePageFull4}{ugUserOnePageEmpty4}
\pastebutton{ugUserOnePageFull4}{\hidepaste}
\tab{5}\spadcommand{s 50\free{s }}
\indentrel{3}\begin{verbatim}
        13943237577224054960759
   (4)
         3099044504245996706400
                                 Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserOnePageEmpty4}
\begin{paste}{ugUserOnePageEmpty4}{ugUserOnePagePatch4}
```

```
\pastebutton{ugUserOnePageEmpty4}{\showpaste}
\tab{5}\spadcommand{s 50\free{s }}
\end{paste}\end{patch}

\begin{patch}{ugUserOnePagePatch5}
\begin{paste}{ugUserOnePageFull5}{ugUserOnePageEmpty5}
\pastebutton{ugUserOnePageFull5}{\hidepaste}
\tab{5}\spadcommand{mersenne i == 2**i - 1\bound{mersenne }}
\indentrel{3}\begin{verbatim}
                                                Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserOnePageEmpty5}
\begin{paste}{ugUserOnePageEmpty5}{ugUserOnePagePatch5}
\pastebutton{ugUserOnePageEmpty5}{\showpaste}
\tab{5}\spadcommand{mersenne i == 2**i - 1\bound{mersenne }}
\end{paste}\end{patch}

\begin{patch}{ugUserOnePagePatch6}
\begin{paste}{ugUserOnePageFull6}{ugUserOnePageEmpty6}
\pastebutton{ugUserOnePageFull6}{\hidepaste}
\tab{5}\spadcommand{mersenne\free{mersenne }}
\indentrel{3}\begin{verbatim}
                       i
   (6)  mersenne i == 2  - 1
                          Type: FunctionCalled mersenne
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserOnePageEmpty6}
\begin{paste}{ugUserOnePageEmpty6}{ugUserOnePagePatch6}
\pastebutton{ugUserOnePageEmpty6}{\showpaste}
\tab{5}\spadcommand{mersenne\free{mersenne }}
\end{paste}\end{patch}

\begin{patch}{ugUserOnePagePatch7}
\begin{paste}{ugUserOnePageFull7}{ugUserOnePageEmpty7}
\pastebutton{ugUserOnePageFull7}{\hidepaste}
\tab{5}\spadcommand{[mersenne i for i in 1..]\free{mersenne }}
\indentrel{3}\begin{verbatim}
   (7)  [1,3,7,15,31,63,127,255,511,1023,...]
                                   Type: Stream Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserOnePageEmpty7}
\begin{paste}{ugUserOnePageEmpty7}{ugUserOnePagePatch7}
\pastebutton{ugUserOnePageEmpty7}{\showpaste}
\tab{5}\spadcommand{[mersenne i for i in 1..]\free{mersenne }}
```

```
\end{paste}\end{patch}

\begin{patch}{ugUserOnePagePatch8}
\begin{paste}{ugUserOnePageFull8}{ugUserOnePageEmpty8}
\pastebutton{ugUserOnePageFull8}{\hidepaste}
\tab{5}\spadcommand{mersenneIndex := [n for n in 1.. | prime?(mersenne(n))]\bound{mersenneIndex }\free{merse
\indentrel{3}\begin{verbatim}
   (8)  [2,3,5,7,13,17,19,31,61,89,...]
                           Type: Stream PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserOnePageEmpty8}
\begin{paste}{ugUserOnePageEmpty8}{ugUserOnePagePatch8}
\pastebutton{ugUserOnePageEmpty8}{\showpaste}
\tab{5}\spadcommand{mersenneIndex := [n for n in 1.. | prime?(mersenne(n))]\bound{mersenneIndex }\free{merse
\end{paste}\end{patch}

\begin{patch}{ugUserOnePagePatch9}
\begin{paste}{ugUserOnePageFull9}{ugUserOnePageEmpty9}
\pastebutton{ugUserOnePageFull9}{\hidepaste}
\tab{5}\spadcommand{mersennePrime n == mersenne mersenneIndex(n)\free{mersenne mersenneIndex }\bound{mersenn
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserOnePageEmpty9}
\begin{paste}{ugUserOnePageEmpty9}{ugUserOnePagePatch9}
\pastebutton{ugUserOnePageEmpty9}{\showpaste}
\tab{5}\spadcommand{mersennePrime n == mersenne mersenneIndex(n)\free{mersenne mersenneIndex }\bound{mersenn
\end{paste}\end{patch}

\begin{patch}{ugUserOnePagePatch10}
\begin{paste}{ugUserOnePageFull10}{ugUserOnePageEmpty10}
\pastebutton{ugUserOnePageFull10}{\hidepaste}
\tab{5}\spadcommand{mersennePrime 5\free{mersennePrime }}
\indentrel{3}\begin{verbatim}
   (10)  8191
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserOnePageEmpty10}
\begin{paste}{ugUserOnePageEmpty10}{ugUserOnePagePatch10}
\pastebutton{ugUserOnePageEmpty10}{\showpaste}
\tab{5}\spadcommand{mersennePrime 5\free{mersennePrime }}
\end{paste}\end{patch}
```

Declared vs. Undeclared Functions

⇒ "notitle" (ugCategoriesPage) 15 on page 2403

— ug06.ht —

```
\begin{page}{ugUserDecUndecPage}{6.6. Declared vs. Undeclared Functions}
\beginscroll

If you declare the type of a function, you can apply
it to any data that can be converted to the source type
of the function.

\labelSpace{2pc}
\xtc{
Define \userfun{f} with type \spadsig{Integer}{Integer}.
}{
\spadpaste{f(x: Integer): Integer == x + 1 \bound{f}}
}
\xtc{
The function
\userfun{f} can be applied to integers, \ldots
}{
\spadpaste{f 9 \free{f}}
}
\xtc{
and to values that convert to integers, \ldots
}{
\spadpaste{f(-2.0) \free{f}}
}
\xtc{
but not to values that cannot be converted to integers.
}{
\spadpaste{f(2/3) \free{f}}
}

To make the function over a wide range of types, do not
declare its type.
\xtc{
Give the same definition with no declaration.
}{
\spadpaste{g x == x + 1 \bound{g}}
}
\xtc{
If \axiom{x + 1} makes sense, you can apply \userfun{g} to \axiom{x}.
}{
\spadpaste{g 9 \free{g}}
```

```
}
\xtc{
A version of \userfun{g} with different argument types
get compiled for each new kind of argument used.
}{
\spadpaste{g(2/3)  \free{g}}
}
\xtc{
Here \axiom{x+1} for \axiom{x = "axiom"} makes no sense.
}{
\spadpaste{g("axiom")\free{g}}
}

As you will see in \downlink{``Categories''}{ugCategoriesPage}
in Chapter 12\ignore{ugCategories},
Axiom has a formal idea of categories for what ``makes sense.''

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugUserDecUndecPagePatch1}
\begin{paste}{ugUserDecUndecPageFull1}{ugUserDecUndecPageEmpty1}
\pastebutton{ugUserDecUndecPageFull1}{\hidepaste}
\tab{5}\spadcommand{f(x: Integer): Integer == x + 1\bound{f }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserDecUndecPageEmpty1}
\begin{paste}{ugUserDecUndecPageEmpty1}{ugUserDecUndecPagePatch1}
\pastebutton{ugUserDecUndecPageEmpty1}{\showpaste}
\tab{5}\spadcommand{f(x: Integer): Integer == x + 1\bound{f }}
\end{paste}\end{patch}

\begin{patch}{ugUserDecUndecPagePatch2}
\begin{paste}{ugUserDecUndecPageFull2}{ugUserDecUndecPageEmpty2}
\pastebutton{ugUserDecUndecPageFull2}{\hidepaste}
\tab{5}\spadcommand{f 9\free{f }}
\indentrel{3}\begin{verbatim}
   (2)  10
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserDecUndecPageEmpty2}
\begin{paste}{ugUserDecUndecPageEmpty2}{ugUserDecUndecPagePatch2}
\pastebutton{ugUserDecUndecPageEmpty2}{\showpaste}
\tab{5}\spadcommand{f 9\free{f }}
```

```
\end{paste}\end{patch}

\begin{patch}{ugUserDecUndecPagePatch3}
\begin{paste}{ugUserDecUndecPageFull3}{ugUserDecUndecPageEmpty3}
\pastebutton{ugUserDecUndecPageFull3}{\hidepaste}
\tab{5}\spadcommand{f(-2.0)\free{f }}
\indentrel{3}\begin{verbatim}
   (3)  - 1
                                             Type: Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserDecUndecPageEmpty3}
\begin{paste}{ugUserDecUndecPageEmpty3}{ugUserDecUndecPagePatch3}
\pastebutton{ugUserDecUndecPageEmpty3}{\showpaste}
\tab{5}\spadcommand{f(-2.0)\free{f }}
\end{paste}\end{patch}

\begin{patch}{ugUserDecUndecPagePatch4}
\begin{paste}{ugUserDecUndecPageFull4}{ugUserDecUndecPageEmpty4}
\pastebutton{ugUserDecUndecPageFull4}{\hidepaste}
\tab{5}\spadcommand{f(2/3)\free{f }}
\indentrel{3}\begin{verbatim}
   2

   3
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserDecUndecPageEmpty4}
\begin{paste}{ugUserDecUndecPageEmpty4}{ugUserDecUndecPagePatch4}
\pastebutton{ugUserDecUndecPageEmpty4}{\showpaste}
\tab{5}\spadcommand{f(2/3)\free{f }}
\end{paste}\end{patch}

\begin{patch}{ugUserDecUndecPagePatch5}
\begin{paste}{ugUserDecUndecPageFull5}{ugUserDecUndecPageEmpty5}
\pastebutton{ugUserDecUndecPageFull5}{\hidepaste}
\tab{5}\spadcommand{g x == x + 1\bound{g }}
\indentrel{3}\begin{verbatim}
                                                Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserDecUndecPageEmpty5}
\begin{paste}{ugUserDecUndecPageEmpty5}{ugUserDecUndecPagePatch5}
\pastebutton{ugUserDecUndecPageEmpty5}{\showpaste}
\tab{5}\spadcommand{g x == x + 1\bound{g }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugUserDecUndecPagePatch6}
\begin{paste}{ugUserDecUndecPageFull6}{ugUserDecUndecPageEmpty6}
\pastebutton{ugUserDecUndecPageFull6}{\hidepaste}
\tab{5}\spadcommand{g 9\free{g }}
\indentrel{3}\begin{verbatim}
   (5)  10
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserDecUndecPageEmpty6}
\begin{paste}{ugUserDecUndecPageEmpty6}{ugUserDecUndecPagePatch6}
\pastebutton{ugUserDecUndecPageEmpty6}{\showpaste}
\tab{5}\spadcommand{g 9\free{g }}
\end{paste}\end{patch}

\begin{patch}{ugUserDecUndecPagePatch7}
\begin{paste}{ugUserDecUndecPageFull7}{ugUserDecUndecPageEmpty7}
\pastebutton{ugUserDecUndecPageFull7}{\hidepaste}
\tab{5}\spadcommand{g(2/3)\free{g }}
\indentrel{3}\begin{verbatim}
        5
   (6)
        3
                                 Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserDecUndecPageEmpty7}
\begin{paste}{ugUserDecUndecPageEmpty7}{ugUserDecUndecPagePatch7}
\pastebutton{ugUserDecUndecPageEmpty7}{\showpaste}
\tab{5}\spadcommand{g(2/3)\free{g }}
\end{paste}\end{patch}

\begin{patch}{ugUserDecUndecPagePatch8}
\begin{paste}{ugUserDecUndecPageFull8}{ugUserDecUndecPageEmpty8}
\pastebutton{ugUserDecUndecPageFull8}{\hidepaste}
\tab{5}\spadcommand{g("axiom")\free{g }}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserDecUndecPageEmpty8}
\begin{paste}{ugUserDecUndecPageEmpty8}{ugUserDecUndecPagePatch8}
\pastebutton{ugUserDecUndecPageEmpty8}{\showpaste}
\tab{5}\spadcommand{g("axiom")\free{g }}
\end{paste}\end{patch}
```

———

Functions vs. Operations

⇒ "notitle" (MappingPackageOneXmpPage) 3.73 on page 965
⇒ "notitle" (ugPackagesAbstractPage) 14 on page 2380
⇒ "notitle" (ugPackagesPage) 14 on page 2375
⇒ "notitle" (ugCategoriesPage) 15 on page 2403

— **ug06.ht** —

```
\begin{page}{ugUserDecOpersPage}{6.7. Functions vs. Operations}
\beginscroll

A function is an object that you can create, manipulate, pass to,
and return from functions (for some interesting examples of
library functions that manipulate functions, see
\downlink{`MappingPackage1'}{MappingPackageOneXmpPage}
\ignore{MappingPackage1}).
Yet, we often seem to use the term \spadgloss{operation} and
function interchangeably in Axiom.
What is the distinction?

First consider values and types associated with some variable
\axiom{n} in your workspace.  You can make the declaration \axiom{n :
Integer}, then assign \axiom{n} an integer value.  You then speak of
the integer \axiom{n}.  However, note that the integer is not the name
\axiom{n} itself, but the value that you assign to \axiom{n}.

Similarly, you can declare a variable \axiom{f} in your workspace to
have type \spadsig{Integer}{Integer}, then assign \axiom{f}, through a
definition or an assignment of an anonymous function.  You then speak
of the function \axiom{f}.  However, the function is not \axiom{f},
but the value that you assign to \axiom{f}.

A function is a value, in fact, some machine code for doing something.
Doing what?  Well, performing some \spadgloss{operation}.  Formally,
an operation consists of the constituent parts of \axiom{f} in your
workspace, excluding the value; thus an operation has a name and a
type.  An operation is what domains and packages export.  Thus
\axiomType{Ring} exports one operation \axiomOp{+}.  Every ring also
exports this operation.  Also, the author of every ring in the system
is obliged under contract (see
\downlink{``Abstract Datatypes''}{ugPackagesAbstractPage} in
Section 11.3\ignore{ugPackagesAbstract}) to
provide an implementation for this operation.

This chapter is all about functions---how you create them
interactively and how you apply them to meet your needs.  In
\downlink{``Packages''}{ugPackagesPage} in Chapter
11\ignore{ugPackages} you will learn how to create them
```

```
for the Axiom library.  Then in
\downlink{``Categories''}{ugCategoriesPage} in Chapter
12\ignore{ugCategories}, you will learn about
categories and exported operations.

\endscroll
\autobuttons
\end{page}
```

———-

Delayed Assignments vs. Functions with No Arguments

⇒ "notitle" (ugLangAssignPage) 9 on page 1745

— **ug06.ht** —

```
\begin{page}{ugUserDelayPage}
{6.8. Delayed Assignments vs. Functions with No Arguments}
\beginscroll

In \downlink{``Immediate and Delayed Assignments''}
{ugLangAssignPage} in Section
5.1\ignore{ugLangAssign} we discussed the difference
between immediate and
delayed assignments.
In this section we show the difference between delayed
assignments and functions of no arguments.

\labelSpace{2pc}
\xtc{
A function of no arguments is sometimes called a {\it nullary function.}
}{
\spadpaste{sin24() == sin(24.0) \bound{sin24}}
}
\xtc{
You must use the parentheses (\axiomSyntax{()}) to evaluate it.
Like a delayed assignment, the right-hand-side of a function evaluation
is not evaluated until the left-hand-side is used.
}{
\spadpaste{sin24() \free{sin24}}
}
\xtc{
If you omit the parentheses, you just get the function definition.
%(Note how the explicit floating point number in the definition
%has been translated into a function call involving a mantissa,
%exponent and radix.)
}{
```

```
\spadpaste{sin24 \free{sin24}}
}
\xtc{
You do not use the parentheses \axiomSyntax{()} in a
delayed assignment\ldots
}{
\spadpaste{cos24 == cos(24.0) \bound{cos24}}
}
\xtc{
nor in the evaluation.
}{
\spadpaste{cos24 \free{cos24}}
}
The only syntactic difference between delayed assignments
and nullary functions is that you use \axiomSyntax{()}
in the latter case.

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugUserDelayPagePatch1}
\begin{paste}{ugUserDelayPageFull1}{ugUserDelayPageEmpty1}
\pastebutton{ugUserDelayPageFull1}{\hidepaste}
\tab{5}\spadcommand{sin24() == sin(24.0)\bound{sin24 }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserDelayPageEmpty1}
\begin{paste}{ugUserDelayPageEmpty1}{ugUserDelayPagePatch1}
\pastebutton{ugUserDelayPageEmpty1}{\showpaste}
\tab{5}\spadcommand{sin24() == sin(24.0)\bound{sin24 }}
\end{paste}\end{patch}

\begin{patch}{ugUserDelayPagePatch2}
\begin{paste}{ugUserDelayPageFull2}{ugUserDelayPageEmpty2}
\pastebutton{ugUserDelayPageFull2}{\hidepaste}
\tab{5}\spadcommand{sin24()\free{sin24 }}
\indentrel{3}\begin{verbatim}
   (2)  - 0.9055783620 0662384514
                                            Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserDelayPageEmpty2}
\begin{paste}{ugUserDelayPageEmpty2}{ugUserDelayPagePatch2}
\pastebutton{ugUserDelayPageEmpty2}{\showpaste}
\tab{5}\spadcommand{sin24()\free{sin24 }}
```

```
\end{paste}\end{patch}

\begin{patch}{ugUserDelayPagePatch3}
\begin{paste}{ugUserDelayPageFull3}{ugUserDelayPageEmpty3}
\pastebutton{ugUserDelayPageFull3}{\hidepaste}
\tab{5}\spadcommand{sin24\free{sin24 }}
\indentrel{3}\begin{verbatim}
   (3)  sin24 () == sin(24.0)
                             Type: FunctionCalled sin24
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserDelayPageEmpty3}
\begin{paste}{ugUserDelayPageEmpty3}{ugUserDelayPagePatch3}
\pastebutton{ugUserDelayPageEmpty3}{\showpaste}
\tab{5}\spadcommand{sin24\free{sin24 }}
\end{paste}\end{patch}

\begin{patch}{ugUserDelayPagePatch4}
\begin{paste}{ugUserDelayPageFull4}{ugUserDelayPageEmpty4}
\pastebutton{ugUserDelayPageFull4}{\hidepaste}
\tab{5}\spadcommand{cos24 == cos(24.0)\bound{cos24 }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserDelayPageEmpty4}
\begin{paste}{ugUserDelayPageEmpty4}{ugUserDelayPagePatch4}
\pastebutton{ugUserDelayPageEmpty4}{\showpaste}
\tab{5}\spadcommand{cos24 == cos(24.0)\bound{cos24 }}
\end{paste}\end{patch}

\begin{patch}{ugUserDelayPagePatch5}
\begin{paste}{ugUserDelayPageFull5}{ugUserDelayPageEmpty5}
\pastebutton{ugUserDelayPageFull5}{\hidepaste}
\tab{5}\spadcommand{cos24\free{cos24 }}
\indentrel{3}\begin{verbatim}
   (5)  0.4241790073 3699697594
                                            Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserDelayPageEmpty5}
\begin{paste}{ugUserDelayPageEmpty5}{ugUserDelayPagePatch5}
\pastebutton{ugUserDelayPageEmpty5}{\showpaste}
\tab{5}\spadcommand{cos24\free{cos24 }}
\end{paste}\end{patch}
```

———-

How Axiom Determines What Function to Use

⇒ "notitle" (ugTypesPkgCallPage) 7 on page 1686
⇒ "notitle" (ugTypesResolvePage) 7 on page 1696

— **ug06.ht** —

```
\begin{page}{ugUserUsePage}
{6.9. How Axiom Determines What Function to Use}
\beginscroll

What happens if you define a function that has the same name as a
library function?  Well, if your function has the same name and number
of arguments (we sometimes say \spadgloss{arity}) as another function
in the library, then your function covers up the library function.  If
you want then to call the library function, you will have to
package-call it.  Axiom can use both the functions you write and those
that come from the library.  Let's do a simple example to illustrate
this.
\xtc{
Suppose you (wrongly!) define \userfun{sin} in this way.
}{
\spadpaste{sin x == 1.0 \bound{sin}}
}
\xtc{
The value \axiom{1.0} is returned for any argument.
}{
\spadpaste{sin 4.3 \free{sin}}
}
\xtc{
If you want the library operation, we have to package-call it
(see \downlink{``Package Calling and Target Types''}{ugTypesPkgCallPage}
in Section 2.9\ignore{ugTypesPkgCall}
for more information).
}{
\spadpaste{sin(4.3)\$Float}
}
\xtc{
}{
\spadpaste{sin(34.6)\$Float}
}
\xtc{
Even worse, say we accidentally used the same name as a library
function in the function.
}{
\spadpaste{sin x == sin x \bound{sin1}}
}
```

```
\xtc{
Then Axiom definitely does not understand us.
}{
\spadpaste{sin 4.3 \free{sin1}}
}
\xtc{
Again, we could package-call the inside function.
}{
\spadpaste{sin x == sin(x)\$Float \bound{sin2}}
}
\xtc{
}{
\spadpaste{sin 4.3 \free{sin2}}
}
Of course, you are unlikely to make such obvious errors.
It is more probable that you would write a function and in the body use a
function that you think is a library function.
If you had also written a function by that same name, the library function
would be invisible.

How does Axiom determine what library function to call?
It very much depends on the particular example, but the simple case of
creating the polynomial
\axiom{x + 2/3} will give you an idea.
\indent{4}
\beginitems
\item[1. ] The \axiom{x} is analyzed and its default type is
\axiomType{Variable(x)}.
\item[2. ] The \axiom{2} is analyzed and its default type is
\axiomType{PositiveInteger}.
\item[3. ] The \axiom{3} is analyzed and its default type is
\axiomType{PositiveInteger}.
\item[4. ] Because the arguments to \axiomOp{/} are integers, Axiom
gives the expression \axiom{2/3} a default target type of
\axiomType{Fraction(Integer)}.
\item[5. ] Axiom looks in \axiomType{PositiveInteger} for \axiomOp{/}.
It is not found.
\item[6. ] Axiom looks in \axiomType{Fraction(Integer)} for \axiomOp{/}.
It is found for arguments of type \axiomType{Integer}.
\item[7. ] The \axiom{2} and \axiom{3} are converted to objects of type
\axiomType{Integer} (this is trivial) and \axiomOp{/} is applied,
creating an object of type \axiomType{Fraction(Integer)}.
\item[8. ] No \axiomOp{+} for arguments of types \axiomType{Variable(x)}
and \axiomType{Fraction(Integer)} are found in either domain.
\item[9. ] Axiom resolves
(see \downlink{``Resolving Types''}{ugTypesResolvePage}
in Section 2.10\ignore{ugTypesResolve})
the types and gets \axiomType{Polynomial (Fraction (Integer))}.
\item[10. ] The \axiom{x} and the \axiom{2/3} are converted to objects
of this type and \axiomOp{+} is applied, yielding the answer, an object
```

```
of type \axiomType{Polynomial (Fraction (Integer))}.
\enditems
\indent{0}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugUserUsePagePatch1}
\begin{paste}{ugUserUsePageFull1}{ugUserUsePageEmpty1}
\pastebutton{ugUserUsePageFull1}{\hidepaste}
\tab{5}\spadcommand{sin x == 1.0\bound{sin }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserUsePageEmpty1}
\begin{paste}{ugUserUsePageEmpty1}{ugUserUsePagePatch1}
\pastebutton{ugUserUsePageEmpty1}{\showpaste}
\tab{5}\spadcommand{sin x == 1.0\bound{sin }}
\end{paste}\end{patch}

\begin{patch}{ugUserUsePagePatch2}
\begin{paste}{ugUserUsePageFull2}{ugUserUsePageEmpty2}
\pastebutton{ugUserUsePageFull2}{\hidepaste}
\tab{5}\spadcommand{sin 4.3\free{sin }}
\indentrel{3}\begin{verbatim}
   (2)  1.0
                                            Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserUsePageEmpty2}
\begin{paste}{ugUserUsePageEmpty2}{ugUserUsePagePatch2}
\pastebutton{ugUserUsePageEmpty2}{\showpaste}
\tab{5}\spadcommand{sin 4.3\free{sin }}
\end{paste}\end{patch}

\begin{patch}{ugUserUsePagePatch3}
\begin{paste}{ugUserUsePageFull3}{ugUserUsePageEmpty3}
\pastebutton{ugUserUsePageFull3}{\hidepaste}
\tab{5}\spadcommand{sin(4.3)$Float}
\indentrel{3}\begin{verbatim}
   (3)  - 0.9161659367 4945498404
                                            Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserUsePageEmpty3}
```

```
\begin{paste}{ugUserUsePageEmpty3}{ugUserUsePagePatch3}
\pastebutton{ugUserUsePageEmpty3}{\showpaste}
\tab{5}\spadcommand{sin(4.3)$Float}
\end{paste}\end{patch}

\begin{patch}{ugUserUsePagePatch4}
\begin{paste}{ugUserUsePageFull4}{ugUserUsePageEmpty4}
\pastebutton{ugUserUsePageFull4}{\hidepaste}
\tab{5}\spadcommand{sin(34.6)$Float}
\indentrel{3}\begin{verbatim}
   (4)  - 0.0424680347 1695010154 3
                                           Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserUsePageEmpty4}
\begin{paste}{ugUserUsePageEmpty4}{ugUserUsePagePatch4}
\pastebutton{ugUserUsePageEmpty4}{\showpaste}
\tab{5}\spadcommand{sin(34.6)$Float}
\end{paste}\end{patch}

\begin{patch}{ugUserUsePagePatch5}
\begin{paste}{ugUserUsePageFull5}{ugUserUsePageEmpty5}
\pastebutton{ugUserUsePageFull5}{\hidepaste}
\tab{5}\spadcommand{sin x == sin x\bound{sin1 }}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserUsePageEmpty5}
\begin{paste}{ugUserUsePageEmpty5}{ugUserUsePagePatch5}
\pastebutton{ugUserUsePageEmpty5}{\showpaste}
\tab{5}\spadcommand{sin x == sin x\bound{sin1 }}
\end{paste}\end{patch}

\begin{patch}{ugUserUsePagePatch6}
\begin{paste}{ugUserUsePageFull6}{ugUserUsePageEmpty6}
\pastebutton{ugUserUsePageFull6}{\hidepaste}
\tab{5}\spadcommand{sin 4.3\free{sin1 }}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserUsePageEmpty6}
\begin{paste}{ugUserUsePageEmpty6}{ugUserUsePagePatch6}
\pastebutton{ugUserUsePageEmpty6}{\showpaste}
\tab{5}\spadcommand{sin 4.3\free{sin1 }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugUserUsePagePatch7}
\begin{paste}{ugUserUsePageFull7}{ugUserUsePageEmpty7}
\pastebutton{ugUserUsePageFull7}{\hidepaste}
\tab{5}\spadcommand{sin x == sin(x)$Float\bound{sin2 }}
\indentrel{3}\begin{verbatim}
                                                Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserUsePageEmpty7}
\begin{paste}{ugUserUsePageEmpty7}{ugUserUsePagePatch7}
\pastebutton{ugUserUsePageEmpty7}{\showpaste}
\tab{5}\spadcommand{sin x == sin(x)$Float\bound{sin2 }}
\end{paste}\end{patch}

\begin{patch}{ugUserUsePagePatch8}
\begin{paste}{ugUserUsePageFull8}{ugUserUsePageEmpty8}
\pastebutton{ugUserUsePageFull8}{\hidepaste}
\tab{5}\spadcommand{sin 4.3\free{sin2 }}
\indentrel{3}\begin{verbatim}
   (7)  - 0.9161659367 4945498404
                                               Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserUsePageEmpty8}
\begin{paste}{ugUserUsePageEmpty8}{ugUserUsePagePatch8}
\pastebutton{ugUserUsePageEmpty8}{\showpaste}
\tab{5}\spadcommand{sin 4.3\free{sin2 }}
\end{paste}\end{patch}
```

———-

Compiling vs. Interpreting

⇒ "notitle" (ugTypesSubdomainsPage) 7 on page 1679

— **ug06.ht** —

```
\begin{page}{ugUserCompIntPage}{6.10. Compiling vs. Interpreting}
\beginscroll

When possible, Axiom completely determines the type of every object in
a function, then translates the function definition to \Lisp{} or to
machine code (see next section).  This translation, called
\spadglossSee{compilation}{compiler}, happens the first time you call
the function and results in a computational delay.  Subsequent
function calls with the same argument types use the compiled version
```

```
of the code without delay.

If Axiom cannot determine the type of everything, the function may
still be executed but in \spadglossSee{interpret-code
mode}{interpreter} : each statement in the function is analyzed and
executed as the control flow indicates.  This process is slower than
executing a compiled function, but it allows the execution of code
that may involve objects whose types change.

\beginImportant
If Axiom decides that it cannot compile the code, it
issues a message stating the problem and then the following
message:
%
\centerline{
{{\bf We will attempt to step through and interpret the code.}}}
%
This is not a time to panic.
Rather, it just means that what you gave to Axiom
is somehow ambiguous: either it is not specific enough to be analyzed
completely, or it is beyond Axiom's present interactive
compilation abilities.
\endImportant

\xtc{
This function runs in interpret-code mode, but it does not compile.
}{
\begin{spadsrc}[\bound{varPolys}]
varPolys(vars) ==
  for var in vars repeat
    output(1 :: UnivariatePolynomial(var,Integer))
\end{spadsrc}
}
\xtc{
For \axiom{vars} equal to \axiom{['x, 'y, 'z]}, this function displays
\axiom{1} three times.
}{
\spadpaste{varPolys ['x,'y,'z] \free{varPolys}}
}
\xtc{
The type of the argument to \axiomFun{output} changes in each iteration,
so Axiom cannot compile the function.
In this case, even the inner loop by itself would have a problem:
}{
\begin{spadsrc}
for var in ['x,'y,'z] repeat
  output(1 :: UnivariatePolynomial(var,Integer))
\end{spadsrc}
}
```

```
Sometimes you can help a function to compile by using an extra
conversion or by using \axiom{pretend}.  \spadkey{pretend} See
\downlink{‘‘Subdomains Again’’}{ugTypesSubdomainsPage} in
Section 2.8\ignore{ugTypesSubdomains} for
details.

When a function is compilable, you have the choice of whether it is
compiled to \Lisp{} and then interpreted by the \Lisp{} interpreter or
then further compiled from \Lisp{} to machine code.  The option is
controlled via \spadcmd{)set functions compile}.  Issue \spadcmd{)set
functions compile on} to compile all the way to machine code.  With
the default setting \spadcmd{)set functions compile off}, Axiom has
its \Lisp{} code interpreted because the overhead of further
compilation is larger than the run-time of most of the functions our
users have defined.  You may find that selectively turning this option
on and off will give you the best performance in your particular
application.  For example, if you are writing functions for graphics
applications where hundreds of points are being computed, it is almost
certainly true that you will get the best performance by issuing
\spadcmd{)set functions compile on}.

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugUserCompIntPagePatch1}
\begin{paste}{ugUserCompIntPageFull1}{ugUserCompIntPageEmpty1}
\pastebutton{ugUserCompIntPageFull1}{\hidepaste}
\tab{5}\spadcommand{varPolys(vars) ==
  for var in vars repeat
    output(1 :: UnivariatePolynomial(var,Integer))
\bound{varPolys }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserCompIntPageEmpty1}
\begin{paste}{ugUserCompIntPageEmpty1}{ugUserCompIntPagePatch1}
\pastebutton{ugUserCompIntPageEmpty1}{\showpaste}
\tab{5}\spadcommand{varPolys(vars) ==
  for var in vars repeat
    output(1 :: UnivariatePolynomial(var,Integer))
\bound{varPolys }}
\end{paste}\end{patch}

\begin{patch}{ugUserCompIntPagePatch2}
\begin{paste}{ugUserCompIntPageFull2}{ugUserCompIntPageEmpty2}
\pastebutton{ugUserCompIntPageFull2}{\hidepaste}
\tab{5}\spadcommand{varPolys [’x,’y,’z]\free{varPolys }}
```

```
\indentrel{3}\begin{verbatim}
   1
   1
   1
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserCompIntPageEmpty2}
\begin{paste}{ugUserCompIntPageEmpty2}{ugUserCompIntPagePatch2}
\pastebutton{ugUserCompIntPageEmpty2}{\showpaste}
\tab{5}\spadcommand{varPolys ['x,'y,'z]\free{varPolys }}
\end{paste}\end{patch}

\begin{patch}{ugUserCompIntPagePatch3}
\begin{paste}{ugUserCompIntPageFull3}{ugUserCompIntPageEmpty3}
\pastebutton{ugUserCompIntPageFull3}{\hidepaste}
\tab{5}\spadcommand{for var in ['x,'y,'z] repeat
  output(1 :: UnivariatePolynomial(var,Integer))
}
\indentrel{3}\begin{verbatim}
   1
   1
   1
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserCompIntPageEmpty3}
\begin{paste}{ugUserCompIntPageEmpty3}{ugUserCompIntPagePatch3}
\pastebutton{ugUserCompIntPageEmpty3}{\showpaste}
\tab{5}\spadcommand{for var in ['x,'y,'z] repeat
  output(1 :: UnivariatePolynomial(var,Integer))
}
\end{paste}\end{patch}
```

———-

Piece-Wise Function Definitions

⇒ "notitle" (ugUserPieceBasicPage) 10 on page 1862
⇒ "notitle" (ugUserPiecePickingPage) 10 on page 1869
⇒ "notitle" (ugUserPiecePredPage) 10 on page 1876

— **ug06.ht** —

```
\begin{page}{ugUserPiecePage}{6.11. Piece-Wise Function Definitions}
```

```
\beginscroll

To move beyond functions defined in one line, we introduce in this
section functions that are defined piece-by-piece.  That is, we say
``use this definition when the argument is such-and-such and use this
other definition when the argument is that-and-that.''

\beginmenu
    \menudownlink{{6.11.1. A Basic Example}}{ugUserPieceBasicPage}
    \menudownlink{{6.11.2. Picking Up the Pieces}}
{ugUserPiecePickingPage}
    \menudownlink{{6.11.3. Predicates}}{ugUserPiecePredPage}
\endmenu
\endscroll
\autobuttons
\end{page}
```

———

A Basic Example

— ug06.ht —

```
\begin{page}{ugUserPieceBasicPage}{6.11.1. A Basic Example}
\beginscroll

There are many other ways to define a factorial function for nonnegative
integers.
You might
say
factorial of \axiom{0} is \axiom{1,} otherwise factorial of \axiom{n} is
\axiom{n} times factorial of \axiom{n-1}.
Here is one way to do this in Axiom.
%
\xtc{
Here is the value for \axiom{n = 0}.
}{
\spadpaste{fact(0) == 1 \bound{fact0}}
}
\xtc{
Here is the value for \axiom{n > 0}.
The vertical bar \axiomSyntax{|} means
``such that''.
}{
\spadpaste{fact(n | n > 0) == n * fact(n - 1) \free{fact0}\bound{factn}}
}
\xtc{
```

```
What is the value for \axiom{n = 3}?
}{
\spadpaste{fact(3) \free{factn}}
}
\xtc{
What is the value for \axiom{n = -3}?
}{
\spadpaste{fact(-3) \free{factn}}
}
\xtc{
Now for a second definition.
Here is the value for \axiom{n = 0}.
}{
\spadpaste{facto(0) == 1 \bound{facto0}}
}
\xtc{
Give an error message if \axiom{n < 0}.
}{
\spadpaste{facto(n | n < 0) ==
error "arguments to facto must be non-negative"
\free{facto0}\bound{factop}}
}
\xtc{
Here is the value otherwise.
}{
\spadpaste{facto(n) == n * facto(n - 1) \free{factop}\bound{facton}}
}
\xtc{
What is the value for \axiom{n = 7}?
}{
\spadpaste{facto(7) \free{facton}}
}
\xtc{
What is the value for \axiom{n = -7}?
}{
\spadpaste{facto(-7) \free{facton}}
}
\xtc{
To see the current piece-wise definition of a function,
use \spadsys{)display value}.
}{
\spadpaste{)display value facto \free{facton}}
}

In general a {\it piece-wise definition} of a function consists of two
or more parts.  Each part gives a ``piece'' of the entire definition.
Axiom collects the pieces of a function as you enter them.  When you
ask for a value of the function, it then ``glues'' the pieces together
to form a function.
```

```
The two piece-wise definitions for the factorial function are examples
of recursive functions, that is, functions that are defined in terms
of themselves.  Here is an interesting doubly-recursive function.
This function returns the value \axiom{11} for all positive integer
arguments.
\xtc{
Here is the first of two pieces.
}{
\spadpaste{eleven(n | n < 1) == n + 11\bound{ff0}}
}
\xtc{
And the general case.
}{
\spadpaste{eleven(m) == eleven(eleven(m - 12))\bound{ff1}\free{ff0}}
}
\xtc{
Compute \axiom{elevens}, the infinite stream
of values of \axiom{eleven}.
}{
\spadpaste{elevens := [eleven(i) for i in 0..]\bound{ff2}\free{ff1}}
}
\xtc{
What is the value at \axiom{n = 200}?
}{
\spadpaste{elevens 200\free{ff2}}
}
\xtc{
What is the Axiom's definition of \axiom{eleven}?
}{
\spadpaste{)display value eleven\free{ff2}}
}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ugUserPieceBasicPagePatch1}
\begin{paste}{ugUserPieceBasicPageFull1}{ugUserPieceBasicPageEmpty1}
\pastebutton{ugUserPieceBasicPageFull1}{\hidepaste}
\tab{5}\spadcommand{fact(0) == 1\bound{fact0 }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserPieceBasicPageEmpty1}
\begin{paste}{ugUserPieceBasicPageEmpty1}{ugUserPieceBasicPagePatch1}
\pastebutton{ugUserPieceBasicPageEmpty1}{\showpaste}
\tab{5}\spadcommand{fact(0) == 1\bound{fact0 }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugUserPieceBasicPagePatch2}
\begin{paste}{ugUserPieceBasicPageFull2}{ugUserPieceBasicPageEmpty2}
\pastebutton{ugUserPieceBasicPageFull2}{\hidepaste}
\tab{5}\spadcommand{fact(n | n > 0) == n * fact(n - 1)\free{fact0 }\bound{factn }}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserPieceBasicPageEmpty2}
\begin{paste}{ugUserPieceBasicPageEmpty2}{ugUserPieceBasicPagePatch2}
\pastebutton{ugUserPieceBasicPageEmpty2}{\showpaste}
\tab{5}\spadcommand{fact(n | n > 0) == n * fact(n - 1)\free{fact0 }\bound{factn }}
\end{paste}\end{patch}

\begin{patch}{ugUserPieceBasicPagePatch3}
\begin{paste}{ugUserPieceBasicPageFull3}{ugUserPieceBasicPageEmpty3}
\pastebutton{ugUserPieceBasicPageFull3}{\hidepaste}
\tab{5}\spadcommand{fact(3)\free{factn }}
\indentrel{3}\begin{verbatim}
   (3)  6
                                 Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserPieceBasicPageEmpty3}
\begin{paste}{ugUserPieceBasicPageEmpty3}{ugUserPieceBasicPagePatch3}
\pastebutton{ugUserPieceBasicPageEmpty3}{\showpaste}
\tab{5}\spadcommand{fact(3)\free{factn }}
\end{paste}\end{patch}

\begin{patch}{ugUserPieceBasicPagePatch4}
\begin{paste}{ugUserPieceBasicPageFull4}{ugUserPieceBasicPageEmpty4}
\pastebutton{ugUserPieceBasicPageFull4}{\hidepaste}
\tab{5}\spadcommand{fact(-3)\free{factn }}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserPieceBasicPageEmpty4}
\begin{paste}{ugUserPieceBasicPageEmpty4}{ugUserPieceBasicPagePatch4}
\pastebutton{ugUserPieceBasicPageEmpty4}{\showpaste}
\tab{5}\spadcommand{fact(-3)\free{factn }}
\end{paste}\end{patch}

\begin{patch}{ugUserPieceBasicPagePatch5}
\begin{paste}{ugUserPieceBasicPageFull5}{ugUserPieceBasicPageEmpty5}
\pastebutton{ugUserPieceBasicPageFull5}{\hidepaste}
\tab{5}\spadcommand{facto(0) == 1\bound{facto0 }}
\indentrel{3}\begin{verbatim}
```

```
                                              Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserPieceBasicPageEmpty5}
\begin{paste}{ugUserPieceBasicPageEmpty5}{ugUserPieceBasicPagePatch5}
\pastebutton{ugUserPieceBasicPageEmpty5}{\showpaste}
\tab{5}\spadcommand{facto(0) == 1\bound{facto0 }}
\end{paste}\end{patch}

\begin{patch}{ugUserPieceBasicPagePatch6}
\begin{paste}{ugUserPieceBasicPageFull6}{ugUserPieceBasicPageEmpty6}
\pastebutton{ugUserPieceBasicPageFull6}{\hidepaste}
\tab{5}\spadcommand{facto(n | n < 0) == error "arguments to facto must be non-negative"\free{fac
\indentrel{3}\begin{verbatim}
                                              Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserPieceBasicPageEmpty6}
\begin{paste}{ugUserPieceBasicPageEmpty6}{ugUserPieceBasicPagePatch6}
\pastebutton{ugUserPieceBasicPageEmpty6}{\showpaste}
\tab{5}\spadcommand{facto(n | n < 0) == error "arguments to facto must be non-negative"\free{fac
\end{paste}\end{patch}

\begin{patch}{ugUserPieceBasicPagePatch7}
\begin{paste}{ugUserPieceBasicPageFull7}{ugUserPieceBasicPageEmpty7}
\pastebutton{ugUserPieceBasicPageFull7}{\hidepaste}
\tab{5}\spadcommand{facto(n) == n * facto(n - 1)\free{factop }\bound{facton }}
\indentrel{3}\begin{verbatim}
                                              Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserPieceBasicPageEmpty7}
\begin{paste}{ugUserPieceBasicPageEmpty7}{ugUserPieceBasicPagePatch7}
\pastebutton{ugUserPieceBasicPageEmpty7}{\showpaste}
\tab{5}\spadcommand{facto(n) == n * facto(n - 1)\free{factop }\bound{facton }}
\end{paste}\end{patch}

\begin{patch}{ugUserPieceBasicPagePatch8}
\begin{paste}{ugUserPieceBasicPageFull8}{ugUserPieceBasicPageEmpty8}
\pastebutton{ugUserPieceBasicPageFull8}{\hidepaste}
\tab{5}\spadcommand{facto(3)\free{facton }}
\indentrel{3}\begin{verbatim}
   (7)  6
                                   Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugUserPieceBasicPageEmpty8}
\begin{paste}{ugUserPieceBasicPageEmpty8}{ugUserPieceBasicPagePatch8}
\pastebutton{ugUserPieceBasicPageEmpty8}{\showpaste}
\tab{5}\spadcommand{facto(3)\free{facton }}
\end{paste}\end{patch}

\begin{patch}{ugUserPieceBasicPagePatch9}
\begin{paste}{ugUserPieceBasicPageFull9}{ugUserPieceBasicPageEmpty9}
\pastebutton{ugUserPieceBasicPageFull9}{\hidepaste}
\tab{5}\spadcommand{facto(-7)\free{facton }}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserPieceBasicPageEmpty9}
\begin{paste}{ugUserPieceBasicPageEmpty9}{ugUserPieceBasicPagePatch9}
\pastebutton{ugUserPieceBasicPageEmpty9}{\showpaste}
\tab{5}\spadcommand{facto(-7)\free{facton }}
\end{paste}\end{patch}

\begin{patch}{ugUserPieceBasicPagePatch10}
\begin{paste}{ugUserPieceBasicPageFull10}{ugUserPieceBasicPageEmpty10}
\pastebutton{ugUserPieceBasicPageFull10}{\hidepaste}
\tab{5}\spadcommand{)display value facto\free{facton }}
\indentrel{3}\begin{verbatim}
   Definition:
     facto 0 == 1
     facto (n | n < 0) ==
       error(arguments to facto must be non-negative)
     facto n == n facto(n - 1)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserPieceBasicPageEmpty10}
\begin{paste}{ugUserPieceBasicPageEmpty10}{ugUserPieceBasicPagePatch10}
\pastebutton{ugUserPieceBasicPageEmpty10}{\showpaste}
\tab{5}\spadcommand{)display value facto\free{facton }}
\end{paste}\end{patch}

\begin{patch}{ugUserPieceBasicPagePatch11}
\begin{paste}{ugUserPieceBasicPageFull11}{ugUserPieceBasicPageEmpty11}
\pastebutton{ugUserPieceBasicPageFull11}{\hidepaste}
\tab{5}\spadcommand{eleven(n | n < 1) == n + 11\bound{ff0 }}
\indentrel{3}\begin{verbatim}
                                          Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserPieceBasicPageEmpty11}
\begin{paste}{ugUserPieceBasicPageEmpty11}{ugUserPieceBasicPagePatch11}
```

```
\pastebutton{ugUserPieceBasicPageEmpty11}{\showpaste}
\tab{5}\spadcommand{eleven(n | n < 1) == n + 11\bound{ff0 }}
\end{paste}\end{patch}

\begin{patch}{ugUserPieceBasicPagePatch12}
\begin{paste}{ugUserPieceBasicPageFull12}{ugUserPieceBasicPageEmpty12}
\pastebutton{ugUserPieceBasicPageFull12}{\hidepaste}
\tab{5}\spadcommand{eleven(m) == eleven(eleven(m - 12))\bound{ff1 }\free{ff0 }}
\indentrel{3}\begin{verbatim}
                                        Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserPieceBasicPageEmpty12}
\begin{paste}{ugUserPieceBasicPageEmpty12}{ugUserPieceBasicPagePatch12}
\pastebutton{ugUserPieceBasicPageEmpty12}{\showpaste}
\tab{5}\spadcommand{eleven(m) == eleven(eleven(m - 12))\bound{ff1 }\free{ff0 }}
\end{paste}\end{patch}

\begin{patch}{ugUserPieceBasicPagePatch13}
\begin{paste}{ugUserPieceBasicPageFull13}{ugUserPieceBasicPageEmpty13}
\pastebutton{ugUserPieceBasicPageFull13}{\hidepaste}
\tab{5}\spadcommand{elevens := [eleven(i) for i in 0..]\bound{ff2 }\free{ff1 }}
\indentrel{3}\begin{verbatim}
   (10)  [11,11,11,11,11,11,11,11,11,11,...]
                              Type: Stream Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserPieceBasicPageEmpty13}
\begin{paste}{ugUserPieceBasicPageEmpty13}{ugUserPieceBasicPagePatch13}
\pastebutton{ugUserPieceBasicPageEmpty13}{\showpaste}
\tab{5}\spadcommand{elevens := [eleven(i) for i in 0..]\bound{ff2 }\free{ff1 }}
\end{paste}\end{patch}

\begin{patch}{ugUserPieceBasicPagePatch14}
\begin{paste}{ugUserPieceBasicPageFull14}{ugUserPieceBasicPageEmpty14}
\pastebutton{ugUserPieceBasicPageFull14}{\hidepaste}
\tab{5}\spadcommand{elevens 200\free{ff2 }}
\indentrel{3}\begin{verbatim}
   (11)  11
                             Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserPieceBasicPageEmpty14}
\begin{paste}{ugUserPieceBasicPageEmpty14}{ugUserPieceBasicPagePatch14}
\pastebutton{ugUserPieceBasicPageEmpty14}{\showpaste}
\tab{5}\spadcommand{elevens 200\free{ff2 }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugUserPieceBasicPagePatch15}
\begin{paste}{ugUserPieceBasicPageFull15}{ugUserPieceBasicPageEmpty15}
\pastebutton{ugUserPieceBasicPageFull15}{\hidepaste}
\tab{5}\spadcommand{)display value eleven\free{ff2 }}
\indentrel{3}\begin{verbatim}
   Definition:
     eleven (m | m < 1) == m + 11
     eleven m == eleven(eleven(m - 12))
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserPieceBasicPageEmpty15}
\begin{paste}{ugUserPieceBasicPageEmpty15}{ugUserPieceBasicPagePatch15}
\pastebutton{ugUserPieceBasicPageEmpty15}{\showpaste}
\tab{5}\spadcommand{)display value eleven\free{ff2 }}
\end{paste}\end{patch}
```

——–

Picking Up the Pieces

⇒ "notitle" (ugUserPieceBasicPage) 10 on page 1862
— **ug06.ht** —

```
\begin{page}{ugUserPiecePickingPage}{6.11.2. Picking Up the Pieces}
\beginscroll

Here are the details about how Axiom creates a function from its
pieces.  Axiom converts the \eth{\axiom{i}} piece of a function
definition into a conditional expression of the form: \axiom{if}
\pred{i} \axiom{then} \expr{i}.  If any new piece has a \pred{i} that
is identical\footnote{after all variables are uniformly named} to an
earlier \pred{j}, the earlier piece is removed.  Otherwise, the new
piece is always added at the end.

\beginImportant
If there are \axiom{n} pieces to a function definition for \axiom{f},
the function defined \axiom{f} is: \newline
%
\texht{\hspace*{3pc}}{\tab{6}}
{\tt if} \pred{1} {\tt then} \expr{1} {\tt else}\newline
\texht{\hspace*{6pc}}{\tab{12}}. . . \newline
\texht{\hspace*{3pc}}{\tab{6}}
{\tt if} \pred{n} {\tt then} \expr{n} {\tt else}\newline
\texht{\hspace*{3pc}}{\tab{6}}
{\tt  error "You did not define f for argument <arg>."}
```

```
%
\endImportant

You can give definitions of any number of mutually recursive function
definitions, piece-wise or otherwise.  No computation is done until
you ask for a value.  When you do ask for a value, all the relevant
definitions are gathered, analyzed, and translated into separate
functions and compiled.

\xtc{
Let's recall the definition of \userfun{eleven} from
\texht{the previous section}{
\downlink{``A Basic Example''}{ugUserPieceBasicPage}
in Section 6.11.1\ignore{ugUserPieceBasic}}.
}{
\spadpaste{eleven(n | n < 1) == n + 11\bound{ff0}}
}
\xtc{
}{
\spadpaste{eleven(m) == eleven(eleven(m - 12))\bound{ff1}\free{ff0}}
}

A similar doubly-recursive function below produces \axiom{-11} for all
negative positive integers.  If you haven't worked out why or how
\userfun{eleven} works, the structure of this definition gives a clue.
\xtc{
This definition we write as a block.
}{
\begin{spadsrc}[\bound{rf1}]
minusEleven(n) ==
  n >= 0 => n - 11
  minusEleven (5 + minusEleven(n + 7))
\end{spadsrc}
}
\xtc{
Define \axiom{s(n)} to be the
sum of plus and minus ``eleven'' functions divided by \axiom{n}.
Since \axiom{11 - 11 = 0}, we define \axiom{s(0)} to be \axiom{1}.
}{
\spadpaste{s(0) == 1\bound{rf2}}
}
\xtc{
And the general term.
}{
\spadpaste{s(n) == (eleven(n) + minusEleven(n))/n
\bound{rf3}\free{rf2 rf1 ff1}}
}
\xtc{
What are the first ten values of \axiom{s}?
}{
```

```
\spadpaste{[s(n) for n in 0..]\free{rf3}}
}
%% interpreter puts the rule at the end - should fix
Axiom can create infinite streams in the positive direction (for
example, for index values \axiom{0,1, \ldots}) or negative direction
(for example, for index values \axiom{0,-1,-2, \ldots}).  Here we
would like a stream of values of \axiom{s(n)} that is infinite in both
directions.  The function \axiom{t(n)} below returns the
\eth{\axiom{n}} term of the infinite stream \axiom{[s(0), s(1), s(-1),
s(2), s(-2), \ldots].}  Its definition has three pieces.
\xtc{
Define the initial term.
}{
\spadpaste{t(1) == s(0)\bound{t1}\free{rf4}}
}
\xtc{
The even numbered terms are the \axiom{s(i)} for positive \axiom{i}.
We use \axiomOp{quo} rather than \axiomOp{/}
since we want the result to be an integer.
}{
\spadpaste{t(n | even?(n)) == s(n quo 2)\free{t1}\bound{t2}}
}
\xtc{
Finally, the odd numbered terms are the
\axiom{s(i)} for negative \axiom{i}.
In piece-wise definitions, you can use different variables
to define different pieces. Axiom will not get confused.
}{
\spadpaste{t(p) == s(- p quo 2)\free{t2}\bound{t3}}
}
\xtc{
Look at the definition of \axiom{t}.
In the first piece, the variable \axiom{n}
was used; in the second piece, \axiom{p}.
Axiom always uses
your last variable to display your definitions
back to you.
}{
\spadpaste{)display value t\free{t2}}
}
\xtc{
Create a series of values of \axiom{s} applied to
alternating positive and negative arguments.
}{
\spadpaste{[t(i) for i in 1..]\free{t3}\bound{t4}}
}
\xtc{
Evidently \axiom{t(n) = 1} for all \axiom{i.}
Check it at \axiom{n= 100}.
}{
```

```
\spadpaste{t(100)\free{t4}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugUserPiecePickingPagePatch1}
\begin{paste}{ugUserPiecePickingPageFull1}{ugUserPiecePickingPageEmpty1}
\pastebutton{ugUserPiecePickingPageFull1}{\hidepaste}
\tab{5}\spadcommand{eleven(n | n < 1) == n + 11\bound{ff0 }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserPiecePickingPageEmpty1}
\begin{paste}{ugUserPiecePickingPageEmpty1}{ugUserPiecePickingPagePatch1}
\pastebutton{ugUserPiecePickingPageEmpty1}{\showpaste}
\tab{5}\spadcommand{eleven(n | n < 1) == n + 11\bound{ff0 }}
\end{paste}\end{patch}

\begin{patch}{ugUserPiecePickingPagePatch2}
\begin{paste}{ugUserPiecePickingPageFull2}{ugUserPiecePickingPageEmpty2}
\pastebutton{ugUserPiecePickingPageFull2}{\hidepaste}
\tab{5}\spadcommand{eleven(m) == eleven(eleven(m - 12))\bound{ff1 }\free{ff0 }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserPiecePickingPageEmpty2}
\begin{paste}{ugUserPiecePickingPageEmpty2}{ugUserPiecePickingPagePatch2}
\pastebutton{ugUserPiecePickingPageEmpty2}{\showpaste}
\tab{5}\spadcommand{eleven(m) == eleven(eleven(m - 12))\bound{ff1 }\free{ff0 }}
\end{paste}\end{patch}

\begin{patch}{ugUserPiecePickingPagePatch3}
\begin{paste}{ugUserPiecePickingPageFull3}{ugUserPiecePickingPageEmpty3}
\pastebutton{ugUserPiecePickingPageFull3}{\hidepaste}
\tab{5}\spadcommand{minusEleven(n) ==
  n >= 0 => n - 11
  minusEleven (5 + minusEleven(n + 7))
\bound{rf1 }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserPiecePickingPageEmpty3}
```

```
\begin{paste}{ugUserPiecePickingPageEmpty3}{ugUserPiecePickingPagePatch3}
\pastebutton{ugUserPiecePickingPageEmpty3}{\showpaste}
\tab{5}\spadcommand{minusEleven(n) ==
  n >= 0 => n - 11
  minusEleven (5 + minusEleven(n + 7))
\bound{rf1 }}
\end{paste}\end{patch}

\begin{patch}{ugUserPiecePickingPagePatch4}
\begin{paste}{ugUserPiecePickingPageFull4}{ugUserPiecePickingPageEmpty4}
\pastebutton{ugUserPiecePickingPageFull4}{\hidepaste}
\tab{5}\spadcommand{s(0) == 1\bound{rf2 }}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserPiecePickingPageEmpty4}
\begin{paste}{ugUserPiecePickingPageEmpty4}{ugUserPiecePickingPagePatch4}
\pastebutton{ugUserPiecePickingPageEmpty4}{\showpaste}
\tab{5}\spadcommand{s(0) == 1\bound{rf2 }}
\end{paste}\end{patch}

\begin{patch}{ugUserPiecePickingPagePatch5}
\begin{paste}{ugUserPiecePickingPageFull5}{ugUserPiecePickingPageEmpty5}
\pastebutton{ugUserPiecePickingPageFull5}{\hidepaste}
\tab{5}\spadcommand{s(n) == (eleven(n) + minusEleven(n))/n\bound{rf3 }\free{rf2 rf1 ff1 }}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserPiecePickingPageEmpty5}
\begin{paste}{ugUserPiecePickingPageEmpty5}{ugUserPiecePickingPagePatch5}
\pastebutton{ugUserPiecePickingPageEmpty5}{\showpaste}
\tab{5}\spadcommand{s(n) == (eleven(n) + minusEleven(n))/n\bound{rf3 }\free{rf2 rf1 ff1 }}
\end{paste}\end{patch}

\begin{patch}{ugUserPiecePickingPagePatch6}
\begin{paste}{ugUserPiecePickingPageFull6}{ugUserPiecePickingPageEmpty6}
\pastebutton{ugUserPiecePickingPageFull6}{\hidepaste}
\tab{5}\spadcommand{[s(n) for n in 0..]\free{rf3 }}
\indentrel{3}\begin{verbatim}
   (6)  [1,1,1,1,1,1,1,1,1,1,...]
                         Type: Stream Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserPiecePickingPageEmpty6}
\begin{paste}{ugUserPiecePickingPageEmpty6}{ugUserPiecePickingPagePatch6}
```

```
\pastebutton{ugUserPiecePickingPageEmpty6}{\showpaste}
\tab{5}\spadcommand{[s(n) for n in 0..]\free{rf3 }}
\end{paste}\end{patch}

\begin{patch}{ugUserPiecePickingPagePatch7}
\begin{paste}{ugUserPiecePickingPageFull7}{ugUserPiecePickingPageEmpty7}
\pastebutton{ugUserPiecePickingPageFull7}{\hidepaste}
\tab{5}\spadcommand{t(1) == s(0)\bound{t1 }\free{rf4 }}
\indentrel{3}\begin{verbatim}
                                        Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserPiecePickingPageEmpty7}
\begin{paste}{ugUserPiecePickingPageEmpty7}{ugUserPiecePickingPagePatch7}
\pastebutton{ugUserPiecePickingPageEmpty7}{\showpaste}
\tab{5}\spadcommand{t(1) == s(0)\bound{t1 }\free{rf4 }}
\end{paste}\end{patch}

\begin{patch}{ugUserPiecePickingPagePatch8}
\begin{paste}{ugUserPiecePickingPageFull8}{ugUserPiecePickingPageEmpty8}
\pastebutton{ugUserPiecePickingPageFull8}{\hidepaste}
\tab{5}\spadcommand{t(n | even?(n)) == s(n quo 2)\free{t1 }\bound{t2 }}
\indentrel{3}\begin{verbatim}
                                        Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserPiecePickingPageEmpty8}
\begin{paste}{ugUserPiecePickingPageEmpty8}{ugUserPiecePickingPagePatch8}
\pastebutton{ugUserPiecePickingPageEmpty8}{\showpaste}
\tab{5}\spadcommand{t(n | even?(n)) == s(n quo 2)\free{t1 }\bound{t2 }}
\end{paste}\end{patch}

\begin{patch}{ugUserPiecePickingPagePatch9}
\begin{paste}{ugUserPiecePickingPageFull9}{ugUserPiecePickingPageEmpty9}
\pastebutton{ugUserPiecePickingPageFull9}{\hidepaste}
\tab{5}\spadcommand{t(p) == s(- p quo 2)\free{t2 }\bound{t3 }}
\indentrel{3}\begin{verbatim}
                                        Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserPiecePickingPageEmpty9}
\begin{paste}{ugUserPiecePickingPageEmpty9}{ugUserPiecePickingPagePatch9}
\pastebutton{ugUserPiecePickingPageEmpty9}{\showpaste}
\tab{5}\spadcommand{t(p) == s(- p quo 2)\free{t2 }\bound{t3 }}
\end{paste}\end{patch}

\begin{patch}{ugUserPiecePickingPagePatch10}
```

```
\begin{paste}{ugUserPiecePickingPageFull10}{ugUserPiecePickingPageEmpty10}
\pastebutton{ugUserPiecePickingPageFull10}{\hidepaste}
\tab{5}\spadcommand{)display value t\free{t2 }}
\indentrel{3}\begin{verbatim}
   Definition:
     t 1 == s(0)
     t (p | even?(p)) == s(p quo 2)
     t p == s(- p quo 2)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserPiecePickingPageEmpty10}
\begin{paste}{ugUserPiecePickingPageEmpty10}{ugUserPiecePickingPagePatch10}
\pastebutton{ugUserPiecePickingPageEmpty10}{\showpaste}
\tab{5}\spadcommand{)display value t\free{t2 }}
\end{paste}\end{patch}

\begin{patch}{ugUserPiecePickingPagePatch11}
\begin{paste}{ugUserPiecePickingPageFull11}{ugUserPiecePickingPageEmpty11}
\pastebutton{ugUserPiecePickingPageFull11}{\hidepaste}
\tab{5}\spadcommand{[t(i) for i in 1..]\free{t3 }\bound{t4 }}
\indentrel{3}\begin{verbatim}
   (10)  [1,1,1,1,1,1,1,1,1,1,...]
                          Type: Stream Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserPiecePickingPageEmpty11}
\begin{paste}{ugUserPiecePickingPageEmpty11}{ugUserPiecePickingPagePatch11}
\pastebutton{ugUserPiecePickingPageEmpty11}{\showpaste}
\tab{5}\spadcommand{[t(i) for i in 1..]\free{t3 }\bound{t4 }}
\end{paste}\end{patch}

\begin{patch}{ugUserPiecePickingPagePatch12}
\begin{paste}{ugUserPiecePickingPageFull12}{ugUserPiecePickingPageEmpty12}
\pastebutton{ugUserPiecePickingPageFull12}{\hidepaste}
\tab{5}\spadcommand{t(100)\free{t4 }}
\indentrel{3}\begin{verbatim}
   (11)  1
                                 Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserPiecePickingPageEmpty12}
\begin{paste}{ugUserPiecePickingPageEmpty12}{ugUserPiecePickingPagePatch12}
\pastebutton{ugUserPiecePickingPageEmpty12}{\showpaste}
\tab{5}\spadcommand{t(100)\free{t4 }}
\end{paste}\end{patch}
```

———-

Predicates

⇒ "notitle" (ugUserPieceBasicPage) 10 on page 1862

— **ug06.ht** —

```
\begin{page}{ugUserPiecePredPage}{6.11.3. Predicates}
\beginscroll

We have already seen some examples of
predicates
(\downlink{``A Basic Example''}{ugUserPieceBasicPage}
in Section 6.11.1\ignore{ugUserPieceBasic}).
Predicates are \axiomType{Boolean}-valued expressions and Axiom uses them
for filtering collections
(see \downlink{``Creating Lists and Streams with Iterators''}
{ugLangItsPage} in Section 5.5\ignore{ugLangIts})
and for placing constraints on function arguments.
In this section we discuss their latter usage.

\xtc{
The simplest use of a predicate is one you don't see at all.
}{
\spadpaste{opposite 'right == 'left}
}
\xtc{
Here is a longer way to give the ``opposite definition.''
}{
\spadpaste{opposite (x | x = 'left) == 'right}
}
\xtc{
Try it out.
}{
\spadpaste{for x in ['right,'left,'inbetween] repeat output opposite x}
}

Explicit predicates tell Axiom that the given function definition
piece is to be applied if the predicate evaluates to {\tt true} for
the arguments to the function.  You can use such ``constant''
arguments for integers, strings, and quoted symbols.  The
\axiomType{Boolean} values \axiom{true} and \axiom{false} can also be
used if qualified with ``\spad{@}'' or ``\spad{\$}'' and
\axiomType{Boolean}.  The following are all valid function definition
fragments using constant arguments.
\begin{verbatim}
a(1) == ...
b("unramified") == ...
```

```
c('untested) == ...
d(true@Boolean) == ...
\end{verbatim}

If a function has more than one argument, each argument can have its
own predicate.  However, if a predicate involves two or more
arguments, it must be given {\it after} all the arguments mentioned in
the predicate have been given.  You are always safe to give a single
predicate at the end of the argument list.
\xtc{
A function involving predicates on two arguments.
}{
\spadpaste{inFirstHalfQuadrant(x | x > 0,y | y < x) == true}
}
\xtc{
This is incorrect as it gives a predicate on \axiom{y}
before the argument \axiom{y} is given.
}{
\spadpaste{inFirstHalfQuadrant(x | x > 0 and y < x,y) == true}
}
\xtc{
It is always correct to write the predicate at the end.
}{
\spadpaste{inFirstHalfQuadrant(x,y | x > 0 and y < x) == true \bound{ifq1a}}
}
\xtc{
Here is the rest of the definition.
}{
\spadpaste{inFirstHalfQuadrant(x,y) == false \bound{ifq1b}}
}
\xtc{
Try it out.
}{
\spadpaste{[inFirstHalfQuadrant(i,3) for i in 1..5]\bound{ifq1b}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugUserPiecePredPagePatch1}
\begin{paste}{ugUserPiecePredPageFull1}{ugUserPiecePredPageEmpty1}
\pastebutton{ugUserPiecePredPageFull1}{\hidepaste}
\tab{5}\spadcommand{opposite 'right == 'left}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserPiecePredPageEmpty1}
```

```
\begin{paste}{ugUserPiecePredPageEmpty1}{ugUserPiecePredPagePatch1}
\pastebutton{ugUserPiecePredPageEmpty1}{\showpaste}
\tab{5}\spadcommand{opposite 'right == 'left}
\end{paste}\end{patch}

\begin{patch}{ugUserPiecePredPagePatch2}
\begin{paste}{ugUserPiecePredPageFull2}{ugUserPiecePredPageEmpty2}
\pastebutton{ugUserPiecePredPageFull2}{\hidepaste}
\tab{5}\spadcommand{opposite (x | x = 'left) == 'right}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserPiecePredPageEmpty2}
\begin{paste}{ugUserPiecePredPageEmpty2}{ugUserPiecePredPagePatch2}
\pastebutton{ugUserPiecePredPageEmpty2}{\showpaste}
\tab{5}\spadcommand{opposite (x | x = 'left) == 'right}
\end{paste}\end{patch}

\begin{patch}{ugUserPiecePredPagePatch3}
\begin{paste}{ugUserPiecePredPageFull3}{ugUserPiecePredPageEmpty3}
\pastebutton{ugUserPiecePredPageFull3}{\hidepaste}
\tab{5}\spadcommand{for x in ['right,'left,'inbetween] repeat output opposite x}
\indentrel{3}\begin{verbatim}
   left
   right
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserPiecePredPageEmpty3}
\begin{paste}{ugUserPiecePredPageEmpty3}{ugUserPiecePredPagePatch3}
\pastebutton{ugUserPiecePredPageEmpty3}{\showpaste}
\tab{5}\spadcommand{for x in ['right,'left,'inbetween] repeat output opposite x}
\end{paste}\end{patch}

\begin{patch}{ugUserPiecePredPagePatch4}
\begin{paste}{ugUserPiecePredPageFull4}{ugUserPiecePredPageEmpty4}
\pastebutton{ugUserPiecePredPageFull4}{\hidepaste}
\tab{5}\spadcommand{inFirstHalfQuadrant(x | x > 0,y | y < x) == true}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserPiecePredPageEmpty4}
\begin{paste}{ugUserPiecePredPageEmpty4}{ugUserPiecePredPagePatch4}
\pastebutton{ugUserPiecePredPageEmpty4}{\showpaste}
\tab{5}\spadcommand{inFirstHalfQuadrant(x | x > 0,y | y < x) == true}
\end{paste}\end{patch}
```

```
\begin{patch}{ugUserPiecePredPagePatch5}
\begin{paste}{ugUserPiecePredPageFull5}{ugUserPiecePredPageEmpty5}
\pastebutton{ugUserPiecePredPageFull5}{\hidepaste}
\tab{5}\spadcommand{inFirstHalfQuadrant(x | x > 0 and y < x,y) == true}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserPiecePredPageEmpty5}
\begin{paste}{ugUserPiecePredPageEmpty5}{ugUserPiecePredPagePatch5}
\pastebutton{ugUserPiecePredPageEmpty5}{\showpaste}
\tab{5}\spadcommand{inFirstHalfQuadrant(x | x > 0 and y < x,y) == true}
\end{paste}\end{patch}

\begin{patch}{ugUserPiecePredPagePatch6}
\begin{paste}{ugUserPiecePredPageFull6}{ugUserPiecePredPageEmpty6}
\pastebutton{ugUserPiecePredPageFull6}{\hidepaste}
\tab{5}\spadcommand{inFirstHalfQuadrant(x,y | x > 0 and y < x) == true\bound{ifq1a }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserPiecePredPageEmpty6}
\begin{paste}{ugUserPiecePredPageEmpty6}{ugUserPiecePredPagePatch6}
\pastebutton{ugUserPiecePredPageEmpty6}{\showpaste}
\tab{5}\spadcommand{inFirstHalfQuadrant(x,y | x > 0 and y < x) == true\bound{ifq1a }}
\end{paste}\end{patch}

\begin{patch}{ugUserPiecePredPagePatch7}
\begin{paste}{ugUserPiecePredPageFull7}{ugUserPiecePredPageEmpty7}
\pastebutton{ugUserPiecePredPageFull7}{\hidepaste}
\tab{5}\spadcommand{inFirstHalfQuadrant(x,y) == false\bound{ifq1b }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserPiecePredPageEmpty7}
\begin{paste}{ugUserPiecePredPageEmpty7}{ugUserPiecePredPagePatch7}
\pastebutton{ugUserPiecePredPageEmpty7}{\showpaste}
\tab{5}\spadcommand{inFirstHalfQuadrant(x,y) == false\bound{ifq1b }}
\end{paste}\end{patch}

\begin{patch}{ugUserPiecePredPagePatch8}
\begin{paste}{ugUserPiecePredPageFull8}{ugUserPiecePredPageEmpty8}
\pastebutton{ugUserPiecePredPageFull8}{\hidepaste}
\tab{5}\spadcommand{[inFirstHalfQuadrant(i,3) for i in 1..5]\bound{ifq1b }}
```

```
\indentrel{3}\begin{verbatim}
   (7)  [false,false,false,true,true]
                                      Type: List Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserPiecePredPageEmpty8}
\begin{paste}{ugUserPiecePredPageEmpty8}{ugUserPiecePredPagePatch8}
\pastebutton{ugUserPiecePredPageEmpty8}{\showpaste}
\tab{5}\spadcommand{[inFirstHalfQuadrant(i,3) for i in 1..5]\bound{ifq1b }}
\end{paste}\end{patch}
```

———-

Caching Previously Computed Results

⇒ "notitle" (ugUserFreeLocalPage) 10 on page 1906

— **ug06.ht** —

```
\begin{page}{ugUserCachePage}{6.12. Caching Previously Computed Results}
\beginscroll

By default, Axiom does not save the values of any function.  You can
cause it to save values and not to recompute unnecessarily by using
\spadcmd{)set functions cache}.  This should be used before the
functions are defined or, at least, before they are executed.  The
word following ‘‘cache’’ should be \axiom{0} to turn off caching, a
positive integer \axiom{n} to save the last \axiom{n} computed values
or ‘‘all’’ to save all computed values.  If you then give a list of
names of functions, the caching only affects those functions.  Use no
list of names or ‘‘all’’ when you want to define the default behavior
for functions not specifically mentioned in other \spadcmd{)set
functions cache} statements.  If you give no list of names, all
functions will have the caching behavior.  If you explicitly turn on
caching for one or more names, you must explicitly turn off caching
for those names when you want to stop saving their values.

\xtc{
This causes the functions \userfun{f} and \userfun{g} to have
the last three computed values saved.
}{
\spadpaste{)set functions cache 3 f g \bound{cache}}
}
\xtc{
This is a sample definition for \userfun{f}.
}{
\spadpaste{f x == factorial(2**x) \bound{fdef}\free{cache}}
```

```
}
\xtc{
A message is displayed stating what \userfun{f} will cache.
}{
\spadpaste{f(4) \free{}\free{cache}}
}
\xtc{
This causes all other functions to have all computed values saved by
default.
}{
\spadpaste{)set functions cache all}
}
\xtc{
This causes all functions that have not been specifically cached in
some way to have no computed values saved.
}{
\spadpaste{)set functions cache 0}
}
\xtc{
We also make \userfun{f} and \userfun{g} uncached.
}{
\spadpaste{)set functions cache 0 f g}
}

\beginImportant
Be careful about caching functions that have \spadglossSee{side
effects}{side effect}.  Such a function might destructively modify the
elements of an array or issue a \axiomFun{draw} command, for example.
A function that you expect to execute every time it is called should
not be cached.  Also, it is highly unlikely that a function with no
arguments should be cached.
\endImportant

You should also be careful about caching functions that depend on
free variables.
See \downlink{``Free and Local Variables''}{ugUserFreeLocalPage} in
Section 6.16\ignore{ugUserFreeLocal}
for an example.

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugUserCachePagePatch1}
\begin{paste}{ugUserCachePageFull1}{ugUserCachePageEmpty1}
\pastebutton{ugUserCachePageFull1}{\hidepaste}
\tab{5}\spadcommand{)set functions cache 3 f g\bound{cache }}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugUserCachePageEmpty1}
\begin{paste}{ugUserCachePageEmpty1}{ugUserCachePagePatch1}
\pastebutton{ugUserCachePageEmpty1}{\showpaste}
\tab{5}\spadcommand{)set functions cache 3 f g\bound{cache }}
\end{paste}\end{patch}

\begin{patch}{ugUserCachePagePatch2}
\begin{paste}{ugUserCachePageFull2}{ugUserCachePageEmpty2}
\pastebutton{ugUserCachePageFull2}{\hidepaste}
\tab{5}\spadcommand{f x == factorial(2**x)\bound{fdef }\free{cache }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserCachePageEmpty2}
\begin{paste}{ugUserCachePageEmpty2}{ugUserCachePagePatch2}
\pastebutton{ugUserCachePageEmpty2}{\showpaste}
\tab{5}\spadcommand{f x == factorial(2**x)\bound{fdef }\free{cache }}
\end{paste}\end{patch}

\begin{patch}{ugUserCachePagePatch3}
\begin{paste}{ugUserCachePageFull3}{ugUserCachePageEmpty3}
\pastebutton{ugUserCachePageFull3}{\hidepaste}
\tab{5}\spadcommand{f(4)\free{}\free{cache }}
\indentrel{3}\begin{verbatim}
   (2)  20922789888000
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserCachePageEmpty3}
\begin{paste}{ugUserCachePageEmpty3}{ugUserCachePagePatch3}
\pastebutton{ugUserCachePageEmpty3}{\showpaste}
\tab{5}\spadcommand{f(4)\free{}\free{cache }}
\end{paste}\end{patch}

\begin{patch}{ugUserCachePagePatch4}
\begin{paste}{ugUserCachePageFull4}{ugUserCachePageEmpty4}
\pastebutton{ugUserCachePageFull4}{\hidepaste}
\tab{5}\spadcommand{)set functions cache all}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserCachePageEmpty4}
\begin{paste}{ugUserCachePageEmpty4}{ugUserCachePagePatch4}
\pastebutton{ugUserCachePageEmpty4}{\showpaste}
\tab{5}\spadcommand{)set functions cache all}
```

```
\end{paste}\end{patch}

\begin{patch}{ugUserCachePagePatch5}
\begin{paste}{ugUserCachePageFull5}{ugUserCachePageEmpty5}
\pastebutton{ugUserCachePageFull5}{\hidepaste}
\tab{5}\spadcommand{)set functions cache 0}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserCachePageEmpty5}
\begin{paste}{ugUserCachePageEmpty5}{ugUserCachePagePatch5}
\pastebutton{ugUserCachePageEmpty5}{\showpaste}
\tab{5}\spadcommand{)set functions cache 0}
\end{paste}\end{patch}

\begin{patch}{ugUserCachePagePatch6}
\begin{paste}{ugUserCachePageFull6}{ugUserCachePageEmpty6}
\pastebutton{ugUserCachePageFull6}{\hidepaste}
\tab{5}\spadcommand{)set functions cache 0 f g}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserCachePageEmpty6}
\begin{paste}{ugUserCachePageEmpty6}{ugUserCachePagePatch6}
\pastebutton{ugUserCachePageEmpty6}{\showpaste}
\tab{5}\spadcommand{)set functions cache 0 f g}
\end{paste}\end{patch}
```

———-

Recurrence Relations

⇒ "notitle" (ugUserFreeLocalPage) 10 on page 1906
⇒ "notitle" (ugUserCachePage) 10 on page 1880

— **ug06.ht** —

```
\begin{page}{ugUserRecurPage}{6.13. Recurrence Relations}
\beginscroll

One of the most useful classes of function are those defined via a
``recurrence relation.''
A {\it recurrence relation} makes each successive
value depend on some or all of the previous values.
A simple example is the ordinary ``factorial'' function:
```

```
\begin{verbatim}
fact(0) == 1
fact(n | n > 0) == n * fact(n-1)
\end{verbatim}

The value of \axiom{fact(10)} depends on the value of \axiom{fact(9)},
\axiom{fact(9)} on \axiom{fact(8)}, and so on.  Because it depends on
only one previous value, it is usually called a {\it first order
recurrence relation.}  You can easily imagine a function based on two,
three or more previous values.  The Fibonacci numbers are probably the
most famous function defined by a second order recurrence relation.
\xtc{
The library function \axiomFun{fibonacci} computes Fibonacci numbers.
It is obviously optimized for speed.
}{
\spadpaste{[fibonacci(i) for i in 0..]}
}
\xtc{
Define the
Fibonacci numbers ourselves using a piece-wise definition.
}{
\spadpaste{fib(1) == 1 \bound{fib0}}
}
\xtc{
}{
\spadpaste{fib(2) == 1 \bound{fib1}\free{fib0}}
}
\xtc{
}{
\spadpaste{fib(n) == fib(n-1) + fib(n-2) \bound{fibn}\free{fib1}}
}

As defined, this recurrence relation is obviously doubly-recursive.
To compute \axiom{fib(10)}, we need to compute \axiom{fib(9)} and
\axiom{fib(8)}.  And to \axiom{fib(9)}, we need to compute
\axiom{fib(8)} and \axiom{fib(7)}.  And so on.  It seems that to
compute \axiom{fib(10)} we need to compute \axiom{fib(9)} once,
\axiom{fib(8)} twice, \axiom{fib(7)} three times.  Look familiar?  The
number of function calls needed to compute {\it any} second order
recurrence relation in the obvious way is exactly \axiom{fib(n)}.
These numbers grow!  For example, if Axiom actually did this, then
\axiom{fib(500)} requires more than
\texht{$10^{104}$}{\axiom{10**104}} function calls.  And, given all
this, our definition of \userfun{fib} obviously could not be used to
calculate the five-hundredth Fibonacci number.
\xtc{
Let's try it anyway.
}{
\spadpaste{fib(500) \free{fibn}}
}
```

```
Since this takes a short time to compute, it obviously didn't do as
many as \texht{$10^{104}$}{\axiom{10**104}} operations!  By default,
Axiom transforms any recurrence relation it recognizes into an
iteration.  Iterations are efficient.  To compute the value of the
\eth{\axiom{n}} term of a recurrence relation using an iteration
requires only \axiom{n} function calls.\footnote{If you compare the
speed of our \userfun{fib} function to the library function, our
version is still slower.  This is because the library
\axiomFunFrom{fibonacci}{IntegerNumberTheoryFunctions} uses a
``powering algorithm'' with a computing time proportional to
\texht{$\log^3(n)$}{\axiom{log(n)**3}} to compute
\axiom{fibonacci(n).}}

To turn off this special recurrence relation compilation, issue
\begin{verbatim}
)set functions recurrence off
\end{verbatim}
To turn it back on, substitute ``{\tt on}'' for ``{\tt off}''.

The transformations that Axiom uses for \userfun{fib} caches the
last two values.\footnote{For a more general \eth{\axiom{k}} order recurrence
relation, Axiom caches the last \axiom{k} values.}
If, after computing a value for \userfun{fib}, you ask
for some larger value, Axiom picks up the cached values
and continues computing from there.
See \downlink{``Free and Local Variables''}{ugUserFreeLocalPage}
in Section 6.16\ignore{ugUserFreeLocal}
for an example of a function definition that has this same behavior.
Also see \downlink{``Caching Previously Computed Results''}
{ugUserCachePage} in Section 6.12\ignore{ugUserCache}
for a more general discussion of how you can cache function values.

Recurrence relations can be used for defining recurrence relations
involving polynomials, rational functions, or anything you like.
Here we compute the infinite stream of Legendre polynomials.
\xtc{
The Legendre polynomial of degree \axiom{0.}
}{
\spadpaste{p(0) == 1\bound{p0}}
}
\xtc{
The Legendre polynomial of degree \axiom{1.}
}{
\spadpaste{p(1) == x\bound{p1}}
}

\xtc{
The Legendre polynomial of degree \axiom{n}.
}{
```

```
\spadpaste{p(n) == ((2*n-1)*x*p(n-1) - (n-1)*p(n-2))/n\bound{pn}\free{p1}}
}
\xtc{
Compute the Legendre polynomial of degree \axiom{6.}
}{
\spadpaste{p(6)\free{pn}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugUserRecurPagePatch1}
\begin{paste}{ugUserRecurPageFull1}{ugUserRecurPageEmpty1}
\pastebutton{ugUserRecurPageFull1}{\hidepaste}
\tab{5}\spadcommand{[fibonacci(i) for i in 0..]}
\indentrel{3}\begin{verbatim}
   (1)  [0,1,1,2,3,5,8,13,21,34,...]
                                   Type: Stream Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserRecurPageEmpty1}
\begin{paste}{ugUserRecurPageEmpty1}{ugUserRecurPagePatch1}
\pastebutton{ugUserRecurPageEmpty1}{\showpaste}
\tab{5}\spadcommand{[fibonacci(i) for i in 0..]}
\end{paste}\end{patch}

\begin{patch}{ugUserRecurPagePatch2}
\begin{paste}{ugUserRecurPageFull2}{ugUserRecurPageEmpty2}
\pastebutton{ugUserRecurPageFull2}{\hidepaste}
\tab{5}\spadcommand{fib(1) == 1\bound{fib0 }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserRecurPageEmpty2}
\begin{paste}{ugUserRecurPageEmpty2}{ugUserRecurPagePatch2}
\pastebutton{ugUserRecurPageEmpty2}{\showpaste}
\tab{5}\spadcommand{fib(1) == 1\bound{fib0 }}
\end{paste}\end{patch}

\begin{patch}{ugUserRecurPagePatch3}
\begin{paste}{ugUserRecurPageFull3}{ugUserRecurPageEmpty3}
\pastebutton{ugUserRecurPageFull3}{\hidepaste}
\tab{5}\spadcommand{fib(2) == 1\bound{fib1 }\free{fib0 }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserRecurPageEmpty3}
\begin{paste}{ugUserRecurPageEmpty3}{ugUserRecurPagePatch3}
\pastebutton{ugUserRecurPageEmpty3}{\showpaste}
\tab{5}\spadcommand{fib(2) == 1\bound{fib1 }\free{fib0 }}
\end{paste}\end{patch}

\begin{patch}{ugUserRecurPagePatch4}
\begin{paste}{ugUserRecurPageFull4}{ugUserRecurPageEmpty4}
\pastebutton{ugUserRecurPageFull4}{\hidepaste}
\tab{5}\spadcommand{fib(n) == fib(n-1) + fib(n-2)\bound{fibn }\free{fib1 }}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserRecurPageEmpty4}
\begin{paste}{ugUserRecurPageEmpty4}{ugUserRecurPagePatch4}
\pastebutton{ugUserRecurPageEmpty4}{\showpaste}
\tab{5}\spadcommand{fib(n) == fib(n-1) + fib(n-2)\bound{fibn }\free{fib1 }}
\end{paste}\end{patch}

\begin{patch}{ugUserRecurPagePatch5}
\begin{paste}{ugUserRecurPageFull5}{ugUserRecurPageEmpty5}
\pastebutton{ugUserRecurPageFull5}{\hidepaste}
\tab{5}\spadcommand{fib(500)\free{fibn }}
\indentrel{3}\begin{verbatim}
   (5)
  139423224561697880139724382870407283950070256587697307_
   264108962948325571622863290691557658876222521294125
                                 Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserRecurPageEmpty5}
\begin{paste}{ugUserRecurPageEmpty5}{ugUserRecurPagePatch5}
\pastebutton{ugUserRecurPageEmpty5}{\showpaste}
\tab{5}\spadcommand{fib(500)\free{fibn }}
\end{paste}\end{patch}

\begin{patch}{ugUserRecurPagePatch6}
\begin{paste}{ugUserRecurPageFull6}{ugUserRecurPageEmpty6}
\pastebutton{ugUserRecurPageFull6}{\hidepaste}
\tab{5}\spadcommand{p(0) == 1\bound{p0 }}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugUserRecurPageEmpty6}
\begin{paste}{ugUserRecurPageEmpty6}{ugUserRecurPagePatch6}
\pastebutton{ugUserRecurPageEmpty6}{\showpaste}
\tab{5}\spadcommand{p(0) == 1\bound{p0 }}
\end{paste}\end{patch}

\begin{patch}{ugUserRecurPagePatch7}
\begin{paste}{ugUserRecurPageFull7}{ugUserRecurPageEmpty7}
\pastebutton{ugUserRecurPageFull7}{\hidepaste}
\tab{5}\spadcommand{p(1) == x\bound{p1 }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserRecurPageEmpty7}
\begin{paste}{ugUserRecurPageEmpty7}{ugUserRecurPagePatch7}
\pastebutton{ugUserRecurPageEmpty7}{\showpaste}
\tab{5}\spadcommand{p(1) == x\bound{p1 }}
\end{paste}\end{patch}

\begin{patch}{ugUserRecurPagePatch8}
\begin{paste}{ugUserRecurPageFull8}{ugUserRecurPageEmpty8}
\pastebutton{ugUserRecurPageFull8}{\hidepaste}
\tab{5}\spadcommand{p(n) == ((2*n-1)*x*p(n-1) - (n-1)*p(n-2))/n\bound{pn }\free{p1 }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserRecurPageEmpty8}
\begin{paste}{ugUserRecurPageEmpty8}{ugUserRecurPagePatch8}
\pastebutton{ugUserRecurPageEmpty8}{\showpaste}
\tab{5}\spadcommand{p(n) == ((2*n-1)*x*p(n-1) - (n-1)*p(n-2))/n\bound{pn }\free{p1 }}
\end{paste}\end{patch}

\begin{patch}{ugUserRecurPagePatch9}
\begin{paste}{ugUserRecurPageFull9}{ugUserRecurPageEmpty9}
\pastebutton{ugUserRecurPageFull9}{\hidepaste}
\tab{5}\spadcommand{p(6)\free{pn }}
\indentrel{3}\begin{verbatim}
        231  6   315  4   105  2    5
   (9)   x  -  x  +  x  -
         16       16       16      16
                      Type: Polynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserRecurPageEmpty9}
\begin{paste}{ugUserRecurPageEmpty9}{ugUserRecurPagePatch9}
```

```
\pastebutton{ugUserRecurPageEmpty9}{\showpaste}
\tab{5}\spadcommand{p(6)\free{pn }}
\end{paste}\end{patch}
```

———-

Making Functions from Objects

⇒ “notitle” (MakeFunctionXmpPage) 3.76 on page 1006

— ug06.ht —

```
\begin{page}{ugUserMakePage}{6.14. Making Functions from Objects}
\beginscroll

There are many times when you compute a complicated expression
and then wish to use that expression as the body of a function.
Axiom provides an operation called \axiomFun{function} to do
this.
It creates a function object and places it into the workspace.
There are several versions, depending on how many arguments the function
has.
The first argument to \axiomFun{function} is always the expression to be
converted into the function body, and the second is always the name to be
used for the function.
For more information, see \downlink{‘MakeFunction’}{MakeFunctionXmpPage}
\ignore{MakeFunction}.

\xtc{
Start with a simple example of a polynomial in three variables.
}{
\spadpaste{p := -x + y**2 - z**3 \bound{p}}
}
\xtc{
To make this into a function of no arguments that
simply returns the polynomial, use the two argument form of
\axiomFun{function}.
}{
\spadpaste{function(p,’f0) \free{p}\bound{f0}}
}
\xtc{
To avoid possible conflicts (see below), it is a good idea to
quote always this second argument.
}{
\spadpaste{f0 \free{f0}}
}
\xtc{
This is what you get when you evaluate the function.
```

```
}{
\spadpaste{f0() \free{f0}}
}
\xtc{
To make a function in \axiom{x}, use a version of
\axiomFun{function} that takes three arguments.
The last argument is the name of the variable to use as the parameter.
Typically, this variable occurs in the expression and, like the function
name, you should quote it to avoid possible confusion.
}{
\spadpaste{function(p,'f1,'x) \free{p}\bound{f1}}
}
\xtc{
This is what the new function looks like.
}{
\spadpaste{f1 \free{f1}}
}
\xtc{
This is the value of \userfun{f1} at \axiom{x = 3}.
Notice that the return type of the function is
\axiomType{Polynomial (Integer)}, the same as \axiom{p}.
}{
\spadpaste{f1(3) \free{f1}}
}
\xtc{
To use \axiom{x} and \axiom{y} as parameters, use the
four argument form of \axiomFun{function}.
}{
\spadpaste{function(p,'f2,'x,'y) \free{p}\bound{f2}}
}
\xtc{
}{
\spadpaste{f2 \free{f2}}
}
\xtc{
Evaluate \axiom{f2} at \axiom{x = 3} and \axiom{y = 0}.
The return type of \userfun{f2} is still
\axiomType{Polynomial(Integer)} because the variable \axiom{z}
is still present and not one of the parameters.
}{
\spadpaste{f2(3,0) \free{f2}}
}
\xtc{
Finally, use all three variables as parameters.
There is no five argument form of \axiomFun{function}, so use the one with
three arguments, the third argument being a list of the parameters.
}{
\spadpaste{function(p,'f3,['x,'y,'z]) \free{p}\bound{f3}}
}
\xtc{
```

```
Evaluate this using the same values for \axiom{x} and \axiom{y}
as above, but let \axiom{z} be \axiom{-6}.
The result type of \userfun{f3} is \axiomType{Integer}.
}{
\spadpaste{f3 \free{f3}}
}
\xtc{
}{
\spadpaste{f3(3,0,-6) \free{f3}}
}

The four functions we have defined via \axiom{p} have been undeclared.
To declare a function whose body is to be generated by
\axiomFun{function},
issue the declaration {\it before} the function is created.
\xtc{
}{
\spadpaste{g: (Integer, Integer) -> Float \bound{g}}
}
\xtc{
}{
\spadpaste{D(sin(x-y)/cos(x+y),x) \bound{prev}}
}
\xtc{
}{
\spadpaste{function(\%,'g,'x,'y) \free{g}\free{prev}}
}
\xtc{
}{
\spadpaste{g \free{g}}
}
It is an error to use \axiom{g} without the quote in the penultimate
expression since \axiom{g} had been declared but did not have a value.
Similarly, since it is common to overuse variable names like
\axiom{x}, \axiom{y}, and so on, you avoid problems if you always
quote the variable names for \axiomFun{function}.  In general, if
\axiom{x} has a value and you use \axiom{x} without a quote in a call
to \axiomFun{function}, then Axiom does not know what you are trying
to do.

What kind of object is allowable as the first argument to
\axiomFun{function}?  Let's use the \Browse{} facility of Hyperdoc to
find out.  At the main \Browse{} menu, enter the string {\tt function}
and then click on {\bf Operations.}  The exposed operations called
\axiomFun{function} all take an object whose type belongs to category
\axiomType{ConvertibleTo InputForm}.  What domains are those?  Go back
to the main \Browse{} menu, erase {\tt function}, enter {\tt
ConvertibleTo} in the input area, and click on {\bf categories} on the
{\bf Constructors} line.  At the bottom of the page, enter {\tt
InputForm} in the input area following {\bf S =}.  Click on {\bf Cross
```

```
Reference} and then on {\bf Domains}.  The list you see contains over
forty domains that belong to the category \axiomType{ConvertibleTo
InputForm}.  Thus you can use \axiomFun{function} for
\axiomType{Integer}, \axiomType{Float}, \axiomType{Symbol},
\axiomType{Complex}, \axiomType{Expression}, and so on.

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugUserMakePagePatch1}
\begin{paste}{ugUserMakePageFull1}{ugUserMakePageEmpty1}
\pastebutton{ugUserMakePageFull1}{\hidepaste}
\tab{5}\spadcommand{p := -x + y**2 - z**3\bound{p }}
\indentrel{3}\begin{verbatim}
           3    2
   (1)  - z  + y  - x
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserMakePageEmpty1}
\begin{paste}{ugUserMakePageEmpty1}{ugUserMakePagePatch1}
\pastebutton{ugUserMakePageEmpty1}{\showpaste}
\tab{5}\spadcommand{p := -x + y**2 - z**3\bound{p }}
\end{paste}\end{patch}

\begin{patch}{ugUserMakePagePatch2}
\begin{paste}{ugUserMakePageFull2}{ugUserMakePageEmpty2}
\pastebutton{ugUserMakePageFull2}{\hidepaste}
\tab{5}\spadcommand{function(p,'f0)\free{p }\bound{f0 }}
\indentrel{3}\begin{verbatim}
   (2)  f0
                                           Type: Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserMakePageEmpty2}
\begin{paste}{ugUserMakePageEmpty2}{ugUserMakePagePatch2}
\pastebutton{ugUserMakePageEmpty2}{\showpaste}
\tab{5}\spadcommand{function(p,'f0)\free{p }\bound{f0 }}
\end{paste}\end{patch}

\begin{patch}{ugUserMakePagePatch3}
\begin{paste}{ugUserMakePageFull3}{ugUserMakePageEmpty3}
\pastebutton{ugUserMakePageFull3}{\hidepaste}
\tab{5}\spadcommand{f0\free{f0 }}
\indentrel{3}\begin{verbatim}
                    3    2
   (3)  f0 () == - z  + y  - x
```

```
                                Type: FunctionCalled f0
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserMakePageEmpty3}
\begin{paste}{ugUserMakePageEmpty3}{ugUserMakePagePatch3}
\pastebutton{ugUserMakePageEmpty3}{\showpaste}
\tab{5}\spadcommand{f0\free{f0 }}
\end{paste}\end{patch}

\begin{patch}{ugUserMakePagePatch4}
\begin{paste}{ugUserMakePageFull4}{ugUserMakePageEmpty4}
\pastebutton{ugUserMakePageFull4}{\hidepaste}
\tab{5}\spadcommand{f0()\free{f0 }}
\indentrel{3}\begin{verbatim}
           3    2
   (4)  - z  + y  - x
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserMakePageEmpty4}
\begin{paste}{ugUserMakePageEmpty4}{ugUserMakePagePatch4}
\pastebutton{ugUserMakePageEmpty4}{\showpaste}
\tab{5}\spadcommand{f0()\free{f0 }}
\end{paste}\end{patch}

\begin{patch}{ugUserMakePagePatch5}
\begin{paste}{ugUserMakePageFull5}{ugUserMakePageEmpty5}
\pastebutton{ugUserMakePageFull5}{\hidepaste}
\tab{5}\spadcommand{function(p,'f1,'x)\free{p }\bound{f1 }}
\indentrel{3}\begin{verbatim}
   (5)  f1
                                           Type: Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserMakePageEmpty5}
\begin{paste}{ugUserMakePageEmpty5}{ugUserMakePagePatch5}
\pastebutton{ugUserMakePageEmpty5}{\showpaste}
\tab{5}\spadcommand{function(p,'f1,'x)\free{p }\bound{f1 }}
\end{paste}\end{patch}

\begin{patch}{ugUserMakePagePatch6}
\begin{paste}{ugUserMakePageFull6}{ugUserMakePageEmpty6}
\pastebutton{ugUserMakePageFull6}{\hidepaste}
\tab{5}\spadcommand{f1\free{f1 }}
\indentrel{3}\begin{verbatim}
                   3    2
   (6)  f1 x == - z  + y  - x
```

```
                                Type: FunctionCalled f1
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserMakePageEmpty6}
\begin{paste}{ugUserMakePageEmpty6}{ugUserMakePagePatch6}
\pastebutton{ugUserMakePageEmpty6}{\showpaste}
\tab{5}\spadcommand{f1\free{f1 }}
\end{paste}\end{patch}

\begin{patch}{ugUserMakePagePatch7}
\begin{paste}{ugUserMakePageFull7}{ugUserMakePageEmpty7}
\pastebutton{ugUserMakePageFull7}{\hidepaste}
\tab{5}\spadcommand{f1(3)\free{f1 }}
\indentrel{3}\begin{verbatim}
          3    2
   (7)  - z  + y  - 3
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserMakePageEmpty7}
\begin{paste}{ugUserMakePageEmpty7}{ugUserMakePagePatch7}
\pastebutton{ugUserMakePageEmpty7}{\showpaste}
\tab{5}\spadcommand{f1(3)\free{f1 }}
\end{paste}\end{patch}

\begin{patch}{ugUserMakePagePatch8}
\begin{paste}{ugUserMakePageFull8}{ugUserMakePageEmpty8}
\pastebutton{ugUserMakePageFull8}{\hidepaste}
\tab{5}\spadcommand{function(p,'f2,'x,'y)\free{p }\bound{f2 }}
\indentrel{3}\begin{verbatim}
   (8)  f2
                                           Type: Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserMakePageEmpty8}
\begin{paste}{ugUserMakePageEmpty8}{ugUserMakePagePatch8}
\pastebutton{ugUserMakePageEmpty8}{\showpaste}
\tab{5}\spadcommand{function(p,'f2,'x,'y)\free{p }\bound{f2 }}
\end{paste}\end{patch}

\begin{patch}{ugUserMakePagePatch9}
\begin{paste}{ugUserMakePageFull9}{ugUserMakePageEmpty9}
\pastebutton{ugUserMakePageFull9}{\hidepaste}
\tab{5}\spadcommand{f2\free{f2 }}
\indentrel{3}\begin{verbatim}
                      3    2
   (9)  f2 (x,y) == - z  + y  - x
```

```
                               Type: FunctionCalled f2
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserMakePageEmpty9}
\begin{paste}{ugUserMakePageEmpty9}{ugUserMakePagePatch9}
\pastebutton{ugUserMakePageEmpty9}{\showpaste}
\tab{5}\spadcommand{f2\free{f2 }}
\end{paste}\end{patch}

\begin{patch}{ugUserMakePagePatch10}
\begin{paste}{ugUserMakePageFull10}{ugUserMakePageEmpty10}
\pastebutton{ugUserMakePageFull10}{\hidepaste}
\tab{5}\spadcommand{f2(3,0)\free{f2 }}
\indentrel{3}\begin{verbatim}
            3
   (10)  - z  - 3
                              Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserMakePageEmpty10}
\begin{paste}{ugUserMakePageEmpty10}{ugUserMakePagePatch10}
\pastebutton{ugUserMakePageEmpty10}{\showpaste}
\tab{5}\spadcommand{f2(3,0)\free{f2 }}
\end{paste}\end{patch}

\begin{patch}{ugUserMakePagePatch11}
\begin{paste}{ugUserMakePageFull11}{ugUserMakePageEmpty11}
\pastebutton{ugUserMakePageFull11}{\hidepaste}
\tab{5}\spadcommand{function(p,'f3,['x,'y,'z])\free{p }\bound{f3 }}
\indentrel{3}\begin{verbatim}
   (11)  f3
                                          Type: Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserMakePageEmpty11}
\begin{paste}{ugUserMakePageEmpty11}{ugUserMakePagePatch11}
\pastebutton{ugUserMakePageEmpty11}{\showpaste}
\tab{5}\spadcommand{function(p,'f3,['x,'y,'z])\free{p }\bound{f3 }}
\end{paste}\end{patch}

\begin{patch}{ugUserMakePagePatch12}
\begin{paste}{ugUserMakePageFull12}{ugUserMakePageEmpty12}
\pastebutton{ugUserMakePageFull12}{\hidepaste}
\tab{5}\spadcommand{f3\free{f3 }}
\indentrel{3}\begin{verbatim}
                          3    2
   (12)  f3 (x,y,z) == - z  + y  - x
```

```
                                Type: FunctionCalled f3
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserMakePageEmpty12}
\begin{paste}{ugUserMakePageEmpty12}{ugUserMakePagePatch12}
\pastebutton{ugUserMakePageEmpty12}{\showpaste}
\tab{5}\spadcommand{f3\free{f3 }}
\end{paste}\end{patch}

\begin{patch}{ugUserMakePagePatch13}
\begin{paste}{ugUserMakePageFull13}{ugUserMakePageEmpty13}
\pastebutton{ugUserMakePageFull13}{\hidepaste}
\tab{5}\spadcommand{f3(3,0,-6)\free{f3 }}
\indentrel{3}\begin{verbatim}
   (13)  213
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserMakePageEmpty13}
\begin{paste}{ugUserMakePageEmpty13}{ugUserMakePagePatch13}
\pastebutton{ugUserMakePageEmpty13}{\showpaste}
\tab{5}\spadcommand{f3(3,0,-6)\free{f3 }}
\end{paste}\end{patch}

\begin{patch}{ugUserMakePagePatch14}
\begin{paste}{ugUserMakePageFull14}{ugUserMakePageEmpty14}
\pastebutton{ugUserMakePageFull14}{\hidepaste}
\tab{5}\spadcommand{g: (Integer, Integer) -> Float\bound{g }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserMakePageEmpty14}
\begin{paste}{ugUserMakePageEmpty14}{ugUserMakePagePatch14}
\pastebutton{ugUserMakePageEmpty14}{\showpaste}
\tab{5}\spadcommand{g: (Integer, Integer) -> Float\bound{g }}
\end{paste}\end{patch}

\begin{patch}{ugUserMakePagePatch15}
\begin{paste}{ugUserMakePageFull15}{ugUserMakePageEmpty15}
\pastebutton{ugUserMakePageFull15}{\hidepaste}
\tab{5}\spadcommand{D(sin(x-y)/cos(x+y),x)\bound{prev }}
\indentrel{3}\begin{verbatim}
         - sin(y - x)sin(y + x) + cos(y - x)cos(y + x)
   (15)  ---------------------------------------------
                                    2
                          cos(y + x)
```

```
                              Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserMakePageEmpty15}
\begin{paste}{ugUserMakePageEmpty15}{ugUserMakePagePatch15}
\pastebutton{ugUserMakePageEmpty15}{\showpaste}
\tab{5}\spadcommand{D(sin(x-y)/cos(x+y),x)\bound{prev }}
\end{paste}\end{patch}

\begin{patch}{ugUserMakePagePatch16}
\begin{paste}{ugUserMakePageFull16}{ugUserMakePageEmpty16}
\pastebutton{ugUserMakePageFull16}{\hidepaste}
\tab{5}\spadcommand{function(\%,'g,'x,'y)\free{g }\free{prev }}
\indentrel{3}\begin{verbatim}
   (16)  g
                                          Type: Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserMakePageEmpty16}
\begin{paste}{ugUserMakePageEmpty16}{ugUserMakePagePatch16}
\pastebutton{ugUserMakePageEmpty16}{\showpaste}
\tab{5}\spadcommand{function(\%,'g,'x,'y)\free{g }\free{prev }}
\end{paste}\end{patch}

\begin{patch}{ugUserMakePagePatch17}
\begin{paste}{ugUserMakePageFull17}{ugUserMakePageEmpty17}
\pastebutton{ugUserMakePageFull17}{\hidepaste}
\tab{5}\spadcommand{g\free{g }}
\indentrel{3}\begin{verbatim}
   (17)
   g (x,y) ==
     - sin(y - x)sin(y + x) + cos(y - x)cos(y + x)

                                2
                      cos(y + x)
                               Type: FunctionCalled g
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserMakePageEmpty17}
\begin{paste}{ugUserMakePageEmpty17}{ugUserMakePagePatch17}
\pastebutton{ugUserMakePageEmpty17}{\showpaste}
\tab{5}\spadcommand{g\free{g }}
\end{paste}\end{patch}
```

Functions Defined with Blocks

⇒ "notitle" (ugLangBlocksPage) 9 on page 1753

— **ug06.ht** —

```
\begin{page}{ugUserBlocksPage}{6.15. Functions Defined with Blocks}
\beginscroll

You need not restrict yourself to functions that only fit on one line
or are written in a piece-wise manner.
The body of the function can be a block, as discussed in
\downlink{``Blocks''}{ugLangBlocksPage}
in Section 5.2\ignore{ugLangBlocks}.

\labelSpace{1pc}
\xtc{
Here is a short function that swaps two elements of a list,
array or vector.
}{
\begin{spadsrc}[\bound{swap}]
swap(m,i,j) ==
  temp := m.i
  m.i := m.j
  m.j := temp
\end{spadsrc}
}
\xtc{
The significance of \userfun{swap} is that it has a destructive
effect on its first argument.
}{
\spadpaste{k := [1,2,3,4,5] \bound{k}}
}
\xtc{
}{
\spadpaste{swap(k,2,4) \free{l swap}\bound{swapk}}
}
\xtc{
You see that the second and fourth elements are interchanged.
}{
\spadpaste{k \free{swapk}}
}

\xtc{
Using this, we write a couple of different sort functions.
First, a simple bubble sort.
The operation \axiomOpFrom{\#}{List} returns the number of elements in
an aggregate.
}{
\begin{spadsrc}[\bound{bubbleSort}]
```

```
bubbleSort(m) ==
  n := #m
  for i in 1..(n-1) repeat
    for j in n..(i+1) by -1 repeat
      if m.j < m.(j-1) then swap(m,j,j-1)
  m
\end{spadsrc}
}
\xtc{
Let this be the list we want to sort.
}{
\spadpaste{m := [8,4,-3,9] \bound{m}}
}
\xtc{
This is the result of sorting.
}{
\spadpaste{bubbleSort(m) \free{m swap bubbleSort}\bound{sortm}}
}
\xtc{
Moreover, \axiom{m} is destructively changed to be the sorted version.
}{
\spadpaste{m \free{sortm}}
}

\xtc{
This function implements an insertion sort.
The basic idea is to traverse the list and insert the \eth{\axiom{i}}
element in its correct position among the \axiom{i-1} previous
elements.
Since we start at the beginning of the list, the list elements before the
\eth{\axiom{i}} element have already been placed in ascending order.
}{
\begin{spadsrc}[\bound{insertionSort}]
insertionSort(m) ==
  for i in 2..#m repeat
    j := i
    while j > 1 and m.j < m.(j-1) repeat
      swap(m,j,j-1)
      j := j - 1
  m
\end{spadsrc}
}
\xtc{
As with our bubble sort, this is a destructive function.
}{
\spadpaste{m := [8,4,-3,9] \bound{m1}}
}
\xtc{
}{
\spadpaste{insertionSort(m) \free{m1 swap insertionSort}\bound{sortm1}}
```

```
}
\xtc{
}{
\spadpaste{m \free{sortm1}}
}

Neither of the above functions is efficient for sorting large lists since
they reference elements by asking for the \eth{\axiom{j}} element of the
structure \axiom{m}.

\xtc{
Here is a more efficient bubble sort for lists.
}{
\begin{spadsrc}[\bound{bubbleSort2}]
bubbleSort2(m: List Integer): List Integer ==
  null m => m
  l := m
  while not null (r := l.rest) repeat
     r := bubbleSort2 r
     x := l.first
     if x < r.first then
       l.first := r.first
       r.first := x
     l.rest := r
     l := l.rest
  m
\end{spadsrc}
}
\xtc{
Try it out.
}{
\spadpaste{bubbleSort2 [3,7,2]\free{bubbleSort2}}
}

This definition is both recursive and iterative, and is tricky!
Unless you are {\it really} curious about this definition,
we suggest you skip immediately to the next section.

Here are the key points in the definition.  First notice that if you
are sorting a list with less than two elements, there is nothing to
do: just return the list.  This definition returns immediately if
there are zero elements, and skips the entire \axiom{while} loop if
there is just one element.

The second point to realize is that on each outer iteration, the
bubble sort ensures that the minimum element is propagated leftmost.
Each iteration of the \axiom{while} loop calls \userfun{bubbleSort2}
recursively to sort all but the first element.  When finished, the
minimum element is either in the first or second position.  The
conditional expression ensures that it comes first.  If it is in the
```

```
second, then a swap occurs.  In any case, the \axiomFun{rest} of the
original list must be updated to hold the result of the recursive
call.

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugUserBlocksPagePatch1}
\begin{paste}{ugUserBlocksPageFull1}{ugUserBlocksPageEmpty1}
\pastebutton{ugUserBlocksPageFull1}{\hidepaste}
\tab{5}\spadcommand{swap(m,i,j) ==
  temp := m.i
  m.i := m.j
  m.j := temp
\bound{swap }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserBlocksPageEmpty1}
\begin{paste}{ugUserBlocksPageEmpty1}{ugUserBlocksPagePatch1}
\pastebutton{ugUserBlocksPageEmpty1}{\showpaste}
\tab{5}\spadcommand{swap(m,i,j) ==
  temp := m.i
  m.i := m.j
  m.j := temp
\bound{swap }}
\end{paste}\end{patch}

\begin{patch}{ugUserBlocksPagePatch2}
\begin{paste}{ugUserBlocksPageFull2}{ugUserBlocksPageEmpty2}
\pastebutton{ugUserBlocksPageFull2}{\hidepaste}
\tab{5}\spadcommand{k := [1,2,3,4,5]\bound{k }}
\indentrel{3}\begin{verbatim}
   (2)  [1,2,3,4,5]
                             Type: List PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserBlocksPageEmpty2}
\begin{paste}{ugUserBlocksPageEmpty2}{ugUserBlocksPagePatch2}
\pastebutton{ugUserBlocksPageEmpty2}{\showpaste}
\tab{5}\spadcommand{k := [1,2,3,4,5]\bound{k }}
\end{paste}\end{patch}

\begin{patch}{ugUserBlocksPagePatch3}
\begin{paste}{ugUserBlocksPageFull3}{ugUserBlocksPageEmpty3}
\pastebutton{ugUserBlocksPageFull3}{\hidepaste}
```

```
\tab{5}\spadcommand{swap(k,2,4)\free{l swap }\bound{swapk }}
\indentrel{3}\begin{verbatim}
   (3)  2
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserBlocksPageEmpty3}
\begin{paste}{ugUserBlocksPageEmpty3}{ugUserBlocksPagePatch3}
\pastebutton{ugUserBlocksPageEmpty3}{\showpaste}
\tab{5}\spadcommand{swap(k,2,4)\free{l swap }\bound{swapk }}
\end{paste}\end{patch}

\begin{patch}{ugUserBlocksPagePatch4}
\begin{paste}{ugUserBlocksPageFull4}{ugUserBlocksPageEmpty4}
\pastebutton{ugUserBlocksPageFull4}{\hidepaste}
\tab{5}\spadcommand{k\free{swapk }}
\indentrel{3}\begin{verbatim}
   (4)  [1,4,3,2,5]
                             Type: List PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserBlocksPageEmpty4}
\begin{paste}{ugUserBlocksPageEmpty4}{ugUserBlocksPagePatch4}
\pastebutton{ugUserBlocksPageEmpty4}{\showpaste}
\tab{5}\spadcommand{k\free{swapk }}
\end{paste}\end{patch}

\begin{patch}{ugUserBlocksPagePatch5}
\begin{paste}{ugUserBlocksPageFull5}{ugUserBlocksPageEmpty5}
\pastebutton{ugUserBlocksPageFull5}{\hidepaste}
\tab{5}\spadcommand{bubbleSort(m) ==
  n := \#m
  for i in 1..(n-1) repeat
    for j in n..(i+1) by -1 repeat
      if m.j < m.(j-1) then swap(m,j,j-1)
  m
\bound{bubbleSort }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserBlocksPageEmpty5}
\begin{paste}{ugUserBlocksPageEmpty5}{ugUserBlocksPagePatch5}
\pastebutton{ugUserBlocksPageEmpty5}{\showpaste}
\tab{5}\spadcommand{bubbleSort(m) ==
  n := \#m
  for i in 1..(n-1) repeat
```

```
    for j in n..(i+1) by -1 repeat
      if m.j < m.(j-1) then swap(m,j,j-1)
  m
\bound{bubbleSort }}
\end{paste}\end{patch}

\begin{patch}{ugUserBlocksPagePatch6}
\begin{paste}{ugUserBlocksPageFull6}{ugUserBlocksPageEmpty6}
\pastebutton{ugUserBlocksPageFull6}{\hidepaste}
\tab{5}\spadcommand{m := [8,4,-3,9]\bound{m }}
\indentrel{3}\begin{verbatim}
   (6)  [8,4,- 3,9]
                                    Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserBlocksPageEmpty6}
\begin{paste}{ugUserBlocksPageEmpty6}{ugUserBlocksPagePatch6}
\pastebutton{ugUserBlocksPageEmpty6}{\showpaste}
\tab{5}\spadcommand{m := [8,4,-3,9]\bound{m }}
\end{paste}\end{patch}

\begin{patch}{ugUserBlocksPagePatch7}
\begin{paste}{ugUserBlocksPageFull7}{ugUserBlocksPageEmpty7}
\pastebutton{ugUserBlocksPageFull7}{\hidepaste}
\tab{5}\spadcommand{bubbleSort(m)\free{m swap bubbleSort }\bound{sortm }}
\indentrel{3}\begin{verbatim}
   (7)  [- 3,4,8,9]
                                    Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserBlocksPageEmpty7}
\begin{paste}{ugUserBlocksPageEmpty7}{ugUserBlocksPagePatch7}
\pastebutton{ugUserBlocksPageEmpty7}{\showpaste}
\tab{5}\spadcommand{bubbleSort(m)\free{m swap bubbleSort }\bound{sortm }}
\end{paste}\end{patch}

\begin{patch}{ugUserBlocksPagePatch8}
\begin{paste}{ugUserBlocksPageFull8}{ugUserBlocksPageEmpty8}
\pastebutton{ugUserBlocksPageFull8}{\hidepaste}
\tab{5}\spadcommand{m\free{sortm }}
\indentrel{3}\begin{verbatim}
   (8)  [- 3,4,8,9]
                                    Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserBlocksPageEmpty8}
\begin{paste}{ugUserBlocksPageEmpty8}{ugUserBlocksPagePatch8}
```

```
\pastebutton{ugUserBlocksPageEmpty8}{\showpaste}
\tab{5}\spadcommand{m\free{sortm }}
\end{paste}\end{patch}

\begin{patch}{ugUserBlocksPagePatch9}
\begin{paste}{ugUserBlocksPageFull9}{ugUserBlocksPageEmpty9}
\pastebutton{ugUserBlocksPageFull9}{\hidepaste}
\tab{5}\spadcommand{insertionSort(m) ==
  for i in 2..\#m repeat
    j := i
    while j > 1 and m.j < m.(j-1) repeat
      swap(m,j,j-1)
      j := j - 1
  m
\bound{insertionSort }}
\indentrel{3}\begin{verbatim}
                                                Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserBlocksPageEmpty9}
\begin{paste}{ugUserBlocksPageEmpty9}{ugUserBlocksPagePatch9}
\pastebutton{ugUserBlocksPageEmpty9}{\showpaste}
\tab{5}\spadcommand{insertionSort(m) ==
  for i in 2..\#m repeat
    j := i
    while j > 1 and m.j < m.(j-1) repeat
      swap(m,j,j-1)
      j := j - 1
  m
\bound{insertionSort }}
\end{paste}\end{patch}

\begin{patch}{ugUserBlocksPagePatch10}
\begin{paste}{ugUserBlocksPageFull10}{ugUserBlocksPageEmpty10}
\pastebutton{ugUserBlocksPageFull10}{\hidepaste}
\tab{5}\spadcommand{m := [8,4,-3,9]\bound{m1 }}
\indentrel{3}\begin{verbatim}
   (10)  [8,4,- 3,9]
                                    Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserBlocksPageEmpty10}
\begin{paste}{ugUserBlocksPageEmpty10}{ugUserBlocksPagePatch10}
\pastebutton{ugUserBlocksPageEmpty10}{\showpaste}
\tab{5}\spadcommand{m := [8,4,-3,9]\bound{m1 }}
\end{paste}\end{patch}

\begin{patch}{ugUserBlocksPagePatch11}
```

```
\begin{paste}{ugUserBlocksPageFull11}{ugUserBlocksPageEmpty11}
\pastebutton{ugUserBlocksPageFull11}{\hidepaste}
\tab{5}\spadcommand{insertionSort(m)\free{m1 swap insertionSort }\bound{sortm1 }}
\indentrel{3}\begin{verbatim}
   (11)  [- 3,4,8,9]
                                     Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserBlocksPageEmpty11}
\begin{paste}{ugUserBlocksPageEmpty11}{ugUserBlocksPagePatch11}
\pastebutton{ugUserBlocksPageEmpty11}{\showpaste}
\tab{5}\spadcommand{insertionSort(m)\free{m1 swap insertionSort }\bound{sortm1 }}
\end{paste}\end{patch}

\begin{patch}{ugUserBlocksPagePatch12}
\begin{paste}{ugUserBlocksPageFull12}{ugUserBlocksPageEmpty12}
\pastebutton{ugUserBlocksPageFull12}{\hidepaste}
\tab{5}\spadcommand{m\free{sortm1 }}
\indentrel{3}\begin{verbatim}
   (12)  [- 3,4,8,9]
                                     Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserBlocksPageEmpty12}
\begin{paste}{ugUserBlocksPageEmpty12}{ugUserBlocksPagePatch12}
\pastebutton{ugUserBlocksPageEmpty12}{\showpaste}
\tab{5}\spadcommand{m\free{sortm1 }}
\end{paste}\end{patch}

\begin{patch}{ugUserBlocksPagePatch13}
\begin{paste}{ugUserBlocksPageFull13}{ugUserBlocksPageEmpty13}
\pastebutton{ugUserBlocksPageFull13}{\hidepaste}
\tab{5}\spadcommand{bubbleSort2(m: List Integer): List Integer ==
  null m => m
  l := m
  while not null (r := l.rest) repeat
     r := bubbleSort2 r
     x := l.first
     if x < r.first then
       l.first := r.first
       r.first := x
     l.rest := r
     l := l.rest
  m
\bound{bubbleSort2 }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserBlocksPageEmpty13}
\begin{paste}{ugUserBlocksPageEmpty13}{ugUserBlocksPagePatch13}
\pastebutton{ugUserBlocksPageEmpty13}{\showpaste}
\tab{5}\spadcommand{bubbleSort2(m: List Integer): List Integer ==
  null m => m
  l := m
  while not null (r := l.rest) repeat
     r := bubbleSort2 r
     x := l.first
     if x < r.first then
       l.first := r.first
       r.first := x
     l.rest := r
     l := l.rest
  m
\bound{bubbleSort2 }}
\end{paste}\end{patch}

\begin{patch}{ugUserBlocksPagePatch14}
\begin{paste}{ugUserBlocksPageFull14}{ugUserBlocksPageEmpty14}
\pastebutton{ugUserBlocksPageFull14}{\hidepaste}
\tab{5}\spadcommand{bubbleSort2 [3,7,2]\free{bubbleSort2 }}
\indentrel{3}\begin{verbatim}
   (14)  [7,3,2]
                                     Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserBlocksPageEmpty14}
\begin{paste}{ugUserBlocksPageEmpty14}{ugUserBlocksPagePatch14}
\pastebutton{ugUserBlocksPageEmpty14}{\showpaste}
\tab{5}\spadcommand{bubbleSort2 [3,7,2]\free{bubbleSort2 }}
\end{paste}\end{patch}
```

————

Free and Local Variables

⇒ “notitle” (ugUserCachePage) 10 on page 1880
⇒ “notitle” (ugUserRecurPage) 10 on page 1883

— **ug06.ht** —

```
\begin{page}{ugUserFreeLocalPage}{6.16. Free and Local Variables}
\beginscroll
```

```
When you want to refer to a variable that is not local to your
function, use a ``\axiom{free}'' declaration.  \spadkey{free}
Variables declared to be \axiom{free} are assumed to be defined
globally in the workspace.

\labelSpace{1pc}
\xtc{
This is a global workspace variable.
}{
\spadpaste{counter := 0 \bound{counter}}
}
\xtc{
This function refers to the global \axiom{counter}.
}{
\begin{spadsrc}[\free{counter}\bound{f}]
f() ==
  free counter
  counter := counter + 1
\end{spadsrc}
}
\xtc{
The global \axiom{counter} is incremented by \axiom{1}.
}{
\spadpaste{f() \free{f}\bound{f1}}
}
\xtc{
}{
\spadpaste{counter \free{f1}}
}

Usually Axiom can tell that you mean to refer to a global
variable and so \axiom{free} isn't always necessary.
However, for clarity and the sake of self-documentation, we encourage
you to use it.

Declare a variable to be ``\axiom{local}'' when you do not want to refer to
a global variable by the same name.

\xtc{
This function uses \axiom{counter} as a local variable.
}{
\begin{spadsrc}[\bound{g}]
g() ==
  local counter
  counter := 7
\end{spadsrc}
}
\xtc{
Apply the function.
```

```
}{
\spadpaste{g() \free{g}}
}
\xtc{
Check that the global value of \axiom{counter} is unchanged.
}{
\spadpaste{counter\free{g f1}}
}

Parameters to a function are local variables in the function.
Even if you issue a \axiom{free} declaration for a parameter, it is
still local.

What happens if you do not declare that a variable \axiom{x} in
the body of your function is \axiom{local} or \axiom{free}?
Well, Axiom decides on this basis:

\indent{4}
\beginitems
\item[1. ] Axiom scans your function line-by-line, from top-to-bottom.
The right-hand side of an assignment is looked at before the left-hand
side.
\item[2. ] If \axiom{x} is referenced before it is assigned a value,
it is a \axiom{free} (global) variable.
\item[3. ] If \axiom{x} is assigned a value before it is referenced,
it is a \axiom{local} variable.
\enditems
\indent{0}

\xtc{
Set two global variables to 1.
}{
\spadpaste{a := b := 1\bound{ab1}}
}
\xtc{
Refer to \axiom{a} before it is assigned a value, but
assign a value to \axiom{b} before it is referenced.
}{
\begin{spadsrc}[\bound{hh}]
h() ==
  b := a + 1
  a := b + a
\end{spadsrc}
}
\xtc{
Can you predict this result?
}{
\spadpaste{h() \free{ab1 hh}\bound{hhh}}
}
\xtc{
```

```
How about this one?
}{
\spadpaste{[a, b] \free{hhh}}
}

What happened?  In the first line of the function body for \axiom{h},
\axiom{a} is referenced on the right-hand side of the assignment.
Thus \axiom{a} is a free variable.  The variable \axiom{b} is not
referenced in that line, but it is assigned a value.  Thus \axiom{b}
is a local variable and is given the value \axiom{a + 1 = 2}.  In the
second line, the free variable \axiom{a} is assigned the value
\axiom{b + a}which equals \axiom{2 + 1 = 3.}  This is the value
returned by the function.  Since \axiom{a} was free in \userfun{h},
the global variable \axiom{a} has value \axiom{3.}  Since \axiom{b}
was local in \userfun{h}, the global variable \axiom{b} is
unchanged---it still has the value \axiom{1.}

It is good programming practice always to declare global variables.
However, by far the most common situation is to have local variables
in your functions.  No declaration is needed for this situation, but
be sure to initialize their values.

Be careful if you use free variables and you cache the value of
your function (see \downlink{``Caching Previously Computed Results''}
{ugUserCachePage} in Section 6.12\ignore{ugUserCache}).
Caching {\it only} checks if the values of the function arguments
are the same as in a function call previously seen.
It does not check if any of the free variables on which the
function depends have changed between function calls.
\xtc{
Turn on caching for \userfun{p}.
}{
\spadpaste{)set fun cache all p \bound{pcache}}
}
\xtc{
Define \userfun{p} to depend on the free variable \axiom{N}.
}{
\spadpaste{p(i,x) == ( free N;
reduce( + , [ (x-i)**n for n in 1..N ] ) ) \free{pcache}\bound{pdef}}
}
\xtc{
Set the value of \axiom{N}.
}{
\spadpaste{N := 1 \bound{Nass}}
}
\xtc{
Evaluate \userfun{p} the first time.
}{
\spadpaste{p(0, x) \free{pdef Nass}\bound{pfirst}}
}
```

```
\xtc{
Change the value of \axiom{N}.
}{
\spadpaste{N := 2 \bound{Nass2}}
}
\xtc{
Evaluate \userfun{p} the second time.
}{
\spadpaste{p(0, x) \free{pfirst Nass2}}
}
If caching had been turned off, the second evaluation would have
reflected the changed value of \axiom{N}.
\xtc{
Turn off caching for \userfun{p}.
}{
\spadpaste{)set fun cache 0 p}
}

Axiom does not allow {\it fluid variables}, that is, variables
\spadglossSee{bound}{binding} by a function \spad{f} that can be
referenced by functions called by \spad{f}.

Values are passed to functions by \spadgloss{reference}: a pointer
to the value is passed rather than a copy of the value or a pointer to
a copy.

\xtc{
This is a global variable that is bound to a record object.
}{
\spadpaste{r : Record(i : Integer) := [1] \free{r}}
}
\xtc{
This function first modifies the one component of its
record argument and then rebinds the parameter to another
record.
}{
\begin{spadsrc}[\bound{resetRecord}]
resetRecord rr ==
  rr.i := 2
  rr := [10]
\end{spadsrc}
}
\xtc{
Pass \axiom{r} as an argument to \userfun{resetRecord}.
}{
\spadpaste{resetRecord r \free{r resetRecord}\bound{rr}}
}
\xtc{
The value of \axiom{r} was changed by the expression
\axiom{rr.i := 2} but not by \axiom{rr := [10]}.
```

```
}{
\spadpaste{r \free{rr}}
}

To conclude this section, we give an iterative definition of
a function that computes Fibonacci numbers.
This definition approximates the definition into which Axiom
transforms the recurrence relation definition of \userfun{fib} in
\downlink{``Recurrence Relations''}{ugUserRecurPage}
in Section 6.13\ignore{ugUserRecur}.

\xtc{
Global variables
\axiom{past} and \axiom{present} are used
to hold the last computed Fibonacci numbers.
}{
\spadpaste{past := present := 1\bound{f0}}
}
\xtc{
Global variable \axiom{index} gives the
current index of \axiom{present}.
}{
\spadpaste{index := 2\bound{f1}\free{f0}}
}
\xtc{
Here is a recurrence relation defined in terms
of these three global variables.
}{
\begin{spadsrc}[\bound{f3}\free{f2}]
fib(n) ==
  free past, present, index
  n < 3 => 1
  n = index - 1 => past
  if n < index-1 then
    (past,present) := (1,1)
    index := 2
  while (index < n) repeat
    (past,present) := (present, past+present)
    index := index + 1
  present
\end{spadsrc}
}
\xtc{
Compute the infinite stream of Fibonacci numbers.
}{
\spadpaste{fibs := [fib(n) for n in 1..] \bound{fibs}\free{f3}}
}
\xtc{
What is the 1000th Fibonacci number?
}{
```

```
\spadpaste{fibs 1000 \free{fibs}}
}

As an exercise, we suggest you write a function in an iterative
style that computes the value of the recurrence relation
\texht{$p(n) = p(n-1) - 2 \, p(n-2) + 4 \, p(n-3)$}
{\axiom{p(n) = p(n-1) - 2*p(n-2) + 4*p(n-3)}}
having the initial values
\texht{$p(1) = 1,\, p(2) = 3 \hbox{ and } p(3) = 9.$}
{\axiom{p(1) = 1, p(2) = 3 {\rm and} p(3) = 9.}}
How would you write the function using an element
\axiomType{OneDimensionalArray} or \axiomType{Vector}
to hold the previously computed values?

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugUserFreeLocalPagePatch1}
\begin{paste}{ugUserFreeLocalPageFull1}{ugUserFreeLocalPageEmpty1}
\pastebutton{ugUserFreeLocalPageFull1}{\hidepaste}
\tab{5}\spadcommand{counter := 0\bound{counter }}
\indentrel{3}\begin{verbatim}
   (1)  0
                               Type: NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPageEmpty1}
\begin{paste}{ugUserFreeLocalPageEmpty1}{ugUserFreeLocalPagePatch1}
\pastebutton{ugUserFreeLocalPageEmpty1}{\showpaste}
\tab{5}\spadcommand{counter := 0\bound{counter }}
\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPagePatch2}
\begin{paste}{ugUserFreeLocalPageFull2}{ugUserFreeLocalPageEmpty2}
\pastebutton{ugUserFreeLocalPageFull2}{\hidepaste}
\tab{5}\spadcommand{f() ==
  free counter
  counter := counter + 1
\free{counter }\bound{f }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPageEmpty2}
\begin{paste}{ugUserFreeLocalPageEmpty2}{ugUserFreeLocalPagePatch2}
\pastebutton{ugUserFreeLocalPageEmpty2}{\showpaste}
\tab{5}\spadcommand{f() ==
```

```
  free counter
  counter := counter + 1
\free{counter }\bound{f }}
\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPagePatch3}
\begin{paste}{ugUserFreeLocalPageFull3}{ugUserFreeLocalPageEmpty3}
\pastebutton{ugUserFreeLocalPageFull3}{\hidepaste}
\tab{5}\spadcommand{f()\free{f }\bound{f1 }}
\indentrel{3}\begin{verbatim}
   (3)  1
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPageEmpty3}
\begin{paste}{ugUserFreeLocalPageEmpty3}{ugUserFreeLocalPagePatch3}
\pastebutton{ugUserFreeLocalPageEmpty3}{\showpaste}
\tab{5}\spadcommand{f()\free{f }\bound{f1 }}
\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPagePatch4}
\begin{paste}{ugUserFreeLocalPageFull4}{ugUserFreeLocalPageEmpty4}
\pastebutton{ugUserFreeLocalPageFull4}{\hidepaste}
\tab{5}\spadcommand{counter\free{f1 }}
\indentrel{3}\begin{verbatim}
   (4)  1
                               Type: NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPageEmpty4}
\begin{paste}{ugUserFreeLocalPageEmpty4}{ugUserFreeLocalPagePatch4}
\pastebutton{ugUserFreeLocalPageEmpty4}{\showpaste}
\tab{5}\spadcommand{counter\free{f1 }}
\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPagePatch5}
\begin{paste}{ugUserFreeLocalPageFull5}{ugUserFreeLocalPageEmpty5}
\pastebutton{ugUserFreeLocalPageFull5}{\hidepaste}
\tab{5}\spadcommand{g() ==
  local counter
  counter := 7
\bound{g }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPageEmpty5}
```

```
\begin{paste}{ugUserFreeLocalPageEmpty5}{ugUserFreeLocalPagePatch5}
\pastebutton{ugUserFreeLocalPageEmpty5}{\showpaste}
\tab{5}\spadcommand{g() ==
  local counter
  counter := 7
\bound{g }}
\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPagePatch6}
\begin{paste}{ugUserFreeLocalPageFull6}{ugUserFreeLocalPageEmpty6}
\pastebutton{ugUserFreeLocalPageFull6}{\hidepaste}
\tab{5}\spadcommand{g()\free{g }}
\indentrel{3}\begin{verbatim}
   (6)  7
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPageEmpty6}
\begin{paste}{ugUserFreeLocalPageEmpty6}{ugUserFreeLocalPagePatch6}
\pastebutton{ugUserFreeLocalPageEmpty6}{\showpaste}
\tab{5}\spadcommand{g()\free{g }}
\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPagePatch7}
\begin{paste}{ugUserFreeLocalPageFull7}{ugUserFreeLocalPageEmpty7}
\pastebutton{ugUserFreeLocalPageFull7}{\hidepaste}
\tab{5}\spadcommand{counter\free{g f1 }}
\indentrel{3}\begin{verbatim}
   (7)  1
                               Type: NonNegativeInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPageEmpty7}
\begin{paste}{ugUserFreeLocalPageEmpty7}{ugUserFreeLocalPagePatch7}
\pastebutton{ugUserFreeLocalPageEmpty7}{\showpaste}
\tab{5}\spadcommand{counter\free{g f1 }}
\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPagePatch8}
\begin{paste}{ugUserFreeLocalPageFull8}{ugUserFreeLocalPageEmpty8}
\pastebutton{ugUserFreeLocalPageFull8}{\hidepaste}
\tab{5}\spadcommand{a := b := 1\bound{ab1 }}
\indentrel{3}\begin{verbatim}
   (8)  1
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugUserFreeLocalPageEmpty8}
\begin{paste}{ugUserFreeLocalPageEmpty8}{ugUserFreeLocalPagePatch8}
\pastebutton{ugUserFreeLocalPageEmpty8}{\showpaste}
\tab{5}\spadcommand{a := b := 1\bound{ab1 }}
\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPagePatch9}
\begin{paste}{ugUserFreeLocalPageFull9}{ugUserFreeLocalPageEmpty9}
\pastebutton{ugUserFreeLocalPageFull9}{\hidepaste}
\tab{5}\spadcommand{h() ==
  b := a + 1
  a := b + a
\bound{hh }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPageEmpty9}
\begin{paste}{ugUserFreeLocalPageEmpty9}{ugUserFreeLocalPagePatch9}
\pastebutton{ugUserFreeLocalPageEmpty9}{\showpaste}
\tab{5}\spadcommand{h() ==
  b := a + 1
  a := b + a
\bound{hh }}
\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPagePatch10}
\begin{paste}{ugUserFreeLocalPageFull10}{ugUserFreeLocalPageEmpty10}
\pastebutton{ugUserFreeLocalPageFull10}{\hidepaste}
\tab{5}\spadcommand{h()\free{ab1 hh }\bound{hhh }}
\indentrel{3}\begin{verbatim}
   (10)  3
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPageEmpty10}
\begin{paste}{ugUserFreeLocalPageEmpty10}{ugUserFreeLocalPagePatch10}
\pastebutton{ugUserFreeLocalPageEmpty10}{\showpaste}
\tab{5}\spadcommand{h()\free{ab1 hh }\bound{hhh }}
\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPagePatch11}
\begin{paste}{ugUserFreeLocalPageFull11}{ugUserFreeLocalPageEmpty11}
\pastebutton{ugUserFreeLocalPageFull11}{\hidepaste}
\tab{5}\spadcommand{[a, b]\free{hhh }}
\indentrel{3}\begin{verbatim}
   (11)  [3,1]
                             Type: List PositiveInteger
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPageEmpty11}
\begin{paste}{ugUserFreeLocalPageEmpty11}{ugUserFreeLocalPagePatch11}
\pastebutton{ugUserFreeLocalPageEmpty11}{\showpaste}
\tab{5}\spadcommand{[a, b]\free{hhh }}
\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPagePatch12}
\begin{paste}{ugUserFreeLocalPageFull12}{ugUserFreeLocalPageEmpty12}
\pastebutton{ugUserFreeLocalPageFull12}{\hidepaste}
\tab{5}\spadcommand{)set fun cache all p\bound{pcache }}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPageEmpty12}
\begin{paste}{ugUserFreeLocalPageEmpty12}{ugUserFreeLocalPagePatch12}
\pastebutton{ugUserFreeLocalPageEmpty12}{\showpaste}
\tab{5}\spadcommand{)set fun cache all p\bound{pcache }}
\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPagePatch13}
\begin{paste}{ugUserFreeLocalPageFull13}{ugUserFreeLocalPageEmpty13}
\pastebutton{ugUserFreeLocalPageFull13}{\hidepaste}
\tab{5}\spadcommand{p(i,x) == ( free N; reduce( + , [ (x-i)**n for n in 1..N ] ) )\free{pcache }
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPageEmpty13}
\begin{paste}{ugUserFreeLocalPageEmpty13}{ugUserFreeLocalPagePatch13}
\pastebutton{ugUserFreeLocalPageEmpty13}{\showpaste}
\tab{5}\spadcommand{p(i,x) == ( free N; reduce( + , [ (x-i)**n for n in 1..N ] ) )\free{pcache }
\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPagePatch14}
\begin{paste}{ugUserFreeLocalPageFull14}{ugUserFreeLocalPageEmpty14}
\pastebutton{ugUserFreeLocalPageFull14}{\hidepaste}
\tab{5}\spadcommand{N := 1\bound{Nass }}
\indentrel{3}\begin{verbatim}
   (13)  1
                                 Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPageEmpty14}
\begin{paste}{ugUserFreeLocalPageEmpty14}{ugUserFreeLocalPagePatch14}
```

```
\pastebutton{ugUserFreeLocalPageEmpty14}{\showpaste}
\tab{5}\spadcommand{N := 1\bound{Nass }}
\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPagePatch15}
\begin{paste}{ugUserFreeLocalPageFull15}{ugUserFreeLocalPageEmpty15}
\pastebutton{ugUserFreeLocalPageFull15}{\hidepaste}
\tab{5}\spadcommand{p(0, x)\free{pdef Nass }\bound{pfirst }}
\indentrel{3}\begin{verbatim}
   (14)  x
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPageEmpty15}
\begin{paste}{ugUserFreeLocalPageEmpty15}{ugUserFreeLocalPagePatch15}
\pastebutton{ugUserFreeLocalPageEmpty15}{\showpaste}
\tab{5}\spadcommand{p(0, x)\free{pdef Nass }\bound{pfirst }}
\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPagePatch16}
\begin{paste}{ugUserFreeLocalPageFull16}{ugUserFreeLocalPageEmpty16}
\pastebutton{ugUserFreeLocalPageFull16}{\hidepaste}
\tab{5}\spadcommand{N := 2\bound{Nass2 }}
\indentrel{3}\begin{verbatim}
   (15)  2
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPageEmpty16}
\begin{paste}{ugUserFreeLocalPageEmpty16}{ugUserFreeLocalPagePatch16}
\pastebutton{ugUserFreeLocalPageEmpty16}{\showpaste}
\tab{5}\spadcommand{N := 2\bound{Nass2 }}
\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPagePatch17}
\begin{paste}{ugUserFreeLocalPageFull17}{ugUserFreeLocalPageEmpty17}
\pastebutton{ugUserFreeLocalPageFull17}{\hidepaste}
\tab{5}\spadcommand{p(0, x)\free{pfirst Nass2 }}
\indentrel{3}\begin{verbatim}
   (16)  x
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPageEmpty17}
\begin{paste}{ugUserFreeLocalPageEmpty17}{ugUserFreeLocalPagePatch17}
\pastebutton{ugUserFreeLocalPageEmpty17}{\showpaste}
\tab{5}\spadcommand{p(0, x)\free{pfirst Nass2 }}
```

```
\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPagePatch18}
\begin{paste}{ugUserFreeLocalPageFull18}{ugUserFreeLocalPageEmpty18}
\pastebutton{ugUserFreeLocalPageFull18}{\hidepaste}
\tab{5}\spadcommand{)set fun cache 0 p}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPageEmpty18}
\begin{paste}{ugUserFreeLocalPageEmpty18}{ugUserFreeLocalPagePatch18}
\pastebutton{ugUserFreeLocalPageEmpty18}{\showpaste}
\tab{5}\spadcommand{)set fun cache 0 p}
\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPagePatch19}
\begin{paste}{ugUserFreeLocalPageFull19}{ugUserFreeLocalPageEmpty19}
\pastebutton{ugUserFreeLocalPageFull19}{\hidepaste}
\tab{5}\spadcommand{r : Record(i : Integer) := [1]\free{r }}
\indentrel{3}\begin{verbatim}
   (17)  [i= 1]
                              Type: Record(i: Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPageEmpty19}
\begin{paste}{ugUserFreeLocalPageEmpty19}{ugUserFreeLocalPagePatch19}
\pastebutton{ugUserFreeLocalPageEmpty19}{\showpaste}
\tab{5}\spadcommand{r : Record(i : Integer) := [1]\free{r }}
\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPagePatch20}
\begin{paste}{ugUserFreeLocalPageFull20}{ugUserFreeLocalPageEmpty20}
\pastebutton{ugUserFreeLocalPageFull20}{\hidepaste}
\tab{5}\spadcommand{resetRecord rr ==
  rr.i := 2
  rr := [10]
\bound{resetRecord }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPageEmpty20}
\begin{paste}{ugUserFreeLocalPageEmpty20}{ugUserFreeLocalPagePatch20}
\pastebutton{ugUserFreeLocalPageEmpty20}{\showpaste}
\tab{5}\spadcommand{resetRecord rr ==
  rr.i := 2
  rr := [10]
```

```
\bound{resetRecord }}
\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPagePatch21}
\begin{paste}{ugUserFreeLocalPageFull21}{ugUserFreeLocalPageEmpty21}
\pastebutton{ugUserFreeLocalPageFull21}{\hidepaste}
\tab{5}\spadcommand{resetRecord r\free{r resetRecord }\bound{rr }}
\indentrel{3}\begin{verbatim}
   (19)  [i= 10]
                              Type: Record(i: Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPageEmpty21}
\begin{paste}{ugUserFreeLocalPageEmpty21}{ugUserFreeLocalPagePatch21}
\pastebutton{ugUserFreeLocalPageEmpty21}{\showpaste}
\tab{5}\spadcommand{resetRecord r\free{r resetRecord }\bound{rr }}
\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPagePatch22}
\begin{paste}{ugUserFreeLocalPageFull22}{ugUserFreeLocalPageEmpty22}
\pastebutton{ugUserFreeLocalPageFull22}{\hidepaste}
\tab{5}\spadcommand{r\free{rr }}
\indentrel{3}\begin{verbatim}
   (20)  [i= 2]
                              Type: Record(i: Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPageEmpty22}
\begin{paste}{ugUserFreeLocalPageEmpty22}{ugUserFreeLocalPagePatch22}
\pastebutton{ugUserFreeLocalPageEmpty22}{\showpaste}
\tab{5}\spadcommand{r\free{rr }}
\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPagePatch23}
\begin{paste}{ugUserFreeLocalPageFull23}{ugUserFreeLocalPageEmpty23}
\pastebutton{ugUserFreeLocalPageFull23}{\hidepaste}
\tab{5}\spadcommand{past := present := 1\bound{f0 }}
\indentrel{3}\begin{verbatim}
   (21)  1
                                 Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPageEmpty23}
\begin{paste}{ugUserFreeLocalPageEmpty23}{ugUserFreeLocalPagePatch23}
\pastebutton{ugUserFreeLocalPageEmpty23}{\showpaste}
\tab{5}\spadcommand{past := present := 1\bound{f0 }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugUserFreeLocalPagePatch24}
\begin{paste}{ugUserFreeLocalPageFull24}{ugUserFreeLocalPageEmpty24}
\pastebutton{ugUserFreeLocalPageFull24}{\hidepaste}
\tab{5}\spadcommand{index := 2\bound{f1 }\free{f0 }}
\indentrel{3}\begin{verbatim}
   (22)  2
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPageEmpty24}
\begin{paste}{ugUserFreeLocalPageEmpty24}{ugUserFreeLocalPagePatch24}
\pastebutton{ugUserFreeLocalPageEmpty24}{\showpaste}
\tab{5}\spadcommand{index := 2\bound{f1 }\free{f0 }}
\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPagePatch25}
\begin{paste}{ugUserFreeLocalPageFull25}{ugUserFreeLocalPageEmpty25}
\pastebutton{ugUserFreeLocalPageFull25}{\hidepaste}
\tab{5}\spadcommand{fib(n) ==
  free past, present, index
  n < 3 => 1
  n = index - 1 => past
  if n < index-1 then
    (past,present) := (1,1)
    index := 2
  while (index < n) repeat
    (past,present) := (present, past+present)
    index := index + 1
  present
\bound{f3 }\free{f2 }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPageEmpty25}
\begin{paste}{ugUserFreeLocalPageEmpty25}{ugUserFreeLocalPagePatch25}
\pastebutton{ugUserFreeLocalPageEmpty25}{\showpaste}
\tab{5}\spadcommand{fib(n) ==
  free past, present, index
  n < 3 => 1
  n = index - 1 => past
  if n < index-1 then
    (past,present) := (1,1)
    index := 2
  while (index < n) repeat
    (past,present) := (present, past+present)
    index := index + 1
```

```
  present
\bound{f3 }\free{f2 }}
\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPagePatch26}
\begin{paste}{ugUserFreeLocalPageFull26}{ugUserFreeLocalPageEmpty26}
\pastebutton{ugUserFreeLocalPageFull26}{\hidepaste}
\tab{5}\spadcommand{fibs := [fib(n) for n in 1..]\bound{fibs }\free{f3 }}
\indentrel{3}\begin{verbatim}
   (24)  [1,1,2,3,5,8,13,21,34,55,...]
                           Type: Stream PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPageEmpty26}
\begin{paste}{ugUserFreeLocalPageEmpty26}{ugUserFreeLocalPagePatch26}
\pastebutton{ugUserFreeLocalPageEmpty26}{\showpaste}
\tab{5}\spadcommand{fibs := [fib(n) for n in 1..]\bound{fibs }\free{f3 }}
\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPagePatch27}
\begin{paste}{ugUserFreeLocalPageFull27}{ugUserFreeLocalPageEmpty27}
\pastebutton{ugUserFreeLocalPageFull27}{\hidepaste}
\tab{5}\spadcommand{fibs 1000\free{fibs }}
\indentrel{3}\begin{verbatim}
   (25)
  434665576869374564356885276750406258025646605173717804_
   024817290895365554179490518904038798400792551692959225_
   930803226347752096896232398733224711616429964409065_
   3187938298969649928516003704476137795166849228875
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserFreeLocalPageEmpty27}
\begin{paste}{ugUserFreeLocalPageEmpty27}{ugUserFreeLocalPagePatch27}
\pastebutton{ugUserFreeLocalPageEmpty27}{\showpaste}
\tab{5}\spadcommand{fibs 1000\free{fibs }}
\end{paste}\end{patch}
```

————-

Anonymous Functions

⇒ "notitle" (ugUserAnonExampPage) 10 on page 1922
⇒ "notitle" (ugUserAnonDeclarePage) 10 on page 1927

— **ug06.ht** —

```
\begin{page}{ugUserAnonPage}{6.17. Anonymous Functions}
\beginscroll

\beginImportant
An {\it anonymous function} is a function that is
defined
by giving a list of parameters, the ‘‘maps-to’’ compound
symbol \axiomSyntax{+->} \texht{(from the mathematical symbol
$\mapsto$)}{},
and by an expression involving the parameters, the evaluation of
which determines the return value of the function.

\centerline{{{\tt ( \subscriptIt{parm}{1}, \subscriptIt{parm}{2},
\ldots, \subscriptIt{parm}{N} ) +-> {\it expression}}}}
\endImportant

You can apply an anonymous function in several ways.
\indent{4}
\beginitems
\item[1. ] Place the anonymous function definition in parentheses
directly followed by a list of arguments.
\item[2. ] Assign the anonymous function to a variable and then
use the variable name when you would normally use a function name.
\item[3. ] Use \axiomSyntax{==} to use the anonymous function
definition as the arguments and body of a regular function definition.
\item[4. ] Have a named function contain a declared anonymous function
and use the result returned by the named function.
\enditems
\indent{0}

\beginmenu
    \menudownlink{{6.17.1. Some Examples}}{ugUserAnonExampPage}
    \menudownlink{{6.17.2. Declaring Anonymous Functions}}
{ugUserAnonDeclarePage}
\endmenu
\endscroll
\autobuttons
\end{page}
```

———-

Some Examples

— ug06.ht —

```
\begin{page}{ugUserAnonExampPage}{6.17.1. Some Examples}
\beginscroll
```

```
Anonymous functions are particularly useful for defining functions
``on the fly.'' That is, they are handy for simple functions that
are used only in one place.
In the following examples, we show how to write some simple
anonymous functions.

\xtc{
This is a simple absolute value function.
}{
\spadpaste{x +-> if x < 0 then -x else x \bound{anon0}}
}
\xtc{
}{
\spadpaste{abs1 := \% \free{anon0}\bound{abs1}}
}
\xtc{
This function returns {\tt true} if the absolute value of
the first argument is greater than the absolute value of the
second, {\tt false} otherwise.
}{
\spadpaste{(x,y) +-> abs1(x) > abs1(y) \bound{anon1}\free{abs1}}
}
\xtc{
We use the above function to ``sort'' a list of integers.
}{
\spadpaste{sort(\%,[3,9,-4,10,-3,-1,-9,5]) \free{anon1}}
}

\xtc{
This function returns \axiom{1} if \axiom{i + j} is even,
\axiom{-1} otherwise.
}{
\spadpaste{ev := ( (i,j) +-> if even?(i+j) then 1 else -1) \bound{ev}}
}
\xtc{
We create a four-by-four matrix containing \axiom{1} or \axiom{-1}
depending on whether the row plus the column index is even or not.
}{
\spadpaste{matrix([[ev(row,col) for row in 1..4] for col in 1..4]) \free{ev}}
}

\xtc{
This function returns {\tt true} if a polynomial in \axiom{x} has multiple
roots, {\tt false} otherwise.
It is defined and applied in the same expression.
}{
\spadpaste{( p +-> not one?(gcd(p,D(p,x))) )(x**2+4*x+4)}
}
```

```
\xtc{
This and the next expression are equivalent.
}{
\spadpaste{g(x,y,z) == cos(x + sin(y + tan(z)))}
}
\xtc{
The one you use is a matter of taste.
}{
\spadpaste{g == (x,y,z) +-> cos(x + sin(y + tan(z)))}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugUserAnonExampPagePatch1}
\begin{paste}{ugUserAnonExampPageFull1}{ugUserAnonExampPageEmpty1}
\pastebutton{ugUserAnonExampPageFull1}{\hidepaste}
\tab{5}\spadcommand{x +-> if x < 0 then -x else x\bound{anon0 }}
\indentrel{3}\begin{verbatim}
   (1)
     x
   +->
     if x < 0
       then - x
       else x
                                 Type: AnonymousFunction
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserAnonExampPageEmpty1}
\begin{paste}{ugUserAnonExampPageEmpty1}{ugUserAnonExampPagePatch1}
\pastebutton{ugUserAnonExampPageEmpty1}{\showpaste}
\tab{5}\spadcommand{x +-> if x < 0 then -x else x\bound{anon0 }}
\end{paste}\end{patch}

\begin{patch}{ugUserAnonExampPagePatch2}
\begin{paste}{ugUserAnonExampPageFull2}{ugUserAnonExampPageEmpty2}
\pastebutton{ugUserAnonExampPageFull2}{\hidepaste}
\tab{5}\spadcommand{abs1 := \%\free{anon0 }\bound{abs1 }}
\indentrel{3}\begin{verbatim}
   (2)
     x
   +->
     if x < 0
       then - x
       else x
                                 Type: AnonymousFunction
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugUserAnonExampPageEmpty2}
\begin{paste}{ugUserAnonExampPageEmpty2}{ugUserAnonExampPagePatch2}
\pastebutton{ugUserAnonExampPageEmpty2}{\showpaste}
\tab{5}\spadcommand{abs1 := \%\free{anon0 }\bound{abs1 }}
\end{paste}\end{patch}

\begin{patch}{ugUserAnonExampPagePatch3}
\begin{paste}{ugUserAnonExampPageFull3}{ugUserAnonExampPageEmpty3}
\pastebutton{ugUserAnonExampPageFull3}{\hidepaste}
\tab{5}\spadcommand{(x,y) +-> abs1(x) > abs1(y)\bound{anon1 }\free{abs1 }}
\indentrel{3}\begin{verbatim}
   (3)  (x,y) +-> abs1(y) < abs1(x)
                                Type: AnonymousFunction
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserAnonExampPageEmpty3}
\begin{paste}{ugUserAnonExampPageEmpty3}{ugUserAnonExampPagePatch3}
\pastebutton{ugUserAnonExampPageEmpty3}{\showpaste}
\tab{5}\spadcommand{(x,y) +-> abs1(x) > abs1(y)\bound{anon1 }\free{abs1 }}
\end{paste}\end{patch}

\begin{patch}{ugUserAnonExampPagePatch4}
\begin{paste}{ugUserAnonExampPageFull4}{ugUserAnonExampPageEmpty4}
\pastebutton{ugUserAnonExampPageFull4}{\hidepaste}
\tab{5}\spadcommand{sort(\%,[3,9,-4,10,-3,-1,-9,5])\free{anon1 }}
\indentrel{3}\begin{verbatim}
   (4)  [10,- 9,9,5,- 4,- 3,3,- 1]
                                     Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserAnonExampPageEmpty4}
\begin{paste}{ugUserAnonExampPageEmpty4}{ugUserAnonExampPagePatch4}
\pastebutton{ugUserAnonExampPageEmpty4}{\showpaste}
\tab{5}\spadcommand{sort(\%,[3,9,-4,10,-3,-1,-9,5])\free{anon1 }}
\end{paste}\end{patch}

\begin{patch}{ugUserAnonExampPagePatch5}
\begin{paste}{ugUserAnonExampPageFull5}{ugUserAnonExampPageEmpty5}
\pastebutton{ugUserAnonExampPageFull5}{\hidepaste}
\tab{5}\spadcommand{ev := ( (i,j) +-> if even?(i+j) then 1 else -1)\bound{ev }}
\indentrel{3}\begin{verbatim}
   (5)
     (i,j)
   +->
     if even?(i + j)
       then 1
       else - 1
```

```
                                   Type: AnonymousFunction
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserAnonExampPageEmpty5}
\begin{paste}{ugUserAnonExampPageEmpty5}{ugUserAnonExampPagePatch5}
\pastebutton{ugUserAnonExampPageEmpty5}{\showpaste}
\tab{5}\spadcommand{ev := ( (i,j) +-> if even?(i+j) then 1 else -1)\bound{ev }}
\end{paste}\end{patch}

\begin{patch}{ugUserAnonExampPagePatch6}
\begin{paste}{ugUserAnonExampPageFull6}{ugUserAnonExampPageEmpty6}
\pastebutton{ugUserAnonExampPageFull6}{\hidepaste}
\tab{5}\spadcommand{matrix([[ev(row,col) for row in 1..4] for col in 1..4])\free{ev }}
\indentrel{3}\begin{verbatim}
         1   - 1   1   - 1

        - 1   1   - 1   1
   (6)
         1   - 1   1   - 1

        - 1   1   - 1   1
                                      Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserAnonExampPageEmpty6}
\begin{paste}{ugUserAnonExampPageEmpty6}{ugUserAnonExampPagePatch6}
\pastebutton{ugUserAnonExampPageEmpty6}{\showpaste}
\tab{5}\spadcommand{matrix([[ev(row,col) for row in 1..4] for col in 1..4])\free{ev }}
\end{paste}\end{patch}

\begin{patch}{ugUserAnonExampPagePatch7}
\begin{paste}{ugUserAnonExampPageFull7}{ugUserAnonExampPageEmpty7}
\pastebutton{ugUserAnonExampPageFull7}{\hidepaste}
\tab{5}\spadcommand{( p +-> not one?(gcd(p,D(p,x))) )(x**2+4*x+4)}
\indentrel{3}\begin{verbatim}
   (7)  true
                                             Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserAnonExampPageEmpty7}
\begin{paste}{ugUserAnonExampPageEmpty7}{ugUserAnonExampPagePatch7}
\pastebutton{ugUserAnonExampPageEmpty7}{\showpaste}
\tab{5}\spadcommand{( p +-> not one?(gcd(p,D(p,x))) )(x**2+4*x+4)}
\end{paste}\end{patch}

\begin{patch}{ugUserAnonExampPagePatch8}
\begin{paste}{ugUserAnonExampPageFull8}{ugUserAnonExampPageEmpty8}
```

```
\pastebutton{ugUserAnonExampPageFull8}{\hidepaste}
\tab{5}\spadcommand{g(x,y,z) == cos(x + sin(y + tan(z)))}
\indentrel{3}\begin{verbatim}
                                          Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserAnonExampPageEmpty8}
\begin{paste}{ugUserAnonExampPageEmpty8}{ugUserAnonExampPagePatch8}
\pastebutton{ugUserAnonExampPageEmpty8}{\showpaste}
\tab{5}\spadcommand{g(x,y,z) == cos(x + sin(y + tan(z)))}
\end{paste}\end{patch}

\begin{patch}{ugUserAnonExampPagePatch9}
\begin{paste}{ugUserAnonExampPageFull9}{ugUserAnonExampPageEmpty9}
\pastebutton{ugUserAnonExampPageFull9}{\hidepaste}
\tab{5}\spadcommand{g == (x,y,z) +-> cos(x + sin(y + tan(z)))}
\indentrel{3}\begin{verbatim}
                                          Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserAnonExampPageEmpty9}
\begin{paste}{ugUserAnonExampPageEmpty9}{ugUserAnonExampPagePatch9}
\pastebutton{ugUserAnonExampPageEmpty9}{\showpaste}
\tab{5}\spadcommand{g == (x,y,z) +-> cos(x + sin(y + tan(z)))}
\end{paste}\end{patch}
```

———-

Declaring Anonymous Functions

— ug06.ht —

```
\begin{page}{ugUserAnonDeclarePage}{6.17.2. Declaring Anonymous Functions}
\beginscroll

If you declare any of the arguments you must declare all of them.
Thus,
\begin{verbatim}
(x: INT,y): FRAC INT +-> (x + 2*y)/(y - 1)
\end{verbatim}
is not legal.

\xtc{
This is an example of a fully declared anonymous
function.
```

```
The output shown just indicates that the object you created is a
particular kind of map, that is, function.
}{
\spadpaste{(x: INT,y: INT): FRAC INT +-> (x + 2*y)/(y - 1)}
}
\xtc{
Axiom allows you to declare the arguments and not declare
the return type.
}{
\spadpaste{(x: INT,y: INT) +-> (x + 2*y)/(y - 1)}
}
The return type is computed from the types of the arguments and the
body of the function.
You cannot declare the return type if you do not declare the arguments.
Therefore,
\begin{verbatim}
(x,y): FRAC INT +-> (x + 2*y)/(y - 1)
\end{verbatim}
is not legal.

\xtc{
This and the next expression are equivalent.
}{
\spadpaste{h(x: INT,y: INT): FRAC INT == (x + 2*y)/(y - 1)}
}
\xtc{
The one you use is a matter of taste.
}{
\spadpaste{h == (x: INT,y: INT): FRAC INT +-> (x + 2*y)/(y - 1)}
}

When should you declare an anonymous function?
\indent{4}
\beginitems
\item[1. ] If you use an anonymous function and Axiom can’t figure
out what you are trying to do, declare the function.
\item[2. ] If the function has nontrivial argument types or a
nontrivial return type that
Axiom may be able to determine eventually, but you are not
willing to wait that long, declare the function.
\item[3. ] If the function will only be used for arguments of specific
types and it is not too much trouble to declare the function, do so.
\item[4. ] If you are using the anonymous function as an argument to
another function (such as \axiomFun{map} or \axiomFun{sort}),
consider declaring the function.
\item[5. ] If you define an anonymous function inside a named function,
you {\it must} declare the anonymous function.
\enditems
\indent{0}
```

```
\xtc{
This is an example of a named function for integers that returns a
function.
}{
\spadpaste{addx x == ((y: Integer): Integer +-> x + y) \bound{addx}}
}
\xtc{
We define \userfun{g} to be a function that adds \axiom{10} to its
argument.
}{
\spadpaste{g := addx 10 \free{addx}\bound{g}}
}
\xtc{
Try it out.
}{
\spadpaste{g 3 \free{g}}
}
\xtc{
}{
\spadpaste{g(-4) \free{g}}
}

An anonymous function cannot be recursive: since it does not have a
name, you cannot even call it within itself!
If you place an anonymous function inside a named function, the
anonymous function must be declared.

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugUserAnonDeclarePagePatch1}
\begin{paste}{ugUserAnonDeclarePageFull1}{ugUserAnonDeclarePageEmpty1}
\pastebutton{ugUserAnonDeclarePageFull1}{\hidepaste}
\tab{5}\spadcommand{(x: INT,y: INT): FRAC INT +-> (x + 2*y)/(y - 1)}
\indentrel{3}\begin{verbatim}
   (1)  theMap(NIL)
          Type: ((Integer,Integer) -> Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserAnonDeclarePageEmpty1}
\begin{paste}{ugUserAnonDeclarePageEmpty1}{ugUserAnonDeclarePagePatch1}
\pastebutton{ugUserAnonDeclarePageEmpty1}{\showpaste}
\tab{5}\spadcommand{(x: INT,y: INT): FRAC INT +-> (x + 2*y)/(y - 1)}
\end{paste}\end{patch}

\begin{patch}{ugUserAnonDeclarePagePatch2}
\begin{paste}{ugUserAnonDeclarePageFull2}{ugUserAnonDeclarePageEmpty2}
\pastebutton{ugUserAnonDeclarePageFull2}{\hidepaste}
```

```
\tab{5}\spadcommand{(x: INT,y: INT) +-> (x + 2*y)/(y - 1)}
\indentrel{3}\begin{verbatim}
   (2)  theMap(NIL)
          Type: ((Integer,Integer) -> Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserAnonDeclarePageEmpty2}
\begin{paste}{ugUserAnonDeclarePageEmpty2}{ugUserAnonDeclarePagePatch2}
\pastebutton{ugUserAnonDeclarePageEmpty2}{\showpaste}
\tab{5}\spadcommand{(x: INT,y: INT) +-> (x + 2*y)/(y - 1)}
\end{paste}\end{patch}

\begin{patch}{ugUserAnonDeclarePagePatch3}
\begin{paste}{ugUserAnonDeclarePageFull3}{ugUserAnonDeclarePageEmpty3}
\pastebutton{ugUserAnonDeclarePageFull3}{\hidepaste}
\tab{5}\spadcommand{h(x: INT,y: INT): FRAC INT == (x + 2*y)/(y - 1)}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserAnonDeclarePageEmpty3}
\begin{paste}{ugUserAnonDeclarePageEmpty3}{ugUserAnonDeclarePagePatch3}
\pastebutton{ugUserAnonDeclarePageEmpty3}{\showpaste}
\tab{5}\spadcommand{h(x: INT,y: INT): FRAC INT == (x + 2*y)/(y - 1)}
\end{paste}\end{patch}

\begin{patch}{ugUserAnonDeclarePagePatch4}
\begin{paste}{ugUserAnonDeclarePageFull4}{ugUserAnonDeclarePageEmpty4}
\pastebutton{ugUserAnonDeclarePageFull4}{\hidepaste}
\tab{5}\spadcommand{h == (x: INT,y: INT): FRAC INT +-> (x + 2*y)/(y - 1)}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserAnonDeclarePageEmpty4}
\begin{paste}{ugUserAnonDeclarePageEmpty4}{ugUserAnonDeclarePagePatch4}
\pastebutton{ugUserAnonDeclarePageEmpty4}{\showpaste}
\tab{5}\spadcommand{h == (x: INT,y: INT): FRAC INT +-> (x + 2*y)/(y - 1)}
\end{paste}\end{patch}

\begin{patch}{ugUserAnonDeclarePagePatch5}
\begin{paste}{ugUserAnonDeclarePageFull5}{ugUserAnonDeclarePageEmpty5}
\pastebutton{ugUserAnonDeclarePageFull5}{\hidepaste}
\tab{5}\spadcommand{addx x == ((y: Integer): Integer +-> x + y)\bound{addx }}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserAnonDeclarePageEmpty5}
\begin{paste}{ugUserAnonDeclarePageEmpty5}{ugUserAnonDeclarePagePatch5}
\pastebutton{ugUserAnonDeclarePageEmpty5}{\showpaste}
\tab{5}\spadcommand{addx x == ((y: Integer): Integer +-> x + y)\bound{addx }}
\end{paste}\end{patch}

\begin{patch}{ugUserAnonDeclarePagePatch6}
\begin{paste}{ugUserAnonDeclarePageFull6}{ugUserAnonDeclarePageEmpty6}
\pastebutton{ugUserAnonDeclarePageFull6}{\hidepaste}
\tab{5}\spadcommand{g := addx 10\free{addx }\bound{g }}
\indentrel{3}\begin{verbatim}
   (6)  theMap(LAMBDA_f647nv_704,826)
                             Type: (Integer -> Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserAnonDeclarePageEmpty6}
\begin{paste}{ugUserAnonDeclarePageEmpty6}{ugUserAnonDeclarePagePatch6}
\pastebutton{ugUserAnonDeclarePageEmpty6}{\showpaste}
\tab{5}\spadcommand{g := addx 10\free{addx }\bound{g }}
\end{paste}\end{patch}

\begin{patch}{ugUserAnonDeclarePagePatch7}
\begin{paste}{ugUserAnonDeclarePageFull7}{ugUserAnonDeclarePageEmpty7}
\pastebutton{ugUserAnonDeclarePageFull7}{\hidepaste}
\tab{5}\spadcommand{g 3\free{g }}
\indentrel{3}\begin{verbatim}
   (7)  13
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserAnonDeclarePageEmpty7}
\begin{paste}{ugUserAnonDeclarePageEmpty7}{ugUserAnonDeclarePagePatch7}
\pastebutton{ugUserAnonDeclarePageEmpty7}{\showpaste}
\tab{5}\spadcommand{g 3\free{g }}
\end{paste}\end{patch}

\begin{patch}{ugUserAnonDeclarePagePatch8}
\begin{paste}{ugUserAnonDeclarePageFull8}{ugUserAnonDeclarePageEmpty8}
\pastebutton{ugUserAnonDeclarePageFull8}{\hidepaste}
\tab{5}\spadcommand{g(-4)\free{g }}
\indentrel{3}\begin{verbatim}
   (8)  6
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugUserAnonDeclarePageEmpty8}
\begin{paste}{ugUserAnonDeclarePageEmpty8}{ugUserAnonDeclarePagePatch8}
\pastebutton{ugUserAnonDeclarePageEmpty8}{\showpaste}
\tab{5}\spadcommand{g(-4)\free{g }}
\end{paste}\end{patch}
```

———-

Example: A Database

— ug06.ht —

```
\begin{page}{ugUserDatabasePage}{6.18. Example: A Database}
\beginscroll

This example shows how you can use Axiom to organize a database of
lineage data and then query the database for relationships.

\labelSpace{1.5pc}
\xtc{
The database is entered as ``assertions'' that are really
pieces of a function definition.
}{
\spadpaste{children("albert") == ["albertJr","richard","diane"]\bound{d1}}
}
\xtc{
Each piece
\axiom{children(x) == y} means
``the children of \axiom{x} are \axiom{y}''.
}{
\spadpaste{children("richard") == ["douglas","daniel","susan"]
\free{d1}\bound{d2}}
}
\xtc{
This family tree thus spans four generations.
}{
\spadpaste{children("douglas") == ["dougie","valerie"]\free{d2}\bound{d3}}
}
\xtc{
Say ``no one else has children.''
}{
\spadpaste{children(x) == []\free{d3}\bound{d4}}
}

\xtc{
We need some functions for computing lineage.
Start with \axiom{childOf}.
```

```
}{
\spadpaste{childOf(x,y) == member?(x,children(y))\bound{d9}\free{d10}}
}
\xtc{
To find the \axiom{parentOf} someone,
you have to scan the database of
people applying \axiom{children}.
}{
\begin{spadsrc}[\bound{d8a}\free{d9}]
parentOf(x) ==
  for y in people repeat
    (if childOf(x,y) then return y)
  "unknown"
\end{spadsrc}
}
\xtc{
And a grandparent of \axiom{x} is just a parent of a parent of \axiom{x}.
}{
\spadpaste{grandParentOf(x) == parentOf parentOf x\bound{d8}\free{d8a}}
}
\xtc{
The grandchildren of \axiom{x}
are the people \axiom{y} such that
\axiom{x} is a grandparent of \axiom{y}.
}{
\spadpaste{grandchildren(x) == [y for y in people | grandParentOf(y) = x]
\free{d7}\bound{d8}}
}
\xtc{
Suppose you want to make a list of all great-grandparents.
Well, a great-grandparent is a grandparent of a person who has children.
}{
\begin{spadsrc}[\free{d6}\bound{d7}]
greatGrandParents == [x for x in people |
  reduce(_or,[not empty? children(y) for y in grandchildren(x)],false)]
\end{spadsrc}
}
\xtc{
Define \axiom{descendants} to include the parent as well.
}{
\begin{spadsrc}[\free{d5}\bound{d6}]
descendants(x) ==
  kids := children(x)
  null kids => [x]
  concat(x,reduce(concat,[descendants(y)
    for y in kids],[]))
\end{spadsrc}
}
\xtc{
Finally, we need a list of people.
```

```
Since all people are descendants of ‘‘albert’’, let’s say so.
}{
\spadpaste{people == descendants "albert"\free{d4}\bound{d5}}
}

We have used \axiomSyntax{==} to define the database and some functions to
query the database.
But no computation is done until we ask for some information.
Then, once and for all, the functions are analyzed and compiled to machine
code for run-time efficiency.
Notice that no types are given anywhere in this example.
They are not needed.

\xtc{
Who are the grandchildren of ‘‘richard’’?
}{
\spadpaste{grandchildren "richard"\bound{d10}\free{d11}}
}
\xtc{
Who are the great-grandparents?
}{
\spadpaste{greatGrandParents\bound{d11}\free{d12}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugUserDatabasePagePatch1}
\begin{paste}{ugUserDatabasePageFull1}{ugUserDatabasePageEmpty1}
\pastebutton{ugUserDatabasePageFull1}{\hidepaste}
\tab{5}\spadcommand{children("albert") == ["albertJr","richard","diane"]\bound{d1 }}
\indentrel{3}\begin{verbatim}
                                                      Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserDatabasePageEmpty1}
\begin{paste}{ugUserDatabasePageEmpty1}{ugUserDatabasePagePatch1}
\pastebutton{ugUserDatabasePageEmpty1}{\showpaste}
\tab{5}\spadcommand{children("albert") == ["albertJr","richard","diane"]\bound{d1 }}
\end{paste}\end{patch}

\begin{patch}{ugUserDatabasePagePatch2}
\begin{paste}{ugUserDatabasePageFull2}{ugUserDatabasePageEmpty2}
\pastebutton{ugUserDatabasePageFull2}{\hidepaste}
\tab{5}\spadcommand{children("richard") == ["douglas","daniel","susan"]\free{d1 }\bound{d2 }}
\indentrel{3}\begin{verbatim}
                                                      Type: Void
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserDatabasePageEmpty2}
\begin{paste}{ugUserDatabasePageEmpty2}{ugUserDatabasePagePatch2}
\pastebutton{ugUserDatabasePageEmpty2}{\showpaste}
\tab{5}\spadcommand{children("richard") == ["douglas","daniel","susan"]\free{d1 }\bound{d2 }}
\end{paste}\end{patch}

\begin{patch}{ugUserDatabasePagePatch3}
\begin{paste}{ugUserDatabasePageFull3}{ugUserDatabasePageEmpty3}
\pastebutton{ugUserDatabasePageFull3}{\hidepaste}
\tab{5}\spadcommand{children("douglas") == ["dougie","valerie"]\free{d2 }\bound{d3 }}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserDatabasePageEmpty3}
\begin{paste}{ugUserDatabasePageEmpty3}{ugUserDatabasePagePatch3}
\pastebutton{ugUserDatabasePageEmpty3}{\showpaste}
\tab{5}\spadcommand{children("douglas") == ["dougie","valerie"]\free{d2 }\bound{d3 }}
\end{paste}\end{patch}

\begin{patch}{ugUserDatabasePagePatch4}
\begin{paste}{ugUserDatabasePageFull4}{ugUserDatabasePageEmpty4}
\pastebutton{ugUserDatabasePageFull4}{\hidepaste}
\tab{5}\spadcommand{children(x) == []\free{d3 }\bound{d4 }}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserDatabasePageEmpty4}
\begin{paste}{ugUserDatabasePageEmpty4}{ugUserDatabasePagePatch4}
\pastebutton{ugUserDatabasePageEmpty4}{\showpaste}
\tab{5}\spadcommand{children(x) == []\free{d3 }\bound{d4 }}
\end{paste}\end{patch}

\begin{patch}{ugUserDatabasePagePatch5}
\begin{paste}{ugUserDatabasePageFull5}{ugUserDatabasePageEmpty5}
\pastebutton{ugUserDatabasePageFull5}{\hidepaste}
\tab{5}\spadcommand{childOf(x,y) == member?(x,children(y))\bound{d9 }\free{d10 }}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserDatabasePageEmpty5}
\begin{paste}{ugUserDatabasePageEmpty5}{ugUserDatabasePagePatch5}
\pastebutton{ugUserDatabasePageEmpty5}{\showpaste}
```

```
\tab{5}\spadcommand{childOf(x,y) == member?(x,children(y))\bound{d9 }\free{d10 }}
\end{paste}\end{patch}

\begin{patch}{ugUserDatabasePagePatch6}
\begin{paste}{ugUserDatabasePageFull6}{ugUserDatabasePageEmpty6}
\pastebutton{ugUserDatabasePageFull6}{\hidepaste}
\tab{5}\spadcommand{parentOf(x) ==
  for y in people repeat
    (if childOf(x,y) then return y)
  "unknown"
\bound{d8a }\free{d9 }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserDatabasePageEmpty6}
\begin{paste}{ugUserDatabasePageEmpty6}{ugUserDatabasePagePatch6}
\pastebutton{ugUserDatabasePageEmpty6}{\showpaste}
\tab{5}\spadcommand{parentOf(x) ==
  for y in people repeat
    (if childOf(x,y) then return y)
  "unknown"
\bound{d8a }\free{d9 }}
\end{paste}\end{patch}

\begin{patch}{ugUserDatabasePagePatch7}
\begin{paste}{ugUserDatabasePageFull7}{ugUserDatabasePageEmpty7}
\pastebutton{ugUserDatabasePageFull7}{\hidepaste}
\tab{5}\spadcommand{grandParentOf(x) == parentOf parentOf x\bound{d8 }\free{d8a }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserDatabasePageEmpty7}
\begin{paste}{ugUserDatabasePageEmpty7}{ugUserDatabasePagePatch7}
\pastebutton{ugUserDatabasePageEmpty7}{\showpaste}
\tab{5}\spadcommand{grandParentOf(x) == parentOf parentOf x\bound{d8 }\free{d8a }}
\end{paste}\end{patch}

\begin{patch}{ugUserDatabasePagePatch8}
\begin{paste}{ugUserDatabasePageFull8}{ugUserDatabasePageEmpty8}
\pastebutton{ugUserDatabasePageFull8}{\hidepaste}
\tab{5}\spadcommand{grandchildren(x) == [y for y in people | grandParentOf(y) = x]\free{d7 }\bou
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugUserDatabasePageEmpty8}
\begin{paste}{ugUserDatabasePageEmpty8}{ugUserDatabasePagePatch8}
\pastebutton{ugUserDatabasePageEmpty8}{\showpaste}
\tab{5}\spadcommand{grandchildren(x) == [y for y in people | grandParentOf(y) = x]\free{d7 }\bound{d8 }}
\end{paste}\end{patch}

\begin{patch}{ugUserDatabasePagePatch9}
\begin{paste}{ugUserDatabasePageFull9}{ugUserDatabasePageEmpty9}
\pastebutton{ugUserDatabasePageFull9}{\hidepaste}
\tab{5}\spadcommand{greatGrandParents == [x for x in people |
  reduce(_or,[not empty? children(y) for y in grandchildren(x)],false)]
\free{d6 }\bound{d7 }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserDatabasePageEmpty9}
\begin{paste}{ugUserDatabasePageEmpty9}{ugUserDatabasePagePatch9}
\pastebutton{ugUserDatabasePageEmpty9}{\showpaste}
\tab{5}\spadcommand{greatGrandParents == [x for x in people |
  reduce(_or,[not empty? children(y) for y in grandchildren(x)],false)]
\free{d6 }\bound{d7 }}
\end{paste}\end{patch}

\begin{patch}{ugUserDatabasePagePatch10}
\begin{paste}{ugUserDatabasePageFull10}{ugUserDatabasePageEmpty10}
\pastebutton{ugUserDatabasePageFull10}{\hidepaste}
\tab{5}\spadcommand{descendants(x) ==
  kids := children(x)
  null kids => [x]
  concat(x,reduce(concat,[descendants(y)
    for y in kids],[]))
\free{d5 }\bound{d6 }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserDatabasePageEmpty10}
\begin{paste}{ugUserDatabasePageEmpty10}{ugUserDatabasePagePatch10}
\pastebutton{ugUserDatabasePageEmpty10}{\showpaste}
\tab{5}\spadcommand{descendants(x) ==
  kids := children(x)
  null kids => [x]
  concat(x,reduce(concat,[descendants(y)
    for y in kids],[]))
\free{d5 }\bound{d6 }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugUserDatabasePagePatch11}
\begin{paste}{ugUserDatabasePageFull11}{ugUserDatabasePageEmpty11}
\pastebutton{ugUserDatabasePageFull11}{\hidepaste}
\tab{5}\spadcommand{people == descendants "albert"\free{d4 }\bound{d5 }}
\indentrel{3}\begin{verbatim}
                                              Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserDatabasePageEmpty11}
\begin{paste}{ugUserDatabasePageEmpty11}{ugUserDatabasePagePatch11}
\pastebutton{ugUserDatabasePageEmpty11}{\showpaste}
\tab{5}\spadcommand{people == descendants "albert"\free{d4 }\bound{d5 }}
\end{paste}\end{patch}

\begin{patch}{ugUserDatabasePagePatch12}
\begin{paste}{ugUserDatabasePageFull12}{ugUserDatabasePageEmpty12}
\pastebutton{ugUserDatabasePageFull12}{\hidepaste}
\tab{5}\spadcommand{grandchildren "richard"\bound{d10 }\free{d11 }}
\indentrel{3}\begin{verbatim}
   (12)  ["dougie","valerie"]
                                       Type: List String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserDatabasePageEmpty12}
\begin{paste}{ugUserDatabasePageEmpty12}{ugUserDatabasePagePatch12}
\pastebutton{ugUserDatabasePageEmpty12}{\showpaste}
\tab{5}\spadcommand{grandchildren "richard"\bound{d10 }\free{d11 }}
\end{paste}\end{patch}

\begin{patch}{ugUserDatabasePagePatch13}
\begin{paste}{ugUserDatabasePageFull13}{ugUserDatabasePageEmpty13}
\pastebutton{ugUserDatabasePageFull13}{\hidepaste}
\tab{5}\spadcommand{greatGrandParents\bound{d11 }\free{d12 }}
\indentrel{3}\begin{verbatim}
   (13)  ["albert"]
                                       Type: List String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserDatabasePageEmpty13}
\begin{paste}{ugUserDatabasePageEmpty13}{ugUserDatabasePagePatch13}
\pastebutton{ugUserDatabasePageEmpty13}{\showpaste}
\tab{5}\spadcommand{greatGrandParents\bound{d11 }\free{d12 }}
\end{paste}\end{patch}
```

Example: A Famous Triangle

⇒ "notitle" (ugTypesExposePage) 7 on page 1699

— **ug06.ht** —

```
\begin{page}{ugUserTrianglePage}{6.19. Example: A Famous Triangle}
\beginscroll

In this example we write some functions that display
Pascal's triangle.
It demonstrates the use of piece-wise definitions and some output
operations you probably haven't seen before.

\labelSpace{1pc}
\xtc{
To make these output operations
available, we have to \spadgloss{expose} the domain
\axiomType{OutputForm}.
See \downlink{``Exposing Domains and Packages''}{ugTypesExposePage}
in Section 2.11\ignore{ugTypesExpose} for
more information about exposing domains
and packages.
}{
\spadpaste{)set expose add constructor OutputForm \bound{expose}}
}
\xtc{
Define the values along the first
row and any column \axiom{i}.
}{
\spadpaste{pascal(1,i) == 1 \bound{pas1}}
}
\xtc{
Define the values for when the row
and column index \axiom{i} are equal.
Repeating the argument name indicates that
the two index values are equal.
}{
\spadpaste{pascal(n,n) == 1 \bound{pas2}\free{pas1}}
}
\xtc{
}{
\begin{spadsrc}[\bound{pas3}\free{pas1 pas2}]
pascal(i,j | 1 < i and i < j) ==
   pascal(i-1,j-1)+pascal(i,j-1)
\end{spadsrc}
}
Now that we have defined the coefficients in Pascal's triangle,
let's write a couple of one-liners to display it.
\xtc{
```

```
First, define a function that gives the \eth{\axiom{n}} row.
}{
\spadpaste{pascalRow(n) == [pascal(i,n) for i in 1..n]
\bound{pascalRow}\free{pas3}}
}
\xtc{
Next, we write the function \userfun{displayRow}
to display the row, separating entries by blanks and centering.
}{
\spadpaste{displayRow(n) == output center blankSeparate pascalRow(n)
\free{pascalRow}\bound{displayRow}\free{expose}}
}
%
Here we have used three output operations.
Operation \axiomFunFrom{output}{OutputForm}
displays the printable form of objects on the screen,
\axiomFunFrom{center}{OutputForm} centers a printable form in the
width of the screen, and \axiomFunFrom{blankSeparate}{OutputForm}
takes a list of printable forms and inserts a blank between
successive elements.
\xtc{
Look at the result.
}{
\spadpaste{for i in 1..7 repeat displayRow i \free{displayRow}}
}
Being purists, we find this less than satisfactory.
Traditionally, elements of Pascal's triangle are centered between
the left and right elements on the line above.
%
\xtc{
To fix this misalignment, we go back and
redefine \userfun{pascalRow} to right adjust the entries within the
triangle within a width of four characters.
}{
\spadpaste{pascalRow(n) == [right(pascal(i,n),4) for i in 1..n]
\bound{pascalRow2}}
}
%
\xtc{
Finally let's look at our purely reformatted triangle.
}{
\spadpaste{for i in 1..7 repeat displayRow i \free{pascalRow2}
\free{displayRow}}
}
\xtc{
Unexpose \axiomType{OutputForm} so we don't get unexpected
results later.
}{
\spadpaste{)set expose drop constructor OutputForm}
}
```

```
\endscroll
\autobuttons
\end{page}

\begin{patch}{ugUserTrianglePagePatch1}
\begin{paste}{ugUserTrianglePageFull1}{ugUserTrianglePageEmpty1}
\pastebutton{ugUserTrianglePageFull1}{\hidepaste}
\tab{5}\spadcommand{)set expose add constructor OutputForm\bound{expose }}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserTrianglePageEmpty1}
\begin{paste}{ugUserTrianglePageEmpty1}{ugUserTrianglePagePatch1}
\pastebutton{ugUserTrianglePageEmpty1}{\showpaste}
\tab{5}\spadcommand{)set expose add constructor OutputForm\bound{expose }}
\end{paste}\end{patch}

\begin{patch}{ugUserTrianglePagePatch2}
\begin{paste}{ugUserTrianglePageFull2}{ugUserTrianglePageEmpty2}
\pastebutton{ugUserTrianglePageFull2}{\hidepaste}
\tab{5}\spadcommand{pascal(1,i) == 1\bound{pas1 }}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserTrianglePageEmpty2}
\begin{paste}{ugUserTrianglePageEmpty2}{ugUserTrianglePagePatch2}
\pastebutton{ugUserTrianglePageEmpty2}{\showpaste}
\tab{5}\spadcommand{pascal(1,i) == 1\bound{pas1 }}
\end{paste}\end{patch}

\begin{patch}{ugUserTrianglePagePatch3}
\begin{paste}{ugUserTrianglePageFull3}{ugUserTrianglePageEmpty3}
\pastebutton{ugUserTrianglePageFull3}{\hidepaste}
\tab{5}\spadcommand{pascal(n,n) == 1\bound{pas2 }\free{pas1 }}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserTrianglePageEmpty3}
\begin{paste}{ugUserTrianglePageEmpty3}{ugUserTrianglePagePatch3}
\pastebutton{ugUserTrianglePageEmpty3}{\showpaste}
\tab{5}\spadcommand{pascal(n,n) == 1\bound{pas2 }\free{pas1 }}
\end{paste}\end{patch}

\begin{patch}{ugUserTrianglePagePatch4}
```

```
\begin{paste}{ugUserTrianglePageFull4}{ugUserTrianglePageEmpty4}
\pastebutton{ugUserTrianglePageFull4}{\hidepaste}
\tab{5}\spadcommand{pascal(i,j | 1 < i and i < j) ==
   pascal(i-1,j-1)+pascal(i,j-1)
\bound{pas3 }\free{pas1 pas2 }}
\indentrel{3}\begin{verbatim}
                                       Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserTrianglePageEmpty4}
\begin{paste}{ugUserTrianglePageEmpty4}{ugUserTrianglePagePatch4}
\pastebutton{ugUserTrianglePageEmpty4}{\showpaste}
\tab{5}\spadcommand{pascal(i,j | 1 < i and i < j) ==
   pascal(i-1,j-1)+pascal(i,j-1)
\bound{pas3 }\free{pas1 pas2 }}
\end{paste}\end{patch}

\begin{patch}{ugUserTrianglePagePatch5}
\begin{paste}{ugUserTrianglePageFull5}{ugUserTrianglePageEmpty5}
\pastebutton{ugUserTrianglePageFull5}{\hidepaste}
\tab{5}\spadcommand{pascalRow(n) == [pascal(i,n) for i in 1..n]\bound{pascalRow }\free{pas3 }}
\indentrel{3}\begin{verbatim}
                                       Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserTrianglePageEmpty5}
\begin{paste}{ugUserTrianglePageEmpty5}{ugUserTrianglePagePatch5}
\pastebutton{ugUserTrianglePageEmpty5}{\showpaste}
\tab{5}\spadcommand{pascalRow(n) == [pascal(i,n) for i in 1..n]\bound{pascalRow }\free{pas3 }}
\end{paste}\end{patch}

\begin{patch}{ugUserTrianglePagePatch6}
\begin{paste}{ugUserTrianglePageFull6}{ugUserTrianglePageEmpty6}
\pastebutton{ugUserTrianglePageFull6}{\hidepaste}
\tab{5}\spadcommand{displayRow(n) == output center blankSeparate pascalRow(n)\free{pascalRow }\b
\indentrel{3}\begin{verbatim}
                                       Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserTrianglePageEmpty6}
\begin{paste}{ugUserTrianglePageEmpty6}{ugUserTrianglePagePatch6}
\pastebutton{ugUserTrianglePageEmpty6}{\showpaste}
\tab{5}\spadcommand{displayRow(n) == output center blankSeparate pascalRow(n)\free{pascalRow }\b
\end{paste}\end{patch}

\begin{patch}{ugUserTrianglePagePatch7}
\begin{paste}{ugUserTrianglePageFull7}{ugUserTrianglePageEmpty7}
```

```
\pastebutton{ugUserTrianglePageFull7}{\hidepaste}
\tab{5}\spadcommand{for i in 1..7 repeat displayRow i\free{displayRow }}
\indentrel{3}\begin{verbatim}
                                   1
                                  1 1
                                 1 2 1
                                1 3 3 1
                               1 4 6 4 1
                             1 5 10 10 5 1
                            1 6 15 20 15 6 1
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserTrianglePageEmpty7}
\begin{paste}{ugUserTrianglePageEmpty7}{ugUserTrianglePagePatch7}
\pastebutton{ugUserTrianglePageEmpty7}{\showpaste}
\tab{5}\spadcommand{for i in 1..7 repeat displayRow i\free{displayRow }}
\end{paste}\end{patch}

\begin{patch}{ugUserTrianglePagePatch8}
\begin{paste}{ugUserTrianglePageFull8}{ugUserTrianglePageEmpty8}
\pastebutton{ugUserTrianglePageFull8}{\hidepaste}
\tab{5}\spadcommand{pascalRow(n) == [right(pascal(i,n),4) for i in 1..n]\bound{pascalRow2 }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserTrianglePageEmpty8}
\begin{paste}{ugUserTrianglePageEmpty8}{ugUserTrianglePagePatch8}
\pastebutton{ugUserTrianglePageEmpty8}{\showpaste}
\tab{5}\spadcommand{pascalRow(n) == [right(pascal(i,n),4) for i in 1..n]\bound{pascalRow2 }}
\end{paste}\end{patch}

\begin{patch}{ugUserTrianglePagePatch9}
\begin{paste}{ugUserTrianglePageFull9}{ugUserTrianglePageEmpty9}
\pastebutton{ugUserTrianglePageFull9}{\hidepaste}
\tab{5}\spadcommand{for i in 1..7 repeat displayRow i\free{pascalRow2 }\free{displayRow }}
\indentrel{3}\begin{verbatim}
                                     1
                                  1    1
                                1    2    1
                             1    3    3    1
                           1    4    6    4    1
                        1    5   10   10    5    1
                      1    6   15   20   15    6    1
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugUserTrianglePageEmpty9}
\begin{paste}{ugUserTrianglePageEmpty9}{ugUserTrianglePagePatch9}
\pastebutton{ugUserTrianglePageEmpty9}{\showpaste}
\tab{5}\spadcommand{for i in 1..7 repeat displayRow i\free{pascalRow2 }\free{displayRow }}
\end{paste}\end{patch}

\begin{patch}{ugUserTrianglePagePatch10}
\begin{paste}{ugUserTrianglePageFull10}{ugUserTrianglePageEmpty10}
\pastebutton{ugUserTrianglePageFull10}{\hidepaste}
\tab{5}\spadcommand{)set expose drop constructor OutputForm}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserTrianglePageEmpty10}
\begin{paste}{ugUserTrianglePageEmpty10}{ugUserTrianglePagePatch10}
\pastebutton{ugUserTrianglePageEmpty10}{\showpaste}
\tab{5}\spadcommand{)set expose drop constructor OutputForm}
\end{paste}\end{patch}
```

———-

Example: Testing for Palindromes

⇒ “notitle” (ugUserTrianglePage) 10 on page 1939

— **ug06.ht** —

```
\begin{page}{ugUserPalPage}{6.20. Example: Testing for Palindromes}
\beginscroll


In this section we define a function \userfun{pal?} that tests whether
its argument is a {\it palindrome}, that is, something that reads the
same backwards and forwards.  For example, the string ‘‘Madam I’m
Adam’’ is a palindrome (excluding blanks and punctuation) and so is
the number \axiom{123454321.}  The definition works for any datatype
that has \axiom{n} components that are accessed by the indices
\axiom{1\ldots n}.

\xtc{
Here is the definition for \userfun{pal?}.
It is simply a call to an auxiliary function called
\userfun{palAux?}.
We are following the convention of ending a function’s name with
\axiomSyntax{?} if the function returns a \axiomType{Boolean} value.
}{
```

```
\spadpaste{pal? s ==  palAux?(s,1,\#s) \bound{pal}}
}
\xtc{
Here is \userfun{palAux?}.
It works by comparing elements that are equidistant from the start and end
of the object.
}{
\begin{spadsrc}[\bound{palAux}]
palAux?(s,i,j) ==
  j > i =>
    (s.i = s.j) and palAux?(s,i+1,i-1)
  true
\end{spadsrc}
}
\xtc{
Try \userfun{pal?} on some examples.
First, a string.
}{
\spadpaste{pal? "Oxford"  \free{pal palAux}}
}
\xtc{
A list of polynomials.
}{
\spadpaste{pal? [4,a,x-1,0,x-1,a,4]  \free{pal palAux}}
}
\xtc{
A list of integers from the example in
\texht{the last section.}{\downlink{``A Famous Triangle''}
{ugUserTrianglePage} in Section 6.19\ignore{ugUserTriangle}.}
}{
\spadpaste{pal? [1,6,15,20,15,6,1] \free{pal palAux}}
}
\xtc{
To use \userfun{pal?} on an integer, first convert it to a string.
}{
\spadpaste{pal?(1441::String)\free{pal palAux}}
}
\xtc{
Compute an infinite stream of decimal numbers,
each of which is an obvious palindrome.
}{
\spadpaste{ones := [reduce(+,[10**j for j in 0..i]) for i in 1..]
\free{pal palAux}\bound{pal5}}
}
\xtc{
How about their squares?
}{
\spadpaste{squares := [x**2 for x in ones]\free{pal5}\bound{pal6}}
}
\xtc{
```

```
Well, let's test them all!
}{
\spadpaste{[pal?(x::String) for x in squares]\free{pal6}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugUserPalPagePatch1}
\begin{paste}{ugUserPalPageFull1}{ugUserPalPageEmpty1}
\pastebutton{ugUserPalPageFull1}{\hidepaste}
\tab{5}\spadcommand{pal? s == palAux?(s,1,\#s)\bound{pal }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserPalPageEmpty1}
\begin{paste}{ugUserPalPageEmpty1}{ugUserPalPagePatch1}
\pastebutton{ugUserPalPageEmpty1}{\showpaste}
\tab{5}\spadcommand{pal? s == palAux?(s,1,\#s)\bound{pal }}
\end{paste}\end{patch}

\begin{patch}{ugUserPalPagePatch2}
\begin{paste}{ugUserPalPageFull2}{ugUserPalPageEmpty2}
\pastebutton{ugUserPalPageFull2}{\hidepaste}
\tab{5}\spadcommand{palAux?(s,i,j) ==
  j > i =>
    (s.i = s.j) and palAux?(s,i+1,i-1)
  true
\bound{palAux }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserPalPageEmpty2}
\begin{paste}{ugUserPalPageEmpty2}{ugUserPalPagePatch2}
\pastebutton{ugUserPalPageEmpty2}{\showpaste}
\tab{5}\spadcommand{palAux?(s,i,j) ==
  j > i =>
    (s.i = s.j) and palAux?(s,i+1,i-1)
  true
\bound{palAux }}
\end{paste}\end{patch}

\begin{patch}{ugUserPalPagePatch3}
\begin{paste}{ugUserPalPageFull3}{ugUserPalPageEmpty3}
\pastebutton{ugUserPalPageFull3}{\hidepaste}
```

```
\tab{5}\spadcommand{pal? "Oxford"\free{pal palAux }}
\indentrel{3}\begin{verbatim}
   (3)  false
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserPalPageEmpty3}
\begin{paste}{ugUserPalPageEmpty3}{ugUserPalPagePatch3}
\pastebutton{ugUserPalPageEmpty3}{\showpaste}
\tab{5}\spadcommand{pal? "Oxford"\free{pal palAux }}
\end{paste}\end{patch}

\begin{patch}{ugUserPalPagePatch4}
\begin{paste}{ugUserPalPageFull4}{ugUserPalPageEmpty4}
\pastebutton{ugUserPalPageFull4}{\hidepaste}
\tab{5}\spadcommand{pal? [4,a,x-1,0,x-1,a,4]\free{pal palAux }}
\indentrel{3}\begin{verbatim}
   (4)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserPalPageEmpty4}
\begin{paste}{ugUserPalPageEmpty4}{ugUserPalPagePatch4}
\pastebutton{ugUserPalPageEmpty4}{\showpaste}
\tab{5}\spadcommand{pal? [4,a,x-1,0,x-1,a,4]\free{pal palAux }}
\end{paste}\end{patch}

\begin{patch}{ugUserPalPagePatch5}
\begin{paste}{ugUserPalPageFull5}{ugUserPalPageEmpty5}
\pastebutton{ugUserPalPageFull5}{\hidepaste}
\tab{5}\spadcommand{pal? [1,6,15,20,15,6,1]\free{pal palAux }}
\indentrel{3}\begin{verbatim}
   (5)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserPalPageEmpty5}
\begin{paste}{ugUserPalPageEmpty5}{ugUserPalPagePatch5}
\pastebutton{ugUserPalPageEmpty5}{\showpaste}
\tab{5}\spadcommand{pal? [1,6,15,20,15,6,1]\free{pal palAux }}
\end{paste}\end{patch}

\begin{patch}{ugUserPalPagePatch6}
\begin{paste}{ugUserPalPageFull6}{ugUserPalPageEmpty6}
\pastebutton{ugUserPalPageFull6}{\hidepaste}
\tab{5}\spadcommand{pal?(1441::String)\free{pal palAux }}
\indentrel{3}\begin{verbatim}
```

```
   (6)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserPalPageEmpty6}
\begin{paste}{ugUserPalPageEmpty6}{ugUserPalPagePatch6}
\pastebutton{ugUserPalPageEmpty6}{\showpaste}
\tab{5}\spadcommand{pal?(1441::String)\free{pal palAux }}
\end{paste}\end{patch}

\begin{patch}{ugUserPalPagePatch7}
\begin{paste}{ugUserPalPageFull7}{ugUserPalPageEmpty7}
\pastebutton{ugUserPalPageFull7}{\hidepaste}
\tab{5}\spadcommand{ones := [reduce(+,[10**j for j in 0..i]) for i in 1..]\free{pal palAux }\bou
\indentrel{3}\begin{verbatim}
   (7)
   [11, 111, 1111, 11111, 111111, 1111111, 11111111,
    111111111, 1111111111, 11111111111, ...]
                           Type: Stream PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserPalPageEmpty7}
\begin{paste}{ugUserPalPageEmpty7}{ugUserPalPagePatch7}
\pastebutton{ugUserPalPageEmpty7}{\showpaste}
\tab{5}\spadcommand{ones := [reduce(+,[10**j for j in 0..i]) for i in 1..]\free{pal palAux }\bou
\end{paste}\end{patch}

\begin{patch}{ugUserPalPagePatch8}
\begin{paste}{ugUserPalPageFull8}{ugUserPalPageEmpty8}
\pastebutton{ugUserPalPageFull8}{\hidepaste}
\tab{5}\spadcommand{squares := [x**2 for x in ones]\free{pal5 }\bound{pal6 }}
\indentrel{3}\begin{verbatim}
   (8)
   [121, 12321, 1234321, 123454321, 12345654321,
    1234567654321, 123456787654321, 12345678987654321,
    1234567900987654321, 123456790120987654321, ...]
                           Type: Stream PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserPalPageEmpty8}
\begin{paste}{ugUserPalPageEmpty8}{ugUserPalPagePatch8}
\pastebutton{ugUserPalPageEmpty8}{\showpaste}
\tab{5}\spadcommand{squares := [x**2 for x in ones]\free{pal5 }\bound{pal6 }}
\end{paste}\end{patch}

\begin{patch}{ugUserPalPagePatch9}
\begin{paste}{ugUserPalPageFull9}{ugUserPalPageEmpty9}
```

```
\pastebutton{ugUserPalPageFull9}{\hidepaste}
\tab{5}\spadcommand{[pal?(x::String) for x in squares]\free{pal6 }}
\indentrel{3}\begin{verbatim}
   (9)
   [true,true,true,true,true,true,true,true,true,true,...]
                                  Type: Stream Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserPalPageEmpty9}
\begin{paste}{ugUserPalPageEmpty9}{ugUserPalPagePatch9}
\pastebutton{ugUserPalPageEmpty9}{\showpaste}
\tab{5}\spadcommand{[pal?(x::String) for x in squares]\free{pal6 }}
\end{paste}\end{patch}
```

Rules and Pattern Matching

— ug06.ht —

```
\begin{page}{ugUserRulesPage}{6.21. Rules and Pattern Matching}
\beginscroll

A common mathematical formula is
\texht{\narrowDisplay{%
\log(x) + \log(y) = \log(x y) \quad\forall \, x \hbox{\ and\ } y.}}{
\axiom{log(x) + log(y) == log(x * y)} for any \axiom{x} and \axiom{y}.}
The presence of
\texht{``$\forall$''}{the word ``any''}
indicates that \axiom{x} and \axiom{y} can stand for arbitrary mathematical
expressions in the above formula.
You can use such mathematical formulas in Axiom to specify ``rewrite
rules''.
Rewrite rules are objects in Axiom that can be assigned to variables for
later use, often for the purpose of simplification.
Rewrite rules look like ordinary function definitions except that they are
preceded by the reserved word \axiom{rule}.
\spadkey{rule}
For example, a rewrite rule for the above formula is:
\begin{verbatim}
rule log(x) + log(y) == log(x * y)
\end{verbatim}
Like function definitions, no action is taken when a rewrite rule is issued.
Think of rewrite rules as functions that take one argument.
When a rewrite rule \axiom{A = B} is applied to an argument \axiom{f}, its
meaning is: ``rewrite every subexpression of \axiom{f} that {\it matches}
```

```
\axiom{A} by \axiom{B.}''
The left-hand side of a rewrite rule is called a \spadgloss{pattern}; its
right-side side is called its \spadgloss{substitution}.

\xtc{
Create a rewrite rule named \userfun{logrule}.
The generated symbol beginning with a \axiomSyntax{\%} is a place-holder
for any other terms that might occur in the sum.
}{
\spadpaste{logrule := rule log(x) + log(y) == log(x * y) \bound{logrule}}
}
\xtc{
Create an expression with logarithms.
}{
\spadpaste{f := log sin x + log x \bound{f}}
}
\xtc{
Apply \userfun{logrule} to \axiom{f}.
}{
\spadpaste{logrule f \free{f}\free{logrule}}
}

The meaning of our example rewrite rule is:
``for all expressions \axiom{x} and \axiom{y}, rewrite
\axiom{log(x) + log(y)} by \axiom{log(x * y)}.''
Patterns generally have both operation names
(here, \axiomFun{log} and \axiomOp{+})
and variables (here, \axiom{x} and \axiom{y}).
By default, every operation name stands for itself.
Thus \axiomFun{log}  matches only ``\axiom{log}'' and not any
other operation such as \axiomFun{sin}.
On the other hand, variables do not stand for themselves.
Rather, a variable denotes a
{\it pattern variable} that is free to match any expression whatsoever.

When a rewrite rule is applied, a process called \spadgloss{pattern
matching} goes to work by systematically scanning the subexpressions
of the argument.  When a subexpression is found that ``matches'' the
pattern, the subexpression is replaced by the right-hand side of the
rule.  The details of what happens will be covered later.

The customary Axiom notation for patterns is actually a shorthand for
a longer, more general notation.  Pattern variables can be made
explicit by using a percent (\axiomSyntax{\%}) as the first character
of the variable name.  To say that a name stands for itself, you can
prefix that name with a quote operator (\axiomSyntax{'}).  Although
the current Axiom parser does not let you quote an operation name,
this more general notation gives you an alternate way of giving the
same rewrite rule:
\begin{verbatim}
```

```
rule log(%x) + log(%y) == log(x * y)
\end{verbatim}
This longer notation gives you patterns that the
standard notation won't handle.
For example, the rule
\texht{\typeout{check this example}}{}
\begin{verbatim}
rule %f(c * 'x) ==  c*%f(x)
\end{verbatim}
means ``for all \axiom{f} and \axiom{c}, replace \axiom{f(y)} by
\axiom{c * f(x)} when \axiom{y} is the product of \axiom{c}
and the explicit variable \axiom{x}.''

Thus the pattern can have several adornments on the names that appear there.
Normally, all these adornments are dropped in the substitution on the
right-hand side.

To summarize:

\beginImportant
To enter a single rule in Axiom, use the following syntax:
\spadkey{rule}
\centerline{{\tt rule {\it leftHandSide} == {\it rightHandSide}}}}
The {\it leftHandSide} is a pattern to be matched and
the {\it rightHandSide} is its substitution.
The rule is an object of type \axiomType{RewriteRule} that can be
assigned to a variable and applied to expressions to transform them.
\endImportant

Rewrite rules can be collected into rulesets so that a set of rules
can be applied at once.  Here is another simplification rule for
logarithms.  \texht{\narrowDisplay{y \log(x) = \log(x^y) \quad\forall
\, x \hbox{\ and\ } y.}}{ \axiom{y * log(x) == log(x ** y)} for any
\axiom{x} and \axiom{y}.}  If instead of giving a single rule
following the reserved word \axiom{rule} you give a ``pile'' of rules,
you create what is called a {\it ruleset.}  Like rules, rulesets are
objects in Axiom and can be assigned to variables.  You will find it
useful to group commonly used rules into input files, and read them in
as needed.
\xtc{
Create a ruleset named \axiom{logrules}.
}{
\begin{spadsrc}[\bound{logrules}]
logrules := rule
  log(x) + log(y) == log(x * y)
  y * log x       == log(x ** y)
\end{spadsrc}
}
\xtc{
Again, create an expression \axiom{f} containing logarithms.
```

```
}{
\spadpaste{f := a * log(sin x) - 2 * log x \bound{f1}}
}
\xtc{
Apply the ruleset \userfun{logrules} to \axiom{f}.
}{
\spadpaste{logrules f \free{f1}\free{logrules}}
}

We have allowed pattern variables to match arbitrary expressions in
the above examples.  Often you want a variable only to match
expressions satisfying some predicate.  For example, we may want to
apply the transformation \texht{\narrowDisplay{y \log(x) =
\log(x^y)}}{\axiom{y * log(x) == log(x ** y)}} only when \axiom{y} is
an integer.
%
The way to restrict a pattern variable \axiom{y} by a predicate
\axiom{f(y)} is by using a vertical bar \axiomSyntax{|}, which means
``such that,'' in much the same way it is used in function
definitions.  You do this only once, but at the earliest (meaning
deepest and leftmost) part of the pattern.
\xtc{
This restricts the logarithmic rule to create integer exponents only.
}{
\begin{spadsrc}[\bound{logrules2}]
logrules2 := rule
  log(x) + log(y)          == log(x * y)
  (y | integer? y) * log x == log(x ** y)
\end{spadsrc}
}
\xtc{
Compare this with the result of applying the previous set of rules.
}{
\spadpaste{f \free{f1}}
}
\xtc{
}{
\spadpaste{logrules2 f \free{f1}\free{logrules2}}
}
You should be aware that you might need to apply a function like
\spadfun{integer} within your predicate expression to actually apply
the test function.
\xtc{
Here we use \spadfun{integer} because \spad{n} has
type \spadtype{Expression Integer} but \spadfun{even?} is an operation
defined on integers.
}{
\spadpaste{
evenRule := rule cos(x)**(n | integer? n and even? integer n)==
(1-sin(x)**2)**(n/2) \bound{evenRule}}
```

```
}
\xtc{
Here is the application of the rule.
}{
\spadpaste{evenRule( cos(x)**2 ) \free{evenRule}}
}
\xtc{
This is an example of some of the usual identities involving products of
sines and cosines.
}{
\begin{spadsrc}[\bound{sinCosProducts}]
sinCosProducts == rule
  sin(x) * sin(y) == (cos(x-y) - cos(x + y))/2
  cos(x) * cos(y) == (cos(x-y) + cos(x+y))/2
  sin(x) * cos(y) == (sin(x-y) + sin(x + y))/2
\end{spadsrc}
}
\xtc{
}{
\spadpaste{g := sin(a)*sin(b) + cos(b)*cos(a) + sin(2*a)*cos(2*a)
\bound{g}}
}
\xtc{
}{
\spadpaste{sinCosProducts g \free{sinCosProducts g}}
}

Another qualification you will often want to use is to allow a pattern
to match an identity element.  Using the pattern \axiom{x + y}, for
example, neither \axiom{x} nor \axiom{y} matches the expression
\axiom{0}.  Similarly, if a pattern contains a product \axiom{x*y} or
an exponentiation \axiom{x**y}, then neither \axiom{x} or \axiom{y}
matches \axiom{1}.

\xtc{
If identical elements were matched, pattern matching would generally loop.
Here is an expansion rule for exponentials.
}{
\spadpaste{exprule := rule exp(a + b) == exp(a) * exp(b)\bound{exprule}}
}
\xtc{
This rule would cause infinite rewriting on this if either \axiom{a} or
\axiom{b} were allowed to match \axiom{0}.
}{
\spadpaste{exprule exp x \free{exprule}}
}

There are occasions when you do want a pattern variable in a sum or
product to match \axiom{0} or \axiom{1}.  If so, prefix its name with
a \axiomSyntax{?} whenever it appears in a left-hand side of a rule.
```

```
For example, consider the following rule for the exponential integral:
\texht{\narrowDisplay{\int \left(\frac{y+e^x}{x}\right)\: dx = \int
\frac{y}{x}\: dx + \hbox{\rm Ei}(x) \quad\forall \, x \hbox{\ and\ }
y.}}{ \axiom{integral((y + exp x)/x, x) == integral(y/x, x) + Ei x}
for any \axiom{x} and \axiom{y}.}  This rule is valid for \axiom{y =
0}.  One solution is to create a \axiomType{Ruleset} with two rules,
one with and one without \axiom{y}.  A better solution is to use an
``optional'' pattern variable.

\xtc{
Define rule \axiom{eirule} with
a pattern variable \axiom{?y} to indicate
that an expression may or may not occur.
}{
\spadpaste{eirule := rule integral((?y + exp x)/x,x) ==
integral(y/x,x) + Ei x \bound{eirule}}
}
\xtc{
Apply rule \axiom{eirule} to an integral without this term.
}{
\spadpaste{eirule integral(exp u/u, u) \free{eirule}}
}
\xtc{
Apply rule \axiom{eirule} to an integral with this term.
}{
\spadpaste{eirule integral(sin u + exp u/u, u) \free{eirule}}
}

Here is one final adornment you will find useful.  When matching a
pattern of the form \axiom{x + y} to an expression containing a long
sum of the form \axiom{a +\ldots+ b}, there is no way to predict in
advance which subset of the sum matches \axiom{x} and which matches
\axiom{y}.  Aside from efficiency, this is generally unimportant since
the rule holds for any possible combination of matches for \axiom{x}
and \axiom{y}.  In some situations, however, you may want to say which
pattern variable is a sum (or product) of several terms, and which
should match only a single term.  To do this, put a prefix colon
\axiomSyntax{:} before the pattern variable that you want to match
multiple terms.

\xtc{
The remaining rules involve operators \axiom{u} and \axiom{v}.
}{
\spadpaste{u := operator 'u \bound{u}}
}
\xtc{
These definitions tell Axiom that
\axiom{u} and \axiom{v} are formal operators to be used in expressions.
}{
\spadpaste{v := operator 'v \bound{v}}
```

```
}
\xtc{
First define \axiom{myRule}
with no restrictions on the pattern variables
\axiom{x} and \axiom{y}.
}{
\spadpaste{myRule := rule u(x + y) == u x + v y \free{u v}\bound{m}}
}
\xtc{
Apply \axiom{myRule} to an expression.
}{
\spadpaste{myRule u(a + b + c + d) \free{m}}
}
\xtc{
Define \axiom{myOtherRule} to match several terms
so that the rule gets applied recursively.
}{
\spadpaste{myOtherRule := rule u(:x + y) == u x + v y \free{u v}\bound{m2}}
}
\xtc{
Apply \axiom{myOtherRule} to the same expression.
}{
\spadpaste{myOtherRule u(a + b + c + d) \free{m2}}
}


Here are some final remarks on pattern matching.
Pattern matching provides a very useful paradigm for solving
certain classes of problems, namely, those that involve
transformations of one form to another and back.
However, it is important to recognize its limitations.

First, pattern matching slows down as the number of rules you have to
apply increases.  Thus it is good practice to organize the sets of
rules you use optimally so that irrelevant rules are never included.

Second, careless use of pattern matching can lead to wrong answers.
You should avoid using pattern matching to handle hidden algebraic
relationships that can go undetected by other programs.  As a simple
example, a symbol such as ``J'' can easily be used to represent the
square root of \axiom{-1} or some other important algebraic quantity.
Many algorithms branch on whether an expression is zero or not, then
divide by that expression if it is not.  If you fail to simplify an
expression involving powers of \axiom{J} to \axiom{-1,} algorithms may
incorrectly assume an expression is non-zero, take a wrong branch, and
produce a meaningless result.

Pattern matching should also not be used as a substitute for a domain.
In Axiom, objects of one domain are transformed to objects of other
domains using well-defined \axiomFun{coerce} operations.
```

```
Pattern matching should be used on objects that are all the same type.
Thus if your application can be handled by type \axiomType{Expression} in
Axiom and you think you need pattern matching, consider this choice
carefully.
You may well be better served by extending an existing domain
or by building a new domain of objects for your application.
\endscroll
\autobuttons
\end{page}

\begin{patch}{ugUserRulesPagePatch1}
\begin{paste}{ugUserRulesPageFull1}{ugUserRulesPageEmpty1}
\pastebutton{ugUserRulesPageFull1}{\hidepaste}
\tab{5}\spadcommand{logrule := rule log(x) + log(y) == log(x * y)\bound{logrule }}
\indentrel{3}\begin{verbatim}
   (1)  log(y) + log(x) + %B == log(x y) + %B
  Type: RewriteRule(Integer,Integer,Expression Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserRulesPageEmpty1}
\begin{paste}{ugUserRulesPageEmpty1}{ugUserRulesPagePatch1}
\pastebutton{ugUserRulesPageEmpty1}{\showpaste}
\tab{5}\spadcommand{logrule := rule log(x) + log(y) == log(x * y)\bound{logrule }}
\end{paste}\end{patch}

\begin{patch}{ugUserRulesPagePatch2}
\begin{paste}{ugUserRulesPageFull2}{ugUserRulesPageEmpty2}
\pastebutton{ugUserRulesPageFull2}{\hidepaste}
\tab{5}\spadcommand{f := log sin x + log x\bound{f }}
\indentrel{3}\begin{verbatim}
   (2)  log(sin(x)) + log(x)
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserRulesPageEmpty2}
\begin{paste}{ugUserRulesPageEmpty2}{ugUserRulesPagePatch2}
\pastebutton{ugUserRulesPageEmpty2}{\showpaste}
\tab{5}\spadcommand{f := log sin x + log x\bound{f }}
\end{paste}\end{patch}

\begin{patch}{ugUserRulesPagePatch3}
\begin{paste}{ugUserRulesPageFull3}{ugUserRulesPageEmpty3}
\pastebutton{ugUserRulesPageFull3}{\hidepaste}
\tab{5}\spadcommand{logrule f\free{f }\free{logrule }}
\indentrel{3}\begin{verbatim}
   (3)  log(x sin(x))
                               Type: Expression Integer
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserRulesPageEmpty3}
\begin{paste}{ugUserRulesPageEmpty3}{ugUserRulesPagePatch3}
\pastebutton{ugUserRulesPageEmpty3}{\showpaste}
\tab{5}\spadcommand{logrule f\free{f }\free{logrule }}
\end{paste}\end{patch}

\begin{patch}{ugUserRulesPagePatch4}
\begin{paste}{ugUserRulesPageFull4}{ugUserRulesPageEmpty4}
\pastebutton{ugUserRulesPageFull4}{\hidepaste}
\tab{5}\spadcommand{logrules := rule
  log(x) + log(y) == log(x * y)
  y * log x       == log(x ** y)
\bound{logrules }}
\indentrel{3}\begin{verbatim}
   (4)
   {log(y) + log(x) + %C == log(x y) + %C,
                     y
    y log(x) == log(x )}
      Type: Ruleset(Integer,Integer,Expression Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserRulesPageEmpty4}
\begin{paste}{ugUserRulesPageEmpty4}{ugUserRulesPagePatch4}
\pastebutton{ugUserRulesPageEmpty4}{\showpaste}
\tab{5}\spadcommand{logrules := rule
  log(x) + log(y) == log(x * y)
  y * log x       == log(x ** y)
\bound{logrules }}
\end{paste}\end{patch}

\begin{patch}{ugUserRulesPagePatch5}
\begin{paste}{ugUserRulesPageFull5}{ugUserRulesPageEmpty5}
\pastebutton{ugUserRulesPageFull5}{\hidepaste}
\tab{5}\spadcommand{f := a * log(sin x) - 2 * log x\bound{f1 }}
\indentrel{3}\begin{verbatim}
   (5)  a log(sin(x)) - 2log(x)
                                Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserRulesPageEmpty5}
\begin{paste}{ugUserRulesPageEmpty5}{ugUserRulesPagePatch5}
\pastebutton{ugUserRulesPageEmpty5}{\showpaste}
\tab{5}\spadcommand{f := a * log(sin x) - 2 * log x\bound{f1 }}
\end{paste}\end{patch}

\begin{patch}{ugUserRulesPagePatch6}
```

```
\begin{paste}{ugUserRulesPageFull6}{ugUserRulesPageEmpty6}
\pastebutton{ugUserRulesPageFull6}{\hidepaste}
\tab{5}\spadcommand{logrules f\free{f1 }\free{logrules }}
\indentrel{3}\begin{verbatim}
                  a
            sin(x)
   (6)  log()
                2
               x
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserRulesPageEmpty6}
\begin{paste}{ugUserRulesPageEmpty6}{ugUserRulesPagePatch6}
\pastebutton{ugUserRulesPageEmpty6}{\showpaste}
\tab{5}\spadcommand{logrules f\free{f1 }\free{logrules }}
\end{paste}\end{patch}

\begin{patch}{ugUserRulesPagePatch7}
\begin{paste}{ugUserRulesPageFull7}{ugUserRulesPageEmpty7}
\pastebutton{ugUserRulesPageFull7}{\hidepaste}
\tab{5}\spadcommand{logrules2 := rule
  log(x) + log(y)          == log(x * y)
  (y | integer? y) * log x == log(x ** y)
\bound{logrules2 }}
\indentrel{3}\begin{verbatim}
   (7)
   {log(y) + log(x) + %E == log(x y) + %E,
                     y
    y log(x) == log(x )}
      Type: Ruleset(Integer,Integer,Expression Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserRulesPageEmpty7}
\begin{paste}{ugUserRulesPageEmpty7}{ugUserRulesPagePatch7}
\pastebutton{ugUserRulesPageEmpty7}{\showpaste}
\tab{5}\spadcommand{logrules2 := rule
  log(x) + log(y)          == log(x * y)
  (y | integer? y) * log x == log(x ** y)
\bound{logrules2 }}
\end{paste}\end{patch}

\begin{patch}{ugUserRulesPagePatch8}
\begin{paste}{ugUserRulesPageFull8}{ugUserRulesPageEmpty8}
\pastebutton{ugUserRulesPageFull8}{\hidepaste}
\tab{5}\spadcommand{f\free{f1 }}
\indentrel{3}\begin{verbatim}
   (8)  a log(sin(x)) - 2log(x)
```

```
                              Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserRulesPageEmpty8}
\begin{paste}{ugUserRulesPageEmpty8}{ugUserRulesPagePatch8}
\pastebutton{ugUserRulesPageEmpty8}{\showpaste}
\tab{5}\spadcommand{f\free{f1 }}
\end{paste}\end{patch}

\begin{patch}{ugUserRulesPagePatch9}
\begin{paste}{ugUserRulesPageFull9}{ugUserRulesPageEmpty9}
\pastebutton{ugUserRulesPageFull9}{\hidepaste}
\tab{5}\spadcommand{logrules2 f\free{f1 }\free{logrules2 }}
\indentrel{3}\begin{verbatim}
                           1
   (9)  a log(sin(x)) + log()
                           2
                          x
                              Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserRulesPageEmpty9}
\begin{paste}{ugUserRulesPageEmpty9}{ugUserRulesPagePatch9}
\pastebutton{ugUserRulesPageEmpty9}{\showpaste}
\tab{5}\spadcommand{logrules2 f\free{f1 }\free{logrules2 }}
\end{paste}\end{patch}

\begin{patch}{ugUserRulesPagePatch10}
\begin{paste}{ugUserRulesPageFull10}{ugUserRulesPageEmpty10}
\pastebutton{ugUserRulesPageFull10}{\hidepaste}
\tab{5}\spadcommand{evenRule := rule cos(x)**(n | integer? n and even? integer n)==(1-sin(x)**2)**(n/2)\bour
\indentrel{3}\begin{verbatim}
                                 n

               n             2     2
   (10)  cos(x)  == (- sin(x)  + 1)
  Type: RewriteRule(Integer,Integer,Expression Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserRulesPageEmpty10}
\begin{paste}{ugUserRulesPageEmpty10}{ugUserRulesPagePatch10}
\pastebutton{ugUserRulesPageEmpty10}{\showpaste}
\tab{5}\spadcommand{evenRule := rule cos(x)**(n | integer? n and even? integer n)==(1-sin(x)**2)**(n/2)\bour
\end{paste}\end{patch}

\begin{patch}{ugUserRulesPagePatch11}
\begin{paste}{ugUserRulesPageFull11}{ugUserRulesPageEmpty11}
```

```
\pastebutton{ugUserRulesPageFull11}{\hidepaste}
\tab{5}\spadcommand{evenRule( cos(x)**2 )\free{evenRule }}
\indentrel{3}\begin{verbatim}
                 2
   (11)  - sin(x)  + 1
                              Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserRulesPageEmpty11}
\begin{paste}{ugUserRulesPageEmpty11}{ugUserRulesPagePatch11}
\pastebutton{ugUserRulesPageEmpty11}{\showpaste}
\tab{5}\spadcommand{evenRule( cos(x)**2 )\free{evenRule }}
\end{paste}\end{patch}

\begin{patch}{ugUserRulesPagePatch12}
\begin{paste}{ugUserRulesPageFull12}{ugUserRulesPageEmpty12}
\pastebutton{ugUserRulesPageFull12}{\hidepaste}
\tab{5}\spadcommand{sinCosProducts == rule
  sin(x) * sin(y) == (cos(x-y) - cos(x + y))/2
  cos(x) * cos(y) == (cos(x-y) + cos(x+y))/2
  sin(x) * cos(y) == (sin(x-y) + sin(x + y))/2
\bound{sinCosProducts }}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserRulesPageEmpty12}
\begin{paste}{ugUserRulesPageEmpty12}{ugUserRulesPagePatch12}
\pastebutton{ugUserRulesPageEmpty12}{\showpaste}
\tab{5}\spadcommand{sinCosProducts == rule
  sin(x) * sin(y) == (cos(x-y) - cos(x + y))/2
  cos(x) * cos(y) == (cos(x-y) + cos(x+y))/2
  sin(x) * cos(y) == (sin(x-y) + sin(x + y))/2
\bound{sinCosProducts }}
\end{paste}\end{patch}

\begin{patch}{ugUserRulesPagePatch13}
\begin{paste}{ugUserRulesPageFull13}{ugUserRulesPageEmpty13}
\pastebutton{ugUserRulesPageFull13}{\hidepaste}
\tab{5}\spadcommand{g := sin(a)*sin(b) + cos(b)*cos(a) + sin(2*a)*cos(2*a)\bound{g }}
\indentrel{3}\begin{verbatim}
   (13)  sin(a)sin(b) + cos(2a)sin(2a) + cos(a)cos(b)
                              Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserRulesPageEmpty13}
\begin{paste}{ugUserRulesPageEmpty13}{ugUserRulesPagePatch13}
```

```
\pastebutton{ugUserRulesPageEmpty13}{\showpaste}
\tab{5}\spadcommand{g := sin(a)*sin(b) + cos(b)*cos(a) + sin(2*a)*cos(2*a)\bound{g }}
\end{paste}\end{patch}

\begin{patch}{ugUserRulesPagePatch14}
\begin{paste}{ugUserRulesPageFull14}{ugUserRulesPageEmpty14}
\pastebutton{ugUserRulesPageFull14}{\hidepaste}
\tab{5}\spadcommand{sinCosProducts g\free{sinCosProducts g }}
\indentrel{3}\begin{verbatim}
         sin(4a) + 2cos(b - a)
   (14)  ---------------------
                   2
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserRulesPageEmpty14}
\begin{paste}{ugUserRulesPageEmpty14}{ugUserRulesPagePatch14}
\pastebutton{ugUserRulesPageEmpty14}{\showpaste}
\tab{5}\spadcommand{sinCosProducts g\free{sinCosProducts g }}
\end{paste}\end{patch}

\begin{patch}{ugUserRulesPagePatch15}
\begin{paste}{ugUserRulesPageFull15}{ugUserRulesPageEmpty15}
\pastebutton{ugUserRulesPageFull15}{\hidepaste}
\tab{5}\spadcommand{exprule := rule exp(a + b) == exp(a) * exp(b)\bound{exprule }}
\indentrel{3}\begin{verbatim}
           b + a      a  b
   (15)  %e      == %e %e
  Type: RewriteRule(Integer,Integer,Expression Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserRulesPageEmpty15}
\begin{paste}{ugUserRulesPageEmpty15}{ugUserRulesPagePatch15}
\pastebutton{ugUserRulesPageEmpty15}{\showpaste}
\tab{5}\spadcommand{exprule := rule exp(a + b) == exp(a) * exp(b)\bound{exprule }}
\end{paste}\end{patch}

\begin{patch}{ugUserRulesPagePatch16}
\begin{paste}{ugUserRulesPageFull16}{ugUserRulesPageEmpty16}
\pastebutton{ugUserRulesPageFull16}{\hidepaste}
\tab{5}\spadcommand{exprule exp x\free{exprule }}
\indentrel{3}\begin{verbatim}
           x
   (16)  %e
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugUserRulesPageEmpty16}
\begin{paste}{ugUserRulesPageEmpty16}{ugUserRulesPagePatch16}
\pastebutton{ugUserRulesPageEmpty16}{\showpaste}
\tab{5}\spadcommand{exprule exp x\free{exprule }}
\end{paste}\end{patch}

\begin{patch}{ugUserRulesPagePatch17}
\begin{paste}{ugUserRulesPageFull17}{ugUserRulesPageEmpty17}
\pastebutton{ugUserRulesPageFull17}{\hidepaste}
\tab{5}\spadcommand{eirule := rule integral((?y + exp x)/x,x) == integral(y/x,x) + Ei x\bound{ei
\indentrel{3}\begin{verbatim}
            x   %N
            %e   + y                   y
   (17)       d%N  == 'integral(,x) + 'Ei(x)
               %N                      x
  Type: RewriteRule(Integer,Integer,Expression Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserRulesPageEmpty17}
\begin{paste}{ugUserRulesPageEmpty17}{ugUserRulesPagePatch17}
\pastebutton{ugUserRulesPageEmpty17}{\showpaste}
\tab{5}\spadcommand{eirule := rule integral((?y + exp x)/x,x) == integral(y/x,x) + Ei x\bound{ei
\end{paste}\end{patch}

\begin{patch}{ugUserRulesPagePatch18}
\begin{paste}{ugUserRulesPageFull18}{ugUserRulesPageEmpty18}
\pastebutton{ugUserRulesPageFull18}{\hidepaste}
\tab{5}\spadcommand{eirule integral(exp u/u, u)\free{eirule }}
\indentrel{3}\begin{verbatim}
   (18)  Ei(u)
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserRulesPageEmpty18}
\begin{paste}{ugUserRulesPageEmpty18}{ugUserRulesPagePatch18}
\pastebutton{ugUserRulesPageEmpty18}{\showpaste}
\tab{5}\spadcommand{eirule integral(exp u/u, u)\free{eirule }}
\end{paste}\end{patch}

\begin{patch}{ugUserRulesPagePatch19}
\begin{paste}{ugUserRulesPageFull19}{ugUserRulesPageEmpty19}
\pastebutton{ugUserRulesPageFull19}{\hidepaste}
\tab{5}\spadcommand{eirule integral(sin u + exp u/u, u)\free{eirule }}
\indentrel{3}\begin{verbatim}
            u

   (19)      sin(%N)d%N  + Ei(u)
```

```
                              Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserRulesPageEmpty19}
\begin{paste}{ugUserRulesPageEmpty19}{ugUserRulesPagePatch19}
\pastebutton{ugUserRulesPageEmpty19}{\showpaste}
\tab{5}\spadcommand{eirule integral(sin u + exp u/u, u)\free{eirule }}
\end{paste}\end{patch}

\begin{patch}{ugUserRulesPagePatch20}
\begin{paste}{ugUserRulesPageFull20}{ugUserRulesPageEmpty20}
\pastebutton{ugUserRulesPageFull20}{\hidepaste}
\tab{5}\spadcommand{u := operator 'u\bound{u }}
\indentrel{3}\begin{verbatim}
   (20)  u
                                   Type: BasicOperator
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserRulesPageEmpty20}
\begin{paste}{ugUserRulesPageEmpty20}{ugUserRulesPagePatch20}
\pastebutton{ugUserRulesPageEmpty20}{\showpaste}
\tab{5}\spadcommand{u := operator 'u\bound{u }}
\end{paste}\end{patch}

\begin{patch}{ugUserRulesPagePatch21}
\begin{paste}{ugUserRulesPageFull21}{ugUserRulesPageEmpty21}
\pastebutton{ugUserRulesPageFull21}{\hidepaste}
\tab{5}\spadcommand{v := operator 'v\bound{v }}
\indentrel{3}\begin{verbatim}
   (21)  v
                                   Type: BasicOperator
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserRulesPageEmpty21}
\begin{paste}{ugUserRulesPageEmpty21}{ugUserRulesPagePatch21}
\pastebutton{ugUserRulesPageEmpty21}{\showpaste}
\tab{5}\spadcommand{v := operator 'v\bound{v }}
\end{paste}\end{patch}

\begin{patch}{ugUserRulesPagePatch22}
\begin{paste}{ugUserRulesPageFull22}{ugUserRulesPageEmpty22}
\pastebutton{ugUserRulesPageFull22}{\hidepaste}
\tab{5}\spadcommand{myRule := rule u(x + y) == u x + v y\free{u v }\bound{m }}
\indentrel{3}\begin{verbatim}
   (22)  u(y + x) == 'v(y) + 'u(x)
  Type: RewriteRule(Integer,Integer,Expression Integer)
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserRulesPageEmpty22}
\begin{paste}{ugUserRulesPageEmpty22}{ugUserRulesPagePatch22}
\pastebutton{ugUserRulesPageEmpty22}{\showpaste}
\tab{5}\spadcommand{myRule := rule u(x + y) == u x + v y\free{u v }\bound{m }}
\end{paste}\end{patch}

\begin{patch}{ugUserRulesPagePatch23}
\begin{paste}{ugUserRulesPageFull23}{ugUserRulesPageEmpty23}
\pastebutton{ugUserRulesPageFull23}{\hidepaste}
\tab{5}\spadcommand{myRule u(a + b + c + d)\free{m }}
\indentrel{3}\begin{verbatim}
   (23)  v(d + c + b) + u(a)
                                 Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserRulesPageEmpty23}
\begin{paste}{ugUserRulesPageEmpty23}{ugUserRulesPagePatch23}
\pastebutton{ugUserRulesPageEmpty23}{\showpaste}
\tab{5}\spadcommand{myRule u(a + b + c + d)\free{m }}
\end{paste}\end{patch}

\begin{patch}{ugUserRulesPagePatch24}
\begin{paste}{ugUserRulesPageFull24}{ugUserRulesPageEmpty24}
\pastebutton{ugUserRulesPageFull24}{\hidepaste}
\tab{5}\spadcommand{myOtherRule := rule u(:x + y) == u x + v y\free{u v }\bound{m2 }}
\indentrel{3}\begin{verbatim}
   (24)  u(y + x) == 'v(y) + 'u(x)
  Type: RewriteRule(Integer,Integer,Expression Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugUserRulesPageEmpty24}
\begin{paste}{ugUserRulesPageEmpty24}{ugUserRulesPagePatch24}
\pastebutton{ugUserRulesPageEmpty24}{\showpaste}
\tab{5}\spadcommand{myOtherRule := rule u(:x + y) == u x + v y\free{u v }\bound{m2 }}
\end{paste}\end{patch}

\begin{patch}{ugUserRulesPagePatch25}
\begin{paste}{ugUserRulesPageFull25}{ugUserRulesPageEmpty25}
\pastebutton{ugUserRulesPageFull25}{\hidepaste}
\tab{5}\spadcommand{myOtherRule u(a + b + c + d)\free{m2 }}
\indentrel{3}\begin{verbatim}
   (25)  v(c) + v(b) + v(a) + u(d)
                                 Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugUserRulesPageEmpty25}
\begin{paste}{ugUserRulesPageEmpty25}{ugUserRulesPagePatch25}
\pastebutton{ugUserRulesPageEmpty25}{\showpaste}
\tab{5}\spadcommand{myOtherRule u(a + b + c + d)\free{m2 }}
\end{paste}\end{patch}
```

Chapter 11

Users Guide Chapter 7 (ug07.ht)

— **ug07.ht** —

```
\newcommand{\optArg}[1]{{{\tt [}{#1}{\tt ]}}}
\newcommand{\argDef}[1]{{\tt ({#1})}}
\newcommand{\funSyntax}[2]{\axiomFun{#1}{\tt ({\small\it{#2}})}}
\newcommand{\funArgs}[1]{{\tt ({\small\it {#1}})}\newline}
```

———-

Graphics

⇒ "notitle" (ugGraphTwoDPage) 11 on page 1968
⇒ "notitle" (ugGraphThreeDPage) 11 on page 2018

— **ug07.ht** —

```
\begin{page}{ugGraphPage}{7. Graphics}
\beginscroll

%

This chapter shows how to use the Axiom graphics facilities under the
X Window System.  Axiom has \twodim{} and \threedim{} drawing and
rendering packages that allow the drawing, coloring, transforming,
mapping, clipping, and combining of graphic output from Axiom
computations.  This facility is particularly useful for investigating
```

```
problems in areas such as topology.  The graphics package is capable
of plotting functions of one or more variables or plotting parametric
surfaces and curves.  Various coordinate systems are also available,
such as polar and spherical.

A graph is displayed in a viewport window and it has a control-panel
that uses interactive mouse commands.  PostScript and other output
forms are available so that Axiom images can be printed or used by
other programs.\footnote{PostScript is a trademark of Adobe Systems
Incorporated, registered in the United States.}

\beginmenu
    \menudownlink{{7.1. Two-Dimensional Graphics}}{ugGraphTwoDPage}
    \menudownlink{{7.2. Three-Dimensional Graphics}}{ugGraphThreeDPage}
\endmenu
\endscroll
\autobuttons
\end{page}
```

———-

Two-Dimensional Graphics

⇒ “notitle” (ugGraphTwoDPlotPage) 11 on page 1969
⇒ “notitle” (ugGraphTwoDParPage) 11 on page 1972
⇒ “notitle” (ugGraphTwoDPlanePage) 11 on page 1976
⇒ “notitle” (ugGraphTwoDOptionsPage) 11 on page 1978
⇒ “notitle” (ugGraphColorPage) 11 on page 1983
⇒ “notitle” (ugGraphColorPalettePage) 11 on page 1985
⇒ “notitle” (ugGraphTwoDControlPage) 11 on page 1988
⇒ “notitle” (ugGraphTwoDopsPage) 11 on page 1991
⇒ “notitle” (ugGraphTwoDbuildPage) 11 on page 1995
⇒ “notitle” (ugGraphTwoDappendPage) 11 on page 2015

— **ug07.ht** —

```
\begin{page}{ugGraphTwoDPage}{7.1. Two-Dimensional Graphics}
\beginscroll
%
The Axiom \twodim{} graphics package provides the ability to
display
%
\indent{4}
\beginitems
%
\item[-] curves defined by functions of a single real variable
%
```

```
\item[-] curves defined by parametric equations
%
\item[-] implicit non-singular curves defined by polynomial equations
%
\item[-] planar graphs generated from lists of point components.
\enditems
\indent{0}
These graphs
can be modified by specifying various options, such as
calculating points in the polar
coordinate system or changing the size of the graph viewport window.

\beginmenu
\menudownlink{{7.1.1. Plotting Two-Dimensional Functions of One Variable}}
{ugGraphTwoDPlotPage}
\menudownlink{{7.1.2. Plotting Two-Dimensional Parametric Plane Curves}}
{ugGraphTwoDParPage}
\menudownlink{{7.1.3. Plotting Plane Algebraic Curves}}
{ugGraphTwoDPlanePage}
\menudownlink{{7.1.4. Two-Dimensional Options}}{ugGraphTwoDOptionsPage}
\menudownlink{{7.1.5. Color}}{ugGraphColorPage}
\menudownlink{{7.1.6. Palette}}{ugGraphColorPalettePage}
\menudownlink{{7.1.7. Two-Dimensional Control-Panel}}
{ugGraphTwoDControlPage}
\menudownlink{{7.1.8. Operations for Two-Dimensional Graphics}}
{ugGraphTwoDopsPage}
\menudownlink{{7.1.9. Addendum: Building Two-Dimensional Graphs}}
{ugGraphTwoDbuildPage}
\menudownlink{
{7.1.10. Addendum: Appending a Graph to a Viewport Window
Containing a Graph}}{ugGraphTwoDappendPage}
\endmenu
\endscroll
\autobuttons
\end{page}
```

———-

Plotting Two-Dimensional Functions of One Variable

⇒ "notitle" (ugGraphTwoDOptionsPage) 11 on page 1978

— **ug07.ht** —

```
\begin{page}{ugGraphTwoDPlotPage}
{7.1.1. Plotting Two-Dimensional Functions of One Variable}
\beginscroll
```

```
The first kind of \twodim{} graph is that of a curve defined by a
function \axiom{y = f(x)} over a finite interval of the \axiom{x}
axis.

%
\beginImportant
The general format for drawing a function defined by a formula
\axiom{f(x)} is:
%
\centerline{{{\tt draw(f(x), x = a..b, {\it options})}}}
where \axiom{a..b} defines the range of \axiom{x}, and where
{\it options} prescribes zero or more options as described in
\downlink{``Two-Dimensional Options''}{ugGraphTwoDOptionsPage}
in Section 7.1.4\ignore{ugGraphTwoDOptions}.
An example of an option is \axiom{curveColor == bright red().}
An alternative format involving functions \axiom{f} and \axiom{g}
is also available.
\endImportant

A simple way to plot a function is to use a formula.  The first
argument is the formula.  For the second argument, write the name of
the independent variable (here, \axiom{x}), followed by an
\spadSyntax{=}, and the range of values.

\psXtc{
Display this formula over the range
\texht{$0 \leq x \leq 6$}{0 <= x <= 6}.
Axiom converts your formula to a compiled
function so that the results can be computed
quickly and efficiently.
}{
\graphpaste{draw(sin(tan(x)) - tan(sin(x)),x = 0..6)}
}{
\epsffile[0 0 295 295]{../ps/2d1vara.ps}
}

Notice that Axiom compiled the function before the graph was put
on the screen.

\psXtc{
Here is the same graph on a different interval.
This time we give the graph a title.
}{
\graphpaste{draw(sin(tan(x)) - tan(sin(x)),x = 10..16)}
}{
%window was 300 x 300
\epsffile[0 0 295 295]{../ps/2d1varb.ps}
}
%
Once again the formula is converted to a compiled function before
```

```
any points were computed.
If you want to graph the same function on several intervals, it is
a good idea to define the function first so that the function has
to be compiled only once.
\xtc{
This time we first define the function.
}{
\spadpaste{f(x) == (x-1)*(x-2)*(x-3) \bound{f}}
}
\psXtc{
To draw the function, the first argument is its name
and the second is just the range with no independent variable.
}{
\graphpaste{draw(f, 0..4) \free{f}}
}{
\epsffile[0 0 295 295]{../ps/2d1vard.ps}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugGraphTwoDPlotPagePatch1}
\begin{paste}{ugGraphTwoDPlotPageFull1}{ugGraphTwoDPlotPageEmpty1}
\pastebutton{ugGraphTwoDPlotPageFull1}{\hidepaste}
\tab{5}\spadgraph{draw(sin(tan(x)) - tan(sin(x)),x = 0..6)}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphtwodplotpage1.view/image}}{viewalone\space
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDPlotPageEmpty1}
\begin{paste}{ugGraphTwoDPlotPageEmpty1}{ugGraphTwoDPlotPagePatch1}
\pastebutton{ugGraphTwoDPlotPageEmpty1}{\showpaste}
\tab{5}\spadgraph{draw(sin(tan(x)) - tan(sin(x)),x = 0..6)}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDPlotPagePatch2}
\begin{paste}{ugGraphTwoDPlotPageFull2}{ugGraphTwoDPlotPageEmpty2}
\pastebutton{ugGraphTwoDPlotPageFull2}{\hidepaste}
\tab{5}\spadgraph{draw(sin(tan(x)) - tan(sin(x)),x = 10..16)}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphtwodplotpage2.view/image}}{viewalone\space
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDPlotPageEmpty2}
\begin{paste}{ugGraphTwoDPlotPageEmpty2}{ugGraphTwoDPlotPagePatch2}
\pastebutton{ugGraphTwoDPlotPageEmpty2}{\showpaste}
\tab{5}\spadgraph{draw(sin(tan(x)) - tan(sin(x)),x = 10..16)}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDPlotPagePatch3}
\begin{paste}{ugGraphTwoDPlotPageFull3}{ugGraphTwoDPlotPageEmpty3}
```

```
\pastebutton{ugGraphTwoDPlotPageFull3}{\hidepaste}
\tab{5}\spadcommand{f(x) == (x-1)*(x-2)*(x-3)\bound{f }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDPlotPageEmpty3}
\begin{paste}{ugGraphTwoDPlotPageEmpty3}{ugGraphTwoDPlotPagePatch3}
\pastebutton{ugGraphTwoDPlotPageEmpty3}{\showpaste}
\tab{5}\spadcommand{f(x) == (x-1)*(x-2)*(x-3)\bound{f }}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDPlotPagePatch4}
\begin{paste}{ugGraphTwoDPlotPageFull4}{ugGraphTwoDPlotPageEmpty4}
\pastebutton{ugGraphTwoDPlotPageFull4}{\hidepaste}
\tab{5}\spadgraph{draw(f, 0..4)\free{f }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphtwodplotpage4.view/image}}{vie
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDPlotPageEmpty4}
\begin{paste}{ugGraphTwoDPlotPageEmpty4}{ugGraphTwoDPlotPagePatch4}
\pastebutton{ugGraphTwoDPlotPageEmpty4}{\showpaste}
\tab{5}\spadgraph{draw(f, 0..4)\free{f }}
\end{paste}\end{patch}
```

———-

Plotting 2D Parametric Plane Curves

⇒ “notitle” (ugGraphThreeDOptionsPage) 11 on page 2028

— **ug07.ht** —

```
\begin{page}{ugGraphTwoDParPage}
{7.1.2. Plotting 2D Parametric Plane Curves}
\beginscroll

The second kind of \twodim{} graph is that of
curves produced by parametric equations.
Let \axiom{x = f(t)} and \axiom{y = g(t)} be formulas or two
functions \axiom{f} and \axiom{g} as the parameter \axiom{t} ranges
over an interval \axiom{[a,b]}.
The function \axiomFun{curve} takes the two functions \axiom{f} and
\axiom{g} as its parameters.

\beginImportant
The general format for drawing a \twodim{} plane curve defined by
```

```
parametric formulas \axiom{x = f(t)} and \axiom{y = g(t)} is:
%
\centerline{{{\tt draw(curve(f(t), g(t)), t = a..b, {\it options})}}}
where \axiom{a..b} defines the range of the independent variable \axiom{t},
and where {\it options} prescribes zero or more options as
described in
\downlink{``Three-Dimensional Options''}{ugGraphThreeDOptionsPage}
in Section 7.2.4\ignore{ugGraphThreeDOptions}.
An example of an option is \axiom{curveColor == bright red().}
\endImportant

Here's an example:

\psXtc{
Define a parametric curve using a range involving
\axiom{\%pi}, Axiom's way of saying \texht{$\pi$}{``pi''}.
For parametric curves, Axiom compiles two
functions, one for each of the functions \axiom{f} and \axiom{g}.
}{
\graphpaste{draw(curve(sin(t)*sin(2*t)*sin(3*t),
sin(4*t)*sin(5*t)*sin(6*t)), t = 0..2*\%pi)}
}{
\epsffile[0 0 295 295]{../ps/2dppca.ps}
}
%
%
\psXtc{
The title may be an arbitrary string and is an
optional argument to the \axiomFun{draw} command.
}{
\graphpaste{draw(curve(cos(t), sin(t)), t = 0..2*\%pi)}
}{
\epsffile[0 0 295 295]{../ps/2dppcb.ps}
}
%
If you plan on plotting \axiom{x = f(t)}, \axiom{y = g(t)} as \axiom{t}
ranges over several intervals, you may want to define functions
\axiom{f} and \axiom{g} first, so
that they need not be recompiled every time you create a new graph.
Here's an example:
\xtc{
As before, you can first define the functions you wish to draw.
}{
\spadpaste{f(t:DFLOAT):DFLOAT == sin(3*t/4) \bound{f}}
}
\xtc{
Axiom compiles them to map \axiomType{DoubleFloat}
values to \axiomType{DoubleFloat} values.
}{
\spadpaste{g(t:DFLOAT):DFLOAT == sin(t) \bound{g}}
```

```
}

\psXtc{
Give to {\tt curve} the names of the functions,
then write the range without the name of the
independent variable.
}{
\graphpaste{draw(curve(f,g),0..\%pi) \free{f g}}
}{
\epsffile[0 0 295 295]{../ps/2dppcc.ps}
}
%
%
\psXtc{
Here is another look at the same curve but over a different
range. Notice that \axiom{f} and \axiom{g} are not recompiled.
Also note that Axiom provides a default title based on
the first function specified in \axiomFun{curve}.
}{
\graphpaste{draw(curve(f,g),-4*\%pi..4*\%pi) \free{f g}}
}{
\epsffile[0 0 295 295]{../ps/2dppce.ps}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugGraphTwoDParPagePatch1}
\begin{paste}{ugGraphTwoDParPageFull1}{ugGraphTwoDParPageEmpty1}
\pastebutton{ugGraphTwoDParPageFull1}{\hidepaste}
\tab{5}\spadgraph{draw(curve(sin(t)*sin(2*t)*sin(3*t), sin(4*t)*sin(5*t)*sin(6*t)), t = 0..2*\%p
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphtwodparpage1.view/image}}{view
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDParPageEmpty1}
\begin{paste}{ugGraphTwoDParPageEmpty1}{ugGraphTwoDParPagePatch1}
\pastebutton{ugGraphTwoDParPageEmpty1}{\showpaste}
\tab{5}\spadgraph{draw(curve(sin(t)*sin(2*t)*sin(3*t), sin(4*t)*sin(5*t)*sin(6*t)), t = 0..2*\%p
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDParPagePatch2}
\begin{paste}{ugGraphTwoDParPageFull2}{ugGraphTwoDParPageEmpty2}
\pastebutton{ugGraphTwoDParPageFull2}{\hidepaste}
\tab{5}\spadgraph{draw(curve(cos(t), sin(t)), t = 0..2*\%pi)}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphtwodparpage2.view/image}}{view
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDParPageEmpty2}
\begin{paste}{ugGraphTwoDParPageEmpty2}{ugGraphTwoDParPagePatch2}
```

```
\pastebutton{ugGraphTwoDParPageEmpty2}{\showpaste}
\tab{5}\spadgraph{draw(curve(cos(t), sin(t)), t = 0..2*\%pi)}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDParPagePatch3}
\begin{paste}{ugGraphTwoDParPageFull3}{ugGraphTwoDParPageEmpty3}
\pastebutton{ugGraphTwoDParPageFull3}{\hidepaste}
\tab{5}\spadcommand{f(t:DFLOAT):DFLOAT == sin(3*t/4)\bound{f }}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDParPageEmpty3}
\begin{paste}{ugGraphTwoDParPageEmpty3}{ugGraphTwoDParPagePatch3}
\pastebutton{ugGraphTwoDParPageEmpty3}{\showpaste}
\tab{5}\spadcommand{f(t:DFLOAT):DFLOAT == sin(3*t/4)\bound{f }}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDParPagePatch4}
\begin{paste}{ugGraphTwoDParPageFull4}{ugGraphTwoDParPageEmpty4}
\pastebutton{ugGraphTwoDParPageFull4}{\hidepaste}
\tab{5}\spadcommand{g(t:DFLOAT):DFLOAT == sin(t)\bound{g }}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDParPageEmpty4}
\begin{paste}{ugGraphTwoDParPageEmpty4}{ugGraphTwoDParPagePatch4}
\pastebutton{ugGraphTwoDParPageEmpty4}{\showpaste}
\tab{5}\spadcommand{g(t:DFLOAT):DFLOAT == sin(t)\bound{g }}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDParPagePatch5}
\begin{paste}{ugGraphTwoDParPageFull5}{ugGraphTwoDParPageEmpty5}
\pastebutton{ugGraphTwoDParPageFull5}{\hidepaste}
\tab{5}\spadgraph{draw(curve(f,g),0..\%pi)\free{f g }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphtwodparpage5.view/image}}{viewalone\space
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDParPageEmpty5}
\begin{paste}{ugGraphTwoDParPageEmpty5}{ugGraphTwoDParPagePatch5}
\pastebutton{ugGraphTwoDParPageEmpty5}{\showpaste}
\tab{5}\spadgraph{draw(curve(f,g),0..\%pi)\free{f g }}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDParPagePatch6}
\begin{paste}{ugGraphTwoDParPageFull6}{ugGraphTwoDParPageEmpty6}
\pastebutton{ugGraphTwoDParPageFull6}{\hidepaste}
```

```
\tab{5}\spadgraph{draw(curve(f,g),-4*\%pi..4*\%pi)\free{f g }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphtwodparpage6.view/image}}{view
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDParPageEmpty6}
\begin{paste}{ugGraphTwoDParPageEmpty6}{ugGraphTwoDParPagePatch6}
\pastebutton{ugGraphTwoDParPageEmpty6}{\showpaste}
\tab{5}\spadgraph{draw(curve(f,g),-4*\%pi..4*\%pi)\free{f g }}
\end{paste}\end{patch}
```

——–

Plotting Plane Algebraic Curves

⇒ "notitle" (ugGraphTwoDOptionsPage) 11 on page 1978

— **ug07.ht** —

```
\begin{page}{ugGraphTwoDPlanePage}{7.1.3. Plotting Plane Algebraic Curves}
\beginscroll

A third kind of \twodim{} graph is a non-singular ‘‘solution curve’’
in a rectangular region of the plane.
A solution curve is a curve defined by a polynomial equation
\axiom{p(x,y) = 0}.
Non-singular means that the curve is ‘‘smooth’’ in that it does not
cross itself or come to a point (cusp).
Algebraically, this means that for any point \axiom{(x,y)} on the curve,
that is, a point such that \axiom{p(x,y) = 0}, the partial derivatives
\texht{${{\partial p}\over{\partial x}}(x,y)$ and
${{\partial p}\over{\partial y}}(x,y)$}{\axiom{dp/dx(x,y)}
and \axiom{dp/dy(a,b)}}
are not both zero.

%
\beginImportant
The general format for drawing a non-singular solution curve
given by a polynomial of the form \axiom{p(x,y) = 0} is:
%
\centerline{{{\tt draw(p(x,y) = 0, x, y, range == [a..b, c..d],
{\it options})}}}
where the second and third arguments name the first and second
independent variables of \axiom{p}.
A {\tt range} option is always given to designate a bounding
rectangular region of the plane \texht{$a \leq x \leq b, c \leq y
\leq d$}{a <= x <= b, c <= y <= d}.
Zero or more additional options as described in
\downlink{‘‘Two-Dimensional Options’’}{ugGraphTwoDOptionsPage}
```

```
in Section 7.1.4\ignore{ugGraphTwoDOptions}
may be given.
\endImportant

\xtc{
We require that the polynomial has rational or integral coefficients.
Here is an algebraic curve example (``Cartesian ovals''):
}{
\spadpaste{p := ((x**2 + y**2 + 1) - 8*x)**2 -
(8*(x**2 + y**2 + 1)-4*x-1) \bound{p}}
}

\psXtc{
The first argument is always expressed as an equation of the form
\axiom{p = 0} where \axiom{p} is a polynomial.
}{
\graphpaste{draw(p = 0, x, y, range == [-1..11, -7..7]) \free{p}}
}{
\epsffile[0 0 295 295]{../ps/2dpaca.ps}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugGraphTwoDPlanePagePatch1}
\begin{paste}{ugGraphTwoDPlanePageFull1}{ugGraphTwoDPlanePageEmpty1}
\pastebutton{ugGraphTwoDPlanePageFull1}{\hidepaste}
\tab{5}\spadcommand{p := ((x**2 + y**2 + 1) - 8*x)**2 - (8*(x**2 + y**2 + 1)-4*x-1)\bound{p }}
\indentrel{3}\begin{verbatim}
   (1)
    4      2             2    4      3      2
   y  + (2x  - 16x - 6)y  + x  - 16x  + 58x  - 12x - 6
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDPlanePageEmpty1}
\begin{paste}{ugGraphTwoDPlanePageEmpty1}{ugGraphTwoDPlanePagePatch1}
\pastebutton{ugGraphTwoDPlanePageEmpty1}{\showpaste}
\tab{5}\spadcommand{p := ((x**2 + y**2 + 1) - 8*x)**2 - (8*(x**2 + y**2 + 1)-4*x-1)\bound{p }}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDPlanePagePatch2}
\begin{paste}{ugGraphTwoDPlanePageFull2}{ugGraphTwoDPlanePageEmpty2}
\pastebutton{ugGraphTwoDPlanePageFull2}{\hidepaste}
\tab{5}\spadgraph{draw(p = 0, x, y, range == [-1..11, -7..7])\free{p }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphtwodplanepage2.view/image}}{viewalone\spa
\end{paste}\end{patch}
```

```
\begin{patch}{ugGraphTwoDPlanePageEmpty2}
\begin{paste}{ugGraphTwoDPlanePageEmpty2}{ugGraphTwoDPlanePagePatch2}
\pastebutton{ugGraphTwoDPlanePageEmpty2}{\showpaste}
\tab{5}\spadgraph{draw(p = 0, x, y, range == [-1..11, -7..7])\free{p }}
\end{paste}\end{patch}
```

———-

Two-Dimensional Options

⇒ "notitle" (ugGraphColorPage) 11 on page 1983
⇒ "notitle" (ugGraphColorPalettePage) 11 on page 1985
⇒ "notitle" (ugGraphColorPage) 11 on page 1983
⇒ "notitle" (ugGraphColorPalettePage) 11 on page 1985

— **ug07.ht** —

```
\begin{page}{ugGraphTwoDOptionsPage}{7.1.4. Two-Dimensional Options}
\beginscroll

The \axiomFun{draw} commands take an optional list of options,
such as {\tt title} shown above.
Each option is given by the syntax: {\it name} {\tt ==} {\it value}.
Here is a list of the available options in the order that they are
described below.

\table{ {adaptive} {clip} {unit} {clip} {curveColor} {range}
{toScale} {pointColor} {coordinates}}

The \axiom{adaptive} option turns adaptive plotting on or off.
Adaptive plotting uses an algorithm that traverses a graph and computes
more points for those parts of the graph with high curvature.
The higher the curvature of a region is, the more points the algorithm
computes.
%
%
\psXtc{
The {\tt adaptive} option is normally on.
Here we turn it off.
}{
\graphpaste{draw(sin(1/x),x=-2*\%pi..2*\%pi, adaptive == false)}
}{
\epsffile[0 0 295 295]{../ps/2doptad.ps}
}
%
%
\psXtc{
```

```
The {\tt clip} option turns clipping on or off.
If on, large values are cut off according to
\axiomFunFrom{clipPointsDefault}{GraphicsDefaults}.
}{
\graphpaste{draw(tan(x),x=-2*\%pi..2*\%pi, clip == true)}
}{
\epsffile[0 0 295 295]{../ps/2doptcp.ps}
}
%
%
\psXtc{
Option {\tt toScale} does plotting to scale if {\tt true} or uses
the entire viewport if {\tt false}.
The default can be determined using
\axiomFunFrom{drawToScale}{GraphicsDefaults}.
}{
\graphpaste{draw(sin(x),x=-\%pi..\%pi, toScale == true, unit == [1.0,1.0])}
}{
\epsffile[0 0 295 295]{../ps/2doptsc.ps}
}
%
%
\psXtc{
Option {\tt clip} with a range sets point clipping of a graph within the
ranges specified in the list \axiom{[x range,y range]}.
If only one range is specified, clipping applies to the y-axis.
}{
\graphpaste{draw(sec(x),x=-2*\%pi..2*\%pi,
clip == [-2*\%pi..2*\%pi,-\%pi..\%pi], unit == [1.0,1.0])}
}{
\epsffile[0 0 295 295]{../ps/2doptcpr.ps}
}
%
\psXtc{
Option {\tt curveColor} sets the color of the graph curves or
lines to be the indicated palette color
(see \downlink{``Color''}{ugGraphColorPage} in Section
7.1.5\ignore{ugGraphColor} and
\downlink{``Palette''}{ugGraphColorPalettePage}
in Section 7.1.6\ignore{ugGraphColorPalette}).
}{
\graphpaste{draw(sin(x),x=-\%pi..\%pi, curveColor == bright red())}
}{
\epsffile[0 0 295 295]{../ps/2doptcvc.ps}
}
%
\psXtc{
Option {\tt pointColor}
sets the color of the graph points to the indicated
palette color
```

```
(see \downlink{‘‘Color’’}{ugGraphColorPage}
in Section 7.1.5\ignore{ugGraphColor} and
\downlink{‘‘Palette’’}{ugGraphColorPalettePage}
in Section 7.1.6\ignore{ugGraphColorPalette}).
}{
\graphpaste{draw(sin(x),x=-\%pi..\%pi, pointColor == pastel yellow())}
}{
\epsffile[0 0 295 295]{../ps/2doptptc.ps}
}
%
\psXtc{
Option {\tt unit} sets the intervals at which the axis units are plotted
according to the indicated steps [\axiom{x} interval, \axiom{y} interval].
}{
\graphpaste{draw(curve(9*sin(3*t/4),8*sin(t)),
t = -4*\%pi..4*\%pi, unit == [2.0,1.0])}
}{
\epsffile[0 0 295 295]{../ps/2doptut.ps}
}
%
%
\psXtc{
Option {\tt range} sets the range of variables in a graph to be
within the ranges
for solving plane algebraic curve plots.
}{
\graphpaste{draw(y**2 + y - (x**3 - x) = 0, x, y,
range == [-2..2,-2..1], unit==[1.0,1.0])}
}{
\epsffile[0 0 295 295]{../ps/2doptrga.ps}
}
%
%
\psXtc{
A second example of a solution plot.
}{
\graphpaste{draw(x**2 + y**2 = 1, x, y,
range == [-3/2..3/2,-3/2..3/2], unit==[0.5,0.5])}
}{
\epsffile[0 0 295 295]{../ps/2doptrgb.ps}
}
%
%
\psXtc{
Option \axiom{coordinates} indicates the coordinate system
in which the graph
is plotted.
The default is to use the Cartesian coordinate system.
For more details, see
\downlink{‘‘Coordinate System Transformations’’}{ugGraphCoordPage} in
```

```
Section 7.2.7\ignore{ugGraphCoord} \texht{.}{or
\axiomType{CoordinateSystems}.}
}{
\graphpaste{draw(curve(sin(5*t),t),t=0..2*\%pi, coordinates == polar)}
}{
\epsffile[0 0 295 295]{../ps/2doptplr.ps}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugGraphTwoDOptionsPagePatch1}
\begin{paste}{ugGraphTwoDOptionsPageFull1}{ugGraphTwoDOptionsPageEmpty1}
\pastebutton{ugGraphTwoDOptionsPageFull1}{\hidepaste}
\tab{5}\spadgraph{draw(sin(1/x),x=-2*\%pi..2*\%pi, adaptive == false)}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphtwodoptionspage1.view/image}}{viewalone\sp
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDOptionsPageEmpty1}
\begin{paste}{ugGraphTwoDOptionsPageEmpty1}{ugGraphTwoDOptionsPagePatch1}
\pastebutton{ugGraphTwoDOptionsPageEmpty1}{\showpaste}
\tab{5}\spadgraph{draw(sin(1/x),x=-2*\%pi..2*\%pi, adaptive == false)}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDOptionsPagePatch2}
\begin{paste}{ugGraphTwoDOptionsPageFull2}{ugGraphTwoDOptionsPageEmpty2}
\pastebutton{ugGraphTwoDOptionsPageFull2}{\hidepaste}
\tab{5}\spadgraph{draw(tan(x),x=-2*\%pi..2*\%pi, clip == true)}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphtwodoptionspage2.view/image}}{viewalone\sp
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDOptionsPageEmpty2}
\begin{paste}{ugGraphTwoDOptionsPageEmpty2}{ugGraphTwoDOptionsPagePatch2}
\pastebutton{ugGraphTwoDOptionsPageEmpty2}{\showpaste}
\tab{5}\spadgraph{draw(tan(x),x=-2*\%pi..2*\%pi, clip == true)}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDOptionsPagePatch3}
\begin{paste}{ugGraphTwoDOptionsPageFull3}{ugGraphTwoDOptionsPageEmpty3}
\pastebutton{ugGraphTwoDOptionsPageFull3}{\hidepaste}
\tab{5}\spadgraph{draw(sin(x),x=-\%pi..\%pi, toScale == true, unit == [1.0,1.0])}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphtwodoptionspage3.view/image}}{viewalone\sp
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDOptionsPageEmpty3}
\begin{paste}{ugGraphTwoDOptionsPageEmpty3}{ugGraphTwoDOptionsPagePatch3}
\pastebutton{ugGraphTwoDOptionsPageEmpty3}{\showpaste}
\tab{5}\spadgraph{draw(sin(x),x=-\%pi..\%pi, toScale == true, unit == [1.0,1.0])}
\end{paste}\end{patch}
```

```
\begin{patch}{ugGraphTwoDOptionsPagePatch4}
\begin{paste}{ugGraphTwoDOptionsPageFull4}{ugGraphTwoDOptionsPageEmpty4}
\pastebutton{ugGraphTwoDOptionsPageFull4}{\hidepaste}
\tab{5}\spadgraph{draw(sec(x),x=-2*\%pi..2*\%pi, clip == [-2*\%pi..2*\%pi,-\%pi..\%pi], unit ==
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphtwodoptionspage4.view/image}}{
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDOptionsPageEmpty4}
\begin{paste}{ugGraphTwoDOptionsPageEmpty4}{ugGraphTwoDOptionsPagePatch4}
\pastebutton{ugGraphTwoDOptionsPageEmpty4}{\showpaste}
\tab{5}\spadgraph{draw(sec(x),x=-2*\%pi..2*\%pi, clip == [-2*\%pi..2*\%pi,-\%pi..\%pi], unit ==
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDOptionsPagePatch5}
\begin{paste}{ugGraphTwoDOptionsPageFull5}{ugGraphTwoDOptionsPageEmpty5}
\pastebutton{ugGraphTwoDOptionsPageFull5}{\hidepaste}
\tab{5}\spadgraph{draw(sin(x),x=-\%pi..\%pi, curveColor == bright red())}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphtwodoptionspage5.view/image}}{
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDOptionsPageEmpty5}
\begin{paste}{ugGraphTwoDOptionsPageEmpty5}{ugGraphTwoDOptionsPagePatch5}
\pastebutton{ugGraphTwoDOptionsPageEmpty5}{\showpaste}
\tab{5}\spadgraph{draw(sin(x),x=-\%pi..\%pi, curveColor == bright red())}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDOptionsPagePatch6}
\begin{paste}{ugGraphTwoDOptionsPageFull6}{ugGraphTwoDOptionsPageEmpty6}
\pastebutton{ugGraphTwoDOptionsPageFull6}{\hidepaste}
\tab{5}\spadgraph{draw(sin(x),x=-\%pi..\%pi, pointColor == pastel yellow())}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphtwodoptionspage6.view/image}}{
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDOptionsPageEmpty6}
\begin{paste}{ugGraphTwoDOptionsPageEmpty6}{ugGraphTwoDOptionsPagePatch6}
\pastebutton{ugGraphTwoDOptionsPageEmpty6}{\showpaste}
\tab{5}\spadgraph{draw(sin(x),x=-\%pi..\%pi, pointColor == pastel yellow())}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDOptionsPagePatch7}
\begin{paste}{ugGraphTwoDOptionsPageFull7}{ugGraphTwoDOptionsPageEmpty7}
\pastebutton{ugGraphTwoDOptionsPageFull7}{\hidepaste}
\tab{5}\spadgraph{draw(curve(9*sin(3*t/4),8*sin(t)), t = -4*\%pi..4*\%pi, unit == [2.0,1.0])}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphtwodoptionspage7.view/image}}{
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDOptionsPageEmpty7}
\begin{paste}{ugGraphTwoDOptionsPageEmpty7}{ugGraphTwoDOptionsPagePatch7}
\pastebutton{ugGraphTwoDOptionsPageEmpty7}{\showpaste}
```

```
\tab{5}\spadgraph{draw(curve(9*sin(3*t/4),8*sin(t)), t = -4*\%pi..4*\%pi, unit == [2.0,1.0])}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDOptionsPagePatch8}
\begin{paste}{ugGraphTwoDOptionsPageFull8}{ugGraphTwoDOptionsPageEmpty8}
\pastebutton{ugGraphTwoDOptionsPageFull8}{\hidepaste}
\tab{5}\spadgraph{draw(y**2 + y - (x**3 - x) = 0, x, y, range == [-2..2,-2..1], unit==[1.0,1.0])}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphtwodoptionspage8.view/image}}{viewalone\sp
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDOptionsPageEmpty8}
\begin{paste}{ugGraphTwoDOptionsPageEmpty8}{ugGraphTwoDOptionsPagePatch8}
\pastebutton{ugGraphTwoDOptionsPageEmpty8}{\showpaste}
\tab{5}\spadgraph{draw(y**2 + y - (x**3 - x) = 0, x, y, range == [-2..2,-2..1], unit==[1.0,1.0])}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDOptionsPagePatch9}
\begin{paste}{ugGraphTwoDOptionsPageFull9}{ugGraphTwoDOptionsPageEmpty9}
\pastebutton{ugGraphTwoDOptionsPageFull9}{\hidepaste}
\tab{5}\spadgraph{draw(x**2 + y**2 = 1, x, y, range == [-3/2..3/2,-3/2..3/2], unit==[0.5,0.5])}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphtwodoptionspage9.view/image}}{viewalone\sp
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDOptionsPageEmpty9}
\begin{paste}{ugGraphTwoDOptionsPageEmpty9}{ugGraphTwoDOptionsPagePatch9}
\pastebutton{ugGraphTwoDOptionsPageEmpty9}{\showpaste}
\tab{5}\spadgraph{draw(x**2 + y**2 = 1, x, y, range == [-3/2..3/2,-3/2..3/2], unit==[0.5,0.5])}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDOptionsPagePatch10}
\begin{paste}{ugGraphTwoDOptionsPageFull10}{ugGraphTwoDOptionsPageEmpty10}
\pastebutton{ugGraphTwoDOptionsPageFull10}{\hidepaste}
\tab{5}\spadgraph{draw(curve(sin(5*t),t),t=0..2*\%pi, coordinates == polar)}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphtwodoptionspage10.view/image}}{viewalone\s
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDOptionsPageEmpty10}
\begin{paste}{ugGraphTwoDOptionsPageEmpty10}{ugGraphTwoDOptionsPagePatch10}
\pastebutton{ugGraphTwoDOptionsPageEmpty10}{\showpaste}
\tab{5}\spadgraph{draw(curve(sin(5*t),t),t=0..2*\%pi, coordinates == polar)}
\end{paste}\end{patch}
```

——-

Color

— **ug07.ht** —

```
\begin{page}{ugGraphColorPage}{7.1.5. Color}
\beginscroll

The domain \axiomType{Color} provides operations for manipulating
colors in \twodim{} graphs.  Colors are objects of \axiomType{Color}.
Each color has a {\it hue} and a {\it weight}.  Hues are represented
by integers that range from \axiom{1} to the
\axiomFunFrom{numberOfHues()}{Color}, normally \axiom{27}.  Weights
are floats and have the value \axiom{1.0} by default.

\indent{0}
\beginitems
%
\item[\axiomFun{color}]\funArgs{integer}
creates a color of hue {\it integer} and weight \axiom{1.0}.
%
\item[\axiomFun{hue}]\funArgs{color}
returns the hue of {\it color} as an integer.
%
\item[\axiomFun{red}]\funArgs{},
\funSyntax{blue}{},
\funSyntax{green}{}, and \funSyntax{yellow}{}
create colors of that hue with weight \axiom{1.0}.
%
\item[\subscriptIt{color}{1} {\tt +} \subscriptIt{color}{2}] returns the
color that results from additively combining the indicated
\subscriptIt{color}{1} and \subscriptIt{color}{2}.
Color addition is not commutative: changing the order of the arguments
produces different results.
%
\item[{\it integer} {\tt *} {\it color}]
changes the weight of {\it color} by {\it integer}
without affecting its hue.
For example,
\axiom{red() + 3*yellow()} produces a color closer to yellow than to red.
Color multiplication is not associative: changing the order of grouping
produces different results.
\enditems
\indent{0}
%
\psXtc{
These functions can be used to change the point and curve colors
for two- and \threedim{} graphs.
Use the {\tt pointColor} option for points.
}{
\graphpaste{draw(x**2,x=-1..1,pointColor == green())}
}{
\epsffile[0 0 295 295]{../ps/23dcola.ps}
}
%
```

```
\psXtc{
Use the {\tt curveColor} option for curves.
}{
\graphpaste{draw(x**2,x=-1..1,curveColor == color(13) + 2*blue())}
}{
\epsffile[0 0 295 295]{../ps/23dcolb.ps}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugGraphColorPagePatch1}
\begin{paste}{ugGraphColorPageFull1}{ugGraphColorPageEmpty1}
\pastebutton{ugGraphColorPageFull1}{\hidepaste}
\tab{5}\spadgraph{draw(x**2,x=-1..1,pointColor == green())}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphcolorpage1.view/image}}{viewalone\space{1
\end{paste}\end{patch}

\begin{patch}{ugGraphColorPageEmpty1}
\begin{paste}{ugGraphColorPageEmpty1}{ugGraphColorPagePatch1}
\pastebutton{ugGraphColorPageEmpty1}{\showpaste}
\tab{5}\spadgraph{draw(x**2,x=-1..1,pointColor == green())}
\end{paste}\end{patch}

\begin{patch}{ugGraphColorPagePatch2}
\begin{paste}{ugGraphColorPageFull2}{ugGraphColorPageEmpty2}
\pastebutton{ugGraphColorPageFull2}{\hidepaste}
\tab{5}\spadgraph{draw(x**2,x=-1..1,curveColor == color(13) + 2*blue())}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphcolorpage2.view/image}}{viewalone\space{1
\end{paste}\end{patch}

\begin{patch}{ugGraphColorPageEmpty2}
\begin{paste}{ugGraphColorPageEmpty2}{ugGraphColorPagePatch2}
\pastebutton{ugGraphColorPageEmpty2}{\showpaste}
\tab{5}\spadgraph{draw(x**2,x=-1..1,curveColor == color(13) + 2*blue())}
\end{paste}\end{patch}
```

———-

Palette

— ug07.ht —

```
\begin{page}{ugGraphColorPalettePage}{7.1.6. Palette}
\beginscroll
```

```
Domain \axiomType{Palette} is the domain of shades of colors:
\axiomFun{dark}, \axiomFun{dim}, \axiomFun{bright},
\axiomFun{pastel}, and \axiomFun{light},
designated by the integers \axiom{1} through \axiom{5}, respectively.
\xtc{
Colors are normally ``bright.''
}{
\spadpaste{shade red()}
}
\xtc{
To change the shade of a color, apply the name of a shade to it.
}{
\spadpaste{myFavoriteColor := dark blue() \bound{mfc}}
}
\xtc{
The expression \axiom{shade(color)}
returns the value of a shade of \axiom{color}.
}{
\spadpaste{shade myFavoriteColor \free{mfc}}
}
\xtc{
The expression \axiom{hue(color)} returns its hue.
}{
\spadpaste{hue myFavoriteColor \free{mfc}}
}
\psXtc{
Palettes can be used in specifying colors in \twodim{} graphs.
}{
\graphpaste{draw(x**2,x=-1..1,curveColor == dark blue())}
}{
\epsffile[0 0 295 295]{../ps/23dpal.ps}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugGraphColorPalettePagePatch1}
\begin{paste}{ugGraphColorPalettePageFull1}{ugGraphColorPalettePageEmpty1}
\pastebutton{ugGraphColorPalettePageFull1}{\hidepaste}
\tab{5}\spadcommand{shade red()}
\indentrel{3}\begin{verbatim}
   (1)  3
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphColorPalettePageEmpty1}
\begin{paste}{ugGraphColorPalettePageEmpty1}{ugGraphColorPalettePagePatch1}
\pastebutton{ugGraphColorPalettePageEmpty1}{\showpaste}
```

```
\tab{5}\spadcommand{shade red()}
\end{paste}\end{patch}

\begin{patch}{ugGraphColorPalettePagePatch2}
\begin{paste}{ugGraphColorPalettePageFull2}{ugGraphColorPalettePageEmpty2}
\pastebutton{ugGraphColorPalettePageFull2}{\hidepaste}
\tab{5}\spadcommand{myFavoriteColor := dark blue()\bound{mfc }}
\indentrel{3}\begin{verbatim}
   (2)  [Hue: 22  Weight: 1.0] from the Dark palette
                                          Type: Palette
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphColorPalettePageEmpty2}
\begin{paste}{ugGraphColorPalettePageEmpty2}{ugGraphColorPalettePagePatch2}
\pastebutton{ugGraphColorPalettePageEmpty2}{\showpaste}
\tab{5}\spadcommand{myFavoriteColor := dark blue()\bound{mfc }}
\end{paste}\end{patch}

\begin{patch}{ugGraphColorPalettePagePatch3}
\begin{paste}{ugGraphColorPalettePageFull3}{ugGraphColorPalettePageEmpty3}
\pastebutton{ugGraphColorPalettePageFull3}{\hidepaste}
\tab{5}\spadcommand{shade myFavoriteColor\free{mfc }}
\indentrel{3}\begin{verbatim}
   (3)  1
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphColorPalettePageEmpty3}
\begin{paste}{ugGraphColorPalettePageEmpty3}{ugGraphColorPalettePagePatch3}
\pastebutton{ugGraphColorPalettePageEmpty3}{\showpaste}
\tab{5}\spadcommand{shade myFavoriteColor\free{mfc }}
\end{paste}\end{patch}

\begin{patch}{ugGraphColorPalettePagePatch4}
\begin{paste}{ugGraphColorPalettePageFull4}{ugGraphColorPalettePageEmpty4}
\pastebutton{ugGraphColorPalettePageFull4}{\hidepaste}
\tab{5}\spadcommand{hue myFavoriteColor\free{mfc }}
\indentrel{3}\begin{verbatim}
   (4)  Hue: 22  Weight: 1.0
                                            Type: Color
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphColorPalettePageEmpty4}
\begin{paste}{ugGraphColorPalettePageEmpty4}{ugGraphColorPalettePagePatch4}
\pastebutton{ugGraphColorPalettePageEmpty4}{\showpaste}
\tab{5}\spadcommand{hue myFavoriteColor\free{mfc }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugGraphColorPalettePagePatch5}
\begin{paste}{ugGraphColorPalettePageFull5}{ugGraphColorPalettePageEmpty5}
\pastebutton{ugGraphColorPalettePageFull5}{\hidepaste}
\tab{5}\spadgraph{draw(x**2,x=-1..1,curveColor == dark blue())}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphcolorpalettepage5.view/image}}}
\end{paste}\end{patch}

\begin{patch}{ugGraphColorPalettePageEmpty5}
\begin{paste}{ugGraphColorPalettePageEmpty5}{ugGraphColorPalettePagePatch5}
\pastebutton{ugGraphColorPalettePageEmpty5}{\showpaste}
\tab{5}\spadgraph{draw(x**2,x=-1..1,curveColor == dark blue())}
\end{paste}\end{patch}
```

———-

Two-Dimensional Control-Panel

— **ug07.ht** —

```
\begin{page}{ugGraphTwoDControlPage}{7.1.7. Two-Dimensional Control-Panel}
\beginscroll
Once you have created a viewport, move your mouse to the viewport and click
with your left mouse button to display a control-panel.
The panel is displayed on the side of the viewport closest to
where you clicked.  Each of the buttons which toggle on and off show the
current state of the graph.


\subsubsection{Transformations}

Object transformations are executed from the control-panel by
mouse-activated potentiometer windows.
%
\indent{0}
\beginitems

\item[Scale:] To scale a graph, click on a mouse button within the
{\bf Scale} window in the upper left corner of the control-panel.  The
axes along which the scaling is to occur are indicated by setting the
toggles above the arrow.  With {\tt X On} and {\tt Y On} appearing,
both axes are selected and scaling is uniform.  If either is not
selected, for example, if {\tt X Off} appears, scaling is non-uniform.

\item[Translate:] To translate a graph, click the mouse in the {\bf
Translate} window in the direction you wish the graph to move.  This
window is located in the upper right corner of the control-panel.
```

Along the top of the {\bf Translate} window are two buttons for selecting the direction of translation. Translation along both coordinate axes results when {\tt X On} and {\tt Y On} appear or along one axis when one is on, for example, {\tt X On} and {\tt Y Off} appear.
\enditems
\indent{0}

\subsubsection{Messages}

The window directly below the transformation potentiometer windows is used to display system messages relating to the viewport and the control-panel. The following format is displayed: \newline

\centerline{{[scaleX, scaleY] \axiom{>}
graph\axiom{<} [translateX, translateY] \newline}}
The two values to the left show the scale factor along the {\tt X} and {\tt Y} coordinate axes. The two values to the right show the distance of translation from the center in the {\tt X} and {\tt Y} directions. The number in the center shows which graph in the viewport this data pertains to. When multiple graphs exist in the same viewport, the graph must be selected (see ‘‘Multiple Graphs,’’ below) in order for its transformation data to be shown, otherwise the number is 1.

\subsubsection{Multiple Graphs}

The {\bf Graphs} window contains buttons that allow the placement of \twodim{} graphs into one of nine available slots in any other \twodim{} viewport. In the center of the window are numeral buttons from one to nine that show whether a graph is displayed in the viewport. Below each number button is a button showing whether a graph that is present is selected for application of some transformation. When the caret symbol is displayed, then the graph in that slot will be manipulated. Initially, the graph for which the viewport is created occupies the first slot, is displayed, and is selected.

%
\indent{0}
\beginitems

\item[Clear:] The {\bf Clear} button deselects every viewport graph slot. A graph slot is reselected by selecting the button below its number.

\item[Query:] The {\bf Query} button is used to display the scale and translate data for the indicated graph. When this button is selected the message ‘‘Click on the graph to query’’ appears. Select a slot number button from the {\bf Graphs} window. The scaling factor and

```
translation offset of the graph are then displayed in the message
window.

\item[Pick:] The {\bf Pick} button is used to select a graph to be
placed or dropped into the indicated viewport.  When this button is
selected, the message ‘‘Click on the graph to pick’’ appears.  Click
on the slot with the graph number of the desired graph.  The graph
information is held waiting for you to execute a {\bf Drop} in some
other graph.

\item[Drop:]  Once a graph has been picked up using the {\bf Pick} button,
the {\bf Drop} button places it into a new viewport slot.
The message ‘‘Click on the graph to drop’’ appears in the message
window when the {\bf Drop} button is selected.
By selecting one of the slot number buttons in the {\bf Graphs}
window, the graph currently being held is dropped into this slot
and displayed.
\enditems
\indent{0}

\subsubsection{Buttons}

%
\indent{0}
\beginitems
%
\item[Axes] turns the coordinate axes on or off.
%
\item[Units] turns the units along the {\tt x}
and {\tt y} axis on or off.
%
\item[Box] encloses the area of the viewport graph
in a bounding box, or removes the box if already enclosed.
%
\item[Pts] turns on or off the display of points.
%
\item[Lines] turns on or off the display
of lines connecting points.
%
\item[PS] writes the current viewport contents to
a file {\bf axiom2D.ps} or to a name specified in the user’s {\bf
.Xdefaults} file.
The file is placed in the directory from which Axiom or the {\bf
viewalone} program was invoked.
%
\item[Reset] resets the object transformation
characteristics and attributes back to their initial states.
%
\item[Hide] makes the control-panel disappear.
%
```

```
\item[Quit] queries whether the current viewport
session should be terminated.
\enditems
\indent{0}

\endscroll
\autobuttons
\end{page}
```

———-

Operations for Two-Dimensional Graphics

— ug07.ht —

```
\begin{page}{ugGraphTwoDopsPage}
{7.1.8. Operations for Two-Dimensional Graphics}
\beginscroll

Here is a summary of useful Axiom operations for \twodim{}
graphics.
Each operation name is followed by a list of arguments.
Each argument is written as a variable informally named according
to the type of the argument (for example, {\it integer}).
If appropriate, a default value for an argument is given in
parentheses immediately following the name.

%
\texht{\bgroup\hbadness = 10001\sloppy}{}
\indent{0}
\beginitems
%
\item[\axiomFun{adaptive}]\funArgs{\optArg{boolean\argDef{true}}}
sets or indicates whether graphs are plotted
according to the adaptive refinement algorithm.

\item[\axiomFun{axesColorDefault}]\funArgs{\optArg{color\argDef{dark
blue()}}} sets or indicates the default color of the axes in a
\twodim{} graph viewport.

\item[\axiomFun{clipPointsDefault}]\funArgs{\optArg{boolean\argDef{false}}}
sets or
indicates whether point clipping is
to be applied as the default for graph plots.
%
\item[\axiomFun{drawToScale}]\funArgs{\optArg{boolean\argDef{false}}}
sets or
```

```
indicates whether the plot of a graph
is ‘‘to scale’’ or uses the entire viewport space as the default.

\item[\axiomFun{lineColorDefault}]\funArgs{\optArg{color\argDef{pastel
yellow()}}} sets or indicates the default color of the lines or curves
in a \twodim{} graph viewport.

\item[\axiomFun{maxPoints}]\funArgs{\optArg{integer\argDef{500}}}
sets or indicates
the default maximum number of
possible points to be used when constructing a \twodim{} graph.
%
\item[\axiomFun{minPoints}]\funArgs{\optArg{integer\argDef{21}}}
sets or indicates the default minimum number of
possible points to be used when constructing a \twodim{} graph.

\item[\axiomFun{pointColorDefault}]\funArgs{\optArg{color\argDef{bright
red()}}} sets or indicates the default color of the points in a
\twodim{} graph viewport.

\item[\axiomFun{pointSizeDefault}]\funArgs{\optArg{integer\argDef{5}}}
sets or indicates the default size of the
dot used to plot points in a \twodim{} graph.
%
\item[\axiomFun{screenResolution}]\funArgs{\optArg{integer\argDef{600}}}
sets or indicates the default screen
resolution constant used in setting the computation limit of adaptively
generated curve plots.

\item[\axiomFun{unitsColorDefault}]\funArgs{\optArg{color\argDef{dim
green()}}} sets or indicates the default color of the unit labels in a
\twodim{} graph viewport.

\item[\axiomFun{viewDefaults}]\funArgs{}
resets the default settings for the following
attributes:  point color, line color, axes color, units color, point size,
viewport upper left-hand corner position, and the viewport size.

\item[\axiomFun{viewPosDefault}]\funArgs{\optArg{list\argDef{[100,100]}}}
sets or indicates the default position of the upper left-hand corner
of a \twodim{} viewport, relative to the display root window.  The
upper left-hand corner of the display is considered to be at the (0,
0) position.

\item[\axiomFun{viewSizeDefault}]\funArgs{\optArg{list\argDef{[200,200]}}}
sets or indicates the default size in which two dimensional viewport
windows are shown.  It is defined by a width and then a height.

\item[\axiomFun{viewWriteAvailable}]\funArgs{\optArg{list\argDef{["pixmap",
"bitmap", "postscript", \"image"}}} indicates the possible file types
```

```
that can be created with the
\axiomFunFrom{write}{TwoDimensionalViewport} function.

\item[\axiomFun{viewWriteDefault}]
\funArgs{\optArg{list\argDef{[]}}}
sets or indicates the default types of files, in
addition to the {\bf data} file, that are created when a
\axiomFun{write} function is executed on a viewport.

\item[\axiomFun{units}]\funArgs{viewport, integer\argDef{1},
string\argDef{"off"}} turns the units on or off for the graph with
index {\it integer}.

\item[\axiomFun{axes}]
\funArgs{viewport, integer\argDef{1}, string\argDef{"on"}}
turns the axes on
or off for the graph with index {\it integer}.
%
\item[\axiomFun{close}]\funArgs{viewport}
closes {\it viewport}.
%
\item[\axiomFun{connect}]
\funArgs{viewport, integer\argDef{1}, string\argDef{"on"}}
declares whether lines
connecting the points are displayed or not.
%
\item[\axiomFun{controlPanel}]\funArgs{viewport, string\argDef{"off"}}
declares
whether the \twodim{} control-panel is automatically displayed
or not.
%
\item[\axiomFun{graphs}]\funArgs{viewport}
returns a list
describing the state of each graph.
If the graph state is not being used this is shown by {\tt "undefined"},
otherwise a description of the graph's contents is shown.
%
\item[\axiomFun{graphStates}]\funArgs{viewport}
displays
a list of all the graph states available for {\it viewport}, giving the
values for every property.
%
\item[\axiomFun{key}]\funArgs{viewport}
returns the process
ID number for {\it viewport}.
%
\item[\axiomFun{move}]\funArgs{viewport,
\subscriptText{integer}{x}(viewPosDefault),
\subscriptText{integer}{y}(viewPosDefault)}
moves {\it viewport} on the screen so that the
```

```
upper left-hand corner of {\it viewport} is at the position {\it (x,y)}.
%
\item[\axiomFun{options}]\funArgs{\it viewport}
returns a list
of all the \axiomType{DrawOption}s used by {\it viewport}.
%
\item[\axiomFun{points}]
\funArgs{viewport, integer\argDef{1}, string\argDef{"on"}}
specifies whether the graph points for graph {\it integer} are
to be displayed or not.
%
\item[\axiomFun{region}]
\funArgs{viewport, integer\argDef{1}, string\argDef{"off"}}
declares whether graph {\it integer} is or is not to be displayed
with a bounding rectangle.
%
\item[\axiomFun{reset}]\funArgs{viewport}
resets all the properties of {\it viewport}.
%
\item[\axiomFun{resize}]\funArgs{viewport,
\subscriptText{integer}{width}, \subscriptText{integer}{height}}
resizes {\it viewport} with a new {\it width} and {\it height}.
%
\item[\axiomFun{scale}]\
funArgs{viewport, \subscriptText{integer}{n}\argDef{1},
\subscriptText{integer}{x}
\argDef{0.9}, \subscriptText{integer}{y}\argDef{0.9}}
scales values for the
{\it x} and {\it y} coordinates of graph {\it n}.
%
\item[\axiomFun{show}]
\funArgs{viewport, \subscriptText{integer}{n}\argDef{1},
string\argDef{"on"}}
indicates if graph {\it n} is shown or not.
%
\item[\axiomFun{title}]\funArgs{viewport, string\argDef{"Axiom 2D"}}
designates the title for {\it viewport}.
%
\item[\axiomFun{translate}]\funArgs{viewport,
\subscriptText{integer}{n}\argDef{1},
\subscriptText{float}{x}
\argDef{0.0}, \subscriptText{float}{y}\argDef{0.0}}
causes graph {\it n} to be moved {\it x}
and {\it y} units in the respective directions.
%
\item[\axiomFun{write}]\funArgs{viewport,
\subscriptText{string}{directory},
\optArg{strings}}
if no third argument is given, writes the
{\bf data} file onto the directory
```

```
with extension {\bf data}.
The third argument can be a single string
or a list of strings with some or
all the entries {\tt "pixmap"}, {\tt "bitmap"},
{\tt "postscript"}, and
{\tt "image"}.
\enditems
\indent{0}
\texht{\egroup}{}

\endscroll
\autobuttons
\end{page}
```

———-

Addendum: Building Two-Dimensional Graphs

— ug07.ht —

```
\begin{page}{ugGraphTwoDbuildPage}
{7.1.9. Addendum: Building Two-Dimensional Graphs}
\beginscroll

In this section we demonstrate how to create \twodim{} graphs from
lists of points and give an example showing how to read the lists
of points from a file.

\subsubsection{Creating a Two-Dimensional Viewport from a List of Points}

Axiom creates lists of points in a \twodim{} viewport by utilizing the
\axiomType{GraphImage} and \axiomType{TwoDimensionalViewport} domains.
In this example, the \axiomFunFrom{makeGraphImage}{GraphImage}
function takes a list of lists of points parameter, a list of colors
for each point in the graph, a list of colors for each line in the
graph, and a list of sizes for each point in the graph.

\xtc{
The following expressions create a list of lists of points which will
be read by Axiom and made into a \twodim{} viewport.
}{
\spadpaste{p1 := point [1,1]\$(Point DFLOAT) \bound{p1}}
}
\xtc{
}{
\spadpaste{p2 := point [0,1]\$(Point DFLOAT) \bound{p2}}
}
```

```
\xtc{
}{
\spadpaste{p3 := point [0,0]\$(Point DFLOAT) \bound{p3}}
}
\xtc{
}{
\spadpaste{p4 := point [1,0]\$(Point DFLOAT) \bound{p4}}
}
\xtc{
}{
\spadpaste{p5 := point [1,.5]\$(Point DFLOAT) \bound{p5}}
}
\xtc{
}{
\spadpaste{p6 := point [.5,0]\$(Point DFLOAT) \bound{p6}}
}
\xtc{
}{
\spadpaste{p7 := point [0,0.5]\$(Point DFLOAT) \bound{p7}}
}
\xtc{
}{
\spadpaste{p8 := point [.5,1]\$(Point DFLOAT) \bound{p8}}
}
\xtc{
}{
\spadpaste{p9 := point [.25,.25]\$(Point DFLOAT) \bound{p9}}
}
\xtc{
}{
\spadpaste{p10 := point [.25,.75]\$(Point DFLOAT) \bound{p10}}
}
\xtc{
}{
\spadpaste{p11 := point [.75,.75]\$(Point DFLOAT) \bound{p11}}
}
\xtc{
}{
\spadpaste{p12 := point [.75,.25]\$(Point DFLOAT) \bound{p12}}
}
\xtc{
Finally, here is the list.
}{
\spadpaste{llp := [[p1,p2], [p2,p3], [p3,p4], [p4,p1], [p5,p6], [p6,p7],
[p7,p8], [p8,p5], [p9,p10], [p10,p11], [p11,p12], [p12,p9]]
\free{p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 p11 p12} \bound{llp}}
}
\xtc{
Now we set the point sizes for all components of the graph.
}{
```

```
\spadpaste{size1 := 6::PositiveInteger \bound{size1}}
}
\xtc{
}{
}
\xtc{
}{
\spadpaste{size2 := 8::PositiveInteger \bound{size2}}
}
\xtc{
}{
\spadpaste{size3 := 10::PositiveInteger \bound{size3}}
}
\xtc{
}{
\spadpaste{lsize := [size1, size1, size1, size1, size2, size2, size2,
size2, size3, size3, size3, size3] \bound{lsize} \free{size1 size2 size3}}
}
\xtc{
Here are the colors for the points.
}{
\spadpaste{pc1 := pastel red() \bound{pc1}}
}
\xtc{
}{
\spadpaste{pc2 := dim green() \bound{pc2}}
}
\xtc{
}{
\spadpaste{pc3 := pastel yellow() \bound{pc3}}
}
\xtc{
}{
\spadpaste{lpc := [pc1, pc1, pc1, pc1, pc2, pc2, pc2, pc2, pc3, pc3,
pc3, pc3] \free{pc1 pc2 pc3} \bound{lpc}}
}
\xtc{
Here are the colors for the lines.
}{
\spadpaste{lc := [pastel blue(), light yellow(), dim green(),
bright red(), light green(), dim yellow(), bright blue(),
dark red(), pastel red(), light blue(), dim green(),
light yellow()] \bound{lc}}
}
\xtc{
Now the \axiomType{GraphImage} is created according to the component
specifications indicated above.
}{
\spadpaste{g := makeGraphImage(llp,lpc,lc,lsize)\$GRIMAGE
\bound{g} \free{llp lpc lc lsize}}
```

```
}
\psXtc{
The \axiomFunFrom{makeViewport2D}{TwoDimensionalViewport} function now
creates a \axiomType{TwoDimensionalViewport} for this graph according to the
list of options specified within the brackets.
}{
\graphpaste{makeViewport2D(g,[title("Lines")])\$VIEW2D \free{g}}
}{
%
}
%See Figure #.#.
\xtc{
This example demonstrates the use of the \axiomType{GraphImage}
functions \axiomFunFrom{component}{GraphImage} and
\axiomFunFrom{appendPoint}{GraphImage} in adding points to an empty
\axiomType{GraphImage}.
}{
\spadpaste{)clear all \bound{clearAll}}
}
\xtc{
}{
\spadpaste{g := graphImage()\$GRIMAGE \bound{Sg}\free{clearAll}}
}
\xtc{
}{
\spadpaste{p1 := point [0,0]\$(Point DFLOAT) \bound{Sp1}}
}
\xtc{
}{
\spadpaste{p2 := point [.25,.25]\$(Point DFLOAT) \bound{Sp2}}
}
\xtc{
}{
\spadpaste{p3 := point [.5,.5]\$(Point DFLOAT) \bound{Sp3}}
}
\xtc{
}{
\spadpaste{p4 := point [.75,.75]\$(Point DFLOAT) \bound{Sp4}}
}
\xtc{
}{
\spadpaste{p5 := point [1,1]\$(Point DFLOAT) \bound{Sp5}}
}
\xtc{
}{
\spadpaste{component(g,p1)\$GRIMAGE\free{Sg Sp1}\bound{gp1}}
}
\xtc{
}{
\spadpaste{component(g,p2)\$GRIMAGE\free{Sg Sp2}\bound{gp2}}
```

```
}
\xtc{
}{
\spadpaste{appendPoint(g,p3)\$GRIMAGE\free{gp1 gp2 Sp3}\bound{gp3}}
}
\xtc{
}{
\spadpaste{appendPoint(g,p4)\$GRIMAGE\free{gp3 Sp4}\bound{gp4}}
}
\xtc{
}{
\spadpaste{appendPoint(g,p5)\$GRIMAGE\free{gp4 Sp5}\bound{gp5}}
}
\xtc{
}{
\spadpaste{g1 := makeGraphImage(g)\$GRIMAGE \bound{Sg1} \free{gp5}}
}
\psXtc{
Here is the graph.
}{
\graphpaste{makeViewport2D(g1,[title("Graph Points")])\$VIEW2D \free{Sg1}}
}{
%
}
%
%See Figure #.#.
%
\xtc{
A list of points can also be made into a \axiomType{GraphImage} by
using the operation \axiomFunFrom{coerce}{GraphImage}.  It is
equivalent to adding each point to \axiom{g2} using
\axiomFunFrom{component}{GraphImage}.
}{
\spadpaste{g2 := coerce([[p1],[p2],[p3],[p4],[p5]])\$GRIMAGE
\free{Sp1 Sp2 Sp3 Sp4 Sp5} \bound{Sg2}}
}
\xtc{
Now, create an empty \axiomType{TwoDimensionalViewport}.
}{
\spadpaste{v := viewport2D()\$VIEW2D \bound{Sv}}
}
\xtc{
}{
\spadpaste{options(v,[title("Just Points")])\$VIEW2D \free{Sv}\bound{Svo}}
}
\xtc{
Place the graph into the viewport.
}{
\spadpaste{putGraph(v,g2,1)\$VIEW2D \free{Sg2 Svo}\bound{Svog2}}
}
```

```
\psXtc{
Take a look.
}{
\graphpaste{makeViewport2D(v)\$VIEW2D \free{Svog2}}
}{
%
}

%See Figure #.#.

\subsubsection{Creating a Two-Dimensional Viewport of a
List of Points from a File}

The following three functions read a list of points from a file and
then draw the points and the connecting lines. The points are stored
in the file in readable form as floating point numbers (specifically,
\axiomType{DoubleFloat} values) as an alternating stream of \axiom{x}-
and \axiom{y}-values. For example,
\begin{verbatim}
0.0 0.0     1.0 1.0     2.0 4.0
3.0 9.0     4.0 16.0    5.0 25.0
\end{verbatim}

\beginImportant

\noindent
{\tt 1.\ \ \ drawPoints(lp:List\ Point\ DoubleFloat):VIEW2D\ ==}\newline
{\tt 2.\ \ \ \ \ g\ :=\ graphImage()\$GRIMAGE}\newline
{\tt 3.\ \ \ \ \ for\ p\ in\ lp\ repeat}\newline
{\tt 4.\ \ \ \ \ \ \
component(g,p,pointColorDefault(),lineColorDefault(),}\newline
{\tt 5.\ \ \ \ \ \ \ \ \ pointSizeDefault())}\newline
{\tt 6.\ \ \ \ \ gi\ :=\ makeGraphImage(g)\$GRIMAGE}\newline
{\tt 7.\ \ \ \ \ makeViewport2D(gi,[title("Points")])\$VIEW2D}\newline
{\tt 8.\ \ \ }\newline
{\tt 9.\ \ \ drawLines(lp:List\ Point\ DoubleFloat):VIEW2D\ ==}\newline
{\tt 10.\ \ \ \ g\ :=\ graphImage()\$GRIMAGE}\newline
{\tt 11.\ \ \ \
component(g,\ lp,\ pointColorDefault(),\ lineColorDefault(),}\newline
{\tt 12.\ \ \ \ \ \ pointSizeDefault())\$GRIMAGE}\newline
{\tt 13.\ \ \ \ gi\ :=\ makeGraphImage(g)\$GRIMAGE}\newline
{\tt 14.\ \ \ \ makeViewport2D(gi,[title("Points")])\$VIEW2D}\newline
{\tt 15.\ \ }\newline
{\tt 16.\ \ plotData2D(name,\ title)\ ==}\newline
{\tt 17.\ \ \ \ f:File(DFLOAT)\ :=\ open(name,"input")}\newline
{\tt 18.\ \ \ \ lp:LIST(Point\ DFLOAT)\ :=\ empty()}\newline
{\tt 19.\ \ \ \
while\ ((x\ :=\ readIfCan!(f))\ case\ DFLOAT)\ repeat}\newline
{\tt 20.\ \ \ \ \ \ y\ :\ DFLOAT\ :=\ read!(f)}\newline
{\tt 21.\ \ \ \ \ \
```

```
lp\ :=\ cons(point\ [x,y]\$(Point\ DFLOAT),\ lp)}\newline
{\tt 22.\ \ \ \ \ \ lp}\newline
{\tt 23.\ \ \ \ close!(f)}\newline
{\tt 24.\ \ \ \ drawPoints(lp)}\newline
{\tt 25.\ \ \ \ drawLines(lp)}\newline
\endImportant
%
This command will actually create the viewport and the graph if
the point data is in the file \axiom{"file.data"}.
\beginImportant

\noindent
{\tt 1.\ \ \ plotData2D("file.data",\ "2D\ Data\ Plot")}\newline
\endImportant

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugGraphTwoDbuildPagePatch1}
\begin{paste}{ugGraphTwoDbuildPageFull1}{ugGraphTwoDbuildPageEmpty1}
\pastebutton{ugGraphTwoDbuildPageFull1}{\hidepaste}
\tab{5}\spadcommand{p1 := point [1,1]$(Point DFLOAT)\bound{p1 }}
\indentrel{3}\begin{verbatim}
   (1)  [1.0,1.0]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPageEmpty1}
\begin{paste}{ugGraphTwoDbuildPageEmpty1}{ugGraphTwoDbuildPagePatch1}
\pastebutton{ugGraphTwoDbuildPageEmpty1}{\showpaste}
\tab{5}\spadcommand{p1 := point [1,1]$(Point DFLOAT)\bound{p1 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPagePatch2}
\begin{paste}{ugGraphTwoDbuildPageFull2}{ugGraphTwoDbuildPageEmpty2}
\pastebutton{ugGraphTwoDbuildPageFull2}{\hidepaste}
\tab{5}\spadcommand{p2 := point [0,1]$(Point DFLOAT)\bound{p2 }}
\indentrel{3}\begin{verbatim}
   (2)  [0.0,1.0]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPageEmpty2}
\begin{paste}{ugGraphTwoDbuildPageEmpty2}{ugGraphTwoDbuildPagePatch2}
\pastebutton{ugGraphTwoDbuildPageEmpty2}{\showpaste}
\tab{5}\spadcommand{p2 := point [0,1]$(Point DFLOAT)\bound{p2 }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugGraphTwoDbuildPagePatch3}
\begin{paste}{ugGraphTwoDbuildPageFull3}{ugGraphTwoDbuildPageEmpty3}
\pastebutton{ugGraphTwoDbuildPageFull3}{\hidepaste}
\tab{5}\spadcommand{p3 := point [0,0]$(Point DFLOAT)\bound{p3 }}
\indentrel{3}\begin{verbatim}
   (3)  [0.0,0.0]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPageEmpty3}
\begin{paste}{ugGraphTwoDbuildPageEmpty3}{ugGraphTwoDbuildPagePatch3}
\pastebutton{ugGraphTwoDbuildPageEmpty3}{\showpaste}
\tab{5}\spadcommand{p3 := point [0,0]$(Point DFLOAT)\bound{p3 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPagePatch4}
\begin{paste}{ugGraphTwoDbuildPageFull4}{ugGraphTwoDbuildPageEmpty4}
\pastebutton{ugGraphTwoDbuildPageFull4}{\hidepaste}
\tab{5}\spadcommand{p4 := point [1,0]$(Point DFLOAT)\bound{p4 }}
\indentrel{3}\begin{verbatim}
   (4)  [1.0,0.0]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPageEmpty4}
\begin{paste}{ugGraphTwoDbuildPageEmpty4}{ugGraphTwoDbuildPagePatch4}
\pastebutton{ugGraphTwoDbuildPageEmpty4}{\showpaste}
\tab{5}\spadcommand{p4 := point [1,0]$(Point DFLOAT)\bound{p4 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPagePatch5}
\begin{paste}{ugGraphTwoDbuildPageFull5}{ugGraphTwoDbuildPageEmpty5}
\pastebutton{ugGraphTwoDbuildPageFull5}{\hidepaste}
\tab{5}\spadcommand{p5 := point [1,.5]$(Point DFLOAT)\bound{p5 }}
\indentrel{3}\begin{verbatim}
   (5)  [1.0,0.5]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPageEmpty5}
\begin{paste}{ugGraphTwoDbuildPageEmpty5}{ugGraphTwoDbuildPagePatch5}
\pastebutton{ugGraphTwoDbuildPageEmpty5}{\showpaste}
\tab{5}\spadcommand{p5 := point [1,.5]$(Point DFLOAT)\bound{p5 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPagePatch6}
```

```
\begin{paste}{ugGraphTwoDbuildPageFull6}{ugGraphTwoDbuildPageEmpty6}
\pastebutton{ugGraphTwoDbuildPageFull6}{\hidepaste}
\tab{5}\spadcommand{p6 := point [.5,0]$(Point DFLOAT)\bound{p6 }}
\indentrel{3}\begin{verbatim}
   (6)  [0.5,0.0]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPageEmpty6}
\begin{paste}{ugGraphTwoDbuildPageEmpty6}{ugGraphTwoDbuildPagePatch6}
\pastebutton{ugGraphTwoDbuildPageEmpty6}{\showpaste}
\tab{5}\spadcommand{p6 := point [.5,0]$(Point DFLOAT)\bound{p6 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPagePatch7}
\begin{paste}{ugGraphTwoDbuildPageFull7}{ugGraphTwoDbuildPageEmpty7}
\pastebutton{ugGraphTwoDbuildPageFull7}{\hidepaste}
\tab{5}\spadcommand{p7 := point [0,0.5]$(Point DFLOAT)\bound{p7 }}
\indentrel{3}\begin{verbatim}
   (7)  [0.0,0.5]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPageEmpty7}
\begin{paste}{ugGraphTwoDbuildPageEmpty7}{ugGraphTwoDbuildPagePatch7}
\pastebutton{ugGraphTwoDbuildPageEmpty7}{\showpaste}
\tab{5}\spadcommand{p7 := point [0,0.5]$(Point DFLOAT)\bound{p7 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPagePatch8}
\begin{paste}{ugGraphTwoDbuildPageFull8}{ugGraphTwoDbuildPageEmpty8}
\pastebutton{ugGraphTwoDbuildPageFull8}{\hidepaste}
\tab{5}\spadcommand{p8 := point [.5,1]$(Point DFLOAT)\bound{p8 }}
\indentrel{3}\begin{verbatim}
   (8)  [0.5,1.0]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPageEmpty8}
\begin{paste}{ugGraphTwoDbuildPageEmpty8}{ugGraphTwoDbuildPagePatch8}
\pastebutton{ugGraphTwoDbuildPageEmpty8}{\showpaste}
\tab{5}\spadcommand{p8 := point [.5,1]$(Point DFLOAT)\bound{p8 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPagePatch9}
\begin{paste}{ugGraphTwoDbuildPageFull9}{ugGraphTwoDbuildPageEmpty9}
\pastebutton{ugGraphTwoDbuildPageFull9}{\hidepaste}
```

```
\tab{5}\spadcommand{p9 := point [.25,.25]$(Point DFLOAT)\bound{p9 }}
\indentrel{3}\begin{verbatim}
   (9)  [0.25,0.25]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPageEmpty9}
\begin{paste}{ugGraphTwoDbuildPageEmpty9}{ugGraphTwoDbuildPagePatch9}
\pastebutton{ugGraphTwoDbuildPageEmpty9}{\showpaste}
\tab{5}\spadcommand{p9 := point [.25,.25]$(Point DFLOAT)\bound{p9 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPagePatch10}
\begin{paste}{ugGraphTwoDbuildPageFull10}{ugGraphTwoDbuildPageEmpty10}
\pastebutton{ugGraphTwoDbuildPageFull10}{\hidepaste}
\tab{5}\spadcommand{p10 := point [.25,.75]$(Point DFLOAT)\bound{p10 }}
\indentrel{3}\begin{verbatim}
   (10)  [0.25,0.75]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPageEmpty10}
\begin{paste}{ugGraphTwoDbuildPageEmpty10}{ugGraphTwoDbuildPagePatch10}
\pastebutton{ugGraphTwoDbuildPageEmpty10}{\showpaste}
\tab{5}\spadcommand{p10 := point [.25,.75]$(Point DFLOAT)\bound{p10 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPagePatch11}
\begin{paste}{ugGraphTwoDbuildPageFull11}{ugGraphTwoDbuildPageEmpty11}
\pastebutton{ugGraphTwoDbuildPageFull11}{\hidepaste}
\tab{5}\spadcommand{p11 := point [.75,.75]$(Point DFLOAT)\bound{p11 }}
\indentrel{3}\begin{verbatim}
   (11)  [0.75,0.75]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPageEmpty11}
\begin{paste}{ugGraphTwoDbuildPageEmpty11}{ugGraphTwoDbuildPagePatch11}
\pastebutton{ugGraphTwoDbuildPageEmpty11}{\showpaste}
\tab{5}\spadcommand{p11 := point [.75,.75]$(Point DFLOAT)\bound{p11 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPagePatch12}
\begin{paste}{ugGraphTwoDbuildPageFull12}{ugGraphTwoDbuildPageEmpty12}
\pastebutton{ugGraphTwoDbuildPageFull12}{\hidepaste}
\tab{5}\spadcommand{p12 := point [.75,.25]$(Point DFLOAT)\bound{p12 }}
\indentrel{3}\begin{verbatim}
```

```
   (12)  [0.75,0.25]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPageEmpty12}
\begin{paste}{ugGraphTwoDbuildPageEmpty12}{ugGraphTwoDbuildPagePatch12}
\pastebutton{ugGraphTwoDbuildPageEmpty12}{\showpaste}
\tab{5}\spadcommand{p12 := point [.75,.25]$(Point DFLOAT)\bound{p12 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPagePatch13}
\begin{paste}{ugGraphTwoDbuildPageFull13}{ugGraphTwoDbuildPageEmpty13}
\pastebutton{ugGraphTwoDbuildPageFull13}{\hidepaste}
\tab{5}\spadcommand{llp := [[p1,p2], [p2,p3], [p3,p4], [p4,p1], [p5,p6], [p6,p7], [p7,p8], [p8,p5], [p9,p10
\indentrel{3}\begin{verbatim}
   (13)
   [[[1.0,1.0],[0.0,1.0]], [[0.0,1.0],[0.0,0.0]],
    [[0.0,0.0],[1.0,0.0]], [[1.0,0.0],[1.0,1.0]],
    [[1.0,0.5],[0.5,0.0]], [[0.5,0.0],[0.0,0.5]],
    [[0.0,0.5],[0.5,1.0]], [[0.5,1.0],[1.0,0.5]],
    [[0.25,0.25],[0.25,0.75]], [[0.25,0.75],[0.75,0.75]],
    [[0.75,0.75],[0.75,0.25]], [[0.75,0.25],[0.25,0.25]]]
                      Type: List List Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPageEmpty13}
\begin{paste}{ugGraphTwoDbuildPageEmpty13}{ugGraphTwoDbuildPagePatch13}
\pastebutton{ugGraphTwoDbuildPageEmpty13}{\showpaste}
\tab{5}\spadcommand{llp := [[p1,p2], [p2,p3], [p3,p4], [p4,p1], [p5,p6], [p6,p7], [p7,p8], [p8,p5], [p9,p10
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPagePatch14}
\begin{paste}{ugGraphTwoDbuildPageFull14}{ugGraphTwoDbuildPageEmpty14}
\pastebutton{ugGraphTwoDbuildPageFull14}{\hidepaste}
\tab{5}\spadcommand{size1 := 6::PositiveInteger\bound{size1 }}
\indentrel{3}\begin{verbatim}
   (14)  6
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPageEmpty14}
\begin{paste}{ugGraphTwoDbuildPageEmpty14}{ugGraphTwoDbuildPagePatch14}
\pastebutton{ugGraphTwoDbuildPageEmpty14}{\showpaste}
\tab{5}\spadcommand{size1 := 6::PositiveInteger\bound{size1 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPagePatch15}
```

```
\begin{paste}{ugGraphTwoDbuildPageFull15}{ugGraphTwoDbuildPageEmpty15}
\pastebutton{ugGraphTwoDbuildPageFull15}{\hidepaste}
\tab{5}\spadcommand{size2 := 8::PositiveInteger\bound{size2 }}
\indentrel{3}\begin{verbatim}
   (15)  8
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPageEmpty15}
\begin{paste}{ugGraphTwoDbuildPageEmpty15}{ugGraphTwoDbuildPagePatch15}
\pastebutton{ugGraphTwoDbuildPageEmpty15}{\showpaste}
\tab{5}\spadcommand{size2 := 8::PositiveInteger\bound{size2 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPagePatch16}
\begin{paste}{ugGraphTwoDbuildPageFull16}{ugGraphTwoDbuildPageEmpty16}
\pastebutton{ugGraphTwoDbuildPageFull16}{\hidepaste}
\tab{5}\spadcommand{size3 := 10::PositiveInteger\bound{size3 }}
\indentrel{3}\begin{verbatim}
   (16)  10
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPageEmpty16}
\begin{paste}{ugGraphTwoDbuildPageEmpty16}{ugGraphTwoDbuildPagePatch16}
\pastebutton{ugGraphTwoDbuildPageEmpty16}{\showpaste}
\tab{5}\spadcommand{size3 := 10::PositiveInteger\bound{size3 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPagePatch17}
\begin{paste}{ugGraphTwoDbuildPageFull17}{ugGraphTwoDbuildPageEmpty17}
\pastebutton{ugGraphTwoDbuildPageFull17}{\hidepaste}
\tab{5}\spadcommand{lsize := [size1, size1, size1, size1, size2, size2, size2, size2, size3, siz
\indentrel{3}\begin{verbatim}
   (17)  [6,6,6,6,8,8,8,8,10,10,10,10]
                             Type: List PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPageEmpty17}
\begin{paste}{ugGraphTwoDbuildPageEmpty17}{ugGraphTwoDbuildPagePatch17}
\pastebutton{ugGraphTwoDbuildPageEmpty17}{\showpaste}
\tab{5}\spadcommand{lsize := [size1, size1, size1, size1, size2, size2, size2, size2, size3, siz
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPagePatch18}
\begin{paste}{ugGraphTwoDbuildPageFull18}{ugGraphTwoDbuildPageEmpty18}
\pastebutton{ugGraphTwoDbuildPageFull18}{\hidepaste}
```

```
\tab{5}\spadcommand{pc1 := pastel red()\bound{pc1 }}
\indentrel{3}\begin{verbatim}
   (18)  [Hue: 1  Weight: 1.0] from the Pastel palette
                                          Type: Palette
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPageEmpty18}
\begin{paste}{ugGraphTwoDbuildPageEmpty18}{ugGraphTwoDbuildPagePatch18}
\pastebutton{ugGraphTwoDbuildPageEmpty18}{\showpaste}
\tab{5}\spadcommand{pc1 := pastel red()\bound{pc1 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPagePatch19}
\begin{paste}{ugGraphTwoDbuildPageFull19}{ugGraphTwoDbuildPageEmpty19}
\pastebutton{ugGraphTwoDbuildPageFull19}{\hidepaste}
\tab{5}\spadcommand{pc2 := dim green()\bound{pc2 }}
\indentrel{3}\begin{verbatim}
   (19)  [Hue: 14  Weight: 1.0] from the Dim palette
                                          Type: Palette
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPageEmpty19}
\begin{paste}{ugGraphTwoDbuildPageEmpty19}{ugGraphTwoDbuildPagePatch19}
\pastebutton{ugGraphTwoDbuildPageEmpty19}{\showpaste}
\tab{5}\spadcommand{pc2 := dim green()\bound{pc2 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPagePatch20}
\begin{paste}{ugGraphTwoDbuildPageFull20}{ugGraphTwoDbuildPageEmpty20}
\pastebutton{ugGraphTwoDbuildPageFull20}{\hidepaste}
\tab{5}\spadcommand{pc3 := pastel yellow()\bound{pc3 }}
\indentrel{3}\begin{verbatim}
   (20)  [Hue: 11  Weight: 1.0] from the Pastel palette
                                          Type: Palette
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPageEmpty20}
\begin{paste}{ugGraphTwoDbuildPageEmpty20}{ugGraphTwoDbuildPagePatch20}
\pastebutton{ugGraphTwoDbuildPageEmpty20}{\showpaste}
\tab{5}\spadcommand{pc3 := pastel yellow()\bound{pc3 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPagePatch21}
\begin{paste}{ugGraphTwoDbuildPageFull21}{ugGraphTwoDbuildPageEmpty21}
\pastebutton{ugGraphTwoDbuildPageFull21}{\hidepaste}
\tab{5}\spadcommand{lpc := [pc1, pc1, pc1, pc1, pc2, pc2, pc2, pc2, pc3, pc3, pc3, pc3]\free{pc1 pc2 pc3 }\
\indentrel{3}\begin{verbatim}
```

```
   (21)
   [[Hue: 1  Weight: 1.0] from the Pastel palette,
    [Hue: 1  Weight: 1.0] from the Pastel palette,
    [Hue: 1  Weight: 1.0] from the Pastel palette,
    [Hue: 1  Weight: 1.0] from the Pastel palette,
    [Hue: 14  Weight: 1.0] from the Dim palette,
    [Hue: 14  Weight: 1.0] from the Dim palette,
    [Hue: 14  Weight: 1.0] from the Dim palette,
    [Hue: 14  Weight: 1.0] from the Dim palette,
    [Hue: 11  Weight: 1.0] from the Pastel palette,
    [Hue: 11  Weight: 1.0] from the Pastel palette,
    [Hue: 11  Weight: 1.0] from the Pastel palette,
    [Hue: 11  Weight: 1.0] from the Pastel palette]
                                     Type: List Palette
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPageEmpty21}
\begin{paste}{ugGraphTwoDbuildPageEmpty21}{ugGraphTwoDbuildPagePatch21}
\pastebutton{ugGraphTwoDbuildPageEmpty21}{\showpaste}
\tab{5}\spadcommand{lpc := [pc1, pc1, pc1, pc1, pc2, pc2, pc2, pc2, pc3, pc3, pc3, pc3]\free{pc1
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPagePatch22}
\begin{paste}{ugGraphTwoDbuildPageFull22}{ugGraphTwoDbuildPageEmpty22}
\pastebutton{ugGraphTwoDbuildPageFull22}{\hidepaste}
\tab{5}\spadcommand{lc := [pastel blue(), light yellow(), dim green(), bright red(), light green
\indentrel{3}\begin{verbatim}
   (22)
   [[Hue: 22  Weight: 1.0] from the Pastel palette,
    [Hue: 11  Weight: 1.0] from the Light palette,
    [Hue: 14  Weight: 1.0] from the Dim palette,
    [Hue: 1  Weight: 1.0] from the Bright palette,
    [Hue: 14  Weight: 1.0] from the Light palette,
    [Hue: 11  Weight: 1.0] from the Dim palette,
    [Hue: 22  Weight: 1.0] from the Bright palette,
    [Hue: 1  Weight: 1.0] from the Dark palette,
    [Hue: 1  Weight: 1.0] from the Pastel palette,
    [Hue: 22  Weight: 1.0] from the Light palette,
    [Hue: 14  Weight: 1.0] from the Dim palette,
    [Hue: 11  Weight: 1.0] from the Light palette]
                                     Type: List Palette
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPageEmpty22}
\begin{paste}{ugGraphTwoDbuildPageEmpty22}{ugGraphTwoDbuildPagePatch22}
\pastebutton{ugGraphTwoDbuildPageEmpty22}{\showpaste}
\tab{5}\spadcommand{lc := [pastel blue(), light yellow(), dim green(), bright red(), light green
\end{paste}\end{patch}
```

```
\begin{patch}{ugGraphTwoDbuildPagePatch23}
\begin{paste}{ugGraphTwoDbuildPageFull23}{ugGraphTwoDbuildPageEmpty23}
\pastebutton{ugGraphTwoDbuildPageFull23}{\hidepaste}
\tab{5}\spadcommand{g := makeGraphImage(llp,lpc,lc,lsize)$GRIMAGE\bound{g }\free{llp lpc lc lsize }}
\indentrel{3}\begin{verbatim}
   (23)  Graph with 12 point lists
                                       Type: GraphImage
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPageEmpty23}
\begin{paste}{ugGraphTwoDbuildPageEmpty23}{ugGraphTwoDbuildPagePatch23}
\pastebutton{ugGraphTwoDbuildPageEmpty23}{\showpaste}
\tab{5}\spadcommand{g := makeGraphImage(llp,lpc,lc,lsize)$GRIMAGE\bound{g }\free{llp lpc lc lsize }}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPagePatch24}
\begin{paste}{ugGraphTwoDbuildPageFull24}{ugGraphTwoDbuildPageEmpty24}
\pastebutton{ugGraphTwoDbuildPageFull24}{\hidepaste}
\tab{5}\spadgraph{makeViewport2D(g,[title("Lines")])$VIEW2D\free{g }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphtwodbuildpage24.view/image}}{viewalone\spa
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPageEmpty24}
\begin{paste}{ugGraphTwoDbuildPageEmpty24}{ugGraphTwoDbuildPagePatch24}
\pastebutton{ugGraphTwoDbuildPageEmpty24}{\showpaste}
\tab{5}\spadgraph{makeViewport2D(g,[title("Lines")])$VIEW2D\free{g }}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPagePatch25}
\begin{paste}{ugGraphTwoDbuildPageFull25}{ugGraphTwoDbuildPageEmpty25}
\pastebutton{ugGraphTwoDbuildPageFull25}{\hidepaste}
\tab{5}\spadcommand{)clear all\bound{clearAll }}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPageEmpty25}
\begin{paste}{ugGraphTwoDbuildPageEmpty25}{ugGraphTwoDbuildPagePatch25}
\pastebutton{ugGraphTwoDbuildPageEmpty25}{\showpaste}
\tab{5}\spadcommand{)clear all\bound{clearAll }}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPagePatch26}
\begin{paste}{ugGraphTwoDbuildPageFull26}{ugGraphTwoDbuildPageEmpty26}
\pastebutton{ugGraphTwoDbuildPageFull26}{\hidepaste}
\tab{5}\spadcommand{g := graphImage()$GRIMAGE\bound{Sg }\free{clearAll }}
\indentrel{3}\begin{verbatim}
   (1)  Graph with 0 point lists
```

```
                                       Type: GraphImage
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPageEmpty26}
\begin{paste}{ugGraphTwoDbuildPageEmpty26}{ugGraphTwoDbuildPagePatch26}
\pastebutton{ugGraphTwoDbuildPageEmpty26}{\showpaste}
\tab{5}\spadcommand{g := graphImage()$GRIMAGE\bound{Sg }\free{clearAll }}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPagePatch27}
\begin{paste}{ugGraphTwoDbuildPageFull27}{ugGraphTwoDbuildPageEmpty27}
\pastebutton{ugGraphTwoDbuildPageFull27}{\hidepaste}
\tab{5}\spadcommand{p1 := point [0,0]$(Point DFLOAT)\bound{Sp1 }}
\indentrel{3}\begin{verbatim}
   (2)  [0.0,0.0]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPageEmpty27}
\begin{paste}{ugGraphTwoDbuildPageEmpty27}{ugGraphTwoDbuildPagePatch27}
\pastebutton{ugGraphTwoDbuildPageEmpty27}{\showpaste}
\tab{5}\spadcommand{p1 := point [0,0]$(Point DFLOAT)\bound{Sp1 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPagePatch28}
\begin{paste}{ugGraphTwoDbuildPageFull28}{ugGraphTwoDbuildPageEmpty28}
\pastebutton{ugGraphTwoDbuildPageFull28}{\hidepaste}
\tab{5}\spadcommand{p2 := point [.25,.25]$(Point DFLOAT)\bound{Sp2 }}
\indentrel{3}\begin{verbatim}
   (3)  [0.25,0.25]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPageEmpty28}
\begin{paste}{ugGraphTwoDbuildPageEmpty28}{ugGraphTwoDbuildPagePatch28}
\pastebutton{ugGraphTwoDbuildPageEmpty28}{\showpaste}
\tab{5}\spadcommand{p2 := point [.25,.25]$(Point DFLOAT)\bound{Sp2 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPagePatch29}
\begin{paste}{ugGraphTwoDbuildPageFull29}{ugGraphTwoDbuildPageEmpty29}
\pastebutton{ugGraphTwoDbuildPageFull29}{\hidepaste}
\tab{5}\spadcommand{p3 := point [.5,.5]$(Point DFLOAT)\bound{Sp3 }}
\indentrel{3}\begin{verbatim}
   (4)  [0.5,0.5]
                                Type: Point DoubleFloat
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPageEmpty29}
\begin{paste}{ugGraphTwoDbuildPageEmpty29}{ugGraphTwoDbuildPagePatch29}
\pastebutton{ugGraphTwoDbuildPageEmpty29}{\showpaste}
\tab{5}\spadcommand{p3 := point [.5,.5]$(Point DFLOAT)\bound{Sp3 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPagePatch30}
\begin{paste}{ugGraphTwoDbuildPageFull30}{ugGraphTwoDbuildPageEmpty30}
\pastebutton{ugGraphTwoDbuildPageFull30}{\hidepaste}
\tab{5}\spadcommand{p4 := point [.75,.75]$(Point DFLOAT)\bound{Sp4 }}
\indentrel{3}\begin{verbatim}
   (5)  [0.75,0.75]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPageEmpty30}
\begin{paste}{ugGraphTwoDbuildPageEmpty30}{ugGraphTwoDbuildPagePatch30}
\pastebutton{ugGraphTwoDbuildPageEmpty30}{\showpaste}
\tab{5}\spadcommand{p4 := point [.75,.75]$(Point DFLOAT)\bound{Sp4 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPagePatch31}
\begin{paste}{ugGraphTwoDbuildPageFull31}{ugGraphTwoDbuildPageEmpty31}
\pastebutton{ugGraphTwoDbuildPageFull31}{\hidepaste}
\tab{5}\spadcommand{p5 := point [1,1]$(Point DFLOAT)\bound{Sp5 }}
\indentrel{3}\begin{verbatim}
   (6)  [1.0,1.0]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPageEmpty31}
\begin{paste}{ugGraphTwoDbuildPageEmpty31}{ugGraphTwoDbuildPagePatch31}
\pastebutton{ugGraphTwoDbuildPageEmpty31}{\showpaste}
\tab{5}\spadcommand{p5 := point [1,1]$(Point DFLOAT)\bound{Sp5 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPagePatch32}
\begin{paste}{ugGraphTwoDbuildPageFull32}{ugGraphTwoDbuildPageEmpty32}
\pastebutton{ugGraphTwoDbuildPageFull32}{\hidepaste}
\tab{5}\spadcommand{component(g,p1)$GRIMAGE\free{Sg Sp1 }\bound{gp1 }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPageEmpty32}
```

```
\begin{paste}{ugGraphTwoDbuildPageEmpty32}{ugGraphTwoDbuildPagePatch32}
\pastebutton{ugGraphTwoDbuildPageEmpty32}{\showpaste}
\tab{5}\spadcommand{component(g,p1)$GRIMAGE\free{Sg Sp1 }\bound{gp1 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPagePatch33}
\begin{paste}{ugGraphTwoDbuildPageFull33}{ugGraphTwoDbuildPageEmpty33}
\pastebutton{ugGraphTwoDbuildPageFull33}{\hidepaste}
\tab{5}\spadcommand{component(g,p2)$GRIMAGE\free{Sg Sp2 }\bound{gp2 }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPageEmpty33}
\begin{paste}{ugGraphTwoDbuildPageEmpty33}{ugGraphTwoDbuildPagePatch33}
\pastebutton{ugGraphTwoDbuildPageEmpty33}{\showpaste}
\tab{5}\spadcommand{component(g,p2)$GRIMAGE\free{Sg Sp2 }\bound{gp2 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPagePatch34}
\begin{paste}{ugGraphTwoDbuildPageFull34}{ugGraphTwoDbuildPageEmpty34}
\pastebutton{ugGraphTwoDbuildPageFull34}{\hidepaste}
\tab{5}\spadcommand{appendPoint(g,p3)$GRIMAGE\free{gp1 gp2 Sp3 }\bound{gp3 }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPageEmpty34}
\begin{paste}{ugGraphTwoDbuildPageEmpty34}{ugGraphTwoDbuildPagePatch34}
\pastebutton{ugGraphTwoDbuildPageEmpty34}{\showpaste}
\tab{5}\spadcommand{appendPoint(g,p3)$GRIMAGE\free{gp1 gp2 Sp3 }\bound{gp3 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPagePatch35}
\begin{paste}{ugGraphTwoDbuildPageFull35}{ugGraphTwoDbuildPageEmpty35}
\pastebutton{ugGraphTwoDbuildPageFull35}{\hidepaste}
\tab{5}\spadcommand{appendPoint(g,p4)$GRIMAGE\free{gp3 Sp4 }\bound{gp4 }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPageEmpty35}
\begin{paste}{ugGraphTwoDbuildPageEmpty35}{ugGraphTwoDbuildPagePatch35}
\pastebutton{ugGraphTwoDbuildPageEmpty35}{\showpaste}
\tab{5}\spadcommand{appendPoint(g,p4)$GRIMAGE\free{gp3 Sp4 }\bound{gp4 }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugGraphTwoDbuildPagePatch36}
\begin{paste}{ugGraphTwoDbuildPageFull36}{ugGraphTwoDbuildPageEmpty36}
\pastebutton{ugGraphTwoDbuildPageFull36}{\hidepaste}
\tab{5}\spadcommand{appendPoint(g,p5)$GRIMAGE\free{gp4 Sp5 }\bound{gp5 }}
\indentrel{3}\begin{verbatim}
                                           Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPageEmpty36}
\begin{paste}{ugGraphTwoDbuildPageEmpty36}{ugGraphTwoDbuildPagePatch36}
\pastebutton{ugGraphTwoDbuildPageEmpty36}{\showpaste}
\tab{5}\spadcommand{appendPoint(g,p5)$GRIMAGE\free{gp4 Sp5 }\bound{gp5 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPagePatch37}
\begin{paste}{ugGraphTwoDbuildPageFull37}{ugGraphTwoDbuildPageEmpty37}
\pastebutton{ugGraphTwoDbuildPageFull37}{\hidepaste}
\tab{5}\spadcommand{g1 := makeGraphImage(g)$GRIMAGE\bound{Sg1 }\free{gp5 }}
\indentrel{3}\begin{verbatim}
   (12)  Graph with 2 point lists
                                      Type: GraphImage
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPageEmpty37}
\begin{paste}{ugGraphTwoDbuildPageEmpty37}{ugGraphTwoDbuildPagePatch37}
\pastebutton{ugGraphTwoDbuildPageEmpty37}{\showpaste}
\tab{5}\spadcommand{g1 := makeGraphImage(g)$GRIMAGE\bound{Sg1 }\free{gp5 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPagePatch38}
\begin{paste}{ugGraphTwoDbuildPageFull38}{ugGraphTwoDbuildPageEmpty38}
\pastebutton{ugGraphTwoDbuildPageFull38}{\hidepaste}
\tab{5}\spadgraph{makeViewport2D(g1,[title("Graph Points")])$VIEW2D\free{Sg1 }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphtwodbuildpage38.view/image}}{viewalone\spa
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPageEmpty38}
\begin{paste}{ugGraphTwoDbuildPageEmpty38}{ugGraphTwoDbuildPagePatch38}
\pastebutton{ugGraphTwoDbuildPageEmpty38}{\showpaste}
\tab{5}\spadgraph{makeViewport2D(g1,[title("Graph Points")])$VIEW2D\free{Sg1 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPagePatch39}
\begin{paste}{ugGraphTwoDbuildPageFull39}{ugGraphTwoDbuildPageEmpty39}
\pastebutton{ugGraphTwoDbuildPageFull39}{\hidepaste}
\tab{5}\spadcommand{g2 := coerce([[p1],[p2],[p3],[p4],[p5]])$GRIMAGE\free{Sp1 Sp2 Sp3 Sp4 Sp5 }\bound{Sg2 }}
\indentrel{3}\begin{verbatim}
   (14)  Graph with 5 point lists
```

```
                                         Type: GraphImage
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPageEmpty39}
\begin{paste}{ugGraphTwoDbuildPageEmpty39}{ugGraphTwoDbuildPagePatch39}
\pastebutton{ugGraphTwoDbuildPageEmpty39}{\showpaste}
\tab{5}\spadcommand{g2 := coerce([[p1],[p2],[p3],[p4],[p5]])$GRIMAGE\free{Sp1 Sp2 Sp3 Sp4 Sp5 }\
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPagePatch40}
\begin{paste}{ugGraphTwoDbuildPageFull40}{ugGraphTwoDbuildPageEmpty40}
\pastebutton{ugGraphTwoDbuildPageFull40}{\hidepaste}
\tab{5}\spadcommand{v := viewport2D()$VIEW2D\bound{Sv }}
\indentrel{3}\begin{verbatim}
   (15)
   Closed or Undefined TwoDimensionalViewport: "AXIOM2D"
                           Type: TwoDimensionalViewport
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPageEmpty40}
\begin{paste}{ugGraphTwoDbuildPageEmpty40}{ugGraphTwoDbuildPagePatch40}
\pastebutton{ugGraphTwoDbuildPageEmpty40}{\showpaste}
\tab{5}\spadcommand{v := viewport2D()$VIEW2D\bound{Sv }}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPagePatch41}
\begin{paste}{ugGraphTwoDbuildPageFull41}{ugGraphTwoDbuildPageEmpty41}
\pastebutton{ugGraphTwoDbuildPageFull41}{\hidepaste}
\tab{5}\spadcommand{options(v,[title("Just Points")])$VIEW2D\free{Sv }\bound{Svo }}
\indentrel{3}\begin{verbatim}
   (16)
   Closed or Undefined TwoDimensionalViewport: "AXIOM2D"
                           Type: TwoDimensionalViewport
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPageEmpty41}
\begin{paste}{ugGraphTwoDbuildPageEmpty41}{ugGraphTwoDbuildPagePatch41}
\pastebutton{ugGraphTwoDbuildPageEmpty41}{\showpaste}
\tab{5}\spadcommand{options(v,[title("Just Points")])$VIEW2D\free{Sv }\bound{Svo }}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPagePatch42}
\begin{paste}{ugGraphTwoDbuildPageFull42}{ugGraphTwoDbuildPageEmpty42}
\pastebutton{ugGraphTwoDbuildPageFull42}{\hidepaste}
\tab{5}\spadcommand{putGraph(v,g2,1)$VIEW2D\free{Sg2 Svo }\bound{Svog2 }}
\indentrel{3}\begin{verbatim}
                                               Type: Void
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPageEmpty42}
\begin{paste}{ugGraphTwoDbuildPageEmpty42}{ugGraphTwoDbuildPagePatch42}
\pastebutton{ugGraphTwoDbuildPageEmpty42}{\showpaste}
\tab{5}\spadcommand{putGraph(v,g2,1)$VIEW2D\free{Sg2 Svo }\bound{Svog2 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPagePatch43}
\begin{paste}{ugGraphTwoDbuildPageFull43}{ugGraphTwoDbuildPageEmpty43}
\pastebutton{ugGraphTwoDbuildPageFull43}{\hidepaste}
\tab{5}\spadgraph{makeViewport2D(v)$VIEW2D\free{Svog2 }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphtwodbuildpage43.view/image}}{viewalone\spa
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDbuildPageEmpty43}
\begin{paste}{ugGraphTwoDbuildPageEmpty43}{ugGraphTwoDbuildPagePatch43}
\pastebutton{ugGraphTwoDbuildPageEmpty43}{\showpaste}
\tab{5}\spadgraph{makeViewport2D(v)$VIEW2D\free{Svog2 }}
\end{paste}\end{patch}
```

———-

Addendum: Appending a Graph to a Viewport Window Containing a Graph

— ug07.ht —

```
\begin{page}{ugGraphTwoDappendPage}
{7.1.10. Addendum: Appending a Graph to a Viewport
Window Containing a Graph}
\beginscroll

This section demonstrates how to append a \twodim{} graph to a
viewport already containing other graphs.  The default \axiomFun{draw}
command places a graph into the first \axiomType{GraphImage} slot
position of the \axiomType{TwoDimensionalViewport}.

\xtc{
This graph is in the first slot in its viewport.
}{
\spadpaste{v1 := draw(sin(x),x=0..2*\%pi) \bound{v1}}
}
\xtc{
So is this graph.
```

```
}{
\spadpaste{v2 := draw(cos(x),x=0..2*\%pi, curveColor==light red())
\bound{v2}}
}
\xtc{
The operation \axiomFunFrom{getGraph}{TwoDimensionalViewport}
retrieves the \axiomType{GraphImage} \axiom{g1} from the first slot
position in the viewport \axiom{v1}.
}{
\spadpaste{g1 := getGraph(v1,1) \bound{g1}\free{v1}}
}
\xtc{
Now \axiomFunFrom{putGraph}{TwoDimensionalViewport}
places \axiom{g1} into the the second slot position of \axiom{v2}.
}{
\spadpaste{putGraph(v2,g1,2) \bound{v22}\free{g1 v2}}
}
\psXtc{
Display the new \axiomType{TwoDimensionalViewport} containing both graphs.
}{
\graphpaste{makeViewport2D(v2) \free{v22}}
}{
%
}
%
%See Figure #.#.
%

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugGraphTwoDappendPagePatch1}
\begin{paste}{ugGraphTwoDappendPageFull1}{ugGraphTwoDappendPageEmpty1}
\pastebutton{ugGraphTwoDappendPageFull1}{\hidepaste}
\tab{5}\spadcommand{v1 := draw(sin(x),x=0..2*\%pi)\bound{v1 }}
\indentrel{3}\begin{verbatim}
   (1)  TwoDimensionalViewport: "sin x"
                              Type: TwoDimensionalViewport
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDappendPageEmpty1}
\begin{paste}{ugGraphTwoDappendPageEmpty1}{ugGraphTwoDappendPagePatch1}
\pastebutton{ugGraphTwoDappendPageEmpty1}{\showpaste}
\tab{5}\spadcommand{v1 := draw(sin(x),x=0..2*\%pi)\bound{v1 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDappendPagePatch2}
\begin{paste}{ugGraphTwoDappendPageFull2}{ugGraphTwoDappendPageEmpty2}
```

```
\pastebutton{ugGraphTwoDappendPageFull2}{\hidepaste}
\tab{5}\spadcommand{v2 := draw(cos(x),x=0..2*\%pi, curveColor==light red())\bound{v2 }}
\indentrel{3}\begin{verbatim}
   (2)  TwoDimensionalViewport: "cos x"
                             Type: TwoDimensionalViewport
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDappendPageEmpty2}
\begin{paste}{ugGraphTwoDappendPageEmpty2}{ugGraphTwoDappendPagePatch2}
\pastebutton{ugGraphTwoDappendPageEmpty2}{\showpaste}
\tab{5}\spadcommand{v2 := draw(cos(x),x=0..2*\%pi, curveColor==light red())\bound{v2 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDappendPagePatch3}
\begin{paste}{ugGraphTwoDappendPageFull3}{ugGraphTwoDappendPageEmpty3}
\pastebutton{ugGraphTwoDappendPageFull3}{\hidepaste}
\tab{5}\spadcommand{g1 := getGraph(v1,1)\bound{g1 }\free{v1 }}
\indentrel{3}\begin{verbatim}
   (3)  Graph with 1 point list
                                         Type: GraphImage
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDappendPageEmpty3}
\begin{paste}{ugGraphTwoDappendPageEmpty3}{ugGraphTwoDappendPagePatch3}
\pastebutton{ugGraphTwoDappendPageEmpty3}{\showpaste}
\tab{5}\spadcommand{g1 := getGraph(v1,1)\bound{g1 }\free{v1 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDappendPagePatch4}
\begin{paste}{ugGraphTwoDappendPageFull4}{ugGraphTwoDappendPageEmpty4}
\pastebutton{ugGraphTwoDappendPageFull4}{\hidepaste}
\tab{5}\spadcommand{putGraph(v2,g1,2)\bound{v22 }\free{g1 v2 }}
\indentrel{3}\begin{verbatim}
                                               Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDappendPageEmpty4}
\begin{paste}{ugGraphTwoDappendPageEmpty4}{ugGraphTwoDappendPagePatch4}
\pastebutton{ugGraphTwoDappendPageEmpty4}{\showpaste}
\tab{5}\spadcommand{putGraph(v2,g1,2)\bound{v22 }\free{g1 v2 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDappendPagePatch5}
\begin{paste}{ugGraphTwoDappendPageFull5}{ugGraphTwoDappendPageEmpty5}
\pastebutton{ugGraphTwoDappendPageFull5}{\hidepaste}
\tab{5}\spadgraph{makeViewport2D(v2)\free{v22 }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphtwodappendpage5.view/image}}{viewalone\spa
```

```
\end{paste}\end{patch}

\begin{patch}{ugGraphTwoDappendPageEmpty5}
\begin{paste}{ugGraphTwoDappendPageEmpty5}{ugGraphTwoDappendPagePatch5}
\pastebutton{ugGraphTwoDappendPageEmpty5}{\showpaste}
\tab{5}\spadgraph{makeViewport2D(v2)\free{v22 }}
\end{paste}\end{patch}
```

———-

Three-Dimensional Graphics

— **ug07.ht** —

```
\begin{page}{ugGraphThreeDPage}{7.2. Three-Dimensional Graphics}
\beginscroll
%
The Axiom \threedim{} graphics package provides the ability to
%
\indent{4}
\beginitems
%
\item[-] generate surfaces defined by a function of two real variables
%
\item[-] generate space curves and tubes defined by parametric equations
%
\item[-] generate surfaces defined by parametric equations
\enditems
\indent{0}
These graphs can be modified by using various options, such as calculating
points in the spherical coordinate system or changing the polygon grid size
of a surface.

\beginmenu
    \menudownlink{
```

```
{7.2.1. Plotting Three-Dimensional Functions of Two Variables}}
{ugGraphThreeDPlotPage}
    \menudownlink{
{7.2.2. Plotting Three-Dimensional Parametric Space Curves}}
{ugGraphThreeDParmPage}
    \menudownlink{
{7.2.3. Plotting Three-Dimensional Parametric Surfaces}}
{ugGraphThreeDParPage}
    \menudownlink{
{7.2.4. Three-Dimensional Options}}{ugGraphThreeDOptionsPage}
    \menudownlink{
{7.2.5. The makeObject Command}}{ugGraphMakeObjectPage}
    \menudownlink{
{7.2.6. Building Three-Dimensional Objects From Primitives}}
{ugGraphThreeDBuildPage}
    \menudownlink{
{7.2.7. Coordinate System Transformations}}{ugGraphCoordPage}
    \menudownlink{
{7.2.8. Three-Dimensional Clipping}}{ugGraphClipPage}
    \menudownlink{
{7.2.9. Three-Dimensional Control-Panel}}
{ugGraphThreeDControlPage}
    \menudownlink{
{7.2.10. Operations for Three-Dimensional Graphics}}
{ugGraphThreeDopsPage}
    \menudownlink{
{7.2.11. Customization using .Xdefaults}}{ugXdefaultsPage}
\endmenu
\endscroll
\autobuttons
\end{page}
```

———-

Plotting Three-Dimensional Functions of Two Variables

⇒ “notitle” (ugGraphThreeDOptionsPage) 11 on page 2028

— **ug07.ht** —

```
\begin{page}{ugGraphThreeDPlotPage}
{7.2.1. Plotting Three-Dimensional Functions of Two Variables}
\beginscroll

The simplest \threedim{} graph is that of a surface defined by a function
of two variables, \axiom{z = f(x,y)}.
```

```
\beginImportant

The general format for drawing a surface defined by a formula
\axiom{f(x,y)} of two variables \axiom{x} and \axiom{y} is:

\centerline{{{\tt draw(f(x,y), x = a..b, y = c..d, {\it options})}}}
where \axiom{a..b} and \axiom{c..d} define the range of \axiom{x}
and \axiom{y}, and where {\it options} prescribes zero or more
options as described in
\downlink{``Three-Dimensional Options''}{ugGraphThreeDOptionsPage}
in Section 7.2.4\ignore{ugGraphThreeDOptions}.
An example of an option is \axiom{title == "Title of Graph".}
An alternative format involving a function \axiom{f} is also
available.
\endImportant

%
\psXtc{
The simplest way to plot a function of two variables is to use a formula.
With formulas you always precede the range specifications with
the variable name and an \spadSyntax{=} sign.
}{
\graphpaste{draw(cos(x*y),x=-3..3,y=-3..3)}
}{
\epsffile[0 0 295 295]{../ps/3d2vara.ps}
}
%
\xtc{
If you intend to use a function more than once,
or it is long and complex, then first
give its definition to Axiom.
}{
\spadpaste{f(x,y) == sin(x)*cos(y) \bound{f}}
}
%
%
\psXtc{
To draw the function, just give its name and drop the variables
from the range specifications.
Axiom compiles your function for efficient computation
of data for the graph.
Notice that Axiom uses the text of your function as a
default title.
}{
\graphpaste{draw(f,-\%pi..\%pi,-\%pi..\%pi) \free{f}}
}{
\epsffile[0 0 295 295]{../ps/3d2varb.ps}
}

\endscroll
```

```
\autobuttons
\end{page}

\begin{patch}{ugGraphThreeDPlotPagePatch1}
\begin{paste}{ugGraphThreeDPlotPageFull1}{ugGraphThreeDPlotPageEmpty1}
\pastebutton{ugGraphThreeDPlotPageFull1}{\hidepaste}
\tab{5}\spadgraph{draw(cos(x*y),x=-3..3,y=-3..3)}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphthreedplotpage1.view/image}}{viewalone\spa
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDPlotPageEmpty1}
\begin{paste}{ugGraphThreeDPlotPageEmpty1}{ugGraphThreeDPlotPagePatch1}
\pastebutton{ugGraphThreeDPlotPageEmpty1}{\showpaste}
\tab{5}\spadgraph{draw(cos(x*y),x=-3..3,y=-3..3)}
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDPlotPagePatch2}
\begin{paste}{ugGraphThreeDPlotPageFull2}{ugGraphThreeDPlotPageEmpty2}
\pastebutton{ugGraphThreeDPlotPageFull2}{\hidepaste}
\tab{5}\spadcommand{f(x,y) == sin(x)*cos(y)\bound{f }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDPlotPageEmpty2}
\begin{paste}{ugGraphThreeDPlotPageEmpty2}{ugGraphThreeDPlotPagePatch2}
\pastebutton{ugGraphThreeDPlotPageEmpty2}{\showpaste}
\tab{5}\spadcommand{f(x,y) == sin(x)*cos(y)\bound{f }}
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDPlotPagePatch3}
\begin{paste}{ugGraphThreeDPlotPageFull3}{ugGraphThreeDPlotPageEmpty3}
\pastebutton{ugGraphThreeDPlotPageFull3}{\hidepaste}
\tab{5}\spadgraph{draw(f,-\%pi..\%pi,-\%pi..\%pi)\free{f }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphthreedplotpage3.view/image}}{viewalone\spa
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDPlotPageEmpty3}
\begin{paste}{ugGraphThreeDPlotPageEmpty3}{ugGraphThreeDPlotPagePatch3}
\pastebutton{ugGraphThreeDPlotPageEmpty3}{\showpaste}
\tab{5}\spadgraph{draw(f,-\%pi..\%pi,-\%pi..\%pi)\free{f }}
\end{paste}\end{patch}
```

Plotting Three-Dimensional Parametric Space Curves

⇒ “notitle” (ugGraphThreeDOptionsPage) 11 on page 2028

— **ug07.ht** —

```
\begin{page}{ugGraphThreeDParmPage}
{7.2.2. Plotting Three-Dimensional Parametric Space Curves}
\beginscroll

A second kind of \threedim{} graph is a \threedim{} space curve
defined by the parametric equations for \axiom{x(t)}, \axiom{y(t)},
and \axiom{z(t)} as a function of an independent variable \axiom{t}.

%
\beginImportant
The general format for drawing a \threedim{} space curve defined by
parametric formulas \axiom{x = f(t)}, \axiom{y = g(t)}, and
\axiom{z = h(t)} is:
%
\centerline{{{\tt draw(curve(f(t),g(t),h(t)), t = a..b, {\it options})}}}
where \axiom{a..b} defines the range of the independent variable
\axiom{t}, and where {\it options} prescribes zero or more options
as described in
\downlink{‘‘Three-Dimensional Options’’}{ugGraphThreeDOptionsPage}
in Section 7.2.4\ignore{ugGraphThreeDOptions}.
An example of an option is \axiom{title == "Title of Graph".}
An alternative format involving functions \axiom{f}, \axiom{g} and
\axiom{h} is also available.
\endImportant

%
\psXtc{
If you use explicit formulas to draw a space curve, always precede
the range specification with the variable name and an
\spadSyntax{=} sign.
}{
\graphpaste{draw(curve(5*cos(t), 5*sin(t),t), t=-12..12)}
}{
\epsffile[0 0 295 295]{../ps/3dpsca.ps}
}
%
\xtc{
Alternatively, you can draw space curves by referring to functions.
}{
\spadpaste{i1(t:DFLOAT):DFLOAT == sin(t)*cos(3*t/5) \bound{i1}}
}
\xtc{
This is useful if the functions are to be used more than once \ldots
}{
```

```
\spadpaste{i2(t:DFLOAT):DFLOAT == cos(t)*cos(3*t/5) \bound{i2}}
}
\xtc{
or if the functions are long and complex.
}{
\spadpaste{i3(t:DFLOAT):DFLOAT == cos(t)*sin(3*t/5) \bound{i3}}
}
%
%
\psXtc{
Give the names of the functions and
drop the variable name specification in the second argument.
Again, Axiom supplies a default title.
}{
\graphpaste{draw(curve(i1,i2,i3),0..15*\%pi) \free{i1 i2 i3}}
}{
\epsffile[0 0 295 295]{../ps/3dpscb.ps}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugGraphThreeDParmPagePatch1}
\begin{paste}{ugGraphThreeDParmPageFull1}{ugGraphThreeDParmPageEmpty1}
\pastebutton{ugGraphThreeDParmPageFull1}{\hidepaste}
\tab{5}\spadgraph{draw(curve(5*cos(t), 5*sin(t),t), t=-12..12)}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphthreedparmpage1.view/image}}{viewalone\spa
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDParmPageEmpty1}
\begin{paste}{ugGraphThreeDParmPageEmpty1}{ugGraphThreeDParmPagePatch1}
\pastebutton{ugGraphThreeDParmPageEmpty1}{\showpaste}
\tab{5}\spadgraph{draw(curve(5*cos(t), 5*sin(t),t), t=-12..12)}
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDParmPagePatch2}
\begin{paste}{ugGraphThreeDParmPageFull2}{ugGraphThreeDParmPageEmpty2}
\pastebutton{ugGraphThreeDParmPageFull2}{\hidepaste}
\tab{5}\spadcommand{i1(t:DFLOAT):DFLOAT == sin(t)*cos(3*t/5)\bound{i1 }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDParmPageEmpty2}
\begin{paste}{ugGraphThreeDParmPageEmpty2}{ugGraphThreeDParmPagePatch2}
\pastebutton{ugGraphThreeDParmPageEmpty2}{\showpaste}
\tab{5}\spadcommand{i1(t:DFLOAT):DFLOAT == sin(t)*cos(3*t/5)\bound{i1 }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugGraphThreeDParmPagePatch3}
\begin{paste}{ugGraphThreeDParmPageFull3}{ugGraphThreeDParmPageEmpty3}
\pastebutton{ugGraphThreeDParmPageFull3}{\hidepaste}
\tab{5}\spadcommand{i2(t:DFLOAT):DFLOAT == cos(t)*cos(3*t/5)\bound{i2 }}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDParmPageEmpty3}
\begin{paste}{ugGraphThreeDParmPageEmpty3}{ugGraphThreeDParmPagePatch3}
\pastebutton{ugGraphThreeDParmPageEmpty3}{\showpaste}
\tab{5}\spadcommand{i2(t:DFLOAT):DFLOAT == cos(t)*cos(3*t/5)\bound{i2 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDParmPagePatch4}
\begin{paste}{ugGraphThreeDParmPageFull4}{ugGraphThreeDParmPageEmpty4}
\pastebutton{ugGraphThreeDParmPageFull4}{\hidepaste}
\tab{5}\spadcommand{i3(t:DFLOAT):DFLOAT == cos(t)*sin(3*t/5)\bound{i3 }}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDParmPageEmpty4}
\begin{paste}{ugGraphThreeDParmPageEmpty4}{ugGraphThreeDParmPagePatch4}
\pastebutton{ugGraphThreeDParmPageEmpty4}{\showpaste}
\tab{5}\spadcommand{i3(t:DFLOAT):DFLOAT == cos(t)*sin(3*t/5)\bound{i3 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDParmPagePatch5}
\begin{paste}{ugGraphThreeDParmPageFull5}{ugGraphThreeDParmPageEmpty5}
\pastebutton{ugGraphThreeDParmPageFull5}{\hidepaste}
\tab{5}\spadgraph{draw(curve(i1,i2,i3),0..15*\%pi)\free{i1 i2 i3 }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphthreedparmpage5.view/image}}{v
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDParmPageEmpty5}
\begin{paste}{ugGraphThreeDParmPageEmpty5}{ugGraphThreeDParmPagePatch5}
\pastebutton{ugGraphThreeDParmPageEmpty5}{\showpaste}
\tab{5}\spadgraph{draw(curve(i1,i2,i3),0..15*\%pi)\free{i1 i2 i3 }}
\end{paste}\end{patch}
```

Plotting 3D Parametric Surfaces

⇒ "notitle" (ugGraphThreeDOptionsPage) 11 on page 2028
⇒ "notitle" (ugGraphCoordPage) 11 on page 2053

— **ug07.ht** —

```
\begin{page}{ugGraphThreeDParPage}
{7.2.3. Plotting 3D Parametric Surfaces}
\beginscroll

A third kind of \threedim{} graph is a surface defined by
parametric equations for \axiom{x(u,v)}, \axiom{y(u,v)}, and
\axiom{z(u,v)} of two independent variables \axiom{u} and \axiom{v}.

%
\beginImportant
The general format for drawing a \threedim{} graph defined by
parametric formulas \axiom{x = f(u,v)}, \axiom{y = g(u,v)},
and \axiom{z = h(u,v)} is:
%
\centerline{{{\tt draw(surface(f(u,v),g(u,v),h(u,v)),
u = a..b, v = c..d, {\it options})}}}
where \axiom{a..b} and \axiom{c..d} define the range of the
independent variables \axiom{u} and \axiom{v}, and where
{\it options} prescribes zero or more options as described in
\downlink{``Three-Dimensional Options''}{ugGraphThreeDOptionsPage}
in Section 7.2.4\ignore{ugGraphThreeDOptions}.
An example of an option is \axiom{title == "Title of Graph".}
An alternative format involving functions \axiom{f}, \axiom{g} and
\axiom{h} is also available.
\endImportant

%
\psXtc{
This example draws a graph of a surface plotted using the
parabolic cylindrical coordinate system option.
The values of the functions supplied to \axiomFun{surface} are
interpreted in coordinates as given by a {\tt coordinates} option,
here as parabolic cylindrical coordinates (see
\downlink{``Coordinate System Transformations''}{ugGraphCoordPage}
in Section 7.2.7\ignore{ugGraphCoord}).
}{
\graphpaste{draw(surface(u*cos(v), u*sin(v), v*cos(u)),
u=-4..4, v=0..\%pi, coordinates== parabolicCylindrical)}
}{
\epsffile[0 0 295 295]{../ps/3dpsa.ps}
}
%
```

```
Again, you can graph these parametric surfaces using functions,
if the functions are long and complex.
\xtc{
Here we declare the types of arguments and values to be of type
\axiomType{DoubleFloat}.
}{
\spadpaste{n1(u:DFLOAT,v:DFLOAT):DFLOAT == u*cos(v) \bound{n1}}
}
\xtc{
As shown by previous examples, these declarations are necessary.
}{
\spadpaste{n2(u:DFLOAT,v:DFLOAT):DFLOAT == u*sin(v) \bound{n2}}
}
\xtc{
In either case, Axiom compiles the functions
when needed to graph a result.
}{
\spadpaste{n3(u:DFLOAT,v:DFLOAT):DFLOAT == u \bound{n3}}
}
\xtc{
Without these declarations, you have to suffix floats
with \axiom{@DFLOAT} to get a \axiomType{DoubleFloat} result.
However, a call here with an unadorned float
produces a \axiomType{DoubleFloat}.
}{
\spadpaste{n3(0.5,1.0)\free{n3}}
}
%
%
\psXtc{
Draw the surface by referencing the function names, this time
choosing the toroidal coordinate system.
}{
\graphpaste{draw(surface(n1,n2,n3), 1..4, 1..2*\%pi,
coordinates == toroidal(1\$DFLOAT)) \free{n1 n2 n3}}
}{
\epsffile[0 0 295 295]{../ps/3dpsb.ps}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugGraphThreeDParPagePatch1}
\begin{paste}{ugGraphThreeDParPageFull1}{ugGraphThreeDParPageEmpty1}
\pastebutton{ugGraphThreeDParPageFull1}{\hidepaste}
\tab{5}\spadgraph{draw(surface(u*cos(v), u*sin(v), v*cos(u)), u=-4..4, v=0..\%pi, coordinates==
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphthreedparpage1.view/image}}{vi
\end{paste}\end{patch}
```

```
\begin{patch}{ugGraphThreeDParPageEmpty1}
\begin{paste}{ugGraphThreeDParPageEmpty1}{ugGraphThreeDParPagePatch1}
\pastebutton{ugGraphThreeDParPageEmpty1}{\showpaste}
\tab{5}\spadgraph{draw(surface(u*cos(v), u*sin(v), v*cos(u)), u=-4..4, v=0..\%pi, coordinates== parabolicCyl
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDParPagePatch2}
\begin{paste}{ugGraphThreeDParPageFull2}{ugGraphThreeDParPageEmpty2}
\pastebutton{ugGraphThreeDParPageFull2}{\hidepaste}
\tab{5}\spadcommand{n1(u:DFLOAT,v:DFLOAT):DFLOAT == u*cos(v)\bound{n1 }}
\indentrel{3}\begin{verbatim}
                                        Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDParPageEmpty2}
\begin{paste}{ugGraphThreeDParPageEmpty2}{ugGraphThreeDParPagePatch2}
\pastebutton{ugGraphThreeDParPageEmpty2}{\showpaste}
\tab{5}\spadcommand{n1(u:DFLOAT,v:DFLOAT):DFLOAT == u*cos(v)\bound{n1 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDParPagePatch3}
\begin{paste}{ugGraphThreeDParPageFull3}{ugGraphThreeDParPageEmpty3}
\pastebutton{ugGraphThreeDParPageFull3}{\hidepaste}
\tab{5}\spadcommand{n2(u:DFLOAT,v:DFLOAT):DFLOAT == u*sin(v)\bound{n2 }}
\indentrel{3}\begin{verbatim}
                                        Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDParPageEmpty3}
\begin{paste}{ugGraphThreeDParPageEmpty3}{ugGraphThreeDParPagePatch3}
\pastebutton{ugGraphThreeDParPageEmpty3}{\showpaste}
\tab{5}\spadcommand{n2(u:DFLOAT,v:DFLOAT):DFLOAT == u*sin(v)\bound{n2 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDParPagePatch4}
\begin{paste}{ugGraphThreeDParPageFull4}{ugGraphThreeDParPageEmpty4}
\pastebutton{ugGraphThreeDParPageFull4}{\hidepaste}
\tab{5}\spadcommand{n3(u:DFLOAT,v:DFLOAT):DFLOAT == u\bound{n3 }}
\indentrel{3}\begin{verbatim}
                                        Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDParPageEmpty4}
\begin{paste}{ugGraphThreeDParPageEmpty4}{ugGraphThreeDParPagePatch4}
\pastebutton{ugGraphThreeDParPageEmpty4}{\showpaste}
\tab{5}\spadcommand{n3(u:DFLOAT,v:DFLOAT):DFLOAT == u\bound{n3 }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugGraphThreeDParPagePatch5}
\begin{paste}{ugGraphThreeDParPageFull5}{ugGraphThreeDParPageEmpty5}
\pastebutton{ugGraphThreeDParPageFull5}{\hidepaste}
\tab{5}\spadcommand{n3(0.5,1.0)\free{n3 }}
\indentrel{3}\begin{verbatim}
   (5)  0.5
                                      Type: DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDParPageEmpty5}
\begin{paste}{ugGraphThreeDParPageEmpty5}{ugGraphThreeDParPagePatch5}
\pastebutton{ugGraphThreeDParPageEmpty5}{\showpaste}
\tab{5}\spadcommand{n3(0.5,1.0)\free{n3 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDParPagePatch6}
\begin{paste}{ugGraphThreeDParPageFull6}{ugGraphThreeDParPageEmpty6}
\pastebutton{ugGraphThreeDParPageFull6}{\hidepaste}
\tab{5}\spadgraph{draw(surface(n1,n2,n3), 1..4, 1..2*\%pi, coordinates == toroidal(1$DFLOAT))\fr
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphthreedparpage6.view/image}}{vi
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDParPageEmpty6}
\begin{paste}{ugGraphThreeDParPageEmpty6}{ugGraphThreeDParPagePatch6}
\pastebutton{ugGraphThreeDParPageEmpty6}{\showpaste}
\tab{5}\spadgraph{draw(surface(n1,n2,n3), 1..4, 1..2*\%pi, coordinates == toroidal(1$DFLOAT))\fr
\end{paste}\end{patch}
```

————-

Three-Dimensional Options

⇒ “notitle” (ugGraphCoordPage) 11 on page 2053

— **ug07.ht** —

```
\begin{page}{ugGraphThreeDOptionsPage}{7.2.4. Three-Dimensional Options}
\beginscroll

The \axiomFun{draw} commands optionally take an optional list of
options such as {\tt coordinates} as shown in the last example.  Each
option is given by the syntax: \axiom{name} {\tt ==} \axiom{value}.
Here is a list of the available options in the order that they are
described below:

\table{ {title} {coordinates} {var1Steps} {style} {tubeRadius} {var2Steps}
```

```
{colorFunction} {tubePoints} {space}}

\psXtc{
The option \axiom{title} gives your graph a title.
}{
\graphpaste{draw(cos(x*y),x=0..2*\%pi,y=0..\%pi,title == "Title of Graph") }
}{
\epsffile[0 0 295 295]{../ps/3dopttitl.ps}
}
%
\psXtc{
The \axiom{style} determines which of four rendering algorithms is used for
the graph.
The choices are
{\tt "wireMesh"}, {\tt "solid"}, {\tt "shade"}, and {\tt "smooth"}.
}{
\graphpaste{draw(cos(x*y),x=-3..3,y=-3..3, style=="smooth",
title=="Smooth Option")}
}{
\epsffile[0 0 295 295]{../ps/3doptsty.ps}
}
%

In all but the wire-mesh style, polygons in a surface or tube plot
are normally colored in a graph according to their
\axiom{z}-coordinate value.  Space curves are colored according to their
parametric variable value.
To change this, you can give a coloring function.
The coloring function is sampled across the range of its arguments, then
normalized onto the standard Axiom colormap.

\xtc{
A function of one variable  makes the color depend on the
value of the parametric variable specified for a tube plot.
}{
\spadpaste{color1(t) == t \bound{colorFxn1}}
}
\psXtc{
}{
\graphpaste{draw(curve(sin(t), cos(t),0), t=0..2*\%pi,
tubeRadius == .3, colorFunction == color1) \free{colorFxn1}}
}{
\epsffile[0 0 295 295]{../ps/3doptcf1.ps}
}
%
\xtc{
A function of two variables makes the color depend on the
values of the independent variables.
}{
\spadpaste{color2(u,v) == u**2 - v**2 \bound{colorFxn2}}
```

```
}
\psXtc{
Use the option {\tt colorFunction} for special coloring.
}{
\graphpaste{draw(cos(u*v), u=-3..3, v=-3..3,
colorFunction == color2) \free{colorFxn2}}
}{
\epsffile[0 0 295 295]{../ps/3doptcf2.ps}
}
%
\xtc{
With a three variable function, the
color also depends on the value of the function.
}{
\spadpaste{color3(x,y,fxy) == sin(x*fxy) + cos(y*fxy) \bound{colorFxn3}}
}
\psXtc{
}{
\graphpaste{draw(cos(x*y), x=-3..3, y=-3..3, colorFunction == color3)
\free{colorFxn3}}
}{
\epsffile[0 0 295 295]{../ps/3doptcf3.ps}
}
%
Normally the Cartesian coordinate system is used.
To change this, use the {\tt coordinates} option.
For details, see
\downlink{``Coordinate System Transformations''}{ugGraphCoordPage}
in Section 7.2.7\ignore{ugGraphCoord}.
%
%
\xtc{
}{
\spadpaste{m(u:DFLOAT,v:DFLOAT):DFLOAT == 1 \bound{m}}
}
\psXtc{
Use the spherical
coordinate system.
}{
\graphpaste{draw(m, 0..2*\%pi,0..\%pi, coordinates == spherical,
style=="shade") \free{m}}
}{
\epsffile[0 0 295 295]{../ps/3doptcrd.ps}
}
%
Space curves may be displayed as tubes with polygonal cross sections.
Two options, {\tt tubeRadius} and {\tt tubePoints},  control the size and
shape of this cross section.
%
\psXtc{
```

```
The {\tt tubeRadius} option specifies the radius of the tube that
encircles the specified space curve.
}{
\graphpaste{draw(curve(sin(t),cos(t),0),t=0..2*\%pi, style=="shade",
tubeRadius == .3)}
}{
\epsffile[0 0 295 295]{../ps/3doptrad.ps}
}
%
%
\psXtc{
The {\tt tubePoints} option specifies the number of vertices
defining the polygon that is used to create a tube around the
specified space curve.
The larger this number is, the more cylindrical the tube becomes.
}{
\graphpaste{draw(curve(sin(t), cos(t), 0), t=0..2*\%pi, style=="shade",
tubeRadius == .25, tubePoints == 3)}
}{
\epsffile[0 0 295 295]{../ps/3doptpts.ps}
}
%
%
%
\psXtc{
Options \axiomFunFrom{var1Steps}{DrawOption} and
\axiomFunFrom{var2Steps}{DrawOption} specify the number of intervals into
which the grid defining a surface plot is subdivided with respect to the
first and second parameters of the surface function(s).
}{
\graphpaste{draw(cos(x*y),x=-3..3,y=-3..3, style=="shade", var1Steps == 30,
var2Steps == 30)}
}{
\epsffile[0 0 295 295]{../ps/3doptvb.ps}
}
%
The {\tt space} option
of a \axiomFun{draw} command lets you build multiple graphs in three space.
To use this option, first create an empty three-space object,
then use the {\tt space} option thereafter.
There is no restriction as to the number or kinds
of graphs that can be combined this way.
\xtc{
Create an empty three-space object.
}{
\spadpaste{s := create3Space()\$(ThreeSpace DFLOAT) \bound{s}}
}
%
%
\xtc{
```

```
}{
\spadpaste{m(u:DFLOAT,v:DFLOAT):DFLOAT == 1 \bound{m}}
}
\psXtc{
Add a graph to this three-space object.
The new graph destructively inserts the graph
into \axiom{s}.
}{
\graphpaste{
draw(m,0..\%pi,0..2*\%pi, coordinates == spherical, space == s) \free{s m}}
}{
\epsffile[0 0 295 295]{../ps/3dmult1a.ps}
}
%
%
\psXtc{
Add a second graph to \axiom{s}.
}{
\graphpaste{v := draw(curve(1.5*sin(t), 1.5*cos(t),0), t=0..2*\%pi,
tubeRadius == .25, space == s) \free{s} \bound{v}}
}{
\epsffile[0 0 295 295]{../ps/3dmult1b.ps}
}
%
A three-space object can also be obtained from an existing
\threedim{} viewport
using the \axiomFunFrom{subspace}{ThreeSpace} command.
You can then use \axiomFun{makeViewport3D} to create a viewport window.
\xtc{
Assign to \axiom{subsp} the three-space object in viewport \axiom{v}.
}{
\spadpaste{subsp := subspace v \free{v} \bound{su}}
}
\xtc{
Reset the space component of \axiom{v} to the value of \axiom{subsp}.
}{
\spadpaste{subspace(v, subsp) \bound{sp} \free{su}}
}
\noOutputXtc{
Create a viewport window from a three-space object.
}{
\graphpaste{makeViewport3D(subsp,"Graphs") \free{sp}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugGraphThreeDOptionsPagePatch1}
\begin{paste}{ugGraphThreeDOptionsPageFull1}{ugGraphThreeDOptionsPageEmpty1}
```

```
\pastebutton{ugGraphThreeDOptionsPageFull1}{\hidepaste}
\tab{5}\spadgraph{draw(cos(x*y),x=0..2*\%pi,y=0..\%pi,title == "Title of Graph")}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/ugGraphThreeDOptionsPage1.view/image}}{viewalone\
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDOptionsPageEmpty1}
\begin{paste}{ugGraphThreeDOptionsPageEmpty1}{ugGraphThreeDOptionsPagePatch1}
\pastebutton{ugGraphThreeDOptionsPageEmpty1}{\showpaste}
\tab{5}\spadgraph{draw(cos(x*y),x=0..2*\%pi,y=0..\%pi,title == "Title of Graph")}
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDOptionsPagePatch2}
\begin{paste}{ugGraphThreeDOptionsPageFull2}{ugGraphThreeDOptionsPageEmpty2}
\pastebutton{ugGraphThreeDOptionsPageFull2}{\hidepaste}
\tab{5}\spadgraph{draw(cos(x*y),x=-3..3,y=-3..3, style=="smooth", title=="Smooth Option")}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphthreedoptionspage2.view/image}}{viewalone\
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDOptionsPageEmpty2}
\begin{paste}{ugGraphThreeDOptionsPageEmpty2}{ugGraphThreeDOptionsPagePatch2}
\pastebutton{ugGraphThreeDOptionsPageEmpty2}{\showpaste}
\tab{5}\spadgraph{draw(cos(x*y),x=-3..3,y=-3..3, style=="smooth", title=="Smooth Option")}
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDOptionsPagePatch3}
\begin{paste}{ugGraphThreeDOptionsPageFull3}{ugGraphThreeDOptionsPageEmpty3}
\pastebutton{ugGraphThreeDOptionsPageFull3}{\hidepaste}
\tab{5}\spadcommand{color1(t) == t\bound{colorFxn1 }}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDOptionsPageEmpty3}
\begin{paste}{ugGraphThreeDOptionsPageEmpty3}{ugGraphThreeDOptionsPagePatch3}
\pastebutton{ugGraphThreeDOptionsPageEmpty3}{\showpaste}
\tab{5}\spadcommand{color1(t) == t\bound{colorFxn1 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDOptionsPagePatch4}
\begin{paste}{ugGraphThreeDOptionsPageFull4}{ugGraphThreeDOptionsPageEmpty4}
\pastebutton{ugGraphThreeDOptionsPageFull4}{\hidepaste}
\tab{5}\spadgraph{draw(curve(sin(t), cos(t),0), t=0..2*\%pi, tubeRadius == .3, colorFunction == color1)\fre
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphthreedoptionspage4.view/image}}{viewalone\
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDOptionsPageEmpty4}
\begin{paste}{ugGraphThreeDOptionsPageEmpty4}{ugGraphThreeDOptionsPagePatch4}
\pastebutton{ugGraphThreeDOptionsPageEmpty4}{\showpaste}
\tab{5}\spadgraph{draw(curve(sin(t), cos(t),0), t=0..2*\%pi, tubeRadius == .3, colorFunction == color1)\fre
```

```
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDOptionsPagePatch5}
\begin{paste}{ugGraphThreeDOptionsPageFull5}{ugGraphThreeDOptionsPageEmpty5}
\pastebutton{ugGraphThreeDOptionsPageFull5}{\hidepaste}
\tab{5}\spadcommand{color2(u,v) == u**2 - v**2\bound{colorFxn2 }}
\indentrel{3}\begin{verbatim}
                                         Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDOptionsPageEmpty5}
\begin{paste}{ugGraphThreeDOptionsPageEmpty5}{ugGraphThreeDOptionsPagePatch5}
\pastebutton{ugGraphThreeDOptionsPageEmpty5}{\showpaste}
\tab{5}\spadcommand{color2(u,v) == u**2 - v**2\bound{colorFxn2 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDOptionsPagePatch6}
\begin{paste}{ugGraphThreeDOptionsPageFull6}{ugGraphThreeDOptionsPageEmpty6}
\pastebutton{ugGraphThreeDOptionsPageFull6}{\hidepaste}
\tab{5}\spadgraph{draw(cos(u*v), u=-3..3, v=-3..3, colorFunction == color2)\free{colorFxn2 }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphthreedoptionspage6.view/image}
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDOptionsPageEmpty6}
\begin{paste}{ugGraphThreeDOptionsPageEmpty6}{ugGraphThreeDOptionsPagePatch6}
\pastebutton{ugGraphThreeDOptionsPageEmpty6}{\showpaste}
\tab{5}\spadgraph{draw(cos(u*v), u=-3..3, v=-3..3, colorFunction == color2)\free{colorFxn2 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDOptionsPagePatch7}
\begin{paste}{ugGraphThreeDOptionsPageFull7}{ugGraphThreeDOptionsPageEmpty7}
\pastebutton{ugGraphThreeDOptionsPageFull7}{\hidepaste}
\tab{5}\spadcommand{color3(x,y,fxy) == sin(x*fxy) + cos(y*fxy)\bound{colorFxn3 }}
\indentrel{3}\begin{verbatim}
                                         Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDOptionsPageEmpty7}
\begin{paste}{ugGraphThreeDOptionsPageEmpty7}{ugGraphThreeDOptionsPagePatch7}
\pastebutton{ugGraphThreeDOptionsPageEmpty7}{\showpaste}
\tab{5}\spadcommand{color3(x,y,fxy) == sin(x*fxy) + cos(y*fxy)\bound{colorFxn3 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDOptionsPagePatch8}
\begin{paste}{ugGraphThreeDOptionsPageFull8}{ugGraphThreeDOptionsPageEmpty8}
\pastebutton{ugGraphThreeDOptionsPageFull8}{\hidepaste}
\tab{5}\spadgraph{draw(cos(x*y), x=-3..3, y=-3..3, colorFunction == color3)\free{colorFxn3 }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphthreedoptionspage8.view/image}
```

```
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDOptionsPageEmpty8}
\begin{paste}{ugGraphThreeDOptionsPageEmpty8}{ugGraphThreeDOptionsPagePatch8}
\pastebutton{ugGraphThreeDOptionsPageEmpty8}{\showpaste}
\tab{5}\spadgraph{draw(cos(x*y), x=-3..3, y=-3..3, colorFunction == color3)\free{colorFxn3 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDOptionsPagePatch9}
\begin{paste}{ugGraphThreeDOptionsPageFull9}{ugGraphThreeDOptionsPageEmpty9}
\pastebutton{ugGraphThreeDOptionsPageFull9}{\hidepaste}
\tab{5}\spadcommand{m(u:DFLOAT,v:DFLOAT):DFLOAT == 1\bound{m }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDOptionsPageEmpty9}
\begin{paste}{ugGraphThreeDOptionsPageEmpty9}{ugGraphThreeDOptionsPagePatch9}
\pastebutton{ugGraphThreeDOptionsPageEmpty9}{\showpaste}
\tab{5}\spadcommand{m(u:DFLOAT,v:DFLOAT):DFLOAT == 1\bound{m }}
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDOptionsPagePatch10}
\begin{paste}{ugGraphThreeDOptionsPageFull10}{ugGraphThreeDOptionsPageEmpty10}
\pastebutton{ugGraphThreeDOptionsPageFull10}{\hidepaste}
\tab{5}\spadgraph{draw(m, 0..2*\%pi,0..\%pi, coordinates == spherical, style=="shade")\free{m }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/ugGraphThreeDOptionsPage10.view/image}}{viewalone
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDOptionsPageEmpty10}
\begin{paste}{ugGraphThreeDOptionsPageEmpty10}{ugGraphThreeDOptionsPagePatch10}
\pastebutton{ugGraphThreeDOptionsPageEmpty10}{\showpaste}
\tab{5}\spadgraph{draw(m, 0..2*\%pi,0..\%pi, coordinates == spherical, style=="shade")\free{m }}
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDOptionsPagePatch11}
\begin{paste}{ugGraphThreeDOptionsPageFull11}{ugGraphThreeDOptionsPageEmpty11}
\pastebutton{ugGraphThreeDOptionsPageFull11}{\hidepaste}
\tab{5}\spadgraph{draw(curve(sin(t),cos(t),0),t=0..2*\%pi, style=="shade", tubeRadius == .3)}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphthreedoptionspage11.view/image}}{viewalone
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDOptionsPageEmpty11}
\begin{paste}{ugGraphThreeDOptionsPageEmpty11}{ugGraphThreeDOptionsPagePatch11}
\pastebutton{ugGraphThreeDOptionsPageEmpty11}{\showpaste}
\tab{5}\spadgraph{draw(curve(sin(t),cos(t),0),t=0..2*\%pi, style=="shade", tubeRadius == .3)}
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDOptionsPagePatch12}
```

```
\begin{paste}{ugGraphThreeDOptionsPageFull12}{ugGraphThreeDOptionsPageEmpty12}
\pastebutton{ugGraphThreeDOptionsPageFull12}{\hidepaste}
\tab{5}\spadgraph{draw(curve(sin(t), cos(t), 0), t=0..2*\%pi, style=="shade", tubeRadius == .25,
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphthreedoptionspage12.view/image
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDOptionsPageEmpty12}
\begin{paste}{ugGraphThreeDOptionsPageEmpty12}{ugGraphThreeDOptionsPagePatch12}
\pastebutton{ugGraphThreeDOptionsPageEmpty12}{\showpaste}
\tab{5}\spadgraph{draw(curve(sin(t), cos(t), 0), t=0..2*\%pi, style=="shade", tubeRadius == .25,
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDOptionsPagePatch13}
\begin{paste}{ugGraphThreeDOptionsPageFull13}{ugGraphThreeDOptionsPageEmpty13}
\pastebutton{ugGraphThreeDOptionsPageFull13}{\hidepaste}
\tab{5}\spadgraph{draw(cos(x*y),x=-3..3,y=-3..3, style=="shade", var1Steps == 30, var2Steps == 3
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphthreedoptionspage13.view/image
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDOptionsPageEmpty13}
\begin{paste}{ugGraphThreeDOptionsPageEmpty13}{ugGraphThreeDOptionsPagePatch13}
\pastebutton{ugGraphThreeDOptionsPageEmpty13}{\showpaste}
\tab{5}\spadgraph{draw(cos(x*y),x=-3..3,y=-3..3, style=="shade", var1Steps == 30, var2Steps == 3
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDOptionsPagePatch14}
\begin{paste}{ugGraphThreeDOptionsPageFull14}{ugGraphThreeDOptionsPageEmpty14}
\pastebutton{ugGraphThreeDOptionsPageFull14}{\hidepaste}
\tab{5}\spadcommand{s := create3Space()$(ThreeSpace DFLOAT)\bound{s }}
\indentrel{3}\begin{verbatim}
   (14)  3-Space with 0 components
                           Type: ThreeSpace DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDOptionsPageEmpty14}
\begin{paste}{ugGraphThreeDOptionsPageEmpty14}{ugGraphThreeDOptionsPagePatch14}
\pastebutton{ugGraphThreeDOptionsPageEmpty14}{\showpaste}
\tab{5}\spadcommand{s := create3Space()$(ThreeSpace DFLOAT)\bound{s }}
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDOptionsPagePatch15}
\begin{paste}{ugGraphThreeDOptionsPageFull15}{ugGraphThreeDOptionsPageEmpty15}
\pastebutton{ugGraphThreeDOptionsPageFull15}{\hidepaste}
\tab{5}\spadcommand{m(u:DFLOAT,v:DFLOAT):DFLOAT == 1\bound{m }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugGraphThreeDOptionsPageEmpty15}
\begin{paste}{ugGraphThreeDOptionsPageEmpty15}{ugGraphThreeDOptionsPagePatch15}
\pastebutton{ugGraphThreeDOptionsPageEmpty15}{\showpaste}
\tab{5}\spadcommand{m(u:DFLOAT,v:DFLOAT):DFLOAT == 1\bound{m }}
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDOptionsPagePatch16}
\begin{paste}{ugGraphThreeDOptionsPageFull16}{ugGraphThreeDOptionsPageEmpty16}
\pastebutton{ugGraphThreeDOptionsPageFull16}{\hidepaste}
\tab{5}\spadgraph{draw(m,0..\%pi,0..2*\%pi, coordinates == spherical, space == s)\free{s m }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphthreedoptionspage16.view/image}}{viewalone
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDOptionsPageEmpty16}
\begin{paste}{ugGraphThreeDOptionsPageEmpty16}{ugGraphThreeDOptionsPagePatch16}
\pastebutton{ugGraphThreeDOptionsPageEmpty16}{\showpaste}
\tab{5}\spadgraph{draw(m,0..\%pi,0..2*\%pi, coordinates == spherical, space == s)\free{s m }}
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDOptionsPagePatch17}
\begin{paste}{ugGraphThreeDOptionsPageFull17}{ugGraphThreeDOptionsPageEmpty17}
\pastebutton{ugGraphThreeDOptionsPageFull17}{\hidepaste}
\tab{5}\spadgraph{v := draw(curve(1.5*sin(t), 1.5*cos(t),0), t=0..2*\%pi, tubeRadius == .25, space == s)\fre
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphthreedoptionspage17.view/image}}{viewalone
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDOptionsPageEmpty17}
\begin{paste}{ugGraphThreeDOptionsPageEmpty17}{ugGraphThreeDOptionsPagePatch17}
\pastebutton{ugGraphThreeDOptionsPageEmpty17}{\showpaste}
\tab{5}\spadgraph{v := draw(curve(1.5*sin(t), 1.5*cos(t),0), t=0..2*\%pi, tubeRadius == .25, space == s)\fre
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDOptionsPagePatch18}
\begin{paste}{ugGraphThreeDOptionsPageFull18}{ugGraphThreeDOptionsPageEmpty18}
\pastebutton{ugGraphThreeDOptionsPageFull18}{\hidepaste}
\tab{5}\spadcommand{subsp := subspace v\free{v }\bound{su }}
\indentrel{3}\begin{verbatim}
   (18)  3-Space with 2 components
                         Type: ThreeSpace DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDOptionsPageEmpty18}
\begin{paste}{ugGraphThreeDOptionsPageEmpty18}{ugGraphThreeDOptionsPagePatch18}
\pastebutton{ugGraphThreeDOptionsPageEmpty18}{\showpaste}
\tab{5}\spadcommand{subsp := subspace v\free{v }\bound{su }}
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDOptionsPagePatch19}
\begin{paste}{ugGraphThreeDOptionsPageFull19}{ugGraphThreeDOptionsPageEmpty19}
```

```
\pastebutton{ugGraphThreeDOptionsPageFull19}{\hidepaste}
\tab{5}\spadcommand{subspace(v, subsp)\bound{sp }\free{su }}
\indentrel{3}\begin{verbatim}
   (19)
   Closed or Undefined ThreeDimensionalViewport: "AXIOM3D"
                         Type: ThreeDimensionalViewport
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDOptionsPageEmpty19}
\begin{paste}{ugGraphThreeDOptionsPageEmpty19}{ugGraphThreeDOptionsPagePatch19}
\pastebutton{ugGraphThreeDOptionsPageEmpty19}{\showpaste}
\tab{5}\spadcommand{subspace(v, subsp)\bound{sp }\free{su }}
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDOptionsPagePatch20}
\begin{paste}{ugGraphThreeDOptionsPageFull20}{ugGraphThreeDOptionsPageEmpty20}
\pastebutton{ugGraphThreeDOptionsPageFull20}{\hidepaste}
\tab{5}\spadgraph{makeViewport3D(subsp,"Graphs")\free{sp }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphthreedoptionspage20.view/image
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDOptionsPageEmpty20}
\begin{paste}{ugGraphThreeDOptionsPageEmpty20}{ugGraphThreeDOptionsPagePatch20}
\pastebutton{ugGraphThreeDOptionsPageEmpty20}{\showpaste}
\tab{5}\spadgraph{makeViewport3D(subsp,"Graphs")\free{sp }}
\end{paste}\end{patch}
```

———-

The makeObject Command

— ug07.ht —

```
\begin{page}{ugGraphMakeObjectPage}{7.2.5. The makeObject Command}
\beginscroll

An alternate way to create multiple graphs is to use
\axiomFun{makeObject}.
The \axiomFun{makeObject} command is similar to the \axiomFun{draw}
command, except that it returns a three-space object rather than a
\axiomType{ThreeDimensionalViewport}.
In fact, \axiomFun{makeObject} is called by the \axiomFun{draw}
command to create the \axiomType{ThreeSpace} then
\axiomFunFrom{makeViewport3D}{ThreeDimensionalViewport} to create a
viewport window.
```

```
\xtc{
}{
\spadpaste{m(u:DFLOAT,v:DFLOAT):DFLOAT == 1 \bound{m}}
}
\noOutputXtc{
Do the last example a new way.
First use \axiomFun{makeObject} to
create a three-space object \axiom{sph}.
}{
\spadpaste{sph := makeObject(m, 0..\%pi, 0..2*\%pi,
coordinates==spherical)\bound{sph}\free{m}}
}
\noOutputXtc{
Add a second object to \axiom{sph}.
}{
\spadpaste{makeObject(curve(1.5*sin(t), 1.5*cos(t), 0),
t=0..2*\%pi, space == sph, tubeRadius == .25) \free{sph}\bound{v1}}
}
\noOutputXtc{
Create and display a viewport
containing \axiom{sph}.
}{
\graphpaste{makeViewport3D(sph,"Multiple Objects") \free{v1}}
}

Note that an undefined \axiomType{ThreeSpace} parameter declared in a
\axiomFun{makeObject} or \axiomFun{draw} command results in an error.
Use the \axiomFunFrom{create3Space}{ThreeSpace} function to define a
\axiomType{ThreeSpace}, or obtain a \axiomType{ThreeSpace} that has been
previously generated before including it in a command line.

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugGraphMakeObjectPagePatch1}
\begin{paste}{ugGraphMakeObjectPageFull1}{ugGraphMakeObjectPageEmpty1}
\pastebutton{ugGraphMakeObjectPageFull1}{\hidepaste}
\tab{5}\spadcommand{m(u:DFLOAT,v:DFLOAT):DFLOAT == 1\bound{m }}
\indentrel{3}\begin{verbatim}
                                              Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphMakeObjectPageEmpty1}
\begin{paste}{ugGraphMakeObjectPageEmpty1}{ugGraphMakeObjectPagePatch1}
\pastebutton{ugGraphMakeObjectPageEmpty1}{\showpaste}
\tab{5}\spadcommand{m(u:DFLOAT,v:DFLOAT):DFLOAT == 1\bound{m }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugGraphMakeObjectPagePatch2}
\begin{paste}{ugGraphMakeObjectPageFull2}{ugGraphMakeObjectPageEmpty2}
\pastebutton{ugGraphMakeObjectPageFull2}{\hidepaste}
\tab{5}\spadcommand{sph := makeObject(m, 0..\%pi, 0..2*\%pi, coordinates==spherical)\bound{sph }
\indentrel{3}\begin{verbatim}
   (2)  3-Space with 1 component
                           Type: ThreeSpace DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphMakeObjectPageEmpty2}
\begin{paste}{ugGraphMakeObjectPageEmpty2}{ugGraphMakeObjectPagePatch2}
\pastebutton{ugGraphMakeObjectPageEmpty2}{\showpaste}
\tab{5}\spadcommand{sph := makeObject(m, 0..\%pi, 0..2*\%pi, coordinates==spherical)\bound{sph }
\end{paste}\end{patch}

\begin{patch}{ugGraphMakeObjectPagePatch3}
\begin{paste}{ugGraphMakeObjectPageFull3}{ugGraphMakeObjectPageEmpty3}
\pastebutton{ugGraphMakeObjectPageFull3}{\hidepaste}
\tab{5}\spadcommand{makeObject(curve(1.5*sin(t), 1.5*cos(t), 0), t=0..2*\%pi, space == sph, tube
\indentrel{3}\begin{verbatim}
   (3)  3-Space with 2 components
                           Type: ThreeSpace DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphMakeObjectPageEmpty3}
\begin{paste}{ugGraphMakeObjectPageEmpty3}{ugGraphMakeObjectPagePatch3}
\pastebutton{ugGraphMakeObjectPageEmpty3}{\showpaste}
\tab{5}\spadcommand{makeObject(curve(1.5*sin(t), 1.5*cos(t), 0), t=0..2*\%pi, space == sph, tube
\end{paste}\end{patch}

\begin{patch}{ugGraphMakeObjectPagePatch4}
\begin{paste}{ugGraphMakeObjectPageFull4}{ugGraphMakeObjectPageEmpty4}
\pastebutton{ugGraphMakeObjectPageFull4}{\hidepaste}
\tab{5}\spadgraph{makeViewport3D(sph,"Multiple Objects")\free{v1 }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphmakeobjectpage4.view/image}}{v
\end{paste}\end{patch}

\begin{patch}{ugGraphMakeObjectPageEmpty4}
\begin{paste}{ugGraphMakeObjectPageEmpty4}{ugGraphMakeObjectPagePatch4}
\pastebutton{ugGraphMakeObjectPageEmpty4}{\showpaste}
\tab{5}\spadgraph{makeViewport3D(sph,"Multiple Objects")\free{v1 }}
\end{paste}\end{patch}
```

Building 3D Objects From Primitives

— ug07.ht —

```
\begin{page}{ugGraphThreeDBuildPage}
{7.2.6. Building 3D Objects From Primitives}
\beginscroll

Rather than using the \axiomFun{draw} and \axiomFun{makeObject} commands,
you can create \threedim{} graphs from primitives.
Operation \axiomFunFrom{create3Space}{ThreeSpace} creates a
three-space object to which points, curves and polygons
can be added using the operations from the \axiomType{ThreeSpace}
domain.
The resulting object can then be displayed in a viewport using
\axiomFunFrom{makeViewport3D}{ThreeDimensionalViewport}.

\xtc{
Create the empty three-space object \axiom{space}.
}{
\spadpaste{space := create3Space()\$(ThreeSpace DFLOAT) \bound{space}}
}

Objects can be sent to this \axiom{space} using the operations
exported by the \axiomType{ThreeSpace} domain.
The following examples place curves into \axiom{space}.

\xtc{
Add these eight curves to the space.
}{
\spadpaste{closedCurve(space,[[0,30,20], [0,30,30], [0,40,30],
[0,40,100], [0,30,100],[0,30,110], [0,60,110], [0,60,100],
[0,50,100], [0,50,30], [0,60,30], [0,60,20]]) \bound{curve1}
\free{space}}
}
\xtc{
}{
\spadpaste{closedCurve(space,[[80,0,30], [80,0,100], [70,0,110],
[40,0,110], [30,0,100], [30,0,90], [40,0,90], [40,0,95],
[45,0,100], [65,0,100], [70,0,95], [70,0,35]]) \bound{curve2}
\free{space}}
}
\xtc{
}{
\spadpaste{closedCurve(space,[[70,0,35], [65,0,30], [45,0,30],
[40,0,35], [40,0,60], [50,0,60], [50,0,70], [30,0,70], [30,0,30],
[40,0,20], [70,0,20], [80,0,30]]) \bound{curve3} \free{space}}
}
\xtc{
```

```
}{
\spadpaste{closedCurve(space,[[0,70,20], [0,70,110], [0,110,110],
[0,120,100], [0,120,70], [0,115,65], [0,120,60], [0,120,30],
[0,110,20], [0,80,20], [0,80,30], [0,80,20]]) \bound{curve4} \free{space}}
}
\xtc{
}{
\spadpaste{closedCurve(space,[[0,105,30], [0,110,35], [0,110,55],
[0,105,60], [0,80,60], [0,80,70], [0,105,70], [0,110,75],
[0,110,95], [0,105,100], [0,80,100], [0,80,20], [0,80,30]])
\bound{curve5} \free{space}}
}
\xtc{
}{
\spadpaste{closedCurve(space,[[140,0,20], [140,0,110], [130,0,110],
[90,0,20], [101,0,20],[114,0,50], [130,0,50], [130,0,60], [119,0,60],
[130,0,85], [130,0,20]]) \bound{curve6} \free{space}}
}
\xtc{
}{
\spadpaste{closedCurve(space,[[0,140,20], [0,140,110], [0,150,110],
[0,170,50], [0,190,110], [0,200,110], [0,200,20], [0,190,20],
[0,190,75], [0,175,35], [0,165,35],[0,150,75], [0,150,20]])
\bound{curve7} \free{space}}
}
\xtc{
}{
\spadpaste{closedCurve(space,[[200,0,20], [200,0,110], [189,0,110],
[160,0,45], [160,0,110], [150,0,110], [150,0,20], [161,0,20],
[190,0,85], [190,0,20]]) \bound{curve8} \free{space}}
}
\psXtc{
Create and display the viewport using \axiomFun{makeViewport3D}.
Options may also be given but here are displayed as a list with values
enclosed in parentheses.
}{
\graphpaste{makeViewport3D(space, title == "Letters")
\free{space curve1 curve2 curve3 curve4 curve5 curve6 curve7 curve8}}
}{
\epsffile[0 0 295 295]{../ps/3dbuilda.ps}
}

\subsubsection{Cube Example}

As a second example of the use of primitives, we generate a cube using a
polygon mesh.
It is important to use a consistent orientation of the polygons for
correct generation of \threedim{} objects.

\xtc{
```

```
Again start with an empty three-space object.
}{
\spadpaste{spaceC := create3Space()\$(ThreeSpace DFLOAT) \bound{spaceC}}
}
\xtc{
For convenience,
give \axiomType{DoubleFloat} values \axiom{+1} and \axiom{-1} names.
}{
\spadpaste{x: DFLOAT := 1 \bound{x}}
}
\xtc{
}{
\spadpaste{y: DFLOAT := -1 \bound{y}}
}
\xtc{
Define the vertices of the cube.
}{
\spadpaste{a := point [x,x,y,1::DFLOAT]\$(Point DFLOAT) \bound{a}
\free{x y}}
}
\xtc{
}{
\spadpaste{b := point [y,x,y,4::DFLOAT]\$(Point DFLOAT) \bound{b}
\free{x y}}
}
\xtc{
}{
\spadpaste{c := point [y,x,x,8::DFLOAT]\$(Point DFLOAT) \bound{c}
\free{x y}}
}
\xtc{
}{
\spadpaste{d := point [x,x,x,12::DFLOAT]\$(Point DFLOAT) \bound{d}
\free{x y}}
}
\xtc{
}{
\spadpaste{e := point [x,y,y,16::DFLOAT]\$(Point DFLOAT) \bound{e}
\free{x y}}
}
\xtc{
}{
\spadpaste{f := point [y,y,y,20::DFLOAT]\$(Point DFLOAT) \bound{f}
\free{x y}}
}
\xtc{
}{
\spadpaste{g := point [y,y,x,24::DFLOAT]\$(Point DFLOAT) \bound{g}
\free{x y}}
}
```

```
\xtc{
}{
\spadpaste{h := point [x,y,x,27::DFLOAT]\$(Point DFLOAT) \bound{h}
\free{x y}}
}
\xtc{
Add the faces of the cube as polygons to the space using a
consistent orientation.
}{
\spadpaste{polygon(spaceC,[d,c,g,h]) \free{d c g h spaceC} \bound{pol1}}
}
\xtc{
}{
\spadpaste{polygon(spaceC,[d,h,e,a]) \free{d h e a spaceC} \bound{pol2}}
}
\xtc{
}{
\spadpaste{polygon(spaceC,[c,d,a,b]) \free{c d a b spaceC} \bound{pol3}}
}
\xtc{
}{
\spadpaste{polygon(spaceC,[g,c,b,f]) \free{g c b f spaceC} \bound{pol4}}
}
\xtc{
}{
\spadpaste{polygon(spaceC,[h,g,f,e]) \free{h g f e spaceC} \bound{pol5}}
}
\xtc{
}{
\spadpaste{polygon(spaceC,[e,f,b,a]) \free{e f b a spaceC} \bound{pol6}}
}
\psXtc{
Create and display the viewport.
}{
\graphpaste{makeViewport3D(spaceC, title == "Cube")
\free{pol1 pol2 pol3 pol4 pol5 pol6}}
}{
\epsffile[0 0 295 295]{../ps/3dbuildb.ps}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugGraphThreeDBuildPagePatch1}
\begin{paste}{ugGraphThreeDBuildPageFull1}{ugGraphThreeDBuildPageEmpty1}
\pastebutton{ugGraphThreeDBuildPageFull1}{\hidepaste}
\tab{5}\spadcommand{space := create3Space()$(ThreeSpace DFLOAT)\bound{space }}
\indentrel{3}\begin{verbatim}
   (1)  3-Space with 0 components
```

```
                           Type: ThreeSpace DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPageEmpty1}
\begin{paste}{ugGraphThreeDBuildPageEmpty1}{ugGraphThreeDBuildPagePatch1}
\pastebutton{ugGraphThreeDBuildPageEmpty1}{\showpaste}
\tab{5}\spadcommand{space := create3Space()$(ThreeSpace DFLOAT)\bound{space }}
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPagePatch2}
\begin{paste}{ugGraphThreeDBuildPageFull2}{ugGraphThreeDBuildPageEmpty2}
\pastebutton{ugGraphThreeDBuildPageFull2}{\hidepaste}
\tab{5}\spadcommand{closedCurve(space,[[0,30,20], [0,30,30], [0,40,30], [0,40,100], [0,30,100],[0,30,110],
\indentrel{3}\begin{verbatim}
   (2)  3-Space with 1 component
                           Type: ThreeSpace DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPageEmpty2}
\begin{paste}{ugGraphThreeDBuildPageEmpty2}{ugGraphThreeDBuildPagePatch2}
\pastebutton{ugGraphThreeDBuildPageEmpty2}{\showpaste}
\tab{5}\spadcommand{closedCurve(space,[[0,30,20], [0,30,30], [0,40,30], [0,40,100], [0,30,100],[0,30,110],
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPagePatch3}
\begin{paste}{ugGraphThreeDBuildPageFull3}{ugGraphThreeDBuildPageEmpty3}
\pastebutton{ugGraphThreeDBuildPageFull3}{\hidepaste}
\tab{5}\spadcommand{closedCurve(space,[[80,0,30], [80,0,100], [70,0,110], [40,0,110], [30,0,100], [30,0,90]
\indentrel{3}\begin{verbatim}
   (3)  3-Space with 2 components
                           Type: ThreeSpace DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPageEmpty3}
\begin{paste}{ugGraphThreeDBuildPageEmpty3}{ugGraphThreeDBuildPagePatch3}
\pastebutton{ugGraphThreeDBuildPageEmpty3}{\showpaste}
\tab{5}\spadcommand{closedCurve(space,[[80,0,30], [80,0,100], [70,0,110], [40,0,110], [30,0,100], [30,0,90]
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPagePatch4}
\begin{paste}{ugGraphThreeDBuildPageFull4}{ugGraphThreeDBuildPageEmpty4}
\pastebutton{ugGraphThreeDBuildPageFull4}{\hidepaste}
\tab{5}\spadcommand{closedCurve(space,[[70,0,35], [65,0,30], [45,0,30], [40,0,35], [40,0,60], [50,0,60], [50
\indentrel{3}\begin{verbatim}
   (4)  3-Space with 3 components
                           Type: ThreeSpace DoubleFloat
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPageEmpty4}
\begin{paste}{ugGraphThreeDBuildPageEmpty4}{ugGraphThreeDBuildPagePatch4}
\pastebutton{ugGraphThreeDBuildPageEmpty4}{\showpaste}
\tab{5}\spadcommand{closedCurve(space,[[70,0,35], [65,0,30], [45,0,30], [40,0,35], [40,0,60], [5
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPagePatch5}
\begin{paste}{ugGraphThreeDBuildPageFull5}{ugGraphThreeDBuildPageEmpty5}
\pastebutton{ugGraphThreeDBuildPageFull5}{\hidepaste}
\tab{5}\spadcommand{closedCurve(space,[[0,70,20], [0,70,110], [0,110,110], [0,120,100], [0,120,7
\indentrel{3}\begin{verbatim}
   (5)  3-Space with 4 components
                           Type: ThreeSpace DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPageEmpty5}
\begin{paste}{ugGraphThreeDBuildPageEmpty5}{ugGraphThreeDBuildPagePatch5}
\pastebutton{ugGraphThreeDBuildPageEmpty5}{\showpaste}
\tab{5}\spadcommand{closedCurve(space,[[0,70,20], [0,70,110], [0,110,110], [0,120,100], [0,120,7
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPagePatch6}
\begin{paste}{ugGraphThreeDBuildPageFull6}{ugGraphThreeDBuildPageEmpty6}
\pastebutton{ugGraphThreeDBuildPageFull6}{\hidepaste}
\tab{5}\spadcommand{closedCurve(space,[[0,105,30], [0,110,35], [0,110,55], [0,105,60], [0,80,60]
\indentrel{3}\begin{verbatim}
   (6)  3-Space with 5 components
                           Type: ThreeSpace DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPageEmpty6}
\begin{paste}{ugGraphThreeDBuildPageEmpty6}{ugGraphThreeDBuildPagePatch6}
\pastebutton{ugGraphThreeDBuildPageEmpty6}{\showpaste}
\tab{5}\spadcommand{closedCurve(space,[[0,105,30], [0,110,35], [0,110,55], [0,105,60], [0,80,60]
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPagePatch7}
\begin{paste}{ugGraphThreeDBuildPageFull7}{ugGraphThreeDBuildPageEmpty7}
\pastebutton{ugGraphThreeDBuildPageFull7}{\hidepaste}
\tab{5}\spadcommand{closedCurve(space,[[140,0,20], [140,0,110], [130,0,110], [90,0,20], [101,0,2
\indentrel{3}\begin{verbatim}
   (7)  3-Space with 6 components
                           Type: ThreeSpace DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugGraphThreeDBuildPageEmpty7}
\begin{paste}{ugGraphThreeDBuildPageEmpty7}{ugGraphThreeDBuildPagePatch7}
\pastebutton{ugGraphThreeDBuildPageEmpty7}{\showpaste}
\tab{5}\spadcommand{closedCurve(space,[[140,0,20], [140,0,110], [130,0,110], [90,0,20], [101,0,20],[114,0,5
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPagePatch8}
\begin{paste}{ugGraphThreeDBuildPageFull8}{ugGraphThreeDBuildPageEmpty8}
\pastebutton{ugGraphThreeDBuildPageFull8}{\hidepaste}
\tab{5}\spadcommand{closedCurve(space,[[0,140,20], [0,140,110], [0,150,110], [0,170,50], [0,190,110], [0,20
\indentrel{3}\begin{verbatim}
   (8)  3-Space with 7 components
                           Type: ThreeSpace DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPageEmpty8}
\begin{paste}{ugGraphThreeDBuildPageEmpty8}{ugGraphThreeDBuildPagePatch8}
\pastebutton{ugGraphThreeDBuildPageEmpty8}{\showpaste}
\tab{5}\spadcommand{closedCurve(space,[[0,140,20], [0,140,110], [0,150,110], [0,170,50], [0,190,110], [0,20
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPagePatch9}
\begin{paste}{ugGraphThreeDBuildPageFull9}{ugGraphThreeDBuildPageEmpty9}
\pastebutton{ugGraphThreeDBuildPageFull9}{\hidepaste}
\tab{5}\spadcommand{closedCurve(space,[[200,0,20], [200,0,110], [189,0,110], [160,0,45], [160,0,110], [150,
\indentrel{3}\begin{verbatim}
   (9)  3-Space with 8 components
                           Type: ThreeSpace DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPageEmpty9}
\begin{paste}{ugGraphThreeDBuildPageEmpty9}{ugGraphThreeDBuildPagePatch9}
\pastebutton{ugGraphThreeDBuildPageEmpty9}{\showpaste}
\tab{5}\spadcommand{closedCurve(space,[[200,0,20], [200,0,110], [189,0,110], [160,0,45], [160,0,110], [150,
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPagePatch10}
\begin{paste}{ugGraphThreeDBuildPageFull10}{ugGraphThreeDBuildPageEmpty10}
\pastebutton{ugGraphThreeDBuildPageFull10}{\hidepaste}
\tab{5}\spadgraph{makeViewport3D(space, title == "Letters")\free{space curve1 curve2 curve3 curve4 curve5 c
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphthreedbuildpage10.view/image}}{viewalone\
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPageEmpty10}
\begin{paste}{ugGraphThreeDBuildPageEmpty10}{ugGraphThreeDBuildPagePatch10}
\pastebutton{ugGraphThreeDBuildPageEmpty10}{\showpaste}
\tab{5}\spadgraph{makeViewport3D(space, title == "Letters")\free{space curve1 curve2 curve3 curve4 curve5 c
\end{paste}\end{patch}
```

```
\begin{patch}{ugGraphThreeDBuildPagePatch11}
\begin{paste}{ugGraphThreeDBuildPageFull11}{ugGraphThreeDBuildPageEmpty11}
\pastebutton{ugGraphThreeDBuildPageFull11}{\hidepaste}
\tab{5}\spadcommand{spaceC := create3Space()$(ThreeSpace DFLOAT)\bound{spaceC }}
\indentrel{3}\begin{verbatim}
   (11)  3-Space with 0 components
                          Type: ThreeSpace DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPageEmpty11}
\begin{paste}{ugGraphThreeDBuildPageEmpty11}{ugGraphThreeDBuildPagePatch11}
\pastebutton{ugGraphThreeDBuildPageEmpty11}{\showpaste}
\tab{5}\spadcommand{spaceC := create3Space()$(ThreeSpace DFLOAT)\bound{spaceC }}
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPagePatch12}
\begin{paste}{ugGraphThreeDBuildPageFull12}{ugGraphThreeDBuildPageEmpty12}
\pastebutton{ugGraphThreeDBuildPageFull12}{\hidepaste}
\tab{5}\spadcommand{x: DFLOAT := 1\bound{x }}
\indentrel{3}\begin{verbatim}
   (12)  1.0
                                     Type: DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPageEmpty12}
\begin{paste}{ugGraphThreeDBuildPageEmpty12}{ugGraphThreeDBuildPagePatch12}
\pastebutton{ugGraphThreeDBuildPageEmpty12}{\showpaste}
\tab{5}\spadcommand{x: DFLOAT := 1\bound{x }}
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPagePatch13}
\begin{paste}{ugGraphThreeDBuildPageFull13}{ugGraphThreeDBuildPageEmpty13}
\pastebutton{ugGraphThreeDBuildPageFull13}{\hidepaste}
\tab{5}\spadcommand{y: DFLOAT := -1\bound{y }}
\indentrel{3}\begin{verbatim}
   (13)  - 1.0
                                     Type: DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPageEmpty13}
\begin{paste}{ugGraphThreeDBuildPageEmpty13}{ugGraphThreeDBuildPagePatch13}
\pastebutton{ugGraphThreeDBuildPageEmpty13}{\showpaste}
\tab{5}\spadcommand{y: DFLOAT := -1\bound{y }}
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPagePatch14}
```

```
\begin{paste}{ugGraphThreeDBuildPageFull14}{ugGraphThreeDBuildPageEmpty14}
\pastebutton{ugGraphThreeDBuildPageFull14}{\hidepaste}
\tab{5}\spadcommand{a := point [x,x,y,1::DFLOAT]$(Point DFLOAT)\bound{a }\free{x y }}
\indentrel{3}\begin{verbatim}
   (14)  [1.0,1.0,- 1.0,1.0]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPageEmpty14}
\begin{paste}{ugGraphThreeDBuildPageEmpty14}{ugGraphThreeDBuildPagePatch14}
\pastebutton{ugGraphThreeDBuildPageEmpty14}{\showpaste}
\tab{5}\spadcommand{a := point [x,x,y,1::DFLOAT]$(Point DFLOAT)\bound{a }\free{x y }}
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPagePatch15}
\begin{paste}{ugGraphThreeDBuildPageFull15}{ugGraphThreeDBuildPageEmpty15}
\pastebutton{ugGraphThreeDBuildPageFull15}{\hidepaste}
\tab{5}\spadcommand{b := point [y,x,y,4::DFLOAT]$(Point DFLOAT)\bound{b }\free{x y }}
\indentrel{3}\begin{verbatim}
   (15)  [- 1.0,1.0,- 1.0,4.0]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPageEmpty15}
\begin{paste}{ugGraphThreeDBuildPageEmpty15}{ugGraphThreeDBuildPagePatch15}
\pastebutton{ugGraphThreeDBuildPageEmpty15}{\showpaste}
\tab{5}\spadcommand{b := point [y,x,y,4::DFLOAT]$(Point DFLOAT)\bound{b }\free{x y }}
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPagePatch16}
\begin{paste}{ugGraphThreeDBuildPageFull16}{ugGraphThreeDBuildPageEmpty16}
\pastebutton{ugGraphThreeDBuildPageFull16}{\hidepaste}
\tab{5}\spadcommand{c := point [y,x,x,8::DFLOAT]$(Point DFLOAT)\bound{c }\free{x y }}
\indentrel{3}\begin{verbatim}
   (16)  [- 1.0,1.0,1.0,8.0]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPageEmpty16}
\begin{paste}{ugGraphThreeDBuildPageEmpty16}{ugGraphThreeDBuildPagePatch16}
\pastebutton{ugGraphThreeDBuildPageEmpty16}{\showpaste}
\tab{5}\spadcommand{c := point [y,x,x,8::DFLOAT]$(Point DFLOAT)\bound{c }\free{x y }}
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPagePatch17}
\begin{paste}{ugGraphThreeDBuildPageFull17}{ugGraphThreeDBuildPageEmpty17}
\pastebutton{ugGraphThreeDBuildPageFull17}{\hidepaste}
```

```
\tab{5}\spadcommand{d := point [x,x,x,12::DFLOAT]$(Point DFLOAT)\bound{d }\free{x y }}
\indentrel{3}\begin{verbatim}
   (17)  [1.0,1.0,1.0,12.0]
                               Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPageEmpty17}
\begin{paste}{ugGraphThreeDBuildPageEmpty17}{ugGraphThreeDBuildPagePatch17}
\pastebutton{ugGraphThreeDBuildPageEmpty17}{\showpaste}
\tab{5}\spadcommand{d := point [x,x,x,12::DFLOAT]$(Point DFLOAT)\bound{d }\free{x y }}
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPagePatch18}
\begin{paste}{ugGraphThreeDBuildPageFull18}{ugGraphThreeDBuildPageEmpty18}
\pastebutton{ugGraphThreeDBuildPageFull18}{\hidepaste}
\tab{5}\spadcommand{e := point [x,y,y,16::DFLOAT]$(Point DFLOAT)\bound{e }\free{x y }}
\indentrel{3}\begin{verbatim}
   (18)  [1.0,- 1.0,- 1.0,16.0]
                               Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPageEmpty18}
\begin{paste}{ugGraphThreeDBuildPageEmpty18}{ugGraphThreeDBuildPagePatch18}
\pastebutton{ugGraphThreeDBuildPageEmpty18}{\showpaste}
\tab{5}\spadcommand{e := point [x,y,y,16::DFLOAT]$(Point DFLOAT)\bound{e }\free{x y }}
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPagePatch19}
\begin{paste}{ugGraphThreeDBuildPageFull19}{ugGraphThreeDBuildPageEmpty19}
\pastebutton{ugGraphThreeDBuildPageFull19}{\hidepaste}
\tab{5}\spadcommand{f := point [y,y,y,20::DFLOAT]$(Point DFLOAT)\bound{f }\free{x y }}
\indentrel{3}\begin{verbatim}
   (19)  [- 1.0,- 1.0,- 1.0,20.0]
                               Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPageEmpty19}
\begin{paste}{ugGraphThreeDBuildPageEmpty19}{ugGraphThreeDBuildPagePatch19}
\pastebutton{ugGraphThreeDBuildPageEmpty19}{\showpaste}
\tab{5}\spadcommand{f := point [y,y,y,20::DFLOAT]$(Point DFLOAT)\bound{f }\free{x y }}
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPagePatch20}
\begin{paste}{ugGraphThreeDBuildPageFull20}{ugGraphThreeDBuildPageEmpty20}
\pastebutton{ugGraphThreeDBuildPageFull20}{\hidepaste}
\tab{5}\spadcommand{g := point [y,y,x,24::DFLOAT]$(Point DFLOAT)\bound{g }\free{x y }}
\indentrel{3}\begin{verbatim}
```

```
   (20)  [- 1.0,- 1.0,1.0,24.0]
                               Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPageEmpty20}
\begin{paste}{ugGraphThreeDBuildPageEmpty20}{ugGraphThreeDBuildPagePatch20}
\pastebutton{ugGraphThreeDBuildPageEmpty20}{\showpaste}
\tab{5}\spadcommand{g := point [y,y,x,24::DFLOAT]$(Point DFLOAT)\bound{g }\free{x y }}
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPagePatch21}
\begin{paste}{ugGraphThreeDBuildPageFull21}{ugGraphThreeDBuildPageEmpty21}
\pastebutton{ugGraphThreeDBuildPageFull21}{\hidepaste}
\tab{5}\spadcommand{h := point [x,y,x,27::DFLOAT]$(Point DFLOAT)\bound{h }\free{x y }}
\indentrel{3}\begin{verbatim}
   (21)  [1.0,- 1.0,1.0,27.0]
                               Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPageEmpty21}
\begin{paste}{ugGraphThreeDBuildPageEmpty21}{ugGraphThreeDBuildPagePatch21}
\pastebutton{ugGraphThreeDBuildPageEmpty21}{\showpaste}
\tab{5}\spadcommand{h := point [x,y,x,27::DFLOAT]$(Point DFLOAT)\bound{h }\free{x y }}
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPagePatch22}
\begin{paste}{ugGraphThreeDBuildPageFull22}{ugGraphThreeDBuildPageEmpty22}
\pastebutton{ugGraphThreeDBuildPageFull22}{\hidepaste}
\tab{5}\spadcommand{polygon(spaceC,[d,c,g,h])\free{d c g h spaceC }\bound{pol1 }}
\indentrel{3}\begin{verbatim}
   (22)  3-Space with 1 component
                          Type: ThreeSpace DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPageEmpty22}
\begin{paste}{ugGraphThreeDBuildPageEmpty22}{ugGraphThreeDBuildPagePatch22}
\pastebutton{ugGraphThreeDBuildPageEmpty22}{\showpaste}
\tab{5}\spadcommand{polygon(spaceC,[d,c,g,h])\free{d c g h spaceC }\bound{pol1 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPagePatch23}
\begin{paste}{ugGraphThreeDBuildPageFull23}{ugGraphThreeDBuildPageEmpty23}
\pastebutton{ugGraphThreeDBuildPageFull23}{\hidepaste}
\tab{5}\spadcommand{polygon(spaceC,[d,h,e,a])\free{d h e a spaceC }\bound{pol2 }}
\indentrel{3}\begin{verbatim}
   (23)  3-Space with 2 components
                          Type: ThreeSpace DoubleFloat
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPageEmpty23}
\begin{paste}{ugGraphThreeDBuildPageEmpty23}{ugGraphThreeDBuildPagePatch23}
\pastebutton{ugGraphThreeDBuildPageEmpty23}{\showpaste}
\tab{5}\spadcommand{polygon(spaceC,[d,h,e,a])\free{d h e a spaceC }\bound{pol2 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPagePatch24}
\begin{paste}{ugGraphThreeDBuildPageFull24}{ugGraphThreeDBuildPageEmpty24}
\pastebutton{ugGraphThreeDBuildPageFull24}{\hidepaste}
\tab{5}\spadcommand{polygon(spaceC,[c,d,a,b])\free{c d a b spaceC }\bound{pol3 }}
\indentrel{3}\begin{verbatim}
   (24)  3-Space with 3 components
                          Type: ThreeSpace DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPageEmpty24}
\begin{paste}{ugGraphThreeDBuildPageEmpty24}{ugGraphThreeDBuildPagePatch24}
\pastebutton{ugGraphThreeDBuildPageEmpty24}{\showpaste}
\tab{5}\spadcommand{polygon(spaceC,[c,d,a,b])\free{c d a b spaceC }\bound{pol3 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPagePatch25}
\begin{paste}{ugGraphThreeDBuildPageFull25}{ugGraphThreeDBuildPageEmpty25}
\pastebutton{ugGraphThreeDBuildPageFull25}{\hidepaste}
\tab{5}\spadcommand{polygon(spaceC,[g,c,b,f])\free{g c b f spaceC }\bound{pol4 }}
\indentrel{3}\begin{verbatim}
   (25)  3-Space with 4 components
                          Type: ThreeSpace DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPageEmpty25}
\begin{paste}{ugGraphThreeDBuildPageEmpty25}{ugGraphThreeDBuildPagePatch25}
\pastebutton{ugGraphThreeDBuildPageEmpty25}{\showpaste}
\tab{5}\spadcommand{polygon(spaceC,[g,c,b,f])\free{g c b f spaceC }\bound{pol4 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPagePatch26}
\begin{paste}{ugGraphThreeDBuildPageFull26}{ugGraphThreeDBuildPageEmpty26}
\pastebutton{ugGraphThreeDBuildPageFull26}{\hidepaste}
\tab{5}\spadcommand{polygon(spaceC,[h,g,f,e])\free{h g f e spaceC }\bound{pol5 }}
\indentrel{3}\begin{verbatim}
   (26)  3-Space with 5 components
                          Type: ThreeSpace DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugGraphThreeDBuildPageEmpty26}
\begin{paste}{ugGraphThreeDBuildPageEmpty26}{ugGraphThreeDBuildPagePatch26}
\pastebutton{ugGraphThreeDBuildPageEmpty26}{\showpaste}
\tab{5}\spadcommand{polygon(spaceC,[h,g,f,e])\free{h g f e spaceC }\bound{pol5 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPagePatch27}
\begin{paste}{ugGraphThreeDBuildPageFull27}{ugGraphThreeDBuildPageEmpty27}
\pastebutton{ugGraphThreeDBuildPageFull27}{\hidepaste}
\tab{5}\spadcommand{polygon(spaceC,[e,f,b,a])\free{e f b a spaceC }\bound{pol6 }}
\indentrel{3}\begin{verbatim}
   (27)  3-Space with 6 components
                           Type: ThreeSpace DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPageEmpty27}
\begin{paste}{ugGraphThreeDBuildPageEmpty27}{ugGraphThreeDBuildPagePatch27}
\pastebutton{ugGraphThreeDBuildPageEmpty27}{\showpaste}
\tab{5}\spadcommand{polygon(spaceC,[e,f,b,a])\free{e f b a spaceC }\bound{pol6 }}
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPagePatch28}
\begin{paste}{ugGraphThreeDBuildPageFull28}{ugGraphThreeDBuildPageEmpty28}
\pastebutton{ugGraphThreeDBuildPageFull28}{\hidepaste}
\tab{5}\spadgraph{makeViewport3D(spaceC, title == "Cube")\free{pol1 pol2 pol3 pol4 pol5 pol6 }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphthreedbuildpage28.view/image}}{viewalone\
\end{paste}\end{patch}

\begin{patch}{ugGraphThreeDBuildPageEmpty28}
\begin{paste}{ugGraphThreeDBuildPageEmpty28}{ugGraphThreeDBuildPagePatch28}
\pastebutton{ugGraphThreeDBuildPageEmpty28}{\showpaste}
\tab{5}\spadgraph{makeViewport3D(spaceC, title == "Cube")\free{pol1 pol2 pol3 pol4 pol5 pol6 }}
\end{paste}\end{patch}
```

———-

Coordinate System Transformations

— **ug07.ht** —

```
\begin{page}{ugGraphCoordPage}{7.2.7. Coordinate System Transformations}
\beginscroll

The \axiomType{CoordinateSystems} package provides coordinate
transformation functions that map a given data point from the
```

```
coordinate system specified into the Cartesian coordinate system.  The
default coordinate system, given a triplet \axiom{(f(u,v), u, v)},
assumes that \axiom{z = f(u, v)}, \axiom{x = u} and \axiom{y = v},
that is, reads the coordinates in \axiom{(z, x, y)} order.

\xtc{
}{
\spadpaste{m(u:DFLOAT,v:DFLOAT):DFLOAT == u**2 \bound{m}}
}
%
\psXtc{
Graph plotted in default coordinate system.
}{
\graphpaste{draw(m,0..3,0..5) \free{m}}
}{
\epsffile[0 0 295 295]{../ps/defcoord.ps}
}

The \axiom{z} coordinate comes first since the first argument of the
\axiomFun{draw} command gives its values.  In general, the coordinate
systems Axiom provides, or any that you make up, must provide a map to
an \axiom{(x, y, z)} triplet in order to be compatible with the
\axiomFunFrom{coordinates}{DrawOption} \axiomType{DrawOption}.  Here
is an example.

\xtc{
Define the identity function.
}{
\spadpaste{cartesian(point:Point DFLOAT):Point DFLOAT == point
\bound{cart}}
}
\psXtc{
Pass \axiom{cartesian} as the \axiomFunFrom{coordinates}{DrawOption}
parameter to the \axiomFun{draw} command.
}{
\graphpaste{draw(m,0..3,0..5,coordinates==cartesian) \free{m cart}}
}{
\epsffile[0 0 295 295]{../ps/cartcoord.ps}
}
%

What happened?  The option {\tt coordinates == cartesian} directs
Axiom to treat the dependent variable \axiom{m} defined by
\texht{$m=u^2$}{m=u**2} as the \axiom{x} coordinate.  Thus the triplet
of values \axiom{(m, u, v)} is transformed to coordinates
\axiom{(x, y, z)} and so we get the graph of \texht{$x=y^2$}{x=y**2}.

Here is another example.
The \axiomFunFrom{cylindrical}{CoordinateSystems} transform takes
input of the form \axiom{(w,u,v)}, interprets it in the order
```

```
\texht{($r$,$\theta$,$z$)}{(\axiom{r}, \axiom{theta}, \axiom{z})}
and maps it to the Cartesian coordinates
\texht{$x=r\cos(\theta)$, $y=r\sin(\theta)$, $z=z$}
{\axiom{x = r * cos(theta)}, \axiom{y = r * sin(theta)}, \axiom{z = z}}
in which
\texht{$r$}{\axiom{r}} is the radius,
\texht{$\theta$}{\axiom{theta}} is the angle and
\texht{$z$}{\axiom{z}} is the z-coordinate.
\xtc{
An example using the \axiomFunFrom{cylindrical}{CoordinateSystems}
coordinates for the constant \axiom{r = 3}.
}{
\spadpaste{f(u:DFLOAT,v:DFLOAT):DFLOAT == 3 \bound{f}}
}
\psXtc{
Graph plotted in cylindrical coordinates.
}{
\graphpaste{draw(f,0..\%pi,0..6,coordinates==cylindrical) \free{f}}
}{
\epsffile[0 0 295 295]{../ps/cylcoord.ps}
}

Suppose you would like to specify \smath{z} as a function of \smath{r}
and \texht{$\theta$}{\axiom{theta}} instead of just \smath{r}?  Well,
you still can use the \axiomFun{cylindrical} Axiom transformation but
we have to reorder the triplet before passing it to the
transformation.

\xtc{
First, let's create a point to
work with and call it \axiom{pt} with some color \axiom{col}.
}{
\spadpaste{col := 5 \bound{c}}
}
\xtc{
}{
\spadpaste{pt := point[1,2,3,col]\$(Point DFLOAT) \free{c} \bound{pt}}
}
The reordering you want is
\texht{$(z,r, \theta)$}{\axiom{(z,r,theta)}} to
\texht{$(r, \theta,z)$}{\axiom{(r,theta,z)}}
so that the first element is moved to the third element, while the second
and third elements move forward and the color element does not change.
\xtc{
Define a function \userfun{reorder} to reorder the point elements.
}{
\spadpaste{
reorder(p:Point DFLOAT):Point DFLOAT == point[p.2, p.3, p.1, p.4]
\bound{freo}}
}
```

```
\xtc{
The function moves the second and third elements
forward but the color does not change.
}{
\spadpaste{reorder pt \free{pt freo}}
}
\xtc{
The function \userfun{newmap} converts our reordered version of
the cylindrical coordinate system to the standard
\texht{$(x,y,z)$}{\axiom{(x,y,z)}} Cartesian system.
}{
\spadpaste{
newmap(pt:Point DFLOAT):Point DFLOAT == cylindrical(reorder pt)
\free{freo} \bound{fnewmap}}
}
\xtc{
}{
\spadpaste{newmap pt \free{fnewmap pt} \bound{new}}
}
%
\psXtc{
Graph the same function \axiom{f} using the coordinate mapping
of the function \axiom{newmap}, so it is now interpreted as
\texht{$z=3$}{\axiom{z = 3}}:
}{
\graphpaste{draw(f,0..3,0..2*\%pi,coordinates==newmap) \free{f new}}
}{
\epsffile[0 0 295 295]{../ps/newmap.ps}
}

{\texht{\sloppy}{}
The \axiomType{CoordinateSystems} package exports the following
operations:
\axiomFun{bipolar},
\axiomFun{bipolarCylindrical},
\axiomFun{cartesian},
\axiomFun{conical},
\axiomFun{cylindrical},
\axiomFun{elliptic},
\axiomFun{ellipticCylindrical},
\axiomFun{oblateSpheroidal},
\axiomFun{parabolic},
\axiomFun{parabolicCylindrical},
\axiomFun{paraboloidal},
\axiomFun{polar},
\axiomFun{prolateSpheroidal},
\axiomFun{spherical}, and
\axiomFun{toroidal}.
Use \Browse{} or the \spadcmd{)show} system command
to get more information.
```

```
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugGraphCoordPagePatch1}
\begin{paste}{ugGraphCoordPageFull1}{ugGraphCoordPageEmpty1}
\pastebutton{ugGraphCoordPageFull1}{\hidepaste}
\tab{5}\spadcommand{m(u:DFLOAT,v:DFLOAT):DFLOAT == u**2\bound{m }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphCoordPageEmpty1}
\begin{paste}{ugGraphCoordPageEmpty1}{ugGraphCoordPagePatch1}
\pastebutton{ugGraphCoordPageEmpty1}{\showpaste}
\tab{5}\spadcommand{m(u:DFLOAT,v:DFLOAT):DFLOAT == u**2\bound{m }}
\end{paste}\end{patch}

\begin{patch}{ugGraphCoordPagePatch2}
\begin{paste}{ugGraphCoordPageFull2}{ugGraphCoordPageEmpty2}
\pastebutton{ugGraphCoordPageFull2}{\hidepaste}
\tab{5}\spadgraph{draw(m,0..3,0..5)\free{m }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphcoordpage2.view/image}}{viewalone\space{1
\end{paste}\end{patch}

\begin{patch}{ugGraphCoordPageEmpty2}
\begin{paste}{ugGraphCoordPageEmpty2}{ugGraphCoordPagePatch2}
\pastebutton{ugGraphCoordPageEmpty2}{\showpaste}
\tab{5}\spadgraph{draw(m,0..3,0..5)\free{m }}
\end{paste}\end{patch}

\begin{patch}{ugGraphCoordPagePatch3}
\begin{paste}{ugGraphCoordPageFull3}{ugGraphCoordPageEmpty3}
\pastebutton{ugGraphCoordPageFull3}{\hidepaste}
\tab{5}\spadcommand{cartesian(point:Point DFLOAT):Point DFLOAT == point\bound{cart }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphCoordPageEmpty3}
\begin{paste}{ugGraphCoordPageEmpty3}{ugGraphCoordPagePatch3}
\pastebutton{ugGraphCoordPageEmpty3}{\showpaste}
\tab{5}\spadcommand{cartesian(point:Point DFLOAT):Point DFLOAT == point\bound{cart }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugGraphCoordPagePatch4}
\begin{paste}{ugGraphCoordPageFull4}{ugGraphCoordPageEmpty4}
\pastebutton{ugGraphCoordPageFull4}{\hidepaste}
\tab{5}\spadgraph{draw(m,0..3,0..5,coordinates==cartesian)\free{m cart }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphcoordpage4.view/image}}{viewal
\end{paste}\end{patch}

\begin{patch}{ugGraphCoordPageEmpty4}
\begin{paste}{ugGraphCoordPageEmpty4}{ugGraphCoordPagePatch4}
\pastebutton{ugGraphCoordPageEmpty4}{\showpaste}
\tab{5}\spadgraph{draw(m,0..3,0..5,coordinates==cartesian)\free{m cart }}
\end{paste}\end{patch}

\begin{patch}{ugGraphCoordPagePatch5}
\begin{paste}{ugGraphCoordPageFull5}{ugGraphCoordPageEmpty5}
\pastebutton{ugGraphCoordPageFull5}{\hidepaste}
\tab{5}\spadcommand{f(u:DFLOAT,v:DFLOAT):DFLOAT == 3\bound{f }}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphCoordPageEmpty5}
\begin{paste}{ugGraphCoordPageEmpty5}{ugGraphCoordPagePatch5}
\pastebutton{ugGraphCoordPageEmpty5}{\showpaste}
\tab{5}\spadcommand{f(u:DFLOAT,v:DFLOAT):DFLOAT == 3\bound{f }}
\end{paste}\end{patch}

\begin{patch}{ugGraphCoordPagePatch6}
\begin{paste}{ugGraphCoordPageFull6}{ugGraphCoordPageEmpty6}
\pastebutton{ugGraphCoordPageFull6}{\hidepaste}
\tab{5}\spadgraph{draw(f,0..\%pi,0..6,coordinates==cylindrical)\free{f }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphcoordpage6.view/image}}{viewal
\end{paste}\end{patch}

\begin{patch}{ugGraphCoordPageEmpty6}
\begin{paste}{ugGraphCoordPageEmpty6}{ugGraphCoordPagePatch6}
\pastebutton{ugGraphCoordPageEmpty6}{\showpaste}
\tab{5}\spadgraph{draw(f,0..\%pi,0..6,coordinates==cylindrical)\free{f }}
\end{paste}\end{patch}

\begin{patch}{ugGraphCoordPagePatch7}
\begin{paste}{ugGraphCoordPageFull7}{ugGraphCoordPageEmpty7}
\pastebutton{ugGraphCoordPageFull7}{\hidepaste}
\tab{5}\spadcommand{col := 5\bound{c }}
\indentrel{3}\begin{verbatim}
   (7)  5
                                 Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugGraphCoordPageEmpty7}
\begin{paste}{ugGraphCoordPageEmpty7}{ugGraphCoordPagePatch7}
\pastebutton{ugGraphCoordPageEmpty7}{\showpaste}
\tab{5}\spadcommand{col := 5\bound{c }}
\end{paste}\end{patch}

\begin{patch}{ugGraphCoordPagePatch8}
\begin{paste}{ugGraphCoordPageFull8}{ugGraphCoordPageEmpty8}
\pastebutton{ugGraphCoordPageFull8}{\hidepaste}
\tab{5}\spadcommand{pt := point[1,2,3,col]$(Point DFLOAT)\free{c }\bound{pt }}
\indentrel{3}\begin{verbatim}
   (8)  [1.0,2.0,3.0,5.0]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphCoordPageEmpty8}
\begin{paste}{ugGraphCoordPageEmpty8}{ugGraphCoordPagePatch8}
\pastebutton{ugGraphCoordPageEmpty8}{\showpaste}
\tab{5}\spadcommand{pt := point[1,2,3,col]$(Point DFLOAT)\free{c }\bound{pt }}
\end{paste}\end{patch}

\begin{patch}{ugGraphCoordPagePatch9}
\begin{paste}{ugGraphCoordPageFull9}{ugGraphCoordPageEmpty9}
\pastebutton{ugGraphCoordPageFull9}{\hidepaste}
\tab{5}\spadcommand{reorder(p:Point DFLOAT):Point DFLOAT == point[p.2, p.3, p.1, p.4]\bound{freo }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphCoordPageEmpty9}
\begin{paste}{ugGraphCoordPageEmpty9}{ugGraphCoordPagePatch9}
\pastebutton{ugGraphCoordPageEmpty9}{\showpaste}
\tab{5}\spadcommand{reorder(p:Point DFLOAT):Point DFLOAT == point[p.2, p.3, p.1, p.4]\bound{freo }}
\end{paste}\end{patch}

\begin{patch}{ugGraphCoordPagePatch10}
\begin{paste}{ugGraphCoordPageFull10}{ugGraphCoordPageEmpty10}
\pastebutton{ugGraphCoordPageFull10}{\hidepaste}
\tab{5}\spadcommand{reorder pt\free{pt freo }}
\indentrel{3}\begin{verbatim}
   (10)  [2.0,3.0,1.0,5.0]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphCoordPageEmpty10}
\begin{paste}{ugGraphCoordPageEmpty10}{ugGraphCoordPagePatch10}
```

```
\pastebutton{ugGraphCoordPageEmpty10}{\showpaste}
\tab{5}\spadcommand{reorder pt\free{pt freo }}
\end{paste}\end{patch}

\begin{patch}{ugGraphCoordPagePatch11}
\begin{paste}{ugGraphCoordPageFull11}{ugGraphCoordPageEmpty11}
\pastebutton{ugGraphCoordPageFull11}{\hidepaste}
\tab{5}\spadcommand{newmap(pt:Point DFLOAT):Point DFLOAT == cylindrical(reorder pt)\free{freo }\
\indentrel{3}\begin{verbatim}
                                                Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphCoordPageEmpty11}
\begin{paste}{ugGraphCoordPageEmpty11}{ugGraphCoordPagePatch11}
\pastebutton{ugGraphCoordPageEmpty11}{\showpaste}
\tab{5}\spadcommand{newmap(pt:Point DFLOAT):Point DFLOAT == cylindrical(reorder pt)\free{freo }\
\end{paste}\end{patch}

\begin{patch}{ugGraphCoordPagePatch12}
\begin{paste}{ugGraphCoordPageFull12}{ugGraphCoordPageEmpty12}
\pastebutton{ugGraphCoordPageFull12}{\hidepaste}
\tab{5}\spadcommand{newmap pt\free{fnewmap pt }\bound{new }}
\indentrel{3}\begin{verbatim}
   (12)
   [- 1.9799849932008908,0.28224001611973443,1.0,5.0]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugGraphCoordPageEmpty12}
\begin{paste}{ugGraphCoordPageEmpty12}{ugGraphCoordPagePatch12}
\pastebutton{ugGraphCoordPageEmpty12}{\showpaste}
\tab{5}\spadcommand{newmap pt\free{fnewmap pt }\bound{new }}
\end{paste}\end{patch}

\begin{patch}{ugGraphCoordPagePatch13}
\begin{paste}{ugGraphCoordPageFull13}{ugGraphCoordPageEmpty13}
\pastebutton{ugGraphCoordPageFull13}{\hidepaste}
\tab{5}\spadgraph{draw(f,0..3,0..2*\%pi,coordinates==newmap)\free{f new }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphcoordpage13.view/image}}{viewa
\end{paste}\end{patch}

\begin{patch}{ugGraphCoordPageEmpty13}
\begin{paste}{ugGraphCoordPageEmpty13}{ugGraphCoordPagePatch13}
\pastebutton{ugGraphCoordPageEmpty13}{\showpaste}
\tab{5}\spadgraph{draw(f,0..3,0..2*\%pi,coordinates==newmap)\free{f new }}
\end{paste}\end{patch}
```

Three-Dimensional Clipping

— ug07.ht —

```
\begin{page}{ugGraphClipPage}{7.2.8. Three-Dimensional Clipping}
\beginscroll

A \threedim{} graph can be explicitly clipped within the \axiomFun{draw}
command by indicating a minimum and maximum threshold for the
given function definition.
These thresholds can be defined using the Axiom \axiomFun{min}
and \axiomFun{max} functions.
\xtc{
}{
\begin{spadsrc}[\bound{g}]
gamma(x,y) ==
  g := Gamma complex(x,y)
  point [x, y, max( min(real g, 4), -4), argument g]
\end{spadsrc}
}
\psXtc{
Here is an example that clips
the gamma function in order to eliminate the extreme divergence it creates.
}{
\graphpaste{
draw(gamma,-\%pi..\%pi,-\%pi..\%pi,var1Steps==50,var2Steps==50) \free{g}}
}{
\epsffile[0 0 295 295]{../ps/clipgamma.ps}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugGraphClipPagePatch1}
\begin{paste}{ugGraphClipPageFull1}{ugGraphClipPageEmpty1}
\pastebutton{ugGraphClipPageFull1}{\hidepaste}
\tab{5}\spadcommand{gamma(x,y) ==
  g := Gamma complex(x,y)
  point [x, y, max( min(real g, 4), -4), argument g]
\bound{g }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugGraphClipPageEmpty1}
\begin{paste}{ugGraphClipPageEmpty1}{ugGraphClipPagePatch1}
\pastebutton{ugGraphClipPageEmpty1}{\showpaste}
\tab{5}\spadcommand{gamma(x,y) ==
  g := Gamma complex(x,y)
  point [x, y, max( min(real g, 4), -4), argument g]
\bound{g }}
\end{paste}\end{patch}

\begin{patch}{ugGraphClipPagePatch2}
\begin{paste}{ugGraphClipPageFull2}{ugGraphClipPageEmpty2}
\pastebutton{ugGraphClipPageFull2}{\hidepaste}
\tab{5}\spadgraph{draw(gamma,-\%pi..\%pi,-\%pi..\%pi,var1Steps==50,var2Steps==50)\free{g }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/uggraphclippage2.view/image}}{viewalo
\end{paste}\end{patch}

\begin{patch}{ugGraphClipPageEmpty2}
\begin{paste}{ugGraphClipPageEmpty2}{ugGraphClipPagePatch2}
\pastebutton{ugGraphClipPageEmpty2}{\showpaste}
\tab{5}\spadgraph{draw(gamma,-\%pi..\%pi,-\%pi..\%pi,var1Steps==50,var2Steps==50)\free{g }}
\end{paste}\end{patch}
```

———-

Three-Dimensional Control-Panel

— ug07.ht —

```
\begin{page}{ugGraphThreeDControlPage}
{7.2.9. Three-Dimensional Control-Panel}
\beginscroll
Once you have created a viewport, move your mouse to the viewport and
click with your left mouse button.  This displays a control-panel on
the side of the viewport that is closest to where you clicked.

\subsubsection{Transformations}

We recommend you first select the {\bf Bounds} button while
executing transformations since the bounding box displayed
indicates the object's position as it changes.
%
\indent{0}
\beginitems
%
\item[Rotate:]  A rotation transformation occurs by clicking the mouse
within the {\bf Rotate} window in the upper left corner of the
control-panel.
```

```
The rotation is computed in spherical coordinates, using the
horizontal mouse position to increment or decrement the value of
the longitudinal angle \texht{$\theta$}{\axiom{theta}} within the
range of 0 to \texht{2$\pi$}{2*pi} and the vertical mouse position
to increment or decrement the value of the latitudinal angle
\texht{$\phi$}{\axiom{phi}} within the range of \texht{-$\pi$}{pi}
to \texht{$\pi$}{pi}.
The active mode of rotation is displayed in green on a color
monitor or in clear text on a black and white monitor, while the
inactive mode is displayed in red for color display or a mottled
pattern for black and white.
%
\indent{0}
\beginitems

\item[origin:] The {\bf origin} button indicates that the rotation is
to occur with respect to the origin of the viewing space, that is
indicated by the axes.

\item[object:]  The {\bf object} button indicates that the
rotation is to occur with respect to the center of volume of the object,
independent of the axes' origin position.
\enditems
\indent{0}
%
\item[Scale:]  A scaling transformation occurs by clicking the mouse
within the {\bf Scale} window in the upper center of the
control-panel, containing a zoom arrow.
The axes along which the scaling is to occur are indicated by
selecting the appropriate button above the zoom arrow window.
The selected axes are displayed in green on a color monitor or in
clear text on a black and white monitor, while the unselected axes
are displayed in red for a color display or a mottled pattern for
black and white.
%
\indent{0}
\beginitems
%
\item[uniform:]  Uniform scaling along the {\tt x}, {\tt y}
and {\tt z} axes occurs when all the axes buttons are selected.

\item[non-uniform:] If any of the axes buttons are not selected,
non-uniform scaling occurs, that is, scaling occurs only in the
direction of the axes that are selected.
\enditems
\indent{0}

\item[Translate:] Translation occurs by indicating with the mouse in
the {\bf Translate} window the direction you want the graph to move.
This window is located in the upper right corner of the control-panel
```

```
and contains a potentiometer with crossed arrows pointing up, down,
left and right.  Along the top of the {\bf Translate} window are three
buttons ({\bf XY}, {\bf XZ}, and {\bf YZ}) indicating the three
orthographic projection planes.  Each orientates the group as a view
into that plane.  Any translation of the graph occurs only along this
plane.
\enditems
\indent{0}

\subsubsection{Messages}

The window directly below the potentiometer windows for
transformations is used to display system messages relating to the
viewport, the control-panel and the current graph displaying status.

\subsubsection{Colormap}

Directly below the message window is the colormap range indicator
window.  The Axiom Colormap shows a sampling of the spectrum from
which hues can be drawn to represent the colors of a surface.  The
Colormap is composed of five shades for each of the hues along this
spectrum.  By moving the markers above and below the Colormap, the
range of hues that are used to color the existing surface are set.
The bottom marker shows the hue for the low end of the color range and
the top marker shows the hue for the upper end of the range.  Setting
the bottom and top markers at the same hue results in monochromatic
smooth shading of the graph when {\bf Smooth} mode is selected.  At
each end of the Colormap are {\bf +} and {\bf -} buttons.  When
clicked on, these increment or decrement the top or bottom marker.

\subsubsection{Buttons}

Below the Colormap window and to the left are located various
buttons that determine the characteristics of a graph.
The buttons along the bottom and right hand side all have special
meanings; the remaining buttons in the first row indicate the mode
or style used to display the graph.
The second row are toggles that turn on or off a property of the
graph.
On a color monitor, the property is on if green (clear text, on a
monochrome monitor) and off if red (mottled pattern, on a
monochrome monitor).
Here is a list of their functions.
%
\indent{0}
\beginitems

\item[Wire] displays surface and tube plots as a wireframe image in a
single color (blue) with no hidden surfaces removed, or displays space
curve plots in colors based upon their parametric variables.  This is
```

the fastest mode for displaying a graph. This is very useful when you
want to find a good orientation of your graph.

\item[Solid] displays the graph with hidden
surfaces removed, drawing each polygon beginning with the furthest
from the viewer.
The edges of the polygons are displayed in the hues specified by
the range in the Colormap window.
%
\item[Shade] displays the graph with hidden
surfaces removed and with the polygons shaded, drawing each
polygon beginning with the furthest from the viewer.
Polygons are shaded in the hues specified by the range in the
Colormap window using the Phong illumination model.
%
\item[Smooth] displays the graph using a
renderer that computes the graph one line at a time.
The location and color of the graph at each visible point on the
screen are determined and displayed using the Phong illumination
model.
Smooth shading is done in one of two ways, depending on the range
selected in the colormap window and the number of colors available
from the hardware and/or window manager.
When the top and bottom markers of the colormap range are set to
different hues, the graph is rendered by dithering between the
transitions in color hue.
When the top and bottom markers of the colormap range are set to
the same hue, the graph is rendered using the Phong smooth shading
model.
However, if enough colors cannot be allocated for this purpose,
the renderer reverts to the color dithering method until a
sufficient color supply is available.
For this reason, it may not be possible to render multiple Phong
smooth shaded graphs at the same time on some systems.

\item[Bounds] encloses the entire volume of the viewgraph within a
bounding box, or removes the box if previously selected. The region
that encloses the entire volume of the viewport graph is displayed.

\item[Axes] displays Cartesian
coordinate axes of the space, or turns them off if previously selected.

\item[Outline] causes quadrilateral polygons forming the graph surface
to be outlined in black when the graph is displayed in {\bf Shade}
mode.

\item[BW] converts a color viewport to black and white, or vice-versa.
When this button is selected the control-panel and viewport switch to
an immutable colormap composed of a range of grey scale patterns or
tiles that are used wherever shading is necessary.

```
\item[Light] takes you to a control-panel described below.
%
\item[ViewVolume] takes you to another control-panel as described below.

\item[Save] creates a menu of the possible file types that can be
written using the control-panel.  The {\bf Exit} button leaves the
save menu.  The {\bf Pixmap} button writes an Axiom pixmap of the
current viewport contents.  The file is called {\bf axiom3D.pixmap}
and is located in the directory from which Axiom or {\bf viewalone}
was started.  The {\bf PS} button writes the current viewport contents
to PostScript output rather than to the viewport window.  By default
the file is called {\bf axiom3D.ps}; however, if a file name is
specified in the user's {\bf .Xdefaults} file it is used.  The file is
placed in the directory from which the Axiom or {\bf viewalone}
session was begun.  See also the
\axiomFunFrom{write}{ThreeDimensionalViewport} function.

\item[Reset] returns the object transformation
characteristics back to their initial states.
%
\item[Hide] causes the control-panel for the
corresponding viewport to disappear from the screen.
%
\item[Quit]  queries whether the current viewport
session should be terminated.
\enditems
\indent{0}

\subsubsection{Light}

The {\bf Light} button changes the control-panel into the
{\bf Lighting Control-Panel}.  At the top of this panel, the three axes
are shown with the same orientation as the object.  A light vector from
the origin of the axes shows the current position of the light source
relative to the object.  At the bottom of the panel is an {\bf Abort}
button that cancels any changes to the lighting that were made, and a
{\bf Return} button that carries out the current set of lighting changes
on the graph.
%
\indent{0}
\beginitems
%
\item[XY:]  The {\bf XY} lighting axes window is below the
{\bf Lighting Control-Panel} title and to the left.
This changes the light vector within the {\bf XY} view plane.
%
\item[Z:]  The {\bf Z} lighting axis window is below the
{\bf Lighting Control-Panel} title and in the center.  This
changes the {\bf Z}
```

```
location of the light vector.
%
\item[Intensity:]
Below the {\bf Lighting Control-Panel} title
and to the right is the light intensity meter.
Moving the intensity indicator down decreases the amount of
light emitted from the light source.
When the indicator is at the top of the meter the light source is
emitting at 100\% intensity.
At the bottom of the meter the light source is emitting at a level
slightly above ambient lighting.
\enditems
\indent{0}

\subsubsection{View Volume}

The {\bf View Volume} button changes the control-panel into
the {\bf Viewing Volume Panel}.
At the bottom of the viewing panel is an {\bf Abort} button that
cancels any changes to the viewing volume that were made and a
{\it Return} button that carries out the current set of
viewing changes to the graph.

\indent{0}
\beginitems

\item[Eye Reference:] At the top of this panel is the {\bf Eye
Reference} window.  It shows a planar projection of the viewing
pyramid from the eye of the viewer relative to the location of the
object.  This has a bounding region represented by the rectangle on
the left.  Below the object rectangle is the {\bf Hither} window.  By
moving the slider in this window the hither clipping plane sets the
front of the view volume.  As a result of this depth clipping all
points of the object closer to the eye than this hither plane are not
shown.  The {\bf Eye Distance} slider to the right of the {\bf Hither}
slider is used to change the degree of perspective in the image.

\item[Clip Volume:] The {\bf Clip Volume} window is at the bottom of
the {\bf Viewing Volume Panel}.  On the right is a {\bf Settings}
menu.  In this menu are buttons to select viewing attributes.
Selecting the {\bf Perspective} button computes the image using
perspective projection.  The {\bf Show Region} button indicates
whether the clipping region of the volume is to be drawn in the
viewport and the {\bf Clipping On} button shows whether the view
volume clipping is to be in effect when the image is drawn.  The left
side of the {\bf Clip Volume} window shows the clipping boundary of
the graph.  Moving the knobs along the {\bf X}, {\bf Y}, and {\bf Z}
sliders adjusts the volume of the clipping region accordingly.
\enditems
\indent{0}
```

```
\endscroll
\autobuttons
\end{page}
```

———-

Operations for Three-Dimensional Graphics

— ug07.ht —

```
\begin{page}{ugGraphThreeDopsPage}
{7.2.10. Operations for Three-Dimensional Graphics}
\beginscroll

Here is a summary of useful Axiom operations for \threedim{} graphics.
Each operation name is followed by a list of arguments.  Each argument
is written as a variable informally named according to the type of the
argument (for example, {\it integer}).  If appropriate, a default
value for an argument is given in parentheses immediately following
the name.

%
\texht{\bgroup\hbadness = 10001\sloppy}{}
\indent{0}
\beginitems
%
\item[\axiomFun{adaptive3D?}]\funArgs{}
tests whether space curves are to be plotted
according to the
adaptive refinement algorithm.

%
\item[\axiomFun{axes}]\funArgs{viewport, string\argDef{"on"}}
turns the axes on and off.

%
\item[\axiomFun{close}]\funArgs{viewport}
closes the viewport.

%
\item[\axiomFun{colorDef}]\funArgs{viewport,
\subscriptIt{color}{1}\argDef{1}, \subscriptIt{color}{2}\argDef{27}}
sets the colormap
range to be from
\subscriptIt{color}{1} to \subscriptIt{color}{2}.
```

```
%
\item[\axiomFun{controlPanel}]\funArgs{viewport, string\argDef{"off"}}
declares whether the
control-panel for the viewport is to be displayed or not.

%
\item[\axiomFun{diagonals}]\funArgs{viewport, string\argDef{"off"}}
declares whether the
polygon outline includes the diagonals or not.

%
\item[\axiomFun{drawStyle}]\funArgs{viewport, style}
selects which of four drawing styles
are used: {\tt "wireMesh", "solid", "shade",} or {\tt "smooth".}

%
\item[\axiomFun{eyeDistance}]\funArgs{viewport,float\argDef{500}}
sets the distance of the eye from the origin of the object
for use in the \axiomFunFrom{perspective}{ThreeDimensionalViewport}.

%
\item[\axiomFun{key}]\funArgs{viewport}
returns the operating
system process ID number for the viewport.

%
\item[\axiomFun{lighting}]\funArgs{viewport,
\subscriptText{float}{x}\argDef{-0.5},
\subscriptText{float}{y}\argDef{0.5}, \subscriptText{float}{z}\argDef{0.5}}
sets the Cartesian
coordinates of the light source.

%
\item[\axiomFun{modifyPointData}]\funArgs{viewport,integer,point}
replaces the coordinates of the point with
the index {\it integer} with {\it point}.

%
\item[\axiomFun{move}]\funArgs{viewport,
\subscriptText{integer}{x}\argDef{viewPosDefault},
\subscriptText{integer}{y}\argDef{viewPosDefault}}
moves the upper
left-hand corner of the viewport to screen position
\allowbreak
({\small \subscriptText{integer}{x}, \subscriptText{integer}{y}}).

%
\item[\axiomFun{options}]\funArgs{viewport}
returns a list of all current draw options.
```

```
%
\item[\axiomFun{outlineRender}]\funArgs{viewport, string\argDef{"off"}}
turns polygon outlining
off or on when drawing in {\tt "shade"} mode.

%
\item[\axiomFun{perspective}]\funArgs{viewport, string\argDef{"on"}}
turns perspective
viewing on and off.

%
\item[\axiomFun{reset}]\funArgs{viewport}
resets the attributes of a viewport to their
initial settings.

%
\item[\axiomFun{resize}]\funArgs{viewport,
\subscriptText{integer}{width} \argDef{viewSizeDefault},
\subscriptText{integer}{height} \argDef{viewSizeDefault}}
resets the width and height
values for a viewport.

%
\item[\axiomFun{rotate}]\funArgs{viewport,
\subscriptText{number}{\texht{$\theta$}{\axiom{theta}}}
\argDef{viewThetaDefault},
\subscriptText{number}{\texht{$\phi$}{\axiom{phi}}}
\argDef{viewPhiDefault}}
rotates the viewport by rotation angles for longitude
({\it \texht{$\theta$}{\axiom{theta}}}) and
latitude ({\it \texht{$\phi$}{\axiom{phi}}}).
Angles designate radians if given as floats, or degrees if given
as integers.

%
\item[\axiomFun{setAdaptive3D}]\funArgs{boolean\argDef{true}}
sets whether space curves are to be plotted
according to the adaptive
refinement algorithm.

%
\item[\axiomFun{setMaxPoints3D}]\funArgs{integer\argDef{1000}}
 sets the default maximum number of possible
points to be used when constructing a \threedim{} space curve.

%
\item[\axiomFun{setMinPoints3D}]\funArgs{integer\argDef{49}}
sets the default minimum number of possible
points to be used when constructing a \threedim{} space curve.
```

```
%
\item[\axiomFun{setScreenResolution3D}]\funArgs{integer\argDef{500}}
sets the default screen resolution constant
used in setting the computation limit of adaptively
generated \threedim{} space curve plots.

%
\item[\axiomFun{showRegion}]\funArgs{viewport, string\argDef{"off"}}
declares whether the bounding
box of a graph is shown or not.
%
\item[\axiomFun{subspace}]\funArgs{viewport}
returns the space component.
%
\item[\axiomFun{subspace}]\funArgs{viewport, subspace}
resets the space component
to {\it subspace}.

%
\item[\axiomFun{title}]\funArgs{viewport, string}
gives the viewport the
title {\it string}.

%
\item[\axiomFun{translate}]\funArgs{viewport,
\subscriptText{float}{x}\argDef{viewDeltaXDefault},
\subscriptText{float}{y}\argDef{viewDeltaYDefault}}
translates the object horizontally and vertically
relative to the center of the viewport.

%
\item[\axiomFun{intensity}]\funArgs{viewport,float\argDef{1.0}}
resets the intensity {\it I} of the light source,
\texht{$0 \le I \le 1.$}{{\it 0 \le I \le 1}.}

%
\item[\axiomFun{tubePointsDefault}]\funArgs{\optArg{integer\argDef{6}}}
sets or indicates the default number of
vertices defining the polygon that is used to create a tube around
a space curve.

%
\item[\axiomFun{tubeRadiusDefault}]\funArgs{\optArg{float\argDef{0.5}}}
sets or indicates the default radius of
the tube that encircles a space curve.

%
\item[\axiomFun{var1StepsDefault}]\funArgs{\optArg{integer\argDef{27}}}
sets or indicates the default number of
increments into which the grid defining a surface plot is subdivided with
```

```
respect to the first parameter declared in the surface function.

%
\item[\axiomFun{var2StepsDefault}]\funArgs{\optArg{integer\argDef{27}}}
sets or indicates the default number of
increments into which the grid defining a surface plot is subdivided with
respect to the second parameter declared in the surface function.

%
\item[\axiomFun{viewDefaults}]\funArgs{{\tt [}\subscriptText{integer}{%
point}, \subscriptText{integer}{line}, \subscriptText{integer}{axes},
\subscriptText{integer}{units}, \subscriptText{float}{point},
\allowbreak\subscriptText{list}{position},
\subscriptText{list}{size}{\tt ]}}
resets the default settings for the
point color, line color, axes color, units color, point size,
viewport upper left-hand corner position, and the viewport size.

%
\item[\axiomFun{viewDeltaXDefault}]\funArgs{\optArg{float\argDef{0}}}
resets the default horizontal offset
from the center of the viewport, or returns the
current default offset if no argument is given.

%
\item[\axiomFun{viewDeltaYDefault}]\funArgs{\optArg{float\argDef{0}}}
resets the default vertical offset
from the center of the viewport, or returns the
current default offset if no argument is given.

%
\item[\axiomFun{viewPhiDefault}]\funArgs{\optArg{float
\argDef{-\texht{$\pi$}{{\it pi}}/4}}}
resets the default latitudinal view angle,
or returns the current default angle if no argument is given.
\texht{$\phi$}{{\it phi}} is set to this value.

%
\item[\axiomFun{viewpoint}]\funArgs{viewport, \subscriptText{float}{x},
\subscriptText{float}{y}, \subscriptText{float}{z}}
sets the viewing position in Cartesian coordinates.

%
\item[\axiomFun{viewpoint}]\funArgs{viewport,
\subscriptText{float}{\texht{$\theta$}{\axiom{theta}}},
\subscriptText{Float}{\texht{$\phi$}{\axiom{phi}}}}
sets the viewing position in spherical coordinates.

%
\item[\axiomFun{viewpoint}]\funArgs{viewport,
```

```
\subscriptText{Float}{\texht{$\theta$}{\axiom{theta}}},
\subscriptText{Float}{\texht{$\phi$}{\axiom{phi}}},
\subscriptText{Float}{scaleFactor},
\subscriptText{Float}{xOffset}, \subscriptText{Float}{yOffset}}
sets the viewing position in spherical coordinates,
the scale factor, and offsets.
\texht{$\theta$}{{\it theta}} (longitude) and
\texht{$\phi$}{{\it phi}} (latitude) are in radians.

%
\item[\axiomFun{viewPosDefault}]\funArgs{\optArg{list\argDef{[0,0]}}}
sets or indicates the position of the upper
left-hand corner of a \twodim{} viewport, relative to the display root
window (the upper left-hand corner of the display is \axiom{[0, 0]}).

%
\item[\axiomFun{viewSizeDefault}]\funArgs{\optArg{list\argDef{[400,400]}}}
sets or indicates the width and height dimensions
of a viewport.

%
\item[\axiomFun{viewThetaDefault}]
\funArgs{\optArg{float\argDef{\texht{$\pi$}{{\it pi}}/4}}}
resets the default longitudinal view angle,
or returns the current default angle if no argument is given.
When a parameter is specified, the default longitudinal view angle
\texht{$\theta$}{{\it theta}} is set to this value.

%
\item[\axiomFun{viewWriteAvailable}]
\funArgs{\optArg{list\argDef{["pixmap",
"bitmap", "postscript", "image"}}}
indicates the possible file types
that can be created with the
\axiomFunFrom{write}{ThreeDimensionalViewport} function.

%
\item[\axiomFun{viewWriteDefault}]\funArgs{\optArg{list\argDef{[]}}}
sets or indicates the default types of files
that are created in addition to the {\bf data} file when a
\axiomFunFrom{write}{ThreeDimensionalViewport} command
is executed on a viewport.

%
\item[\axiomFun{viewScaleDefault}]\funArgs{\optArg{float}}
sets the default scaling factor, or returns
the current factor if no argument is given.

%
\item[\axiomFun{write}]\funArgs{viewport, directory, \optArg{option}}
```

```
writes the file {\bf data} for {\it viewport}
in the directory {\it directory}.
An optional third argument specifies a file type (one of {\tt
pixmap}, {\tt bitmap}, {\tt postscript}, or {\tt image}), or a
list of file types.
An additional file is written for each file type listed.

%
\item[\axiomFun{scale}]\funArgs{viewport, float\argDef{2.5}}
specifies the scaling factor.
\enditems
\indent{0}
\texht{\egroup}{}

\endscroll
\autobuttons
\end{page}
```

———-

Customization using .Xdefaults

— ug07.ht —

```
\begin{page}{ugXdefaultsPage}{7.2.11. Customization using .Xdefaults}
\beginscroll

Both the \twodim{} and \threedim{} drawing facilities consult
the {\bf .Xdefaults} file for various defaults.
The list of defaults that are recognized by the graphing routines
is discussed in this section.
These defaults are preceded by {\tt Axiom.3D.}
for \threedim{} viewport defaults, {\tt Axiom.2D.}
for \twodim{} viewport defaults, or {\tt Axiom*} (no dot) for
those defaults that are acceptable to either viewport type.

%
\indent{0}
\beginitems
%
\item[{\tt Axiom*buttonFont:\ \it font}] \ \newline
This indicates which
font type is used for the button text on the control-panel.
\xdefault{Rom11}
%
\item[{\tt Axiom.2D.graphFont:\ \it font}] \quad (2D only) \newline
This indicates
```

```
which font type is used for displaying the graph numbers and
slots in the {\bf Graphs} section of the \twodim{} control-panel.
\xdefault{Rom22}
%
\item[{\tt Axiom.3D.headerFont:\ \it font}] \ \newline
This indicates which
font type is used for the axes labels and potentiometer
header names on \threedim{} viewport windows.
This is also used for \twodim{} control-panels for indicating
which font type is used for potentionmeter header names and
multiple graph title headers.
%for example, {\tt Axiom.2D.headerFont: 8x13}.
\xdefault{Itl14}
%
\item[{\tt Axiom*inverse:\ \it switch}] \ \newline
This indicates whether the
background color is to be inverted from white to black.
If {\tt on}, the graph viewports use black as the background
color.
If {\tt off} or no declaration is made, the graph viewports use a
white background.
\xdefault{off}
%
\item[{\tt Axiom.3D.lightingFont:\ \it font}] \quad (3D only) \newline
This indicates which font type is used for the {\bf x},
{\bf y}, and {\bf z} labels of the two lighting axes potentiometers,
and for the {\bf Intensity} title on the lighting control-panel.
\xdefault{Rom10}
%
\item[
{\tt Axiom.2D.messageFont, Axiom.3D.messageFont:\ \it font}] \ \newline
These indicate the font type
to be used for the text in the control-panel message window.
\xdefault{Rom14}
%
\item[{\tt Axiom*monochrome:\ \it switch}] \ \newline
This indicates whether the
graph viewports are to be displayed as if the monitor is black and
white, that is, a 1 bit plane.
If {\tt on} is specified, the viewport display is black and white.
If {\tt off} is specified, or no declaration for this default is
given, the viewports are displayed in the normal fashion for the
monitor in use.
\xdefault{off}
%
\item[{\tt Axiom.2D.postScript:\ \it filename}] \ \newline
This specifies
the name of the file that is generated when a 2D PostScript graph
is saved.
\xdefault{axiom2D.ps}
```

```
%
\item[{\tt Axiom.3D.postScript:\ \it filename}] \ \newline
This specifies
the name of the file that is generated when a 3D PostScript graph
is saved.
\xdefault{axiom3D.ps}
%
\item[{\tt Axiom*titleFont \it font}] \ \newline
This
indicates which font type is used
for the title text and, for \threedim{} graphs,
in the lighting and viewing-volume control-panel windows.
\xdefault{Rom14}
%
\item[{\tt Axiom.2D.unitFont:\ \it font}] \quad (2D only) \newline
This indicates
which font type is used for displaying the unit labels on
\twodim{} viewport graphs.
\xdefault{6x10}
%
\item[{\tt Axiom.3D.volumeFont:\ \it font}] \quad (3D only) \newline
This indicates which font type is used for the {\bf x},
{\bf y}, and {\bf z} labels of the clipping region sliders; for the
{\bf Perspective}, {\bf Show Region}, and {\bf Clipping On} buttons under
{\bf Settings}, and above the windows for the {\bf Hither} and
{\bf Eye Distance} sliders in the {\bf Viewing Volume Panel} of the
\threedim{} control-panel.
\xdefault{Rom8}
\enditems
\indent{0}
\endscroll
\autobuttons
\end{page}
```

Chapter 12

Users Guide Chapter 8 (ug08.ht)

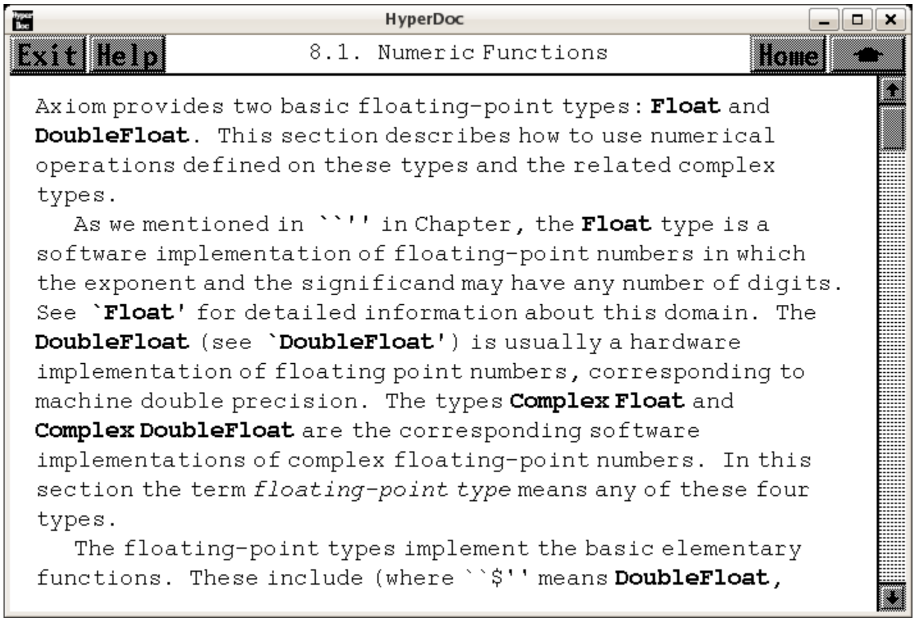
Advanced Problem Solving

— **ug08.ht** —

```
\begin{page}{ugProblemPage}{8. Advanced Problem Solving}
\beginscroll

In this chapter we describe techniques useful in solving advanced problems
with Axiom.

\beginmenu
    \menudownlink{{8.1. Numeric Functions}}
{ugProblemNumericPage}
```

```
    \menudownlink{{8.2. Polynomial Factorization}}
{ugProblemFactorPage}
    \menudownlink{{8.3. Manipulating Symbolic Roots of a Polynomial}}
{ugProblemSymRootPage}
    \menudownlink{{8.4. Computation of Eigenvalues and Eigenvectors}}
{ugProblemEigenPage}
    \menudownlink{{8.5. Solution of Linear and Polynomial Equations}}
{ugProblemLinPolEqnPage}
    \menudownlink{{8.6. Limits}}
{ugProblemLimitsPage}
    \menudownlink{{8.7. Laplace Transforms}}
{ugProblemLaplacePage}
    \menudownlink{{8.8. Integration}}
{ugProblemIntegrationPage}
    \menudownlink{{8.9. Working with Power Series}}
{ugProblemSeriesPage}
    \menudownlink{{8.10. Solution of Differential Equations}}
{ugProblemDEQPage}
    \menudownlink{{8.11. Finite Fields}}
{ugProblemFinitePage}
    \menudownlink{{8.12. Primary Decomposition of Ideals}}
{ugProblemIdealPage}
    \menudownlink{{8.13. Computation of Galois Groups}}
{ugProblemGaloisPage}
    \menudownlink{
{8.14. Non-Associative Algebras and Modelling Genetic Laws}}
{ugProblemGeneticPage}
\endmenu
\endscroll
\autobuttons
\end{page}
```

Numeric Functions



⇐ "Axiom Number Types" (NumberPage) 3.80 on page 1021
⇐ "Complex" (ComplexXmpPage) 3.16 on page 254
⇒ "An Overview of Axiom" (ugIntroPage) 6 on page 1483
⇒ "Float" (FloatXmpPage) 3.41 on page 487
⇒ "DoubleFloat" (DoubleFloatXmpPage) 3.23 on page 369

— **ug08.ht** —

```
\begin{page}{ugProblemNumericPage}{8.1. Numeric Functions}
\beginscroll
Axiom provides two basic floating-point types: \axiomType{Float} and
\axiomType{DoubleFloat}.  This section describes how to use numerical
operations defined on these types and the related complex types.

As we mentioned in \downlink{``An Overview of Axiom''}{ugIntroPage} in
Chapter 1\ignore{ugIntro}, the \axiomType{Float} type is
a software implementation of floating-point numbers in which the
exponent and the significand may have any number of digits.  See
\downlink{`Float'}{FloatXmpPage}\ignore{Float} for detailed
information about this domain.  The \axiomType{DoubleFloat} (see
\downlink{`DoubleFloat'}{DoubleFloatXmpPage}\ignore{DoubleFloat}) is
usually a hardware implementation of floating point numbers,
corresponding to machine double precision.  The types
\axiomType{Complex Float} and \axiomType{Complex DoubleFloat} are the
corresponding software implementations of complex floating-point
numbers.  In this section the term {\it floating-point type} means any
```

```
of these four types.

The floating-point types implement the basic elementary functions.
These include (where \axiomSyntax{\$} means
\axiomType{DoubleFloat},
\axiomType{Float},
\axiomType{Complex DoubleFloat}, or
\axiomType{Complex Float}):

\noindent
\axiomFun{exp},  \axiomFun{log}: \axiom{\$ -> \$} \newline
\axiomFun{sin},  \axiomFun{cos}, \axiomFun{tan}, \axiomFun{cot},
\axiomFun{sec}, \axiomFun{csc}: \axiom{\$ -> \$} \newline
\axiomFun{sin},  \axiomFun{cos}, \axiomFun{tan}, \axiomFun{cot},
\axiomFun{sec}, \axiomFun{csc}: \axiom{\$ -> \$}  \newline
\axiomFun{asin}, \axiomFun{acos}, \axiomFun{atan}, \axiomFun{acot},
\axiomFun{asec}, \axiomFun{acsc}: \axiom{\$ -> \$}  \newline
\axiomFun{sinh},  \axiomFun{cosh}, \axiomFun{tanh}, \axiomFun{coth},
\axiomFun{sech}, \axiomFun{csch}: \axiom{\$ -> \$}  \newline
\axiomFun{asinh}, \axiomFun{acosh}, \axiomFun{atanh}, \axiomFun{acoth},
\axiomFun{asech}, \axiomFun{acsch}: \axiom{\$ -> \$}  \newline
\axiomFun{pi}: \axiom{() -> \$}  \newline
\axiomFun{sqrt}: \axiom{\$ -> \$} \newline
\axiomFun{nthRoot}: \axiom{(\$, Integer) -> \$}  \newline
\axiomFunFrom{**}{Float}: \axiom{(\$, Fraction Integer) -> \$} \newline
\axiomFunFrom{**}{Float}: \axiom{(\$,\$) -> \$}  \newline

The handling of roots depends on whether the floating-point type
is real or complex: for the real floating-point types,
\axiomType{DoubleFloat} and \axiomType{Float}, if a real root exists
the one with the same sign as the radicand is returned; for the
complex floating-point types, the principal value is returned.
Also, for real floating-point types the inverse functions
produce errors if the results are not real.
This includes cases such as \axiom{asin(1.2)}, \axiom{log(-3.2)},
\axiom{sqrt(-1.1)}.
%
\xtc{
The default floating-point type is \axiomType{Float} so to evaluate
functions using \axiomType{Float} or \axiomType{Complex Float}, just
use normal decimal notation.
}{
\spadpaste{exp(3.1)}
}
\xtc{
}{
\spadpaste{exp(3.1 + 4.5 * \%i)}
}
\xtc{
To evaluate functions using \axiomType{DoubleFloat}
```

```
or \axiomType{Complex DoubleFloat},
a declaration or conversion is required.
}{
\spadpaste{r: DFLOAT := 3.1; t: DFLOAT := 4.5; exp(r + t*\%i)}
}
\xtc{
}{
\spadpaste{exp(3.1::DFLOAT + 4.5::DFLOAT * \%i)}
}

A number of special functions are provided by the package
\axiomType{DoubleFloatSpecialFunctions} for the machine-precision
floating-point types.  The special functions provided are listed
below, where \axiom{F} stands for the types \axiomType{DoubleFloat}
and \axiomType{Complex DoubleFloat}.  The real versions of the
functions yield an error if the result is not real.

\noindent
\axiomFun{Gamma}: \axiom{F -> F}\hfill\newline
\axiom{Gamma(z)} is the Euler gamma function,
   \texht{$\Gamma(z)$}{\axiom{Gamma(z)}},
   defined by
   \texht{
\narrowDisplay{\Gamma(z) = \int_{0}^{\infty} t^{z-1} e^{-t} dt.}%
   }{
\newline
\centerline{{
\axiom{Gamma(z) = integrate(t**(z-1)*exp(-t), t=0..\%infinity).}}}
   }

\noindent
\axiomFun{Beta}: \axiom{F -> F}\hfill\newline
   \axiom{Beta(u, v)} is the Euler Beta function,
   \texht{$B(u,v)$}{\axiom{B(u,v)}}, defined by
   \texht{\narrowDisplay{B(u,v) = \int_{0}^{1} t^{u-1} (1-t)^{v-1} dt.}%
   }{
\newline
\centerline{{
\axiom{Beta(u,v) = integrate(t**(u-1)*(1-t)**(b-1), t=0..1).}}}
   }
   This is related to \texht{$\Gamma(z)$}{\axiom{Gamma(z)}} by
   \texht{\narrowDisplay{B(u,v) =
\frac{\Gamma(u) \Gamma(v)}{\Gamma(u + v)}.}%
   }{
\newline
\centerline{{      \axiom{Beta(u,v) = Gamma(u)*Gamma(v)/Gamma(u + v).}}}
   }%

\noindent
\axiomFun{logGamma}: \axiom{F -> F}\hfill\newline
```

```
   \axiom{logGamma(z)} is the natural logarithm of
\texht{$\Gamma(z)$}{\axiom{Gamma(z)}}.
   This can often be computed even if
\texht{$\Gamma(z)$}{\axiom{Gamma(z)}}
cannot.
%

\noindent
\axiomFun{digamma}: \axiom{F -> F}\hfill\newline
   \axiom{digamma(z)}, also called \axiom{psi(z)},
is the function \texht{$\psi(z)$}{\axiom{psi(z)}},
   defined by \texht{\narrowDisplay{\psi(z) = \Gamma'(z)/\Gamma(z).}%
   }{
\newline
\centerline{{      \axiom{psi(z) = Gamma'(z)/Gamma(z).}}}
  }%

\noindent
\axiomFun{polygamma}:
\axiom{(NonNegativeInteger, F) -> F}\hfill\newline
   \axiom{polygamma(n, z)} is the \eth{\axiom{n}} derivative of
   \texht{$\psi(z)$}{
\axiom{digamma(z)}}\texht{, written $\psi^{(n)}(z)$}{}.

\noindent
\axiomFun{besselJ}: \axiom{(F,F) -> F}\hfill\newline
   \axiom{besselJ(v,z)} is the Bessel function of the first kind,
   \texht{$J_\nu (z)$}{\axiom{J(v,z)}}.
   This function satisfies the differential equation
   \texht{\narrowDisplay{z^2 w''(z) + z w'(z) + (z^2-\nu^2)w(z) = 0.}%
   }{
\newline
\centerline{{      \axiom{z**2*w''(z) + z*w'(z) + (z**2-v**2)*w(z) = 0.}}}
  }%

\noindent
\axiomFun{besselY}: \axiom{(F,F) -> F}\hfill\newline
   \axiom{besselY(v,z)} is the Bessel function of the second kind,
   \texht{$Y_\nu (z)$}{\axiom{Y(v,z)}}.
   This function satisfies the same differential equation as
   \axiomFun{besselJ}.
   The implementation simply uses the relation
   \texht{\narrowDisplay{Y_\nu (z) =
\frac{J_\nu (z) \cos(\nu \pi) - J_{-\nu} (z)}{\sin(\nu \pi)}.}%
   }{
\newline
\centerline{{      \axiom{Y(v,z) =
(J(v,z)*cos(v*\%pi) - J(-v,z))/sin(v*\%pi).}}}
   }%
```

```
\noindent
\axiomFun{besselI}: \axiom{(F,F) -> F}\hfill\newline
   \axiom{besselI(v,z)} is the modified Bessel function of the first kind,
   \texht{$I_\nu (z)$}{\axiom{I(v,z)}}.
   This function satisfies the differential equation
   \texht{\narrowDisplay{z^2 w''(z) + z w'(z) - (z^2+\nu^2)w(z) = 0.}%
   }{
\newline
\centerline{{
\axiom{z**2 * w''(z) + z * w'(z) - (z**2+v**2)*w(z) = 0.}}}
   }%

\noindent
\axiomFun{besselK}: \axiom{(F,F) -> F}\hfill\newline
   \axiom{besselK(v,z)} is the modified Bessel
function of the second kind,
   \texht{$K_\nu (z)$}{\axiom{K(v,z)}}.
   This function satisfies the same differential
equation as \axiomFun{besselI}.
   The implementation simply uses the relation
   \texht{\narrowDisplay{K_\nu (z) =
\pi \frac{I_{-\nu} (z) - I_{\nu} (z)}{2 \sin(\nu \pi)}.}%
   }{
\newline
\centerline{{      \axiom{K(v,z) =
\%pi*(I(v,z) - I(-v,z))/(2*sin(v*\%pi)).}}}
   }

\noindent
\axiomFun{airyAi}: \axiom{F -> F}\hfill\newline
   \axiom{airyAi(z)} is the Airy function \texht{$Ai(z)$}{\axiom{Ai(z)}}.
   This function satisfies the differential equation
   \texht{$w''(z) - z w(z) = 0.$}{\axiom{w''(z) - z * w(z) = 0.}}
   The implementation simply uses the relation
   \texht{\narrowDisplay{Ai(-z) =
\frac{1}{3}\sqrt{z} ( J_{-1/3}
(\frac{2}{3}z^{3/2}) + J_{1/3} (\frac{2}{3}z^{3/2}) ).}%
   }{
\newline
\centerline{{      \axiom{Ai(-z) =
1/3 * sqrt(z) * (J(-1/3, 2/3*z**(3/2)) + J(1/3, 2/3*z**(3/2)) ).}}}
   }%

\noindent
\axiomFun{airyBi}: \axiom{F -> F}\hfill\newline
   \axiom{airyBi(z)} is the Airy function
\texht{$Bi(z)$}{\axiom{Bi(z)}}.
   This function satisfies the same differential
equation as \axiomFun{airyAi}.
   The implementation simply uses the relation
```

```
   \texht{\narrowDisplay{Bi(-z) =
\frac{1}{3}\sqrt{3 z} ( J_{-1/3}
(\frac{2}{3}z^{3/2}) - J_{1/3} (\frac{2}{3}z^{3/2}) ).}%
   }{
\newline
\centerline{{      \axiom{Bi(-z) = 1/3 *sqrt(3*z) *
(J(-1/3, 2/3*z**(3/2)) - J(1/3, 2/3*z**(3/2)) ).}}}
   }

\noindent
\axiomFun{hypergeometric0F1}: \axiom{(F,F) -> F}\hfill\newline
   \axiom{hypergeometric0F1(c,z)} is the hypergeometric function
   \texht{${}_0 F_1 ( ; c; z)$}{\axiom{0F1(; c; z)}}.%

\xtc{
The above special functions are defined only for small floating-point
types.  If you give \axiomType{Float} arguments, they are converted to
\axiomType{DoubleFloat} by Axiom.
}{
\spadpaste{Gamma(0.5)**2}
}
\xtc{
}{
\spadpaste{a := 2.1; b := 1.1; besselI(a + \%i*b, b*a + 1)}
}

A number of additional operations may be used to compute numerical
values.  These are special polynomial functions that can be evaluated
for values in any commutative ring \axiom{R}, and in particular for
values in any floating-point type.  The following operations are
provided by the package \axiomType{OrthogonalPolynomialFunctions}:

\noindent
\axiomFun{chebyshevT}:
\axiom{(NonNegativeInteger, R) -> R}\hbox{}\hfill\newline
   \axiom{chebyshevT(n,z)} is the \eth{\axiom{n}}
Chebyshev polynomial of the first
   kind, \texht{$T_n (z)$}{\axiom{T[n](z)}}.  These are defined by
   \texht{\narrowDisplay{\frac{1-t z}{1-2 t z+t^2} =
\sum_{n=0}^{\infty} T_n (z) t^n.}%
   }{
\newline
\centerline{{   \axiom{(1-t*z)/(1-2*t*z+t**2) =
sum(T[n](z) *t**n, n = 0..).}}}
   }%

\noindent
\axiomFun{chebyshevU}: \axiom{(NonNegativeInteger, R) -> R}
\hbox{}\hfill\newline
   \axiom{chebyshevU(n,z)} is the \eth{\axiom{n}}
```

```
Chebyshev polynomial of the second
   kind, \texht{$U_n (z)$}{\axiom{U[n](z)}}. These are defined by
   \texht{\narrowDisplay{\frac{1}{1-2 t z+t^2} =
\sum_{n=0}^{\infty} U_n (z) t^n.}%
   }{
\newline
\centerline{{   \axiom{1/(1-2*t*z+t**2) = sum(U[n](z) *t**n, n = 0..).}}}
   }%

\noindent
\axiomFun{hermiteH}:   \axiom{(NonNegativeInteger, R) -> R}
\hbox{}\hfill\newline
   \axiom{hermiteH(n,z)} is the \eth{\axiom{n}} Hermite polynomial,
   \texht{$H_n (z)$}{\axiom{H[n](z)}}.
   These are defined by
   \texht{\narrowDisplay{e^{2 t z - t^2} =
\sum_{n=0}^{\infty} H_n (z) \frac{t^n}{n!}.}%
   }{
\newline
\centerline{{   \axiom{exp(2*t*z-t**2) = sum(H[n](z)*t**n/n!, n = 0..).}}}
   }%

\noindent
\axiomFun{laguerreL}:  \axiom{(NonNegativeInteger, R) -> R}
\hbox{}\hfill\newline
   \axiom{laguerreL(n,z)} is the \eth{\axiom{n}} Laguerre polynomial,
   \texht{$L_n (z)$}{\axiom{L[n](z)}}.
   These are defined by
   \texht{\narrowDisplay{\frac{e^{-\frac{t z}{1-t}}}{1-t} =
\sum_{n=0}^{\infty} L_n (z) \frac{t^n}{n!}.}%
   }{
\newline
\centerline{{   \axiom{exp(-t*z/(1-t))/(1-t) =
sum(L[n](z)*t**n/n!, n = 0..).}}}
   }%

\noindent
\axiomFun{laguerreL}:
\axiom{(NonNegativeInteger, NonNegativeInteger, R) -> R}\hbox{}\hfill\newline
   \axiom{laguerreL(m,n,z)} is the associated Laguerre polynomial,
   \texht{$L^m_n (z)$}{\axiom{L<m>[n](z)}}.
   This is the
\eth{\axiom{m}} derivative of \texht{$L_n (z)$}{\axiom{L[n](z)}}.

\noindent
\axiomFun{legendreP}:
\axiom{(NonNegativeInteger, R) -> R}\hbox{}\hfill\newline
   \axiom{legendreP(n,z)} is the \eth{\axiom{n}} Legendre polynomial,
   \texht{$P_n (z)$}{\axiom{P[n](z)}}.  These are defined by
   \texht{\narrowDisplay{
```

```
\frac{1}{\sqrt{1-2 t z+t^2}} = \sum_{n=0}^{\infty} P_n (z) t^n.}%
   }{
\newline
\centerline{{   \axiom{1/sqrt(1-2*z*t+t**2) =
sum(P[n](z)*t**n, n = 0..).}}}
   }%

%
\xtc{
These operations require non-negative integers for the indices,
but otherwise the argument can be given as desired.
}{
\spadpaste{[chebyshevT(i, z) for i in 0..5]}
}
\xtc{
The expression \axiom{chebyshevT(n,z)} evaluates to the
\eth{\axiom{n}} Chebyshev polynomial of the first kind.
}{
\spadpaste{chebyshevT(3, 5.0 + 6.0*\%i)}
}
\xtc{
}{
\spadpaste{chebyshevT(3, 5.0::DoubleFloat)}
}
\xtc{
The expression \axiom{chebyshevU(n,z)} evaluates to the
\eth{\axiom{n}} Chebyshev polynomial of the second kind.
}{
\spadpaste{[chebyshevU(i, z) for i in 0..5]}
}
\xtc{
}{
\spadpaste{chebyshevU(3, 0.2)}
}
\xtc{
The expression \axiom{hermiteH(n,z)} evaluates to the
\eth{\axiom{n}} Hermite polynomial.
}{
\spadpaste{[hermiteH(i, z) for i in 0..5]}
}
\xtc{
}{
\spadpaste{hermiteH(100, 1.0)}
}
\xtc{
The expression \axiom{laguerreL(n,z)} evaluates to the
\eth{\axiom{n}} Laguerre polynomial.
}{
\spadpaste{[laguerreL(i, z) for i in 0..4]}
}
```

```
\xtc{
}{
\spadpaste{laguerreL(4, 1.2)}
}
\xtc{
}{
\spadpaste{[laguerreL(j, 3, z) for j in 0..4]}
}
\xtc{
}{
\spadpaste{laguerreL(1, 3, 2.1)}
}
\xtc{
The expression
\axiom{legendreP(n,z)} evaluates to the
\eth{\axiom{n}} Legendre polynomial,
}{
\spadpaste{[legendreP(i,z) for i in 0..5]}
}
\xtc{
}{
\spadpaste{legendreP(3, 3.0*\%i)}
}
%

Finally, three number-theoretic polynomial operations may be evaluated.
The following operations are provided by the package
\axiomType{NumberTheoreticPolynomialFunctions}.

\noindent
\axiomFun{bernoulliB}: \axiom{(NonNegativeInteger, R) -> R}
\hbox{}\hfill\newline
   \axiom{bernoulliB(n,z)} is the \eth{\axiom{n}} Bernoulli polynomial,
   \texht{$B_n (z)$}{\axiom{B[n](z)}}.  These are defined by
   \texht{\narrowDisplay{\frac{t e^{z t}}{e^t - 1} =
\sum_{n=0}^{\infty} B_n (z) \frac{t^n}{n!}.}
   }{
\newline
\centerline{{   \axiom{t*exp(z*t)/(exp t - 1) =
sum(B[n](z)*t**n/n! for n - 0..)}}}
   }%

\noindent
\axiomFun{eulerE}: \axiom{(NonNegativeInteger, R) -> R}
\hbox{}\hfill\newline
   \axiom{eulerE(n,z)} is the \eth{\axiom{n}} Euler polynomial,
   \texht{$E_n (z)$}{\axiom{E[n](z)}}.  These are defined by
   \texht{\narrowDisplay{\frac{2 e^{z t}}{e^t + 1} =
\sum_{n=0}^{\infty} E_n (z) \frac{t^n}{n!}.}%
   }{
```

```
\newline
\centerline{{   \axiom{2*exp(z*t)/(exp t + 1) =
sum(E[n](z)*t**n/n! for n - 0..)}}}
   }%

\noindent
\axiomFun{cyclotomic}: \axiom{(NonNegativeInteger, R) -> R}
\hbox{}\hfill\newline
   \axiom{cyclotomic(n,z)} is the \eth{\axiom{n}} cyclotomic polynomial
   \texht{$\Phi_n (z)$}{\axiom{phi(n,z)}}.  This is the polynomial whose
   roots are precisely the primitive \eth{\axiom{n}} roots of unity.
   This polynomial has degree given by the Euler totient function
   \texht{$\phi(n)$}{\axiom{phi(n)}}.

\xtc{
The expression \axiom{bernoulliB(n,z)} evaluates to the
\eth{\axiom{n}} Bernoulli polynomial.
}{
\spadpaste{bernoulliB(3, z)}
}
\xtc{
}{
\spadpaste{bernoulliB(3, 0.7 + 0.4 * \%i)}
}
\xtc{
The expression
\axiom{eulerE(n,z)} evaluates to the \eth{\axiom{n}} Euler polynomial.
}{
\spadpaste{eulerE(3, z)}
}
\xtc{
}{
\spadpaste{eulerE(3, 0.7 + 0.4 * \%i)}
}
\xtc{
The expression
\axiom{cyclotomic(n,z)} evaluates to the
\eth{\axiom{n}} cyclotomic polynomial.
}{
\spadpaste{cyclotomic(3, z)}
}
\xtc{
}{
\spadpaste{cyclotomic(3, (-1.0 + 0.0 * \%i)**(2/3))}
}

Drawing complex functions in Axiom is presently somewhat
awkward compared to drawing real functions.
It is necessary to use the \axiomFun{draw} operations that operate
on functions rather than expressions.
```

```
\psXtc{
This is the complex exponential function\texht{ (rotated
interactively).}{.}  When this is displayed in color, the height is
the value of the real part of the function and the color is the
imaginary part.  Red indicates large negative imaginary values, green
indicates imaginary values near zero and blue/violet indicates large
positive imaginary values.
}{
\graphpaste{draw((x,y)+-> real exp complex(x,y), -2..2,
-2*\%pi..2*\%pi, colorFunction ==
(x, y) +->  imag exp complex(x,y),
title=="exp(x+\%i*y)", style=="smooth")}
}{
\epsffile[0 0 295 295]{../ps/compexp.ps}
}

\psXtc{
This is the complex arctangent function.
Again, the height is the real part of the function value but here
the color indicates the function value's phase.
The position of the branch cuts are clearly visible and one can
see that the function is real only for a real argument.
}{
\graphpaste{vp := draw((x,y) +-> real atan complex(x,y),
-\%pi..\%pi, -\%pi..\%pi, colorFunction==
(x,y) +->argument atan complex(x,y),
title=="atan(x+\%i*y)", style=="shade"); rotate(vp,-160,-45); vp}
}{
\epsffile[0 0 295 295]{../ps/compatan.ps}
}

\psXtc{
This is the complex Gamma function.
}{
\graphpaste{draw((x,y) +-> max(min(real Gamma complex(x,y),4),-4),
-\%pi..\%pi, -\%pi..\%pi, style=="shade", colorFunction ==
(x,y) +-> argument Gamma complex(x,y),
title == "Gamma(x+\%i*y)", var1Steps == 50, var2Steps== 50)}
}{
\epsffile[0 0 295 295]{../ps/compgamm.ps}
}

\psXtc{
This shows the real Beta function near the origin.
}{
\graphpaste{draw(Beta(x,y)/100, x=-1.6..1.7, y = -1.6..1.7,
style=="shade", title=="Beta(x,y)", var1Steps==40, var2Steps==40)}
}{
\epsffile[0 0 295 295]{../ps/realbeta.ps}
```

```
}

\psXtc{
This is the Bessel function \texht{$J_\alpha (x)$}{\axiom{J(alpha,x)}}
for index \texht{$\alpha$}{\axiom{alpha}} in the range \axiom{-6..4} and
argument \texht{$x$}{\axiom{x}} in the range \axiom{2..14}.
}{
\graphpaste{draw((alpha,x) +-> min(max(besselJ(alpha, x+8), -6), 6),
-6..4, -6..6, title=="besselJ(alpha,x)", style=="shade",
var1Steps==40, var2Steps==40)}
}{
\epsffile[0 0 295 295]{../ps/bessel.ps}
}

\psXtc{
This is the modified Bessel function
\texht{$I_\alpha (x)$}{\axiom{I(alpha,x)}}
evaluated for various real values of the index
\texht{$\alpha$}{\axiom{alpha}}
and fixed argument \texht{$x = 5$}{\axiom{x = 5}}.
}{
\graphpaste{draw(besselI(alpha, 5), alpha = -12..12, unit==[5,20])}
}{
\epsffile[0 0 295 295]{../ps/modbess.ps}
}

\psXtc{
This is similar to the last example
except the index \texht{$\alpha$}{\axiom{alpha}}
takes on complex values in a \axiom{6 x 6} rectangle
centered on the origin.
}{
\graphpaste{draw((x,y) +-> real besselI(complex(x/20, y/20),5),
-60..60, -60..60, colorFunction ==
(x,y)+-> argument besselI(complex(x/20,y/20),5),
title=="besselI(x+i*y,5)", style=="shade")}
}{
\epsffile[0 0 295 295]{../ps/modbessc.ps}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugProblemNumericPagePatch1}
\begin{paste}{ugProblemNumericPageFull1}{ugProblemNumericPageEmpty1}
\pastebutton{ugProblemNumericPageFull1}{\hidepaste}
\tab{5}\spadcommand{exp(3.1)}
\indentrel{3}\begin{verbatim}
   (1)  22.1979512814 41633405
```

```
                                          Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPageEmpty1}
\begin{paste}{ugProblemNumericPageEmpty1}{ugProblemNumericPagePatch1}
\pastebutton{ugProblemNumericPageEmpty1}{\showpaste}
\tab{5}\spadcommand{exp(3.1)}
\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPagePatch2}
\begin{paste}{ugProblemNumericPageFull2}{ugProblemNumericPageEmpty2}
\pastebutton{ugProblemNumericPageFull2}{\hidepaste}
\tab{5}\spadcommand{exp(3.1 + 4.5 * \%i)}
\indentrel{3}\begin{verbatim}
   (2)
   - 4.6792348860 969899118 - 21.6991659280 71731864 %i
                                    Type: Complex Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPageEmpty2}
\begin{paste}{ugProblemNumericPageEmpty2}{ugProblemNumericPagePatch2}
\pastebutton{ugProblemNumericPageEmpty2}{\showpaste}
\tab{5}\spadcommand{exp(3.1 + 4.5 * \%i)}
\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPagePatch3}
\begin{paste}{ugProblemNumericPageFull3}{ugProblemNumericPageEmpty3}
\pastebutton{ugProblemNumericPageFull3}{\hidepaste}
\tab{5}\spadcommand{r: DFLOAT := 3.1; t: DFLOAT := 4.5; exp(r + t*\%i)}
\indentrel{3}\begin{verbatim}
   (3)  - 4.6792348860969906 - 21.699165928071732%i
                              Type: Complex DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPageEmpty3}
\begin{paste}{ugProblemNumericPageEmpty3}{ugProblemNumericPagePatch3}
\pastebutton{ugProblemNumericPageEmpty3}{\showpaste}
\tab{5}\spadcommand{r: DFLOAT := 3.1; t: DFLOAT := 4.5; exp(r + t*\%i)}
\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPagePatch4}
\begin{paste}{ugProblemNumericPageFull4}{ugProblemNumericPageEmpty4}
\pastebutton{ugProblemNumericPageFull4}{\hidepaste}
\tab{5}\spadcommand{exp(3.1::DFLOAT + 4.5::DFLOAT * \%i)}
\indentrel{3}\begin{verbatim}
   (4)  - 4.6792348860969906 - 21.699165928071732%i
                              Type: Complex DoubleFloat
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPageEmpty4}
\begin{paste}{ugProblemNumericPageEmpty4}{ugProblemNumericPagePatch4}
\pastebutton{ugProblemNumericPageEmpty4}{\showpaste}
\tab{5}\spadcommand{exp(3.1::DFLOAT + 4.5::DFLOAT * \%i)}
\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPagePatch5}
\begin{paste}{ugProblemNumericPageFull5}{ugProblemNumericPageEmpty5}
\pastebutton{ugProblemNumericPageFull5}{\hidepaste}
\tab{5}\spadcommand{Gamma(0.5)**2}
\indentrel{3}\begin{verbatim}
   (5)  3.14159265358979
                                      Type: DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPageEmpty5}
\begin{paste}{ugProblemNumericPageEmpty5}{ugProblemNumericPagePatch5}
\pastebutton{ugProblemNumericPageEmpty5}{\showpaste}
\tab{5}\spadcommand{Gamma(0.5)**2}
\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPagePatch6}
\begin{paste}{ugProblemNumericPageFull6}{ugProblemNumericPageEmpty6}
\pastebutton{ugProblemNumericPageFull6}{\hidepaste}
\tab{5}\spadcommand{a := 2.1; b := 1.1; besselI(a + \%i*b, b*a + 1)}
\indentrel{3}\begin{verbatim}
   (6)  2.489482417547372 - 2.3658460381468371%i
                              Type: Complex DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPageEmpty6}
\begin{paste}{ugProblemNumericPageEmpty6}{ugProblemNumericPagePatch6}
\pastebutton{ugProblemNumericPageEmpty6}{\showpaste}
\tab{5}\spadcommand{a := 2.1; b := 1.1; besselI(a + \%i*b, b*a + 1)}
\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPagePatch7}
\begin{paste}{ugProblemNumericPageFull7}{ugProblemNumericPageEmpty7}
\pastebutton{ugProblemNumericPageFull7}{\hidepaste}
\tab{5}\spadcommand{[chebyshevT(i, z) for i in 0..5]}
\indentrel{3}\begin{verbatim}
   (7)
          2       3        4     2        5      3
   [1,z,2z  - 1,4z  - 3z,8z  - 8z  + 1,16z  - 20z  + 5z]
                          Type: List Polynomial Integer
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPageEmpty7}
\begin{paste}{ugProblemNumericPageEmpty7}{ugProblemNumericPagePatch7}
\pastebutton{ugProblemNumericPageEmpty7}{\showpaste}
\tab{5}\spadcommand{[chebyshevT(i, z) for i in 0..5]}
\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPagePatch8}
\begin{paste}{ugProblemNumericPageFull8}{ugProblemNumericPageEmpty8}
\pastebutton{ugProblemNumericPageFull8}{\hidepaste}
\tab{5}\spadcommand{chebyshevT(3, 5.0 + 6.0*\%i)}
\indentrel{3}\begin{verbatim}
   (8)  - 1675.0 + 918.0 %i
                                    Type: Complex Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPageEmpty8}
\begin{paste}{ugProblemNumericPageEmpty8}{ugProblemNumericPagePatch8}
\pastebutton{ugProblemNumericPageEmpty8}{\showpaste}
\tab{5}\spadcommand{chebyshevT(3, 5.0 + 6.0*\%i)}
\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPagePatch9}
\begin{paste}{ugProblemNumericPageFull9}{ugProblemNumericPageEmpty9}
\pastebutton{ugProblemNumericPageFull9}{\hidepaste}
\tab{5}\spadcommand{chebyshevT(3, 5.0::DoubleFloat)}
\indentrel{3}\begin{verbatim}
   (9)  485.0
                                      Type: DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPageEmpty9}
\begin{paste}{ugProblemNumericPageEmpty9}{ugProblemNumericPagePatch9}
\pastebutton{ugProblemNumericPageEmpty9}{\showpaste}
\tab{5}\spadcommand{chebyshevT(3, 5.0::DoubleFloat)}
\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPagePatch10}
\begin{paste}{ugProblemNumericPageFull10}{ugProblemNumericPageEmpty10}
\pastebutton{ugProblemNumericPageFull10}{\hidepaste}
\tab{5}\spadcommand{[chebyshevU(i, z) for i in 0..5]}
\indentrel{3}\begin{verbatim}
   (10)
             2         3           4      2
   [1, 2z, 4z  - 1, 8z  - 4z, 16z  - 12z  + 1,
       5       3
```

```
    32z  - 32z  + 6z]
                          Type: List Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPageEmpty10}
\begin{paste}{ugProblemNumericPageEmpty10}{ugProblemNumericPagePatch10}
\pastebutton{ugProblemNumericPageEmpty10}{\showpaste}
\tab{5}\spadcommand{[chebyshevU(i, z) for i in 0..5]}
\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPagePatch11}
\begin{paste}{ugProblemNumericPageFull11}{ugProblemNumericPageEmpty11}
\pastebutton{ugProblemNumericPageFull11}{\hidepaste}
\tab{5}\spadcommand{chebyshevU(3, 0.2)}
\indentrel{3}\begin{verbatim}
   (11)  - 0.736
                                            Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPageEmpty11}
\begin{paste}{ugProblemNumericPageEmpty11}{ugProblemNumericPagePatch11}
\pastebutton{ugProblemNumericPageEmpty11}{\showpaste}
\tab{5}\spadcommand{chebyshevU(3, 0.2)}
\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPagePatch12}
\begin{paste}{ugProblemNumericPageFull12}{ugProblemNumericPageEmpty12}
\pastebutton{ugProblemNumericPageFull12}{\hidepaste}
\tab{5}\spadcommand{[hermiteH(i, z) for i in 0..5]}
\indentrel{3}\begin{verbatim}
   (12)
            2        3           4      2
   [1, 2z, 4z  - 2, 8z  - 12z, 16z  - 48z  + 12,
       5       3
    32z  - 160z  + 120z]
                          Type: List Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPageEmpty12}
\begin{paste}{ugProblemNumericPageEmpty12}{ugProblemNumericPagePatch12}
\pastebutton{ugProblemNumericPageEmpty12}{\showpaste}
\tab{5}\spadcommand{[hermiteH(i, z) for i in 0..5]}
\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPagePatch13}
\begin{paste}{ugProblemNumericPageFull13}{ugProblemNumericPageEmpty13}
\pastebutton{ugProblemNumericPageFull13}{\hidepaste}
```

```
\tab{5}\spadcommand{hermiteH(100, 1.0)}
\indentrel{3}\begin{verbatim}
   (13)  - 0.1448706729 337934088 E 93
                                             Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPageEmpty13}
\begin{paste}{ugProblemNumericPageEmpty13}{ugProblemNumericPagePatch13}
\pastebutton{ugProblemNumericPageEmpty13}{\showpaste}
\tab{5}\spadcommand{hermiteH(100, 1.0)}
\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPagePatch14}
\begin{paste}{ugProblemNumericPageFull14}{ugProblemNumericPageEmpty14}
\pastebutton{ugProblemNumericPageFull14}{\hidepaste}
\tab{5}\spadcommand{[laguerreL(i, z) for i in 0..4]}
\indentrel{3}\begin{verbatim}
   (14)
                 2              3     2
   [1, - z + 1, z  - 4z + 2, - z  + 9z  - 18z + 6,
     4      3      2
    z  - 16z  + 72z  - 96z + 24]
                          Type: List Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPageEmpty14}
\begin{paste}{ugProblemNumericPageEmpty14}{ugProblemNumericPagePatch14}
\pastebutton{ugProblemNumericPageEmpty14}{\showpaste}
\tab{5}\spadcommand{[laguerreL(i, z) for i in 0..4]}
\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPagePatch15}
\begin{paste}{ugProblemNumericPageFull15}{ugProblemNumericPageEmpty15}
\pastebutton{ugProblemNumericPageFull15}{\hidepaste}
\tab{5}\spadcommand{laguerreL(4, 1.2)}
\indentrel{3}\begin{verbatim}
   (15)  - 13.0944
                                             Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPageEmpty15}
\begin{paste}{ugProblemNumericPageEmpty15}{ugProblemNumericPagePatch15}
\pastebutton{ugProblemNumericPageEmpty15}{\showpaste}
\tab{5}\spadcommand{laguerreL(4, 1.2)}
\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPagePatch16}
```

```
\begin{paste}{ugProblemNumericPageFull16}{ugProblemNumericPageEmpty16}
\pastebutton{ugProblemNumericPageFull16}{\hidepaste}
\tab{5}\spadcommand{[laguerreL(j, 3, z) for j in 0..4]}
\indentrel{3}\begin{verbatim}
   (16)
       3     2                2
   [- z  + 9z  - 18z + 6,- 3z  + 18z - 18,- 6z + 18,- 6,0]
                         Type: List Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPageEmpty16}
\begin{paste}{ugProblemNumericPageEmpty16}{ugProblemNumericPagePatch16}
\pastebutton{ugProblemNumericPageEmpty16}{\showpaste}
\tab{5}\spadcommand{[laguerreL(j, 3, z) for j in 0..4]}
\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPagePatch17}
\begin{paste}{ugProblemNumericPageFull17}{ugProblemNumericPageEmpty17}
\pastebutton{ugProblemNumericPageFull17}{\hidepaste}
\tab{5}\spadcommand{laguerreL(1, 3, 2.1)}
\indentrel{3}\begin{verbatim}
   (17)  6.57
                                             Type: Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPageEmpty17}
\begin{paste}{ugProblemNumericPageEmpty17}{ugProblemNumericPagePatch17}
\pastebutton{ugProblemNumericPageEmpty17}{\showpaste}
\tab{5}\spadcommand{laguerreL(1, 3, 2.1)}
\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPagePatch18}
\begin{paste}{ugProblemNumericPageFull18}{ugProblemNumericPageEmpty18}
\pastebutton{ugProblemNumericPageFull18}{\hidepaste}
\tab{5}\spadcommand{[legendreP(i,z) for i in 0..5]}
\indentrel{3}\begin{verbatim}
   (18)
          3  2   1  5  3   3    35  4   15  2   3
   [1, z,  z  - ,  z  -  z,  z  -  z  + ,
          2      2  2      2     8       4      8
    63  5   35  3   15
     z  -  z  +  z]
     8       4      8
                Type: List Polynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPageEmpty18}
```

```
\begin{paste}{ugProblemNumericPageEmpty18}{ugProblemNumericPagePatch18}
\pastebutton{ugProblemNumericPageEmpty18}{\showpaste}
\tab{5}\spadcommand{[legendreP(i,z) for i in 0..5]}
\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPagePatch19}
\begin{paste}{ugProblemNumericPageFull19}{ugProblemNumericPageEmpty19}
\pastebutton{ugProblemNumericPageFull19}{\hidepaste}
\tab{5}\spadcommand{legendreP(3, 3.0*\%i)}
\indentrel{3}\begin{verbatim}
   (19)  - 72.0 %i
                                   Type: Complex Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPageEmpty19}
\begin{paste}{ugProblemNumericPageEmpty19}{ugProblemNumericPagePatch19}
\pastebutton{ugProblemNumericPageEmpty19}{\showpaste}
\tab{5}\spadcommand{legendreP(3, 3.0*\%i)}
\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPagePatch20}
\begin{paste}{ugProblemNumericPageFull20}{ugProblemNumericPageEmpty20}
\pastebutton{ugProblemNumericPageFull20}{\hidepaste}
\tab{5}\spadcommand{bernoulliB(3, z)}
\indentrel{3}\begin{verbatim}
          3   3  2   1
   (20)  z  -  z  +  z
              2      2
                     Type: Polynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPageEmpty20}
\begin{paste}{ugProblemNumericPageEmpty20}{ugProblemNumericPagePatch20}
\pastebutton{ugProblemNumericPageEmpty20}{\showpaste}
\tab{5}\spadcommand{bernoulliB(3, z)}
\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPagePatch21}
\begin{paste}{ugProblemNumericPageFull21}{ugProblemNumericPageEmpty21}
\pastebutton{ugProblemNumericPageFull21}{\hidepaste}
\tab{5}\spadcommand{bernoulliB(3, 0.7 + 0.4 * \%i)}
\indentrel{3}\begin{verbatim}
   (21)  - 0.138 - 0.116 %i
                                   Type: Complex Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPageEmpty21}
```

```
\begin{paste}{ugProblemNumericPageEmpty21}{ugProblemNumericPagePatch21}
\pastebutton{ugProblemNumericPageEmpty21}{\showpaste}
\tab{5}\spadcommand{bernoulliB(3, 0.7 + 0.4 * \%i)}
\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPagePatch22}
\begin{paste}{ugProblemNumericPageFull22}{ugProblemNumericPageEmpty22}
\pastebutton{ugProblemNumericPageFull22}{\hidepaste}
\tab{5}\spadcommand{eulerE(3, z)}
\indentrel{3}\begin{verbatim}
          3   3  2   1
   (22)  z  -  z  +
              2      4
                      Type: Polynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPageEmpty22}
\begin{paste}{ugProblemNumericPageEmpty22}{ugProblemNumericPagePatch22}
\pastebutton{ugProblemNumericPageEmpty22}{\showpaste}
\tab{5}\spadcommand{eulerE(3, z)}
\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPagePatch23}
\begin{paste}{ugProblemNumericPageFull23}{ugProblemNumericPageEmpty23}
\pastebutton{ugProblemNumericPageFull23}{\hidepaste}
\tab{5}\spadcommand{eulerE(3, 0.7 + 0.4 * \%i)}
\indentrel{3}\begin{verbatim}
   (23)  - 0.238 - 0.316 %i
                                    Type: Complex Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPageEmpty23}
\begin{paste}{ugProblemNumericPageEmpty23}{ugProblemNumericPagePatch23}
\pastebutton{ugProblemNumericPageEmpty23}{\showpaste}
\tab{5}\spadcommand{eulerE(3, 0.7 + 0.4 * \%i)}
\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPagePatch24}
\begin{paste}{ugProblemNumericPageFull24}{ugProblemNumericPageEmpty24}
\pastebutton{ugProblemNumericPageFull24}{\hidepaste}
\tab{5}\spadcommand{cyclotomic(3, z)}
\indentrel{3}\begin{verbatim}
          2
   (24)  z  + z + 1
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugProblemNumericPageEmpty24}
\begin{paste}{ugProblemNumericPageEmpty24}{ugProblemNumericPagePatch24}
\pastebutton{ugProblemNumericPageEmpty24}{\showpaste}
\tab{5}\spadcommand{cyclotomic(3, z)}
\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPagePatch25}
\begin{paste}{ugProblemNumericPageFull25}{ugProblemNumericPageEmpty25}
\pastebutton{ugProblemNumericPageFull25}{\hidepaste}
\tab{5}\spadcommand{cyclotomic(3, (-1.0 + 0.0 * \%i)**(2/3))}
\indentrel{3}\begin{verbatim}
   (25)  0.0
                                  Type: Complex Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPageEmpty25}
\begin{paste}{ugProblemNumericPageEmpty25}{ugProblemNumericPagePatch25}
\pastebutton{ugProblemNumericPageEmpty25}{\showpaste}
\tab{5}\spadcommand{cyclotomic(3, (-1.0 + 0.0 * \%i)**(2/3))}
\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPagePatch26}
\begin{paste}{ugProblemNumericPageFull26}{ugProblemNumericPageEmpty26}
\pastebutton{ugProblemNumericPageFull26}{\hidepaste}
\tab{5}\spadgraph{draw((x,y)+-> real exp complex(x,y), -2..2, -2*\%pi..2*\%pi, colorFunction == (x, y) +-> i
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/ugproblemnumericpage26.view/image}}{viewalone\spa
\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPageEmpty26}
\begin{paste}{ugProblemNumericPageEmpty26}{ugProblemNumericPagePatch26}
\pastebutton{ugProblemNumericPageEmpty26}{\showpaste}
\tab{5}\spadgraph{draw((x,y)+-> real exp complex(x,y), -2..2, -2*\%pi..2*\%pi, colorFunction == (x, y) +-> i
\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPagePatch27}
\begin{paste}{ugProblemNumericPageFull27}{ugProblemNumericPageEmpty27}
\pastebutton{ugProblemNumericPageFull27}{\hidepaste}
\tab{5}\spadgraph{vp := draw((x,y) +-> real atan complex(x,y), -\%pi..\%pi, -\%pi..\%pi, colorFunction==(x,y
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/ugproblemnumericpage27.view/image}}{viewalone\spa
\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPageEmpty27}
\begin{paste}{ugProblemNumericPageEmpty27}{ugProblemNumericPagePatch27}
\pastebutton{ugProblemNumericPageEmpty27}{\showpaste}
\tab{5}\spadgraph{vp := draw((x,y) +-> real atan complex(x,y), -\%pi..\%pi, -\%pi..\%pi, colorFunction==(x,y
\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPagePatch28}
\begin{paste}{ugProblemNumericPageFull28}{ugProblemNumericPageEmpty28}
```

```
\pastebutton{ugProblemNumericPageFull28}{\hidepaste}
\tab{5}\spadgraph{draw((x,y) +-> max(min(real Gamma complex(x,y),4),-4), -\%pi..\%pi, -\%pi..\%p
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/ugproblemnumericpage28.view/image}}{v
\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPageEmpty28}
\begin{paste}{ugProblemNumericPageEmpty28}{ugProblemNumericPagePatch28}
\pastebutton{ugProblemNumericPageEmpty28}{\showpaste}
\tab{5}\spadgraph{draw((x,y) +-> max(min(real Gamma complex(x,y),4),-4), -\%pi..\%pi, -\%pi..\%p
\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPagePatch29}
\begin{paste}{ugProblemNumericPageFull29}{ugProblemNumericPageEmpty29}
\pastebutton{ugProblemNumericPageFull29}{\hidepaste}
\tab{5}\spadgraph{draw(Beta(x,y)/100, x=-1.6..1.7, y = -1.6..1.7, style=="shade", title=="Beta(x
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/ugproblemnumericpage29.view/image}}{v
\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPageEmpty29}
\begin{paste}{ugProblemNumericPageEmpty29}{ugProblemNumericPagePatch29}
\pastebutton{ugProblemNumericPageEmpty29}{\showpaste}
\tab{5}\spadgraph{draw(Beta(x,y)/100, x=-1.6..1.7, y = -1.6..1.7, style=="shade", title=="Beta(x
\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPagePatch30}
\begin{paste}{ugProblemNumericPageFull30}{ugProblemNumericPageEmpty30}
\pastebutton{ugProblemNumericPageFull30}{\hidepaste}
\tab{5}\spadgraph{draw((alpha,x) +-> min(max(besselJ(alpha, x+8), -6), 6), -6..4, -6..6, title==
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/ugproblemnumericpage30.view/image}}{v
\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPageEmpty30}
\begin{paste}{ugProblemNumericPageEmpty30}{ugProblemNumericPagePatch30}
\pastebutton{ugProblemNumericPageEmpty30}{\showpaste}
\tab{5}\spadgraph{draw((alpha,x) +-> min(max(besselJ(alpha, x+8), -6), 6), -6..4, -6..6, title==
\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPagePatch31}
\begin{paste}{ugProblemNumericPageFull31}{ugProblemNumericPageEmpty31}
\pastebutton{ugProblemNumericPageFull31}{\hidepaste}
\tab{5}\spadgraph{draw(besselI(alpha, 5), alpha = -12..12, unit==[5,20])}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/ugproblemnumericpage31.view/image}}{v
\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPageEmpty31}
\begin{paste}{ugProblemNumericPageEmpty31}{ugProblemNumericPagePatch31}
\pastebutton{ugProblemNumericPageEmpty31}{\showpaste}
\tab{5}\spadgraph{draw(besselI(alpha, 5), alpha = -12..12, unit==[5,20])}
\end{paste}\end{patch}
```

```
\begin{patch}{ugProblemNumericPagePatch32}
\begin{paste}{ugProblemNumericPageFull32}{ugProblemNumericPageEmpty32}
\pastebutton{ugProblemNumericPageFull32}{\hidepaste}
\tab{5}\spadgraph{draw((x,y) +-> real besselI(complex(x/20, y/20),5), -60..60, -60..60, colorFunction == (x
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/ugproblemnumericpage32.view/image}}{viewalone\spa
\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPageEmpty32}
\begin{paste}{ugProblemNumericPageEmpty32}{ugProblemNumericPagePatch32}
\pastebutton{ugProblemNumericPageEmpty32}{\showpaste}
\tab{5}\spadgraph{draw((x,y) +-> real besselI(complex(x/20, y/20),5), -60..60, -60..60, colorFunction == (x
\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPagePatch33}
\begin{paste}{ugProblemNumericPageFull33}{ugProblemNumericPageEmpty33}
\pastebutton{ugProblemNumericPageFull33}{\hidepaste}
\tab{5}\spadcommand{all}
\indentrel{3}\begin{verbatim}
   (33)  all
                                   Type: Variable all
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemNumericPageEmpty33}
\begin{paste}{ugProblemNumericPageEmpty33}{ugProblemNumericPagePatch33}
\pastebutton{ugProblemNumericPageEmpty33}{\showpaste}
\tab{5}\spadcommand{all}
\end{paste}\end{patch}
```

———-

Polynomial Factorization

⇒ “notitle” (ugProblemFactorIntRatPage) 12 on page 2102
⇒ “notitle” (ugProblemFactorFFPage) 12 on page 2104
⇒ “notitle” (ugProblemFactorAlgPage) 12 on page 2106
⇒ “notitle” (ugProblemFactorRatFunPage) 12 on page 2111

— **ug08.ht** —

```
\begin{page}{ugProblemFactorPage}{8.2. Polynomial Factorization}
\beginscroll
%
The Axiom polynomial factorization
facilities are available for all polynomial types and a wide variety of
coefficient domains.
Here are some examples.
```

```
\beginmenu
    \menudownlink{{8.2.1. Integer and Rational Number Coefficients}}
{ugProblemFactorIntRatPage}
    \menudownlink{{8.2.2. Finite Field Coefficients}}
{ugProblemFactorFFPage}
    \menudownlink{{8.2.3. Simple Algebraic Extension Field Coefficients}}
{ugProblemFactorAlgPage}
    \menudownlink{{8.2.4. Factoring Rational Functions}}
{ugProblemFactorRatFunPage}
\endmenu
\endscroll
\autobuttons
\end{page}
```

——–

Integer and Rational Number Coefficients

— ug08.ht —

```
\begin{page}{ugProblemFactorIntRatPage}
{8.2.1. Integer and Rational Number Coefficients}
\beginscroll

\labelSpace{4pc}
\xtc{
Polynomials with integer
coefficients can be be factored.
}{
\spadpaste{v := (4*x**3+2*y**2+1)*(12*x**5-x**3*y+12) \bound{v}}
}
\xtc{
}{
\spadpaste{factor v \free{v}}
}
\xtc{
Also, Axiom can factor polynomials with
rational number coefficients.
}{
\spadpaste{w := (4*x**3+(2/3)*x**2+1)*(12*x**5-(1/2)*x**3+12) \bound{w}}
}
\xtc{
}{
\spadpaste{factor w \free{w}}
}
```

```
\endscroll
\autobuttons
\end{page}

\begin{patch}{ugProblemFactorIntRatPagePatch1}
\begin{paste}{ugProblemFactorIntRatPageFull1}{ugProblemFactorIntRatPageEmpty1}
\pastebutton{ugProblemFactorIntRatPageFull1}{\hidepaste}
\tab{5}\spadcommand{v := (4*x**3+2*y**2+1)*(12*x**5-x**3*y+12)\bound{v }}
\indentrel{3}\begin{verbatim}
   (1)
         3 3       5       2        6    3        8      5
     - 2x y  + (24x  + 24)y  + (- 4x  - x )y + 48x  + 12x
   +
        3
     48x  + 12
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemFactorIntRatPageEmpty1}
\begin{paste}{ugProblemFactorIntRatPageEmpty1}{ugProblemFactorIntRatPagePatch1}
\pastebutton{ugProblemFactorIntRatPageEmpty1}{\showpaste}
\tab{5}\spadcommand{v := (4*x**3+2*y**2+1)*(12*x**5-x**3*y+12)\bound{v }}
\end{paste}\end{patch}

\begin{patch}{ugProblemFactorIntRatPagePatch2}
\begin{paste}{ugProblemFactorIntRatPageFull2}{ugProblemFactorIntRatPageEmpty2}
\pastebutton{ugProblemFactorIntRatPageFull2}{\hidepaste}
\tab{5}\spadcommand{factor v\free{v }}
\indentrel{3}\begin{verbatim}
            3       5         2     3
   (2)  - (x y - 12x  - 12)(2y  + 4x  + 1)
                      Type: Factored Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemFactorIntRatPageEmpty2}
\begin{paste}{ugProblemFactorIntRatPageEmpty2}{ugProblemFactorIntRatPagePatch2}
\pastebutton{ugProblemFactorIntRatPageEmpty2}{\showpaste}
\tab{5}\spadcommand{factor v\free{v }}
\end{paste}\end{patch}

\begin{patch}{ugProblemFactorIntRatPagePatch3}
\begin{paste}{ugProblemFactorIntRatPageFull3}{ugProblemFactorIntRatPageEmpty3}
\pastebutton{ugProblemFactorIntRatPageFull3}{\hidepaste}
\tab{5}\spadcommand{w := (4*x**3+(2/3)*x**2+1)*(12*x**5-(1/2)*x**3+12)\bound{w }}
\indentrel{3}\begin{verbatim}
            8     7     6   35  5   95  3     2
   (3)  48x  + 8x  - 2x  +  x  +  x  + 8x  + 12
                            3      2
```

```
                      Type: Polynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemFactorIntRatPageEmpty3}
\begin{paste}{ugProblemFactorIntRatPageEmpty3}{ugProblemFactorIntRatPagePatch3}
\pastebutton{ugProblemFactorIntRatPageEmpty3}{\showpaste}
\tab{5}\spadcommand{w := (4*x**3+(2/3)*x**2+1)*(12*x**5-(1/2)*x**3+12)\bound{w }}
\end{paste}\end{patch}

\begin{patch}{ugProblemFactorIntRatPagePatch4}
\begin{paste}{ugProblemFactorIntRatPageFull4}{ugProblemFactorIntRatPageEmpty4}
\pastebutton{ugProblemFactorIntRatPageFull4}{\hidepaste}
\tab{5}\spadcommand{factor w\free{w }}
\indentrel{3}\begin{verbatim}
            3   1  2   1   5    1  3
   (4)  48(x  +  x  + )(x  -  x  + 1)
                6      4       24
             Type: Factored Polynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemFactorIntRatPageEmpty4}
\begin{paste}{ugProblemFactorIntRatPageEmpty4}{ugProblemFactorIntRatPagePatch4}
\pastebutton{ugProblemFactorIntRatPageEmpty4}{\showpaste}
\tab{5}\spadcommand{factor w\free{w }}
\end{paste}\end{patch}
```

——–

Finite Field Coefficients

⇒ "notitle" (ugProblemFinitePage) 12 on page 2235

— **ug08.ht** —

```
\begin{page}{ugProblemFactorFFPage}{8.2.2. Finite Field Coefficients}
\beginscroll

Polynomials with coefficients in a finite field
can be also be factored.

\labelSpace{3pc}
\xtc{
}{
\spadpaste{u : POLY(PF(19)) :=3*x**4+2*x**2+15*x+18 \bound{u}}
}
\xtc{
```

```
These include the integers mod \axiom{p}, where \axiom{p} is prime, and
extensions of these fields.
}{
\spadpaste{factor u \free{u}}
}
\xtc{
Convert this to have coefficients in the finite field with
\texht{$19^3$}{\axiom{19**3}} elements.  See
\downlink{``Finite Fields''}{ugProblemFinitePage} in Section
8.11\ignore{ugProblemFinite} for more information
about finite fields.
}{
\spadpaste{factor(u :: POLY FFX(PF 19,3)) \free{u}}
}
%

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugProblemFactorFFPagePatch1}
\begin{paste}{ugProblemFactorFFPageFull1}{ugProblemFactorFFPageEmpty1}
\pastebutton{ugProblemFactorFFPageFull1}{\hidepaste}
\tab{5}\spadcommand{u : POLY(PF(19)) :=3*x**4+2*x**2+15*x+18\bound{u }}
\indentrel{3}\begin{verbatim}
          4     2
   (1)  3x  + 2x  + 15x + 18
                         Type: Polynomial PrimeField 19
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemFactorFFPageEmpty1}
\begin{paste}{ugProblemFactorFFPageEmpty1}{ugProblemFactorFFPagePatch1}
\pastebutton{ugProblemFactorFFPageEmpty1}{\showpaste}
\tab{5}\spadcommand{u : POLY(PF(19)) :=3*x**4+2*x**2+15*x+18\bound{u }}
\end{paste}\end{patch}

\begin{patch}{ugProblemFactorFFPagePatch2}
\begin{paste}{ugProblemFactorFFPageFull2}{ugProblemFactorFFPageEmpty2}
\pastebutton{ugProblemFactorFFPageFull2}{\hidepaste}
\tab{5}\spadcommand{factor u\free{u }}
\indentrel{3}\begin{verbatim}
                   3    2
   (2)  3(x + 18)(x  + x  + 8x + 13)
                Type: Factored Polynomial PrimeField 19
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemFactorFFPageEmpty2}
\begin{paste}{ugProblemFactorFFPageEmpty2}{ugProblemFactorFFPagePatch2}
```

```
\pastebutton{ugProblemFactorFFPageEmpty2}{\showpaste}
\tab{5}\spadcommand{factor u\free{u }}
\end{paste}\end{patch}

\begin{patch}{ugProblemFactorFFPagePatch3}
\begin{paste}{ugProblemFactorFFPageFull3}{ugProblemFactorFFPageEmpty3}
\pastebutton{ugProblemFactorFFPageFull3}{\hidepaste}
\tab{5}\spadcommand{factor(u :: POLY FFX(PF 19,3))\free{u }}
\indentrel{3}\begin{verbatim}
   (3)
                         2
     3(x + 18)(x + 5%BC  + 3%BC + 13)
  *
                2                         2
     (x + 16%BC  + 14%BC + 13)(x + 17%BC  + 2%BC + 13)
Type: Factored Polynomial FiniteFieldExtension(PrimeField 19,3)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemFactorFFPageEmpty3}
\begin{paste}{ugProblemFactorFFPageEmpty3}{ugProblemFactorFFPagePatch3}
\pastebutton{ugProblemFactorFFPageEmpty3}{\showpaste}
\tab{5}\spadcommand{factor(u :: POLY FFX(PF 19,3))\free{u }}
\end{paste}\end{patch}
```

———

Simple Algebraic Extension Field Coefficients

— ug08.ht —

```
\begin{page}{ugProblemFactorAlgPage}
{8.2.3. Simple Algebraic Extension Field Coefficients}
\beginscroll

Polynomials with coefficients in simple algebraic extensions
of the rational numbers can be factored.

\labelSpace{2pc}
\xtc{
Here, \axiom{aa} and \axiom{bb} are symbolic roots of polynomials.
}{
\spadpaste{aa := rootOf(aa**2+aa+1) \bound{aa}}
}
\xtc{
}{
\spadpaste{p:=(x**3+aa**2*x+y)*(aa*x**2+aa*x+aa*y**2)**2
```

```
\free{aa}\bound{p}}
}
\xtc{
Note that the second argument to factor can be a list of
algebraic extensions to factor over.
}{
\spadpaste{factor(p,[aa]) \free{p aa}}
}
\xtc{
This factors \axiom{x**2+3} over the integers.
}{
\spadpaste{factor(x**2+3)}
}
\xtc{
Factor the same polynomial over the field obtained by adjoining
\axiom{aa} to the rational numbers.
}{
\spadpaste{factor(x**2+3,[aa]) \free{aa}}
}
\xtc{
Factor \axiom{x**6+108} over the same field.
}{
\spadpaste{factor(x**6+108,[aa]) \free{aa}}
}
\xtc{
}{
\spadpaste{bb:=rootOf(bb**3-2) \bound{bb}}
}
\xtc{
}{
\spadpaste{factor(x**6+108,[bb]) \free{bb}}
}
\xtc{
Factor again over the field obtained by adjoining both \axiom{aa}
and \axiom{bb} to the rational numbers.
}{
\spadpaste{factor(x**6+108,[aa,bb]) \free{aa bb}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugProblemFactorAlgPagePatch1}
\begin{paste}{ugProblemFactorAlgPageFull1}{ugProblemFactorAlgPageEmpty1}
\pastebutton{ugProblemFactorAlgPageFull1}{\hidepaste}
\tab{5}\spadcommand{aa := rootOf(aa**2+aa+1)\bound{aa }}
\indentrel{3}\begin{verbatim}
   (1)  aa
                                  Type: AlgebraicNumber
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemFactorAlgPageEmpty1}
\begin{paste}{ugProblemFactorAlgPageEmpty1}{ugProblemFactorAlgPagePatch1}
\pastebutton{ugProblemFactorAlgPageEmpty1}{\showpaste}
\tab{5}\spadcommand{aa := rootOf(aa**2+aa+1)\bound{aa }}
\end{paste}\end{patch}

\begin{patch}{ugProblemFactorAlgPagePatch2}
\begin{paste}{ugProblemFactorAlgPageFull2}{ugProblemFactorAlgPageEmpty2}
\pastebutton{ugProblemFactorAlgPageFull2}{\hidepaste}
\tab{5}\spadcommand{p:=(x**3+aa**2*x+y)*(aa*x**2+aa*x+aa*y**2)**2\free{aa }\bound{p }}
\indentrel{3}\begin{verbatim}
   (2)
                5              3         4
     (- aa - 1)y  + ((- aa - 1)x  + aa x)y
   +
                  2                 3
     ((- 2aa - 2)x  + (- 2aa - 2)x)y
   +
                  5                4        3        2  2
     ((- 2aa - 2)x  + (- 2aa - 2)x  + 2aa x  + 2aa x )y
   +
                 4              3              2
     ((- aa - 1)x  + (- 2aa - 2)x  + (- aa - 1)x )y
   +
                7               6    5        4       3
     (- aa - 1)x  + (- 2aa - 2)x  - x  + 2aa x  + aa x
                       Type: Polynomial AlgebraicNumber
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemFactorAlgPageEmpty2}
\begin{paste}{ugProblemFactorAlgPageEmpty2}{ugProblemFactorAlgPagePatch2}
\pastebutton{ugProblemFactorAlgPageEmpty2}{\showpaste}
\tab{5}\spadcommand{p:=(x**3+aa**2*x+y)*(aa*x**2+aa*x+aa*y**2)**2\free{aa }\bound{p }}
\end{paste}\end{patch}

\begin{patch}{ugProblemFactorAlgPagePatch3}
\begin{paste}{ugProblemFactorAlgPageFull3}{ugProblemFactorAlgPageEmpty3}
\pastebutton{ugProblemFactorAlgPageFull3}{\hidepaste}
\tab{5}\spadcommand{factor(p,[aa])\free{p aa }}
\indentrel{3}\begin{verbatim}
                        3                 2    2     2
   (3)  (- aa - 1)(y + x  + (- aa - 1)x)(y  + x  + x)
              Type: Factored Polynomial AlgebraicNumber
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugProblemFactorAlgPageEmpty3}
\begin{paste}{ugProblemFactorAlgPageEmpty3}{ugProblemFactorAlgPagePatch3}
\pastebutton{ugProblemFactorAlgPageEmpty3}{\showpaste}
\tab{5}\spadcommand{factor(p,[aa])\free{p aa }}
\end{paste}\end{patch}

\begin{patch}{ugProblemFactorAlgPagePatch4}
\begin{paste}{ugProblemFactorAlgPageFull4}{ugProblemFactorAlgPageEmpty4}
\pastebutton{ugProblemFactorAlgPageFull4}{\hidepaste}
\tab{5}\spadcommand{factor(x**2+3)}
\indentrel{3}\begin{verbatim}
         2
   (4)  x  + 3
                      Type: Factored Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemFactorAlgPageEmpty4}
\begin{paste}{ugProblemFactorAlgPageEmpty4}{ugProblemFactorAlgPagePatch4}
\pastebutton{ugProblemFactorAlgPageEmpty4}{\showpaste}
\tab{5}\spadcommand{factor(x**2+3)}
\end{paste}\end{patch}

\begin{patch}{ugProblemFactorAlgPagePatch5}
\begin{paste}{ugProblemFactorAlgPageFull5}{ugProblemFactorAlgPageEmpty5}
\pastebutton{ugProblemFactorAlgPageFull5}{\hidepaste}
\tab{5}\spadcommand{factor(x**2+3,[aa])\free{aa }}
\indentrel{3}\begin{verbatim}
   (5)  (x - 2aa - 1)(x + 2aa + 1)
              Type: Factored Polynomial AlgebraicNumber
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemFactorAlgPageEmpty5}
\begin{paste}{ugProblemFactorAlgPageEmpty5}{ugProblemFactorAlgPagePatch5}
\pastebutton{ugProblemFactorAlgPageEmpty5}{\showpaste}
\tab{5}\spadcommand{factor(x**2+3,[aa])\free{aa }}
\end{paste}\end{patch}

\begin{patch}{ugProblemFactorAlgPagePatch6}
\begin{paste}{ugProblemFactorAlgPageFull6}{ugProblemFactorAlgPageEmpty6}
\pastebutton{ugProblemFactorAlgPageFull6}{\hidepaste}
\tab{5}\spadcommand{factor(x**6+108,[aa])\free{aa }}
\indentrel{3}\begin{verbatim}
          3              3
   (6)  (x  - 12aa - 6)(x  + 12aa + 6)
              Type: Factored Polynomial AlgebraicNumber
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugProblemFactorAlgPageEmpty6}
\begin{paste}{ugProblemFactorAlgPageEmpty6}{ugProblemFactorAlgPagePatch6}
\pastebutton{ugProblemFactorAlgPageEmpty6}{\showpaste}
\tab{5}\spadcommand{factor(x**6+108,[aa])\free{aa }}
\end{paste}\end{patch}

\begin{patch}{ugProblemFactorAlgPagePatch7}
\begin{paste}{ugProblemFactorAlgPageFull7}{ugProblemFactorAlgPageEmpty7}
\pastebutton{ugProblemFactorAlgPageFull7}{\hidepaste}
\tab{5}\spadcommand{bb:=rootOf(bb**3-2)\bound{bb }}
\indentrel{3}\begin{verbatim}
   (7)  bb
                                 Type: AlgebraicNumber
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemFactorAlgPageEmpty7}
\begin{paste}{ugProblemFactorAlgPageEmpty7}{ugProblemFactorAlgPagePatch7}
\pastebutton{ugProblemFactorAlgPageEmpty7}{\showpaste}
\tab{5}\spadcommand{bb:=rootOf(bb**3-2)\bound{bb }}
\end{paste}\end{patch}

\begin{patch}{ugProblemFactorAlgPagePatch8}
\begin{paste}{ugProblemFactorAlgPageFull8}{ugProblemFactorAlgPageEmpty8}
\pastebutton{ugProblemFactorAlgPageFull8}{\hidepaste}
\tab{5}\spadcommand{factor(x**6+108,[bb])\free{bb }}
\indentrel{3}\begin{verbatim}
   (8)
     2              2   2      2   2              2
   (x  - 3bb x + 3bb )(x  + 3bb )(x  + 3bb x + 3bb )
             Type: Factored Polynomial AlgebraicNumber
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemFactorAlgPageEmpty8}
\begin{paste}{ugProblemFactorAlgPageEmpty8}{ugProblemFactorAlgPagePatch8}
\pastebutton{ugProblemFactorAlgPageEmpty8}{\showpaste}
\tab{5}\spadcommand{factor(x**6+108,[bb])\free{bb }}
\end{paste}\end{patch}

\begin{patch}{ugProblemFactorAlgPagePatch9}
\begin{paste}{ugProblemFactorAlgPageFull9}{ugProblemFactorAlgPageEmpty9}
\pastebutton{ugProblemFactorAlgPageFull9}{\hidepaste}
\tab{5}\spadcommand{factor(x**6+108,[aa,bb])\free{aa bb }}
\indentrel{3}\begin{verbatim}
   (9)
     (x + (- 2aa - 1)bb)(x + (- aa - 2)bb)
  *
     (x + (- aa + 1)bb)(x + (aa - 1)bb)(x + (aa + 2)bb)
  *
```

```
     (x + (2aa + 1)bb)
              Type: Factored Polynomial AlgebraicNumber
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemFactorAlgPageEmpty9}
\begin{paste}{ugProblemFactorAlgPageEmpty9}{ugProblemFactorAlgPagePatch9}
\pastebutton{ugProblemFactorAlgPageEmpty9}{\showpaste}
\tab{5}\spadcommand{factor(x**6+108,[aa,bb])\free{aa bb }}
\end{paste}\end{patch}
```

———-

Factoring Rational Functions

— ug08.ht —

```
\begin{page}{ugProblemFactorRatFunPage}
{8.2.4. Factoring Rational Functions}
\beginscroll

Since fractions of polynomials form a field, every element (other than
zero) divides any other, so there is no useful notion of irreducible
factors.  Thus the \axiomFun{factor} operation is not very useful for
fractions of polynomials.

\xtc{
There is, instead, a specific operation \axiomFun{factorFraction}
that separately factors the numerator and denominator and returns
a fraction of the factored results.
}{
\spadpaste{factorFraction((x**2-4)/(y**2-4))}
}
\xtc{
You can also use \axiomFun{map}. This expression
applies the \axiomFun{factor} operation
to the numerator and denominator.
}{
\spadpaste{map(factor,(x**2-4)/(y**2-4))}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugProblemFactorRatFunPagePatch1}
\begin{paste}{ugProblemFactorRatFunPageFull1}{ugProblemFactorRatFunPageEmpty1}
```

```
\pastebutton{ugProblemFactorRatFunPageFull1}{\hidepaste}
\tab{5}\spadcommand{factorFraction((x**2-4)/(y**2-4))}
\indentrel{3}\begin{verbatim}
        (x - 2)(x + 2)
   (1)
        (y - 2)(y + 2)
             Type: Fraction Factored Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemFactorRatFunPageEmpty1}
\begin{paste}{ugProblemFactorRatFunPageEmpty1}{ugProblemFactorRatFunPagePatch1}
\pastebutton{ugProblemFactorRatFunPageEmpty1}{\showpaste}
\tab{5}\spadcommand{factorFraction((x**2-4)/(y**2-4))}
\end{paste}\end{patch}

\begin{patch}{ugProblemFactorRatFunPagePatch2}
\begin{paste}{ugProblemFactorRatFunPageFull2}{ugProblemFactorRatFunPageEmpty2}
\pastebutton{ugProblemFactorRatFunPageFull2}{\hidepaste}
\tab{5}\spadcommand{map(factor,(x**2-4)/(y**2-4))}
\indentrel{3}\begin{verbatim}
        (x - 2)(x + 2)
   (2)
        (y - 2)(y + 2)
             Type: Fraction Factored Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemFactorRatFunPageEmpty2}
\begin{paste}{ugProblemFactorRatFunPageEmpty2}{ugProblemFactorRatFunPagePatch2}
\pastebutton{ugProblemFactorRatFunPageEmpty2}{\showpaste}
\tab{5}\spadcommand{map(factor,(x**2-4)/(y**2-4))}
\end{paste}\end{patch}
```

——–

Manipulating Symbolic Roots of a Polynomial

⇒ "notitle" (ugxProblemOnePolPage) 12 on page 2135
⇒ "notitle" (ugxProblemPolSysPage) 12 on page 2140
⇒ "notitle" (ugxProblemSymRootOnePage) 12 on page 2113
⇒ "notitle" (ugxProblemSymRootAllPage) 12 on page 2117

— **ug08.ht** —

```
\begin{page}{ugProblemSymRootPage}
{8.3. Manipulating Symbolic Roots of a Polynomial}
```

```
\beginscroll
%
In this section we show you how to work with one root or all roots
of a polynomial.
These roots are represented symbolically (as opposed to being
numeric approximations).
See \downlink{``Solution of a Single Polynomial Equation''}
{ugxProblemOnePolPage}
in Section 8.5.2\ignore{ugxProblemOnePol} and
\downlink{``Solution of Systems of Polynomial Equations''}
{ugxProblemPolSysPage} in
Section 8.5.3\ignore{ugxProblemPolSys} for
information about solving for the roots of one or more
polynomials.

\beginmenu
    \menudownlink{{8.3.1. Using a Single Root of a Polynomial}}
{ugxProblemSymRootOnePage}
    \menudownlink{{8.3.2. Using All Roots of a Polynomial}}
{ugxProblemSymRootAllPage}
\endmenu
\endscroll
\autobuttons
\end{page}
```

———-

Using a Single Root of a Polynomial

— ug08.ht —

```
\begin{page}{ugxProblemSymRootOnePage}
{8.3.1. Using a Single Root of a Polynomial}
\beginscroll

Use \axiomFun{rootOf} to get a symbolic root of a polynomial:
\axiom{rootOf(p, x)} returns a root of \axiom{p(x)}.

\labelSpace{2pc}
\xtc{
This creates an algebraic number \axiom{a}.
}{
\spadpaste{a := rootOf(a**4+1,a) \bound{a}}
}
\xtc{
To find the algebraic relation that defines \axiom{a},
use \axiomFun{definingPolynomial}.
```

```
}{
\spadpaste{definingPolynomial a \free{a}}
}
\xtc{
You can use \axiom{a} in any further expression,
including a nested \axiomFun{rootOf}.
}{
\spadpaste{b := rootOf(b**2-a-1,b) \free{a}\bound{b}}
}
\xtc{
Higher powers of the roots are automatically reduced during
calculations.
}{
\spadpaste{a + b \free{a b}\bound{c}}
}
\xtc{
}{
\spadpaste{\% ** 5 \free{c}}
}
\xtc{
The operation \axiomFun{zeroOf} is similar to \axiomFun{rootOf},
except that it may express the root using radicals in some cases.
}{
\spadpaste{rootOf(c**2+c+1,c)}
}
\xtc{
}{
\spadpaste{zeroOf(d**2+d+1,d)}
}
\xtc{
}{
\spadpaste{rootOf(e**5-2,e)}
}
\xtc{
}{
\spadpaste{zeroOf(f**5-2,f)}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxProblemSymRootOnePagePatch1}
\begin{paste}{ugxProblemSymRootOnePageFull1}{ugxProblemSymRootOnePageEmpty1}
\pastebutton{ugxProblemSymRootOnePageFull1}{\hidepaste}
\tab{5}\spadcommand{a := rootOf(a**4+1,a)\bound{a }}
\indentrel{3}\begin{verbatim}
   (1)  a
                              Type: Expression Integer
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSymRootOnePageEmpty1}
\begin{paste}{ugxProblemSymRootOnePageEmpty1}{ugxProblemSymRootOnePagePatch1}
\pastebutton{ugxProblemSymRootOnePageEmpty1}{\showpaste}
\tab{5}\spadcommand{a := rootOf(a**4+1,a)\bound{a }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSymRootOnePagePatch2}
\begin{paste}{ugxProblemSymRootOnePageFull2}{ugxProblemSymRootOnePageEmpty2}
\pastebutton{ugxProblemSymRootOnePageFull2}{\hidepaste}
\tab{5}\spadcommand{definingPolynomial a\free{a }}
\indentrel{3}\begin{verbatim}
         4
   (2)  a  + 1
                              Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSymRootOnePageEmpty2}
\begin{paste}{ugxProblemSymRootOnePageEmpty2}{ugxProblemSymRootOnePagePatch2}
\pastebutton{ugxProblemSymRootOnePageEmpty2}{\showpaste}
\tab{5}\spadcommand{definingPolynomial a\free{a }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSymRootOnePagePatch3}
\begin{paste}{ugxProblemSymRootOnePageFull3}{ugxProblemSymRootOnePageEmpty3}
\pastebutton{ugxProblemSymRootOnePageFull3}{\hidepaste}
\tab{5}\spadcommand{b := rootOf(b**2-a-1,b)\free{a }\bound{b }}
\indentrel{3}\begin{verbatim}
   (3)  b
                              Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSymRootOnePageEmpty3}
\begin{paste}{ugxProblemSymRootOnePageEmpty3}{ugxProblemSymRootOnePagePatch3}
\pastebutton{ugxProblemSymRootOnePageEmpty3}{\showpaste}
\tab{5}\spadcommand{b := rootOf(b**2-a-1,b)\free{a }\bound{b }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSymRootOnePagePatch4}
\begin{paste}{ugxProblemSymRootOnePageFull4}{ugxProblemSymRootOnePageEmpty4}
\pastebutton{ugxProblemSymRootOnePageFull4}{\hidepaste}
\tab{5}\spadcommand{a + b\free{a b }\bound{c }}
\indentrel{3}\begin{verbatim}
   (4)  b + a
                              Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugxProblemSymRootOnePageEmpty4}
\begin{paste}{ugxProblemSymRootOnePageEmpty4}{ugxProblemSymRootOnePagePatch4}
\pastebutton{ugxProblemSymRootOnePageEmpty4}{\showpaste}
\tab{5}\spadcommand{a + b\free{a b }\bound{c }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSymRootOnePagePatch5}
\begin{paste}{ugxProblemSymRootOnePageFull5}{ugxProblemSymRootOnePageEmpty5}
\pastebutton{ugxProblemSymRootOnePageFull5}{\hidepaste}
\tab{5}\spadcommand{\% ** 5\free{c }}
\indentrel{3}\begin{verbatim}
            3      2                  3      2
   (5)  (10a  + 11a  + 2a - 4)b + 15a  + 10a  + 4a - 10
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSymRootOnePageEmpty5}
\begin{paste}{ugxProblemSymRootOnePageEmpty5}{ugxProblemSymRootOnePagePatch5}
\pastebutton{ugxProblemSymRootOnePageEmpty5}{\showpaste}
\tab{5}\spadcommand{\% ** 5\free{c }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSymRootOnePagePatch6}
\begin{paste}{ugxProblemSymRootOnePageFull6}{ugxProblemSymRootOnePageEmpty6}
\pastebutton{ugxProblemSymRootOnePageFull6}{\hidepaste}
\tab{5}\spadcommand{rootOf(c**2+c+1,c)}
\indentrel{3}\begin{verbatim}
   (6)  c
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSymRootOnePageEmpty6}
\begin{paste}{ugxProblemSymRootOnePageEmpty6}{ugxProblemSymRootOnePagePatch6}
\pastebutton{ugxProblemSymRootOnePageEmpty6}{\showpaste}
\tab{5}\spadcommand{rootOf(c**2+c+1,c)}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSymRootOnePagePatch7}
\begin{paste}{ugxProblemSymRootOnePageFull7}{ugxProblemSymRootOnePageEmpty7}
\pastebutton{ugxProblemSymRootOnePageFull7}{\hidepaste}
\tab{5}\spadcommand{zeroOf(d**2+d+1,d)}
\indentrel{3}\begin{verbatim}

        \- 3  - 1
   (7)
             2
                               Type: Expression Integer
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSymRootOnePageEmpty7}
\begin{paste}{ugxProblemSymRootOnePageEmpty7}{ugxProblemSymRootOnePagePatch7}
\pastebutton{ugxProblemSymRootOnePageEmpty7}{\showpaste}
\tab{5}\spadcommand{zeroOf(d**2+d+1,d)}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSymRootOnePagePatch8}
\begin{paste}{ugxProblemSymRootOnePageFull8}{ugxProblemSymRootOnePageEmpty8}
\pastebutton{ugxProblemSymRootOnePageFull8}{\hidepaste}
\tab{5}\spadcommand{rootOf(e**5-2,e)}
\indentrel{3}\begin{verbatim}
   (8)  e
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSymRootOnePageEmpty8}
\begin{paste}{ugxProblemSymRootOnePageEmpty8}{ugxProblemSymRootOnePagePatch8}
\pastebutton{ugxProblemSymRootOnePageEmpty8}{\showpaste}
\tab{5}\spadcommand{rootOf(e**5-2,e)}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSymRootOnePagePatch9}
\begin{paste}{ugxProblemSymRootOnePageFull9}{ugxProblemSymRootOnePageEmpty9}
\pastebutton{ugxProblemSymRootOnePageFull9}{\hidepaste}
\tab{5}\spadcommand{zeroOf(f**5-2,f)}
\indentrel{3}\begin{verbatim}
        5
   (9)  \2
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSymRootOnePageEmpty9}
\begin{paste}{ugxProblemSymRootOnePageEmpty9}{ugxProblemSymRootOnePagePatch9}
\pastebutton{ugxProblemSymRootOnePageEmpty9}{\showpaste}
\tab{5}\spadcommand{zeroOf(f**5-2,f)}
\end{paste}\end{patch}
```

Using All Roots of a Polynomial

⇒ "notitle" (ugxProblemOnePolPage) 12 on page 2135

— ug08.ht —

```
\begin{page}{ugxProblemSymRootAllPage}
{8.3.2. Using All Roots of a Polynomial}
\beginscroll

Use \axiomFun{rootsOf} to get all symbolic roots of a polynomial:
\axiom{rootsOf(p, x)} returns a
list of all the roots of \axiom{p(x)}.
If \axiom{p(x)} has a multiple root of order \axiom{n}, then that root
appears \axiom{n} times in the list.
\texht{\typeout{Make sure these variables are x0 etc}}{}

\xtc{
Compute all the roots of \axiom{x**4 + 1}.
}{
\spadpaste{l := rootsOf(x**4+1,x) \bound{l}}
}
\xtc{
As a side effect, the variables \axiom{\%x0, \%x1} and \axiom{\%x2}
are bound to the first three roots of \axiom{x**4+1}.
}{
\spadpaste{\%x0**5 \free{l}}
}
\xtc{
Although they all satisfy \axiom{x**4 + 1 = 0, \%x0, \%x1,}
and \axiom{\%x2} are different algebraic numbers.
To find the algebraic relation that defines each of them,
use \axiomFun{definingPolynomial}.
}{
\spadpaste{definingPolynomial \%x0 \free{l}}
}
\xtc{
}{
\spadpaste{definingPolynomial \%x1 \free{l}}
}
\xtc{
}{
\spadpaste{definingPolynomial \%x2 \free{l}}
}
\xtc{
We can check that the sum and product of the roots of \axiom{x**4+1} are
its trace and norm.
}{
\spadpaste{x3 := last l \free{l} \bound{x3}}
}
\xtc{
}{
\spadpaste{\%x0 + \%x1 + \%x2 + x3 \free{x3}}
}
```

```
\xtc{
}{
\spadpaste{\%x0 * \%x1 * \%x2 * x3 \free{x3}}
}
\xtc{
Corresponding to the pair of operations
\axiomFun{rootOf}/\axiomFun{zeroOf} in
\downlink{``Solution of a Single Polynomial Equation''}
{ugxProblemOnePolPage} in Section
8.5.2\ignore{ugxProblemOnePol}, there is an
operation \axiomFun{zerosOf} that, like \axiomFun{rootsOf}, computes
all the roots of a given polynomial, but which expresses some of them
in terms of radicals.
}{
\spadpaste{zerosOf(y**4+1,y) \bound{z}}
}
\xtc{
As you see, only one implicit algebraic number was created
(\axiom{\%y1}), and its defining equation is this.
The other three roots are expressed in radicals.
}{
\spadpaste{definingPolynomial \%y1 \free{z}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxProblemSymRootAllPagePatch1}
\begin{paste}{ugxProblemSymRootAllPageFull1}{ugxProblemSymRootAllPageEmpty1}
\pastebutton{ugxProblemSymRootAllPageFull1}{\hidepaste}
\tab{5}\spadcommand{l := rootsOf(x**4+1,x)\bound{l }}
\indentrel{3}\begin{verbatim}
   (1)  [%x0,%x0 %x1,- %x0,- %x0 %x1]
                          Type: List Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSymRootAllPageEmpty1}
\begin{paste}{ugxProblemSymRootAllPageEmpty1}{ugxProblemSymRootAllPagePatch1}
\pastebutton{ugxProblemSymRootAllPageEmpty1}{\showpaste}
\tab{5}\spadcommand{l := rootsOf(x**4+1,x)\bound{l }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSymRootAllPagePatch2}
\begin{paste}{ugxProblemSymRootAllPageFull2}{ugxProblemSymRootAllPageEmpty2}
\pastebutton{ugxProblemSymRootAllPageFull2}{\hidepaste}
\tab{5}\spadcommand{\%x0**5\free{l }}
\indentrel{3}\begin{verbatim}
   (2)  - %x0
```

```
                              Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSymRootAllPageEmpty2}
\begin{paste}{ugxProblemSymRootAllPageEmpty2}{ugxProblemSymRootAllPagePatch2}
\pastebutton{ugxProblemSymRootAllPageEmpty2}{\showpaste}
\tab{5}\spadcommand{\%x0**5\free{l }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSymRootAllPagePatch3}
\begin{paste}{ugxProblemSymRootAllPageFull3}{ugxProblemSymRootAllPageEmpty3}
\pastebutton{ugxProblemSymRootAllPageFull3}{\hidepaste}
\tab{5}\spadcommand{definingPolynomial \%x0\free{l }}
\indentrel{3}\begin{verbatim}
          4
   (3)  %x0  + 1
                              Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSymRootAllPageEmpty3}
\begin{paste}{ugxProblemSymRootAllPageEmpty3}{ugxProblemSymRootAllPagePatch3}
\pastebutton{ugxProblemSymRootAllPageEmpty3}{\showpaste}
\tab{5}\spadcommand{definingPolynomial \%x0\free{l }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSymRootAllPagePatch4}
\begin{paste}{ugxProblemSymRootAllPageFull4}{ugxProblemSymRootAllPageEmpty4}
\pastebutton{ugxProblemSymRootAllPageFull4}{\hidepaste}
\tab{5}\spadcommand{definingPolynomial \%x1\free{l }}
\indentrel{3}\begin{verbatim}
          2
   (4)  %x1  + 1
                              Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSymRootAllPageEmpty4}
\begin{paste}{ugxProblemSymRootAllPageEmpty4}{ugxProblemSymRootAllPagePatch4}
\pastebutton{ugxProblemSymRootAllPageEmpty4}{\showpaste}
\tab{5}\spadcommand{definingPolynomial \%x1\free{l }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSymRootAllPagePatch5}
\begin{paste}{ugxProblemSymRootAllPageFull5}{ugxProblemSymRootAllPageEmpty5}
\pastebutton{ugxProblemSymRootAllPageFull5}{\hidepaste}
\tab{5}\spadcommand{definingPolynomial \%x2\free{l }}
\indentrel{3}\begin{verbatim}
   (5)  - %x2 + %%var
```

```
                              Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSymRootAllPageEmpty5}
\begin{paste}{ugxProblemSymRootAllPageEmpty5}{ugxProblemSymRootAllPagePatch5}
\pastebutton{ugxProblemSymRootAllPageEmpty5}{\showpaste}
\tab{5}\spadcommand{definingPolynomial \%x2\free{l }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSymRootAllPagePatch6}
\begin{paste}{ugxProblemSymRootAllPageFull6}{ugxProblemSymRootAllPageEmpty6}
\pastebutton{ugxProblemSymRootAllPageFull6}{\hidepaste}
\tab{5}\spadcommand{x3 := last l\free{l }\bound{x3 }}
\indentrel{3}\begin{verbatim}
   (6)  - %x0 %x1
                              Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSymRootAllPageEmpty6}
\begin{paste}{ugxProblemSymRootAllPageEmpty6}{ugxProblemSymRootAllPagePatch6}
\pastebutton{ugxProblemSymRootAllPageEmpty6}{\showpaste}
\tab{5}\spadcommand{x3 := last l\free{l }\bound{x3 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSymRootAllPagePatch7}
\begin{paste}{ugxProblemSymRootAllPageFull7}{ugxProblemSymRootAllPageEmpty7}
\pastebutton{ugxProblemSymRootAllPageFull7}{\hidepaste}
\tab{5}\spadcommand{\%x0 + \%x1 + \%x2 + x3\free{x3 }}
\indentrel{3}\begin{verbatim}
   (7)  (- %x0 + 1)%x1 + %x0 + %x2
                              Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSymRootAllPageEmpty7}
\begin{paste}{ugxProblemSymRootAllPageEmpty7}{ugxProblemSymRootAllPagePatch7}
\pastebutton{ugxProblemSymRootAllPageEmpty7}{\showpaste}
\tab{5}\spadcommand{\%x0 + \%x1 + \%x2 + x3\free{x3 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSymRootAllPagePatch8}
\begin{paste}{ugxProblemSymRootAllPageFull8}{ugxProblemSymRootAllPageEmpty8}
\pastebutton{ugxProblemSymRootAllPageFull8}{\hidepaste}
\tab{5}\spadcommand{\%x0 * \%x1 * \%x2 * x3\free{x3 }}
\indentrel{3}\begin{verbatim}
               2
   (8)  %x2 %x0
                              Type: Expression Integer
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSymRootAllPageEmpty8}
\begin{paste}{ugxProblemSymRootAllPageEmpty8}{ugxProblemSymRootAllPagePatch8}
\pastebutton{ugxProblemSymRootAllPageEmpty8}{\showpaste}
\tab{5}\spadcommand{\%x0 * \%x1 * \%x2 * x3\free{x3 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSymRootAllPagePatch9}
\begin{paste}{ugxProblemSymRootAllPageFull9}{ugxProblemSymRootAllPageEmpty9}
\pastebutton{ugxProblemSymRootAllPageFull9}{\hidepaste}
\tab{5}\spadcommand{zerosOf(y**4+1,y)\bound{z }}
\indentrel{3}\begin{verbatim}
   (9)

    \- 1  + 1 \- 1  - 1 - \- 1  - 1 - \- 1  + 1
   [,,,]

       \2        \2         \2          \2
                          Type: List Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSymRootAllPageEmpty9}
\begin{paste}{ugxProblemSymRootAllPageEmpty9}{ugxProblemSymRootAllPagePatch9}
\pastebutton{ugxProblemSymRootAllPageEmpty9}{\showpaste}
\tab{5}\spadcommand{zerosOf(y**4+1,y)\bound{z }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSymRootAllPagePatch10}
\begin{paste}{ugxProblemSymRootAllPageFull10}{ugxProblemSymRootAllPageEmpty10}
\pastebutton{ugxProblemSymRootAllPageFull10}{\hidepaste}
\tab{5}\spadcommand{definingPolynomial \%y1\free{z }}
\indentrel{3}\begin{verbatim}
              2
   (10)  %%var  + 1
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSymRootAllPageEmpty10}
\begin{paste}{ugxProblemSymRootAllPageEmpty10}{ugxProblemSymRootAllPagePatch10}
\pastebutton{ugxProblemSymRootAllPageEmpty10}{\showpaste}
\tab{5}\spadcommand{definingPolynomial \%y1\free{z }}
\end{paste}\end{patch}
```

Computation of Eigenvalues and Eigenvectors

— ug08.ht —

```
\begin{page}{ugProblemEigenPage}
{8.4. Computation of Eigenvalues and Eigenvectors}
\beginscroll
%
In this section we show you
some of Axiom's facilities for computing and
manipulating eigenvalues and eigenvectors, also called
characteristic values and characteristic vectors,
respectively.

\texht{\vskip 4pc}{}

\xtc{
Let's first create a matrix with integer entries.
}{
\spadpaste{m1 := matrix [[1,2,1],[2,1,-2],[1,-2,4]] \bound{m1}}
}
\xtc{
To get a list of the {\it rational} eigenvalues,
use the operation \axiomFun{eigenvalues}.
}{
\spadpaste{leig := eigenvalues(m1) \free{m1} \bound{leig}}
}
\xtc{
Given an explicit eigenvalue, \axiomFun{eigenvector} computes the
eigenvectors corresponding to it.
}{
\spadpaste{eigenvector(first(leig),m1) \free{m1 leig}}
}

The operation \axiomFun{eigenvectors} returns a list of pairs of values
and vectors. When an eigenvalue is rational, Axiom gives you
the value explicitly; otherwise, its minimal polynomial is given,
(the polynomial of lowest degree with the eigenvalues as roots),
together with a parametric representation of the eigenvector using the
eigenvalue.
This means that if you ask Axiom to \axiomFun{solve}
the minimal polynomial, then you can substitute these roots
into the parametric form of the corresponding eigenvectors.

\xtc{
You must be aware that unless an exact eigenvalue has been computed,
the eigenvector may be badly in error.
}{
\spadpaste{eigenvectors(m1) \free{m1}}
```

```
}
\xtc{
Another possibility is to use the operation
\axiomFun{radicalEigenvectors}
tries to compute explicitly the eigenvectors
in terms of radicals.
}{
\spadpaste{radicalEigenvectors(m1) \free{m1}}
}

Alternatively, Axiom can compute real or complex approximations to the
eigenvectors and eigenvalues using the operations
\axiomFun{realEigenvectors} or \axiomFun{complexEigenvectors}.  They
each take an additional argument \texht{$\epsilon$}{\axiom{epsilon}}
to specify the ``precision'' required.  In the real case, this means
that each approximation will be within \texht{$\pm\epsilon$}{plus or
minus \axiom{epsilon}} of the actual result.  In the complex case,
this means that each approximation will be within
\texht{$\pm\epsilon$}{plus or minus \axiom{epsilon}} of the actual
result in each of the real and imaginary parts.

\xtc{
The precision can be specified as a \axiomType{Float} if the results
are desired in floating-point notation, or as \axiomType{Fraction
Integer} if the results are to be expressed using rational (or complex
rational) numbers.
}{
\spadpaste{realEigenvectors(m1,1/1000) \free{m1}}
}
\xtc{
If an \axiom{n} by \axiom{n} matrix has \axiom{n} distinct eigenvalues
(and therefore \axiom{n} eigenvectors) the operation
\axiomFun{eigenMatrix} gives you a matrix of the eigenvectors.
}{
\spadpaste{eigenMatrix(m1) \free{m1}}
}
\xtc{
}{
\spadpaste{m2 := matrix [[-5,-2],[18,7]] \bound{m2}}
}
\xtc{
}{
\spadpaste{eigenMatrix(m2) \free{m2}}
}
%
%
\xtc{
If a symmetric matrix
has a basis of orthonormal eigenvectors, then
\axiomFun{orthonormalBasis} computes a list of these vectors.
```

```
}{
\spadpaste{m3 := matrix [[1,2],[2,1]] \bound{m3}}
}
\xtc{
}{
\spadpaste{orthonormalBasis(m3) \free{m3}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugProblemEigenPagePatch1}
\begin{paste}{ugProblemEigenPageFull1}{ugProblemEigenPageEmpty1}
\pastebutton{ugProblemEigenPageFull1}{\hidepaste}
\tab{5}\spadcommand{m1 := matrix [[1,2,1],[2,1,-2],[1,-2,4]]\bound{m1 }}
\indentrel{3}\begin{verbatim}
        1   2    1

   (1)  2   1   - 2

        1  - 2   4
                                   Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemEigenPageEmpty1}
\begin{paste}{ugProblemEigenPageEmpty1}{ugProblemEigenPagePatch1}
\pastebutton{ugProblemEigenPageEmpty1}{\showpaste}
\tab{5}\spadcommand{m1 := matrix [[1,2,1],[2,1,-2],[1,-2,4]]\bound{m1 }}
\end{paste}\end{patch}

\begin{patch}{ugProblemEigenPagePatch2}
\begin{paste}{ugProblemEigenPageFull2}{ugProblemEigenPageEmpty2}
\pastebutton{ugProblemEigenPageFull2}{\hidepaste}
\tab{5}\spadcommand{leig := eigenvalues(m1)\free{m1 }\bound{leig }}
\indentrel{3}\begin{verbatim}
                   2
   (2)  [5,%DA | %DA  - %DA - 5]
Type: List Union(Fraction Polynomial Integer,SuchThat(Symbol,Polynomial Integer))
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemEigenPageEmpty2}
\begin{paste}{ugProblemEigenPageEmpty2}{ugProblemEigenPagePatch2}
\pastebutton{ugProblemEigenPageEmpty2}{\showpaste}
\tab{5}\spadcommand{leig := eigenvalues(m1)\free{m1 }\bound{leig }}
\end{paste}\end{patch}

\begin{patch}{ugProblemEigenPagePatch3}
```

```
\begin{paste}{ugProblemEigenPageFull3}{ugProblemEigenPageEmpty3}
\pastebutton{ugProblemEigenPageFull3}{\hidepaste}
\tab{5}\spadcommand{eigenvector(first(leig),m1)\free{m1 leig }}
\indentrel{3}\begin{verbatim}
          0

           1
   (3)  [- ]
           2

          1
 Type: List Matrix Fraction Polynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemEigenPageEmpty3}
\begin{paste}{ugProblemEigenPageEmpty3}{ugProblemEigenPagePatch3}
\pastebutton{ugProblemEigenPageEmpty3}{\showpaste}
\tab{5}\spadcommand{eigenvector(first(leig),m1)\free{m1 leig }}
\end{paste}\end{patch}

\begin{patch}{ugProblemEigenPagePatch4}
\begin{paste}{ugProblemEigenPageFull4}{ugProblemEigenPageEmpty4}
\pastebutton{ugProblemEigenPageFull4}{\hidepaste}
\tab{5}\spadcommand{eigenvectors(m1)\free{m1 }}
\indentrel{3}\begin{verbatim}
   (4)
                                    0

                                     1
   [[eigval= 5,eigmult= 1,eigvec= [- ]],
                                     2

                                    1

                        2
     [eigval= (%DB | %DB  - %DB - 5), eigmult= 1,
               %DB

      eigvec= [ 2 ]]

               1
     ]
Type: List Record(eigval: Union(Fraction Polynomial Integer,SuchThat(Symbol,Polynomial Integer))
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemEigenPageEmpty4}
\begin{paste}{ugProblemEigenPageEmpty4}{ugProblemEigenPagePatch4}
\pastebutton{ugProblemEigenPageEmpty4}{\showpaste}
```

```
\tab{5}\spadcommand{eigenvectors(m1)\free{m1 }}
\end{paste}\end{patch}

\begin{patch}{ugProblemEigenPagePatch5}
\begin{paste}{ugProblemEigenPageFull5}{ugProblemEigenPageEmpty5}
\pastebutton{ugProblemEigenPageFull5}{\hidepaste}
\tab{5}\spadcommand{radicalEigenvectors(m1)\free{m1 }}
\indentrel{3}\begin{verbatim}
   (5)

                                    \21  + 1
            \21  + 1
   [[radval= ,radmult= 1,radvect= [    2    ]],
                 2
                                              2

                                              1



             - \21  + 1
     [radval= , radmult= 1,
                   2

               - \21  + 1

      radvect= [     2     ]]

                     2

                     1
     ,
                                   0

                                    1
    [radval= 5,radmult= 1,radvect= [- ]]]
                                    2

                                   1
Type: List Record(radval: Expression Integer,radmult: Integer,radvect: List Matrix Expression Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemEigenPageEmpty5}
\begin{paste}{ugProblemEigenPageEmpty5}{ugProblemEigenPagePatch5}
\pastebutton{ugProblemEigenPageEmpty5}{\showpaste}
\tab{5}\spadcommand{radicalEigenvectors(m1)\free{m1 }}
\end{paste}\end{patch}

\begin{patch}{ugProblemEigenPagePatch6}
\begin{paste}{ugProblemEigenPageFull6}{ugProblemEigenPageEmpty6}
```

```
\pastebutton{ugProblemEigenPageFull6}{\hidepaste}
\tab{5}\spadcommand{realEigenvectors(m1,1/1000)\free{m1 }}
\indentrel{3}\begin{verbatim}
   (6)
                                     0

                                      1
   [[outval= 5,outmult= 1,outvect= [- ]],
                                      2

                                     1
                                       5717

             5717                      2048
    [outval= ,outmult= 1,outvect= [    ]],
             2048                       2

                                        1
                                           3669
                                         -
               3669                        2048
    [outval= - ,outmult= 1,outvect= [      ]]]
               2048                        2

                                           1
Type: List Record(outval: Fraction Integer,outmult: Integer,outvect: List Matrix Fraction Intege
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemEigenPageEmpty6}
\begin{paste}{ugProblemEigenPageEmpty6}{ugProblemEigenPagePatch6}
\pastebutton{ugProblemEigenPageEmpty6}{\showpaste}
\tab{5}\spadcommand{realEigenvectors(m1,1/1000)\free{m1 }}
\end{paste}\end{patch}

\begin{patch}{ugProblemEigenPagePatch7}
\begin{paste}{ugProblemEigenPageFull7}{ugProblemEigenPageEmpty7}
\pastebutton{ugProblemEigenPageFull7}{\hidepaste}
\tab{5}\spadcommand{eigenMatrix(m1)\free{m1 }}
\indentrel{3}\begin{verbatim}

        \21  + 1  - \21  + 1
             0
            2           2
   (7)
                                1
            2           2      -
                                2

            1           1       1
```

```
            Type: Union(Matrix Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemEigenPageEmpty7}
\begin{paste}{ugProblemEigenPageEmpty7}{ugProblemEigenPagePatch7}
\pastebutton{ugProblemEigenPageEmpty7}{\showpaste}
\tab{5}\spadcommand{eigenMatrix(m1)\free{m1 }}
\end{paste}\end{patch}

\begin{patch}{ugProblemEigenPagePatch8}
\begin{paste}{ugProblemEigenPageFull8}{ugProblemEigenPageEmpty8}
\pastebutton{ugProblemEigenPageFull8}{\hidepaste}
\tab{5}\spadcommand{m2 := matrix [[-5,-2],[18,7]]\bound{m2 }}
\indentrel{3}\begin{verbatim}
        - 5  - 2
   (8)
        18    7
                                   Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemEigenPageEmpty8}
\begin{paste}{ugProblemEigenPageEmpty8}{ugProblemEigenPagePatch8}
\pastebutton{ugProblemEigenPageEmpty8}{\showpaste}
\tab{5}\spadcommand{m2 := matrix [[-5,-2],[18,7]]\bound{m2 }}
\end{paste}\end{patch}

\begin{patch}{ugProblemEigenPagePatch9}
\begin{paste}{ugProblemEigenPageFull9}{ugProblemEigenPageEmpty9}
\pastebutton{ugProblemEigenPageFull9}{\hidepaste}
\tab{5}\spadcommand{eigenMatrix(m2)\free{m2 }}
\indentrel{3}\begin{verbatim}
   (9)  "failed"
                              Type: Union("failed",...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemEigenPageEmpty9}
\begin{paste}{ugProblemEigenPageEmpty9}{ugProblemEigenPagePatch9}
\pastebutton{ugProblemEigenPageEmpty9}{\showpaste}
\tab{5}\spadcommand{eigenMatrix(m2)\free{m2 }}
\end{paste}\end{patch}

\begin{patch}{ugProblemEigenPagePatch10}
\begin{paste}{ugProblemEigenPageFull10}{ugProblemEigenPageEmpty10}
\pastebutton{ugProblemEigenPageFull10}{\hidepaste}
\tab{5}\spadcommand{m3 := matrix [[1,2],[2,1]]\bound{m3 }}
\indentrel{3}\begin{verbatim}
         1  2
```

```
   (10)
         2  1
                                      Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemEigenPageEmpty10}
\begin{paste}{ugProblemEigenPageEmpty10}{ugProblemEigenPagePatch10}
\pastebutton{ugProblemEigenPageEmpty10}{\showpaste}
\tab{5}\spadcommand{m3 := matrix [[1,2],[2,1]]\bound{m3 }}
\end{paste}\end{patch}

\begin{patch}{ugProblemEigenPagePatch11}
\begin{paste}{ugProblemEigenPageFull11}{ugProblemEigenPageEmpty11}
\pastebutton{ugProblemEigenPageFull11}{\hidepaste}
\tab{5}\spadcommand{orthonormalBasis(m3)\free{m3 }}
\indentrel{3}\begin{verbatim}
              1    1
          -


            \2  \2
   (11)  [      ,    ]
             1     1


           \2   \2
                      Type: List Matrix Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemEigenPageEmpty11}
\begin{paste}{ugProblemEigenPageEmpty11}{ugProblemEigenPagePatch11}
\pastebutton{ugProblemEigenPageEmpty11}{\showpaste}
\tab{5}\spadcommand{orthonormalBasis(m3)\free{m3 }}
\end{paste}\end{patch}
```

—————-

Solution of Linear and Polynomial Equations

⇒ “notitle” (ugProblemDEQPage) 12 on page 2211
⇒ “notitle” (ugxProblemLinSysPage) 12 on page 2131
⇒ “notitle” (ugxProblemOnePolPage) 12 on page 2135
⇒ “notitle” (ugxProblemPolSysPage) 12 on page 2140

— ug08.ht —

```
\begin{page}{ugProblemLinPolEqnPage}
```

```
{8.5. Solution of Linear and Polynomial Equations}
\beginscroll
%
In this section we discuss the Axiom facilities for solving
systems of linear equations, finding the roots of polynomials and
solving systems of polynomial equations.
For a discussion of the solution of differential equations, see
\downlink{``Solution of Differential Equations''}{ugProblemDEQPage} in
Section 8.10\ignore{ugProblemDEQ}.

\beginmenu
    \menudownlink{{8.5.1. Solution of Systems of Linear Equations}}
{ugxProblemLinSysPage}
    \menudownlink{{8.5.2. Solution of a Single Polynomial Equation}}
{ugxProblemOnePolPage}
    \menudownlink{{8.5.3. Solution of Systems of Polynomial Equations}}
{ugxProblemPolSysPage}
\endmenu
\endscroll
\autobuttons
\end{page}
```

———-

Solution of Systems of Linear Equations

— ug08.ht —

```
\begin{page}{ugxProblemLinSysPage}
{8.5.1. Solution of Systems of Linear Equations}
\beginscroll

You can use the operation \axiomFun{solve} to solve systems of linear
equations.

The operation \axiomFun{solve} takes two arguments, the list of
equations and the list of the unknowns to be solved for.  A system of
linear equations need not have a unique solution.

\xtc{
To solve the linear system:
\centerline{{\axiom{  x +   y +   z = 8} }}
\centerline{{\axiom{3*x - 2*y +   z = 0} }}
\centerline{{\axiom{  x + 2*y + 2*z = 17}}}
evaluate this expression.
}{
\spadpaste{solve([x+y+z=8,3*x-2*y+z=0,x+2*y+2*z=17],[x,y,z])}
```

```
}

Parameters are given as new variables starting with a percent sign and
\spadSyntax{\%} and
the variables are expressed in terms of the parameters.
If the system has no solutions then the empty list is returned.

\xtc{
When you solve the linear system
\centerline{{\axiom{  x + 2*y + 3*z = 2} }}
\centerline{{\axiom{2*x + 3*y + 4*z = 2} }}
\centerline{{\axiom{3*x + 4*y + 5*z = 2}}}
with this expression
you get a solution involving a parameter.
}{
\spadpaste{solve([x+2*y+3*z=2,2*x+3*y+4*z=2,3*x+4*y+5*z=2],[x,y,z])}
}

The system can also be presented as a matrix and a vector.
The matrix contains the coefficients of the linear equations and
the vector contains the numbers appearing on the right-hand sides
of the equations.
You may input the matrix as a list of rows and the vector as a
list of its elements.

\xtc{
To solve the system:
\centerline{{\axiom{  x +   y +   z = 8} }}
\centerline{{\axiom{3*x - 2*y +   z = 0} }}
\centerline{{\axiom{  x + 2*y + 2*z = 17}}}
in matrix form you would evaluate this expression.
}{
\spadpaste{solve([[1,1,1],[3,-2,1],[1,2,2]],[8,0,17])}
}

The solutions are presented as a \pspadtype{Record} with two
components: the component \texht{{\it particular}}{{\tt
particular}} contains a particular solution of the given system or
the item {\tt "failed"} if there are no solutions, the component
\texht{{\it basis}}{{\tt basis}} contains a list of vectors that
are a basis for the space of solutions of the corresponding
homogeneous system.
If the system of linear equations does not have a unique solution,
then the \texht{{\it basis}}{{\tt basis}} component contains
non-trivial vectors.

\xtc{
This happens when you solve the linear system
\centerline{{\axiom{  x + 2*y + 3*z = 2} }}
\centerline{{\axiom{2*x + 3*y + 4*z = 2} }}
```

```
\centerline{{\axiom{3*x + 4*y + 5*z = 2}}}
with this command.
}{
\spadpaste{solve([[1,2,3],[2,3,4],[3,4,5]],[2,2,2])}
}

All solutions of this system are obtained by adding the particular
solution with a linear combination of the \texht{{\it basis}}{{\tt
basis}} vectors.

When no solution exists then {\tt "failed"} is returned as the
\texht{{\it particular}}{{\tt particular}} component, as follows:

\xtc{
}{
\spadpaste{solve([[1,2,3],[2,3,4],[3,4,5]],[2,3,2])}
}

When you want to solve a system of homogeneous equations (that is,
a system where the numbers on the right-hand sides of the
equations are all zero) in the matrix form you can omit the second
argument and use the \axiomFun{nullSpace} operation.

\xtc{
This computes the solutions of the following system of equations:
\centerline{{\axiom{  x + 2*y + 3*z = 0} }}
\centerline{{\axiom{2*x + 3*y + 4*z = 0} }}
\centerline{{\axiom{3*x + 4*y + 5*z = 0}}}
The result is given as a list of vectors and
these vectors form a basis for the solution space.
}{
\spadpaste{nullSpace([[1,2,3],[2,3,4],[3,4,5]])}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxProblemLinSysPagePatch1}
\begin{paste}{ugxProblemLinSysPageFull1}{ugxProblemLinSysPageEmpty1}
\pastebutton{ugxProblemLinSysPageFull1}{\hidepaste}
\tab{5}\spadcommand{solve([x+y+z=8,3*x-2*y+z=0,x+2*y+2*z=17],[x,y,z])}
\indentrel{3}\begin{verbatim}
   (1)  [[x= - 1,y= 2,z= 7]]
   Type: List List Equation Fraction Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemLinSysPageEmpty1}
\begin{paste}{ugxProblemLinSysPageEmpty1}{ugxProblemLinSysPagePatch1}
```

```
\pastebutton{ugxProblemLinSysPageEmpty1}{\showpaste}
\tab{5}\spadcommand{solve([x+y+z=8,3*x-2*y+z=0,x+2*y+2*z=17],[x,y,z])}
\end{paste}\end{patch}

\begin{patch}{ugxProblemLinSysPagePatch2}
\begin{paste}{ugxProblemLinSysPageFull2}{ugxProblemLinSysPageEmpty2}
\pastebutton{ugxProblemLinSysPageFull2}{\hidepaste}
\tab{5}\spadcommand{solve([x+2*y+3*z=2,2*x+3*y+4*z=2,3*x+4*y+5*z=2],[x,y,z])}
\indentrel{3}\begin{verbatim}
   (2)  [[x= %Y - 2,y= - 2%Y + 2,z= %Y]]
   Type: List List Equation Fraction Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemLinSysPageEmpty2}
\begin{paste}{ugxProblemLinSysPageEmpty2}{ugxProblemLinSysPagePatch2}
\pastebutton{ugxProblemLinSysPageEmpty2}{\showpaste}
\tab{5}\spadcommand{solve([x+2*y+3*z=2,2*x+3*y+4*z=2,3*x+4*y+5*z=2],[x,y,z])}
\end{paste}\end{patch}

\begin{patch}{ugxProblemLinSysPagePatch3}
\begin{paste}{ugxProblemLinSysPageFull3}{ugxProblemLinSysPageEmpty3}
\pastebutton{ugxProblemLinSysPageFull3}{\hidepaste}
\tab{5}\spadcommand{solve([[1,1,1],[3,-2,1],[1,2,2]],[8,0,17])}
\indentrel{3}\begin{verbatim}
   (3)  [particular= [- 1,2,7],basis= [[0,0,0]]]
Type: Record(particular: Union(Vector Fraction Integer,"failed"),basis: List Vector Fraction Int
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemLinSysPageEmpty3}
\begin{paste}{ugxProblemLinSysPageEmpty3}{ugxProblemLinSysPagePatch3}
\pastebutton{ugxProblemLinSysPageEmpty3}{\showpaste}
\tab{5}\spadcommand{solve([[1,1,1],[3,-2,1],[1,2,2]],[8,0,17])}
\end{paste}\end{patch}

\begin{patch}{ugxProblemLinSysPagePatch4}
\begin{paste}{ugxProblemLinSysPageFull4}{ugxProblemLinSysPageEmpty4}
\pastebutton{ugxProblemLinSysPageFull4}{\hidepaste}
\tab{5}\spadcommand{solve([[1,2,3],[2,3,4],[3,4,5]],[2,2,2])}
\indentrel{3}\begin{verbatim}
   (4)  [particular= [- 2,2,0],basis= [[1,- 2,1]]]
Type: Record(particular: Union(Vector Fraction Integer,"failed"),basis: List Vector Fraction Int
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemLinSysPageEmpty4}
\begin{paste}{ugxProblemLinSysPageEmpty4}{ugxProblemLinSysPagePatch4}
\pastebutton{ugxProblemLinSysPageEmpty4}{\showpaste}
\tab{5}\spadcommand{solve([[1,2,3],[2,3,4],[3,4,5]],[2,2,2])}
```

```
\end{paste}\end{patch}

\begin{patch}{ugxProblemLinSysPagePatch5}
\begin{paste}{ugxProblemLinSysPageFull5}{ugxProblemLinSysPageEmpty5}
\pastebutton{ugxProblemLinSysPageFull5}{\hidepaste}
\tab{5}\spadcommand{solve([[1,2,3],[2,3,4],[3,4,5]],[2,3,2])}
\indentrel{3}\begin{verbatim}
   (5)  [particular= "failed",basis= [[1,- 2,1]]]
Type: Record(particular: Union(Vector Fraction Integer,"failed"),basis: List Vector Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemLinSysPageEmpty5}
\begin{paste}{ugxProblemLinSysPageEmpty5}{ugxProblemLinSysPagePatch5}
\pastebutton{ugxProblemLinSysPageEmpty5}{\showpaste}
\tab{5}\spadcommand{solve([[1,2,3],[2,3,4],[3,4,5]],[2,3,2])}
\end{paste}\end{patch}

\begin{patch}{ugxProblemLinSysPagePatch6}
\begin{paste}{ugxProblemLinSysPageFull6}{ugxProblemLinSysPageEmpty6}
\pastebutton{ugxProblemLinSysPageFull6}{\hidepaste}
\tab{5}\spadcommand{nullSpace([[1,2,3],[2,3,4],[3,4,5]])}
\indentrel{3}\begin{verbatim}
   (6)  [[1,- 2,1]]
                              Type: List Vector Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemLinSysPageEmpty6}
\begin{paste}{ugxProblemLinSysPageEmpty6}{ugxProblemLinSysPagePatch6}
\pastebutton{ugxProblemLinSysPageEmpty6}{\showpaste}
\tab{5}\spadcommand{nullSpace([[1,2,3],[2,3,4],[3,4,5]])}
\end{paste}\end{patch}
```

———-

Solution of a Single Polynomial Equation

— **ug08.ht** —

```
\begin{page}{ugxProblemOnePolPage}
{8.5.2. Solution of a Single Polynomial Equation}
\beginscroll

Axiom can solve polynomial equations producing either approximate or
exact solutions.  Exact solutions are either members of the ground
field or can be presented symbolically as roots of irreducible
```

```
polynomials.

\xtc{
This returns the one rational root along with an irreducible
polynomial describing the other solutions.
}{
\spadpaste{solve(x**3  = 8,x)}
}
\xtc{
If you want solutions expressed in terms of radicals you would use this
instead.
}{
\spadpaste{radicalSolve(x**3  = 8,x)}
}

The \axiomFun{solve} command always returns a value but
\axiomFun{radicalSolve} returns only the solutions that it is
able to express in terms of radicals.

If the polynomial equation has rational coefficients
you can ask for approximations to its real roots by calling
solve with a second argument that specifies the ``precision''
\texht{$\epsilon$}{\axiom{epsilon}}.
This means that each approximation will be within
\texht{$\pm\epsilon$}{plus or minus \axiom{epsilon}} of the actual
result.

\xtc{
Notice that the type of second argument controls the type of the result.
}{
\spadpaste{solve(x**4 - 10*x**3 + 35*x**2 - 50*x + 25,.0001)}
}
\xtc{
If you give a floating-point precision you get a floating-point result;
if you give the precision as a rational number you get a rational result.
}{
\spadpaste{solve(x**3-2,1/1000)}
}
\xtc{
If you want approximate complex results you should use the
command \axiomFun{complexSolve} that takes the same precision argument
\texht{$\epsilon$}{\axiom{epsilon}}.
}{
\spadpaste{complexSolve(x**3-2,.0001)}
}
\xtc{
Each approximation will be within
\texht{$\pm\epsilon$}{plus or minus \axiom{epsilon}} of the actual result
in each of the real and imaginary parts.
}{
```

```
\spadpaste{complexSolve(x**2-2*\%i+1,1/100)}
}

Note that if you omit the \axiomOp{=} from the first argument
Axiom generates an equation by equating the first argument to zero.
Also, when only one variable is present in the equation, you
do not need to specify the variable to be solved for, that is,
you can omit the second argument.

\xtc{
Axiom can also solve equations involving rational functions.
Solutions where the denominator vanishes are discarded.
}{
\spadpaste{radicalSolve(1/x**3 + 1/x**2 + 1/x = 0,x)}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxProblemOnePolPagePatch1}
\begin{paste}{ugxProblemOnePolPageFull1}{ugxProblemOnePolPageEmpty1}
\pastebutton{ugxProblemOnePolPageFull1}{\hidepaste}
\tab{5}\spadcommand{solve(x**3 = 8,x)}
\indentrel{3}\begin{verbatim}
               2
   (1)  [x= 2,x  + 2x + 4= 0]
        Type: List Equation Fraction Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemOnePolPageEmpty1}
\begin{paste}{ugxProblemOnePolPageEmpty1}{ugxProblemOnePolPagePatch1}
\pastebutton{ugxProblemOnePolPageEmpty1}{\showpaste}
\tab{5}\spadcommand{solve(x**3 = 8,x)}
\end{paste}\end{patch}

\begin{patch}{ugxProblemOnePolPagePatch2}
\begin{paste}{ugxProblemOnePolPageFull2}{ugxProblemOnePolPageEmpty2}
\pastebutton{ugxProblemOnePolPageFull2}{\hidepaste}
\tab{5}\spadcommand{radicalSolve(x**3 = 8,x)}
\indentrel{3}\begin{verbatim}

   (2)  [x= - \- 3  - 1,x= \- 3  - 1,x= 2]
                 Type: List Equation Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemOnePolPageEmpty2}
\begin{paste}{ugxProblemOnePolPageEmpty2}{ugxProblemOnePolPagePatch2}
```

```
\pastebutton{ugxProblemOnePolPageEmpty2}{\showpaste}
\tab{5}\spadcommand{radicalSolve(x**3 = 8,x)}
\end{paste}\end{patch}

\begin{patch}{ugxProblemOnePolPagePatch3}
\begin{paste}{ugxProblemOnePolPageFull3}{ugxProblemOnePolPageEmpty3}
\pastebutton{ugxProblemOnePolPageFull3}{\hidepaste}
\tab{5}\spadcommand{solve(x**4 - 10*x**3 + 35*x**2 - 50*x + 25,.0001)}
\indentrel{3}\begin{verbatim}
   (3)  [x= 3.6180114746 09375,x= 1.3819885253 90625]
                   Type: List Equation Polynomial Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemOnePolPageEmpty3}
\begin{paste}{ugxProblemOnePolPageEmpty3}{ugxProblemOnePolPagePatch3}
\pastebutton{ugxProblemOnePolPageEmpty3}{\showpaste}
\tab{5}\spadcommand{solve(x**4 - 10*x**3 + 35*x**2 - 50*x + 25,.0001)}
\end{paste}\end{patch}

\begin{patch}{ugxProblemOnePolPagePatch4}
\begin{paste}{ugxProblemOnePolPageFull4}{ugxProblemOnePolPageEmpty4}
\pastebutton{ugxProblemOnePolPageFull4}{\hidepaste}
\tab{5}\spadcommand{solve(x**3-2,1/1000)}
\indentrel{3}\begin{verbatim}
            2581
   (4)  [x= ]
            2048
        Type: List Equation Polynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemOnePolPageEmpty4}
\begin{paste}{ugxProblemOnePolPageEmpty4}{ugxProblemOnePolPagePatch4}
\pastebutton{ugxProblemOnePolPageEmpty4}{\showpaste}
\tab{5}\spadcommand{solve(x**3-2,1/1000)}
\end{paste}\end{patch}

\begin{patch}{ugxProblemOnePolPagePatch5}
\begin{paste}{ugxProblemOnePolPageFull5}{ugxProblemOnePolPageEmpty5}
\pastebutton{ugxProblemOnePolPageFull5}{\hidepaste}
\tab{5}\spadcommand{complexSolve(x**3-2,.0001)}
\indentrel{3}\begin{verbatim}
   (5)
   [x= 1.2599182128 90625,
    x= - 0.6298943279 5395613131 - 1.0910949707 03125 %i,
    x= - 0.6298943279 5395613131 + 1.0910949707 03125 %i]
           Type: List Equation Polynomial Complex Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugxProblemOnePolPageEmpty5}
\begin{paste}{ugxProblemOnePolPageEmpty5}{ugxProblemOnePolPagePatch5}
\pastebutton{ugxProblemOnePolPageEmpty5}{\showpaste}
\tab{5}\spadcommand{complexSolve(x**3-2,.0001)}
\end{paste}\end{patch}

\begin{patch}{ugxProblemOnePolPagePatch6}
\begin{paste}{ugxProblemOnePolPageFull6}{ugxProblemOnePolPageEmpty6}
\pastebutton{ugxProblemOnePolPageFull6}{\hidepaste}
\tab{5}\spadcommand{complexSolve(x**2-2*\%i+1,1/100)}
\indentrel{3}\begin{verbatim}
              13028925   325       13028925   325
   (6)  [x= -  -  %i,x=  +  %i]
              16777216   256       16777216   256
Type: List Equation Polynomial Complex Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemOnePolPageEmpty6}
\begin{paste}{ugxProblemOnePolPageEmpty6}{ugxProblemOnePolPagePatch6}
\pastebutton{ugxProblemOnePolPageEmpty6}{\showpaste}
\tab{5}\spadcommand{complexSolve(x**2-2*\%i+1,1/100)}
\end{paste}\end{patch}

\begin{patch}{ugxProblemOnePolPagePatch7}
\begin{paste}{ugxProblemOnePolPageFull7}{ugxProblemOnePolPageEmpty7}
\pastebutton{ugxProblemOnePolPageFull7}{\hidepaste}
\tab{5}\spadcommand{radicalSolve(1/x**3 + 1/x**2 + 1/x = 0,x)}
\indentrel{3}\begin{verbatim}

            - \- 3  - 1    \- 3  - 1
   (7)  [x= ,x= ]
                  2              2
                 Type: List Equation Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemOnePolPageEmpty7}
\begin{paste}{ugxProblemOnePolPageEmpty7}{ugxProblemOnePolPagePatch7}
\pastebutton{ugxProblemOnePolPageEmpty7}{\showpaste}
\tab{5}\spadcommand{radicalSolve(1/x**3 + 1/x**2 + 1/x = 0,x)}
\end{paste}\end{patch}
```

Solution of Systems of Polynomial Equations

⇒ "notitle" (ugxProblemOnePolPage) 12 on page 2135
— ug08.ht —

```
\begin{page}{ugxProblemPolSysPage}
{8.5.3. Solution of Systems of Polynomial Equations}
\beginscroll

Given a system of equations of rational functions with exact coefficients:
\centerline{{\axiom{p1(x1,...,xn)} }}
\centerline{{\axiom{.}             }}
\centerline{{\axiom{.}             }}
\centerline{{\axiom{.}             }}
\centerline{{\axiom{pm(x1,...,xn)}}}
Axiom can find
numeric or symbolic solutions.
The system is first split into irreducible components, then for
each component, a triangular system of equations is found that reduces
the problem to sequential solution of univariate polynomials resulting
from substitution of partial solutions from the previous stage.
\centerline{{\axiom{q1(x1,...,xn)} }}
\centerline{{\axiom{.} }}
\centerline{{\axiom{.} }}
\centerline{{\axiom{.} }}
\centerline{{\axiom{qm(xn)}}}

Symbolic solutions can be presented using ``implicit'' algebraic numbers
defined as roots of irreducible polynomials or in terms of radicals.
Axiom can also find approximations to the real or complex roots
of a system of polynomial equations to any user-specified accuracy.

The operation \axiomFun{solve} for systems is used in a way similar
to \axiomFun{solve} for single equations.
Instead of a polynomial equation, one has to give a list of
equations and instead of a single variable to solve for, a list of
variables.
For solutions of single equations see
\downlink{``Solution of a Single Polynomial Equation''}
{ugxProblemOnePolPage} in Section 8.5.2\ignore{ugxProblemOnePol}.

%
\xtc{
Use the operation \axiomFun{solve} if you want implicitly presented
solutions.
}{
\spadpaste{solve([3*x**3 + y + 1,y**2 -4],[x,y])}
}
\xtc{
```

```
}{
\spadpaste{solve([x = y**2-19,y = z**2+x+3,z = 3*x],[x,y,z])}
}
\xtc{
Use \axiomFun{radicalSolve} if you want your solutions expressed
in terms of radicals.
}{
\spadpaste{radicalSolve([3*x**3 + y + 1,y**2 -4],[x,y])}
}

To get numeric solutions you only need to give the list of
equations and the precision desired.
The list of variables would be redundant information since there
can be no parameters for the numerical solver.

\xtc{
If the precision is expressed as a floating-point number you get
results expressed as floats.
}{
\spadpaste{solve([x**2*y - 1,x*y**2 - 2],.01)}
}
\xtc{
To get complex numeric solutions, use the operation \axiomFun{complexSolve},
which takes the same arguments as in the real case.
}{
\spadpaste{complexSolve([x**2*y - 1,x*y**2 - 2],1/1000)}
}
\xtc{
It is also possible to solve systems of equations in rational functions
over the rational numbers.
Note that \axiom{[x = 0.0, a = 0.0]} is not returned as a solution since
the denominator vanishes there.
}{
\spadpaste{solve([x**2/a = a,a = a*x],.001)}
}
\xtc{
When solving equations with
denominators, all solutions where the denominator vanishes are
discarded.
}{
\spadpaste{radicalSolve([x**2/a + a + y**3 - 1,a*y + a + 1],[x,y])}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxProblemPolSysPagePatch1}
\begin{paste}{ugxProblemPolSysPageFull1}{ugxProblemPolSysPageEmpty1}
\pastebutton{ugxProblemPolSysPageFull1}{\hidepaste}
```

```
\tab{5}\spadcommand{solve([3*x**3 + y + 1,y**2 -4],[x,y])}
\indentrel{3}\begin{verbatim}
   (1)
                    2
   [[x= - 1,y= 2], [x  - x + 1= 0,y= 2],
       3
    [3x  - 1= 0,y= - 2]]
   Type: List List Equation Fraction Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemPolSysPageEmpty1}
\begin{paste}{ugxProblemPolSysPageEmpty1}{ugxProblemPolSysPagePatch1}
\pastebutton{ugxProblemPolSysPageEmpty1}{\showpaste}
\tab{5}\spadcommand{solve([3*x**3 + y + 1,y**2 -4],[x,y])}
\end{paste}\end{patch}

\begin{patch}{ugxProblemPolSysPagePatch2}
\begin{paste}{ugxProblemPolSysPageFull2}{ugxProblemPolSysPageEmpty2}
\pastebutton{ugxProblemPolSysPageFull2}{\hidepaste}
\tab{5}\spadcommand{solve([x = y**2-19,y = z**2+x+3,z = 3*x],[x,y,z])}
\indentrel{3}\begin{verbatim}
   (2)
              2
        z    3z  + z + 9   4     3      2
   [[x= ,y= ,9z  + 6z  + 55z  + 15z - 90= 0]]
        3         3
   Type: List List Equation Fraction Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemPolSysPageEmpty2}
\begin{paste}{ugxProblemPolSysPageEmpty2}{ugxProblemPolSysPagePatch2}
\pastebutton{ugxProblemPolSysPageEmpty2}{\showpaste}
\tab{5}\spadcommand{solve([x = y**2-19,y = z**2+x+3,z = 3*x],[x,y,z])}
\end{paste}\end{patch}

\begin{patch}{ugxProblemPolSysPagePatch3}
\begin{paste}{ugxProblemPolSysPageFull3}{ugxProblemPolSysPageEmpty3}
\pastebutton{ugxProblemPolSysPageFull3}{\hidepaste}
\tab{5}\spadcommand{radicalSolve([3*x**3 + y + 1,y**2 -4],[x,y])}
\indentrel{3}\begin{verbatim}
   (3)

        \- 3  + 1            - \- 3  + 1
   [[x= ,y= 2], [x= ,y= 2],
           2                       2

        - \- 1 \3  - 1
    [x= ,y= - 2],
```

```
             3
            2\3

      \- 1 \3  - 1                1
    [x= ,y= - 2], [x= ,y= - 2],
            3                  3
           2\3                  \3
    [x= - 1,y= 2]]
           Type: List List Equation Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemPolSysPageEmpty3}
\begin{paste}{ugxProblemPolSysPageEmpty3}{ugxProblemPolSysPagePatch3}
\pastebutton{ugxProblemPolSysPageEmpty3}{\showpaste}
\tab{5}\spadcommand{radicalSolve([3*x**3 + y + 1,y**2 -4],[x,y])}
\end{paste}\end{patch}

\begin{patch}{ugxProblemPolSysPagePatch4}
\begin{paste}{ugxProblemPolSysPageFull4}{ugxProblemPolSysPageEmpty4}
\pastebutton{ugxProblemPolSysPageFull4}{\hidepaste}
\tab{5}\spadcommand{solve([x**2*y - 1,x*y**2 - 2],.01)}
\indentrel{3}\begin{verbatim}
   (4)  [[y= 1.5859375,x= 0.79296875]]
             Type: List List Equation Polynomial Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemPolSysPageEmpty4}
\begin{paste}{ugxProblemPolSysPageEmpty4}{ugxProblemPolSysPagePatch4}
\pastebutton{ugxProblemPolSysPageEmpty4}{\showpaste}
\tab{5}\spadcommand{solve([x**2*y - 1,x*y**2 - 2],.01)}
\end{paste}\end{patch}

\begin{patch}{ugxProblemPolSysPagePatch5}
\begin{paste}{ugxProblemPolSysPageFull5}{ugxProblemPolSysPageEmpty5}
\pastebutton{ugxProblemPolSysPageFull5}{\hidepaste}
\tab{5}\spadcommand{complexSolve([x**2*y - 1,x*y**2 - 2],1/1000)}
\indentrel{3}\begin{verbatim}
   (5)
        1625    1625
   [[y= ,x= ],
        1024    2048

           435445573689   1407
     [y= -  -  %i,
           549755813888   1024
            435445573689   1407
      x= -  -  %i]
           1099511627776   2048
```

```
     ,

           435445573689   1407
     [y= -  +  %i,
           549755813888   1024
            435445573689   1407
      x= -  +  %i]
           1099511627776   2048
     ]
Type: List List Equation Polynomial Complex Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemPolSysPageEmpty5}
\begin{paste}{ugxProblemPolSysPageEmpty5}{ugxProblemPolSysPagePatch5}
\pastebutton{ugxProblemPolSysPageEmpty5}{\showpaste}
\tab{5}\spadcommand{complexSolve([x**2*y - 1,x*y**2 - 2],1/1000)}
\end{paste}\end{patch}

\begin{patch}{ugxProblemPolSysPagePatch6}
\begin{paste}{ugxProblemPolSysPageFull6}{ugxProblemPolSysPageEmpty6}
\pastebutton{ugxProblemPolSysPageFull6}{\hidepaste}
\tab{5}\spadcommand{solve([x**2/a = a,a = a*x],.001)}
\indentrel{3}\begin{verbatim}
   (6)  [[x= 1.0,a= - 1.0],[x= 1.0,a= 1.0]]
              Type: List List Equation Polynomial Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemPolSysPageEmpty6}
\begin{paste}{ugxProblemPolSysPageEmpty6}{ugxProblemPolSysPagePatch6}
\pastebutton{ugxProblemPolSysPageEmpty6}{\showpaste}
\tab{5}\spadcommand{solve([x**2/a = a,a = a*x],.001)}
\end{paste}\end{patch}

\begin{patch}{ugxProblemPolSysPagePatch7}
\begin{paste}{ugxProblemPolSysPageFull7}{ugxProblemPolSysPageEmpty7}
\pastebutton{ugxProblemPolSysPageFull7}{\hidepaste}
\tab{5}\spadcommand{radicalSolve([x**2/a + a + y**3 - 1,a*y + a + 1],[x,y])}
\indentrel{3}\begin{verbatim}
   (7)

              4     3     2
           - a  + 2a  + 3a  + 3a + 1     - a - 1
   [[x= -   ,y= ],
                         2                   a
          \             a

            4     3     2
         - a  + 2a  + 3a  + 3a + 1     - a - 1
```

```
    [x=   ,y= ]]
                     2                   a
       \           a
           Type: List List Equation Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemPolSysPageEmpty7}
\begin{paste}{ugxProblemPolSysPageEmpty7}{ugxProblemPolSysPagePatch7}
\pastebutton{ugxProblemPolSysPageEmpty7}{\showpaste}
\tab{5}\spadcommand{radicalSolve([x**2/a + a + y**3 - 1,a*y + a + 1],[x,y])}
\end{paste}\end{patch}
```

———-

Limits

— ug08.ht —

```
\begin{page}{ugProblemLimitsPage}{8.6. Limits}
\beginscroll
%
To compute a limit, you must specify a functional expression,
a variable, and a limiting value for that variable.
If you do not specify a direction, Axiom attempts to
compute a two-sided limit.

\xtc{
Issue this to compute the limit
\texht{$$\lim_{x \rightarrow 1}{{\displaystyle x^2 - 3x +
2}\over{\displaystyle x^2 - 1}}.$$}{
of \axiom{(x**2 - 3*x + 2)/(x**2 - 1)} as \axiom{x} approaches \axiom{1}.}
}{
\spadpaste{limit((x**2 - 3*x + 2)/(x**2 - 1),x = 1)}
}

Sometimes the limit when approached from the left is different from
the limit from the right and, in this case, you may wish to ask for a
one-sided limit.  Also, if you have a function that is only defined on
one side of a particular value, you can compute a one-sided limit.

\xtc{
The function \axiom{log(x)} is only defined to the right of zero,
that is, for \axiom{x > 0}.
Thus, when computing limits of functions involving \axiom{log(x)},
you probably want a ``right-hand'' limit.
}{
```

```
\spadpaste{limit(x * log(x),x = 0,"right")}
}
\xtc{
When you do not specify \axiom{"right"} or \axiom{"left"} as the
optional fourth argument, \axiomFun{limit} tries to compute a
two-sided limit.
Here the limit from the left does not exist, as Axiom
indicates when you try to take a two-sided limit.
}{
\spadpaste{limit(x * log(x),x = 0)}
}

A function can be defined on both sides of a particular value, but
tend to different limits as its variable approaches that value from the
left and from the right.
We can construct an example of this as follows:
Since
\texht{$\sqrt{y^2}$}{\axiom{sqrt(y**2)}}
is simply the absolute value of \axiom{y},
the function
\texht{$\sqrt{y^2} / y$}{\axiom{sqrt(y**2) / y}}
is simply the sign (\axiom{+1} or \axiom{-1}) of the nonzero
real number \axiom{y}.
Therefore,
\texht{$\sqrt{y^2} / y = -1$}{\axiom{sqrt(y**2) / y = -1}}
for \axiom{y < 0} and
\texht{$\sqrt{y^2} / y = +1$}{\axiom{sqrt(y**2) / y = +1}}
for \axiom{y > 0}.
\xtc{
This is what happens when we take the limit at \axiom{y = 0}.
The answer returned by Axiom gives both a
``left-hand'' and a ``right-hand'' limit.
}{
\spadpaste{limit(sqrt(y**2)/y,y = 0)}
}
\xtc{
Here is another example, this time using a more complicated function.
}{
\spadpaste{limit(sqrt(1 - cos(t))/t,t = 0)}
}

You can compute limits at infinity by passing either
\texht{$+\infty$ or $-\infty$}{``plus infinity'' or ``minus infinity''}
as the third argument of \axiomFun{limit}.
\xtc{
To do this, use the constants
\axiom{\%plusInfinity} and \axiom{\%minusInfinity}.
}{
\spadpaste{limit(sqrt(3*x**2 + 1)/(5*x),x = \%plusInfinity)}
}
```

```
\xtc{
}{
\spadpaste{limit(sqrt(3*x**2 + 1)/(5*x),x = \%minusInfinity)}
}

\xtc{
You can take limits of functions with parameters.
As you can see, the limit is expressed in terms of the parameters.
}{
\spadpaste{limit(sinh(a*x)/tan(b*x),x = 0)}
}

When you use \axiomFun{limit}, you are taking the limit of a real
function of a real variable.
\xtc{
When you compute this, Axiom returns \axiom{0} because, as a function
of a real variable, \axiom{sin(1/z)} is always between \axiom{-1} and
\axiom{1}, so \axiom{z * sin(1/z)} tends to \axiom{0} as \axiom{z}
tends to \axiom{0}.
}{
\spadpaste{limit(z * sin(1/z),z = 0)}
}
However, as a function of a {\it complex} variable, \axiom{sin(1/z)}
is badly behaved near \axiom{0} (one says that \axiom{sin(1/z)} has an
{\it essential singularity} at \axiom{z = 0}).
\xtc{
When viewed as a function of a complex variable, \axiom{z * sin(1/z)}
does not approach any limit as \axiom{z} tends to \axiom{0} in the
complex plane.  Axiom indicates this when we call
\axiomFun{complexLimit}.
}{
\spadpaste{complexLimit(z * sin(1/z),z = 0)}
}

You can also take complex limits at infinity, that is, limits of a
function of \axiom{z} as \axiom{z} approaches infinity on the Riemann
sphere.  Use the symbol \axiom{\%infinity} to denote ``complex
infinity.''
\xtc{
As above, to compute complex limits rather than real limits, use
\axiomFun{complexLimit}.
}{
\spadpaste{complexLimit((2 + z)/(1 - z),z = \%infinity)}
}
\xtc{
In many cases, a limit of a real function of a real variable
exists when the corresponding complex limit does not.
This limit exists.
}{
\spadpaste{limit(sin(x)/x,x = \%plusInfinity)}
```

```
}
\xtc{
But this limit does not.
}{
\spadpaste{complexLimit(sin(x)/x,x = \%infinity)}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugProblemLimitsPagePatch1}
\begin{paste}{ugProblemLimitsPageFull1}{ugProblemLimitsPageEmpty1}
\pastebutton{ugProblemLimitsPageFull1}{\hidepaste}
\tab{5}\spadcommand{limit((x**2 - 3*x + 2)/(x**2 - 1),x = 1)}
\indentrel{3}\begin{verbatim}
          1
   (1)  -
          2
Type: Union(OrderedCompletion Fraction Polynomial Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemLimitsPageEmpty1}
\begin{paste}{ugProblemLimitsPageEmpty1}{ugProblemLimitsPagePatch1}
\pastebutton{ugProblemLimitsPageEmpty1}{\showpaste}
\tab{5}\spadcommand{limit((x**2 - 3*x + 2)/(x**2 - 1),x = 1)}
\end{paste}\end{patch}

\begin{patch}{ugProblemLimitsPagePatch2}
\begin{paste}{ugProblemLimitsPageFull2}{ugProblemLimitsPageEmpty2}
\pastebutton{ugProblemLimitsPageFull2}{\hidepaste}
\tab{5}\spadcommand{limit(x * log(x),x = 0,"right")}
\indentrel{3}\begin{verbatim}
   (2)  0
  Type: Union(OrderedCompletion Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemLimitsPageEmpty2}
\begin{paste}{ugProblemLimitsPageEmpty2}{ugProblemLimitsPagePatch2}
\pastebutton{ugProblemLimitsPageEmpty2}{\showpaste}
\tab{5}\spadcommand{limit(x * log(x),x = 0,"right")}
\end{paste}\end{patch}

\begin{patch}{ugProblemLimitsPagePatch3}
\begin{paste}{ugProblemLimitsPageFull3}{ugProblemLimitsPageEmpty3}
\pastebutton{ugProblemLimitsPageFull3}{\hidepaste}
\tab{5}\spadcommand{limit(x * log(x),x = 0)}
\indentrel{3}\begin{verbatim}
```

```
   (3)  [leftHandLimit= "failed",rightHandLimit= 0]
Type: Union(Record(leftHandLimit: Union(OrderedCompletion Expression Integer,"failed"),rightHandLimit: Unior
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemLimitsPageEmpty3}
\begin{paste}{ugProblemLimitsPageEmpty3}{ugProblemLimitsPagePatch3}
\pastebutton{ugProblemLimitsPageEmpty3}{\showpaste}
\tab{5}\spadcommand{limit(x * log(x),x = 0)}
\end{paste}\end{patch}

\begin{patch}{ugProblemLimitsPagePatch4}
\begin{paste}{ugProblemLimitsPageFull4}{ugProblemLimitsPageEmpty4}
\pastebutton{ugProblemLimitsPageFull4}{\hidepaste}
\tab{5}\spadcommand{limit(sqrt(y**2)/y,y = 0)}
\indentrel{3}\begin{verbatim}
   (4)  [leftHandLimit= - 1,rightHandLimit= 1]
Type: Union(Record(leftHandLimit: Union(OrderedCompletion Expression Integer,"failed"),rightHandLimit: Unior
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemLimitsPageEmpty4}
\begin{paste}{ugProblemLimitsPageEmpty4}{ugProblemLimitsPagePatch4}
\pastebutton{ugProblemLimitsPageEmpty4}{\showpaste}
\tab{5}\spadcommand{limit(sqrt(y**2)/y,y = 0)}
\end{paste}\end{patch}

\begin{patch}{ugProblemLimitsPagePatch5}
\begin{paste}{ugProblemLimitsPageFull5}{ugProblemLimitsPageEmpty5}
\pastebutton{ugProblemLimitsPageFull5}{\hidepaste}
\tab{5}\spadcommand{limit(sqrt(1 - cos(t))/t,t = 0)}
\indentrel{3}\begin{verbatim}
                            1                    1
   (5)  [leftHandLimit= - ,rightHandLimit= ]

                          \2                  \2
Type: Union(Record(leftHandLimit: Union(OrderedCompletion Expression Integer,"failed"),rightHandLimit: Unior
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemLimitsPageEmpty5}
\begin{paste}{ugProblemLimitsPageEmpty5}{ugProblemLimitsPagePatch5}
\pastebutton{ugProblemLimitsPageEmpty5}{\showpaste}
\tab{5}\spadcommand{limit(sqrt(1 - cos(t))/t,t = 0)}
\end{paste}\end{patch}

\begin{patch}{ugProblemLimitsPagePatch6}
\begin{paste}{ugProblemLimitsPageFull6}{ugProblemLimitsPageEmpty6}
\pastebutton{ugProblemLimitsPageFull6}{\hidepaste}
\tab{5}\spadcommand{limit(sqrt(3*x**2 + 1)/(5*x),x = \%plusInfinity)}
```

```
\indentrel{3}\begin{verbatim}

        \3
   (6)
          5
  Type: Union(OrderedCompletion Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemLimitsPageEmpty6}
\begin{paste}{ugProblemLimitsPageEmpty6}{ugProblemLimitsPagePatch6}
\pastebutton{ugProblemLimitsPageEmpty6}{\showpaste}
\tab{5}\spadcommand{limit(sqrt(3*x**2 + 1)/(5*x),x = \%plusInfinity)}
\end{paste}\end{patch}

\begin{patch}{ugProblemLimitsPagePatch7}
\begin{paste}{ugProblemLimitsPageFull7}{ugProblemLimitsPageEmpty7}
\pastebutton{ugProblemLimitsPageFull7}{\hidepaste}
\tab{5}\spadcommand{limit(sqrt(3*x**2 + 1)/(5*x),x = \%minusInfinity)}
\indentrel{3}\begin{verbatim}

          \3
   (7)  -
            5
  Type: Union(OrderedCompletion Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemLimitsPageEmpty7}
\begin{paste}{ugProblemLimitsPageEmpty7}{ugProblemLimitsPagePatch7}
\pastebutton{ugProblemLimitsPageEmpty7}{\showpaste}
\tab{5}\spadcommand{limit(sqrt(3*x**2 + 1)/(5*x),x = \%minusInfinity)}
\end{paste}\end{patch}

\begin{patch}{ugProblemLimitsPagePatch8}
\begin{paste}{ugProblemLimitsPageFull8}{ugProblemLimitsPageEmpty8}
\pastebutton{ugProblemLimitsPageFull8}{\hidepaste}
\tab{5}\spadcommand{limit(sinh(a*x)/tan(b*x),x = 0)}
\indentrel{3}\begin{verbatim}
        a
   (8)
        b
  Type: Union(OrderedCompletion Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemLimitsPageEmpty8}
\begin{paste}{ugProblemLimitsPageEmpty8}{ugProblemLimitsPagePatch8}
\pastebutton{ugProblemLimitsPageEmpty8}{\showpaste}
\tab{5}\spadcommand{limit(sinh(a*x)/tan(b*x),x = 0)}
```

```
\end{paste}\end{patch}

\begin{patch}{ugProblemLimitsPagePatch9}
\begin{paste}{ugProblemLimitsPageFull9}{ugProblemLimitsPageEmpty9}
\pastebutton{ugProblemLimitsPageFull9}{\hidepaste}
\tab{5}\spadcommand{limit(z * sin(1/z),z = 0)}
\indentrel{3}\begin{verbatim}
   (9)  0
  Type: Union(OrderedCompletion Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemLimitsPageEmpty9}
\begin{paste}{ugProblemLimitsPageEmpty9}{ugProblemLimitsPagePatch9}
\pastebutton{ugProblemLimitsPageEmpty9}{\showpaste}
\tab{5}\spadcommand{limit(z * sin(1/z),z = 0)}
\end{paste}\end{patch}

\begin{patch}{ugProblemLimitsPagePatch10}
\begin{paste}{ugProblemLimitsPageFull10}{ugProblemLimitsPageEmpty10}
\pastebutton{ugProblemLimitsPageFull10}{\hidepaste}
\tab{5}\spadcommand{complexLimit(z * sin(1/z),z = 0)}
\indentrel{3}\begin{verbatim}
   (10)  "failed"
                            Type: Union("failed",...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemLimitsPageEmpty10}
\begin{paste}{ugProblemLimitsPageEmpty10}{ugProblemLimitsPagePatch10}
\pastebutton{ugProblemLimitsPageEmpty10}{\showpaste}
\tab{5}\spadcommand{complexLimit(z * sin(1/z),z = 0)}
\end{paste}\end{patch}

\begin{patch}{ugProblemLimitsPagePatch11}
\begin{paste}{ugProblemLimitsPageFull11}{ugProblemLimitsPageEmpty11}
\pastebutton{ugProblemLimitsPageFull11}{\hidepaste}
\tab{5}\spadcommand{complexLimit((2 + z)/(1 - z),z = \%infinity)}
\indentrel{3}\begin{verbatim}
   (11)  - 1
   Type: OnePointCompletion Fraction Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemLimitsPageEmpty11}
\begin{paste}{ugProblemLimitsPageEmpty11}{ugProblemLimitsPagePatch11}
\pastebutton{ugProblemLimitsPageEmpty11}{\showpaste}
\tab{5}\spadcommand{complexLimit((2 + z)/(1 - z),z = \%infinity)}
\end{paste}\end{patch}
```

```
\begin{patch}{ugProblemLimitsPagePatch12}
\begin{paste}{ugProblemLimitsPageFull12}{ugProblemLimitsPageEmpty12}
\pastebutton{ugProblemLimitsPageFull12}{\hidepaste}
\tab{5}\spadcommand{limit(sin(x)/x,x = \%plusInfinity)}
\indentrel{3}\begin{verbatim}
   (12)  0
  Type: Union(OrderedCompletion Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemLimitsPageEmpty12}
\begin{paste}{ugProblemLimitsPageEmpty12}{ugProblemLimitsPagePatch12}
\pastebutton{ugProblemLimitsPageEmpty12}{\showpaste}
\tab{5}\spadcommand{limit(sin(x)/x,x = \%plusInfinity)}
\end{paste}\end{patch}

\begin{patch}{ugProblemLimitsPagePatch13}
\begin{paste}{ugProblemLimitsPageFull13}{ugProblemLimitsPageEmpty13}
\pastebutton{ugProblemLimitsPageFull13}{\hidepaste}
\tab{5}\spadcommand{complexLimit(sin(x)/x,x = \%infinity)}
\indentrel{3}\begin{verbatim}
   (13)  "failed"
                              Type: Union("failed",...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemLimitsPageEmpty13}
\begin{paste}{ugProblemLimitsPageEmpty13}{ugProblemLimitsPagePatch13}
\pastebutton{ugProblemLimitsPageEmpty13}{\showpaste}
\tab{5}\spadcommand{complexLimit(sin(x)/x,x = \%infinity)}
\end{paste}\end{patch}
```

———-

Laplace Transforms

— ug08.ht —

```
\begin{page}{ugProblemLaplacePage}{8.7. Laplace Transforms}
\beginscroll
%
Axiom can compute some forward Laplace transforms, mostly
of elementary
functions
not involving logarithms, although some cases of
special functions are handled.
\xtc{
```

```
To compute the forward Laplace transform of \axiom{F(t)} with respect to
\axiom{t} and express the result as \axiom{f(s)}, issue the command
\axiom{laplace(F(t), t, s)}.
}{
\spadpaste{laplace(sin(a*t)*cosh(a*t)-cos(a*t)*sinh(a*t), t, s)}
}
\xtc{
Here are some other non-trivial examples.
}{
\spadpaste{laplace((exp(a*t) - exp(b*t))/t, t, s)}
}
\xtc{
}{
\spadpaste{laplace(2/t * (1 - cos(a*t)), t, s)}
}
\xtc{
}{
\spadpaste{laplace(exp(-a*t) * sin(b*t) / b**2, t, s)}
}
\xtc{
}{
\spadpaste{laplace((cos(a*t) - cos(b*t))/t, t, s)}
}
\xtc{
Axiom also knows about a few special functions.
}{
\spadpaste{laplace(exp(a*t+b)*Ei(c*t), t, s)}
}
\xtc{
}{
\spadpaste{laplace(a*Ci(b*t) + c*Si(d*t), t, s)}
}
\xtc{
When Axiom does not know about a particular transform,
it keeps it as a formal transform in the answer.
}{
\spadpaste{laplace(sin(a*t) - a*t*cos(a*t) + exp(t**2), t, s)}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugProblemLaplacePagePatch1}
\begin{paste}{ugProblemLaplacePageFull1}{ugProblemLaplacePageEmpty1}
\pastebutton{ugProblemLaplacePageFull1}{\hidepaste}
\tab{5}\spadcommand{laplace(sin(a*t)*cosh(a*t)-cos(a*t)*sinh(a*t), t, s)}
\indentrel{3}\begin{verbatim}
             3
           4a
```

```
   (1)
        4     4
       s  + 4a
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemLaplacePageEmpty1}
\begin{paste}{ugProblemLaplacePageEmpty1}{ugProblemLaplacePagePatch1}
\pastebutton{ugProblemLaplacePageEmpty1}{\showpaste}
\tab{5}\spadcommand{laplace(sin(a*t)*cosh(a*t)-cos(a*t)*sinh(a*t), t, s)}
\end{paste}\end{patch}

\begin{patch}{ugProblemLaplacePagePatch2}
\begin{paste}{ugProblemLaplacePageFull2}{ugProblemLaplacePageEmpty2}
\pastebutton{ugProblemLaplacePageFull2}{\hidepaste}
\tab{5}\spadcommand{laplace((exp(a*t) - exp(b*t))/t, t, s)}
\indentrel{3}\begin{verbatim}
   (2)  - log(s - a) + log(s - b)
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemLaplacePageEmpty2}
\begin{paste}{ugProblemLaplacePageEmpty2}{ugProblemLaplacePagePatch2}
\pastebutton{ugProblemLaplacePageEmpty2}{\showpaste}
\tab{5}\spadcommand{laplace((exp(a*t) - exp(b*t))/t, t, s)}
\end{paste}\end{patch}

\begin{patch}{ugProblemLaplacePagePatch3}
\begin{paste}{ugProblemLaplacePageFull3}{ugProblemLaplacePageEmpty3}
\pastebutton{ugProblemLaplacePageFull3}{\hidepaste}
\tab{5}\spadcommand{laplace(2/t * (1 - cos(a*t)), t, s)}
\indentrel{3}\begin{verbatim}
            2    2
   (3)  log(s  + a ) - 2log(s)
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemLaplacePageEmpty3}
\begin{paste}{ugProblemLaplacePageEmpty3}{ugProblemLaplacePagePatch3}
\pastebutton{ugProblemLaplacePageEmpty3}{\showpaste}
\tab{5}\spadcommand{laplace(2/t * (1 - cos(a*t)), t, s)}
\end{paste}\end{patch}

\begin{patch}{ugProblemLaplacePagePatch4}
\begin{paste}{ugProblemLaplacePageFull4}{ugProblemLaplacePageEmpty4}
\pastebutton{ugProblemLaplacePageFull4}{\hidepaste}
\tab{5}\spadcommand{laplace(exp(-a*t) * sin(b*t) / b**2, t, s)}
```

```
\indentrel{3}\begin{verbatim}
                    1
   (4)
           2             3    2
        b s  + 2a b s + b  + a b
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemLaplacePageEmpty4}
\begin{paste}{ugProblemLaplacePageEmpty4}{ugProblemLaplacePagePatch4}
\pastebutton{ugProblemLaplacePageEmpty4}{\showpaste}
\tab{5}\spadcommand{laplace(exp(-a*t) * sin(b*t) / b**2, t, s)}
\end{paste}\end{patch}

\begin{patch}{ugProblemLaplacePagePatch5}
\begin{paste}{ugProblemLaplacePageFull5}{ugProblemLaplacePageEmpty5}
\pastebutton{ugProblemLaplacePageFull5}{\hidepaste}
\tab{5}\spadcommand{laplace((cos(a*t) - cos(b*t))/t, t, s)}
\indentrel{3}\begin{verbatim}
             2    2         2    2
        log(s  + b ) - log(s  + a )
   (5)
                     2
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemLaplacePageEmpty5}
\begin{paste}{ugProblemLaplacePageEmpty5}{ugProblemLaplacePagePatch5}
\pastebutton{ugProblemLaplacePageEmpty5}{\showpaste}
\tab{5}\spadcommand{laplace((cos(a*t) - cos(b*t))/t, t, s)}
\end{paste}\end{patch}

\begin{patch}{ugProblemLaplacePagePatch6}
\begin{paste}{ugProblemLaplacePageFull6}{ugProblemLaplacePageEmpty6}
\pastebutton{ugProblemLaplacePageFull6}{\hidepaste}
\tab{5}\spadcommand{laplace(exp(a*t+b)*Ei(c*t), t, s)}
\indentrel{3}\begin{verbatim}
          b    s + c - a
        %e log()
                   c
   (6)
              s - a
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemLaplacePageEmpty6}
\begin{paste}{ugProblemLaplacePageEmpty6}{ugProblemLaplacePagePatch6}
```

```
\pastebutton{ugProblemLaplacePageEmpty6}{\showpaste}
\tab{5}\spadcommand{laplace(exp(a*t+b)*Ei(c*t), t, s)}
\end{paste}\end{patch}

\begin{patch}{ugProblemLaplacePagePatch7}
\begin{paste}{ugProblemLaplacePageFull7}{ugProblemLaplacePageEmpty7}
\pastebutton{ugProblemLaplacePageFull7}{\hidepaste}
\tab{5}\spadcommand{laplace(a*Ci(b*t) + c*Si(d*t), t, s)}
\indentrel{3}\begin{verbatim}
               2    2
              s  + b            d
        a log() + 2c atan()
                  2             s
                 b
   (7)
                     2s
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemLaplacePageEmpty7}
\begin{paste}{ugProblemLaplacePageEmpty7}{ugProblemLaplacePagePatch7}
\pastebutton{ugProblemLaplacePageEmpty7}{\showpaste}
\tab{5}\spadcommand{laplace(a*Ci(b*t) + c*Si(d*t), t, s)}
\end{paste}\end{patch}

\begin{patch}{ugProblemLaplacePagePatch8}
\begin{paste}{ugProblemLaplacePageFull8}{ugProblemLaplacePageEmpty8}
\pastebutton{ugProblemLaplacePageFull8}{\hidepaste}
\tab{5}\spadcommand{laplace(sin(a*t) - a*t*cos(a*t) + exp(t**2), t, s)}
\indentrel{3}\begin{verbatim}
                                   2
          4     2 2    4          t           3
        (s  + 2a s  + a )laplace(%e  ,t,s) + 2a
   (8)
                      4     2 2    4
                     s  + 2a s  + a
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemLaplacePageEmpty8}
\begin{paste}{ugProblemLaplacePageEmpty8}{ugProblemLaplacePagePatch8}
\pastebutton{ugProblemLaplacePageEmpty8}{\showpaste}
\tab{5}\spadcommand{laplace(sin(a*t) - a*t*cos(a*t) + exp(t**2), t, s)}
\end{paste}\end{patch}
```

Integration

⇒ "notitle" (ugxProblemSymRootAllPage) 12 on page 2117

— ug08.ht —

```
\begin{page}{ugProblemIntegrationPage}{8.8. Integration}
\beginscroll
%
Integration is the reverse process of differentiation, that is,
an {\it integral} of a function \axiom{f} with respect to a variable
\axiom{x} is any function \axiom{g} such that \axiom{D(g,x)}
is equal to \axiom{f}.
\xtc{
The package \axiomType{FunctionSpaceIntegration} provides the top-level
integration operation, \axiomFunFrom{integrate}{FunctionSpaceIntegration},
for integrating real-valued elementary functions.
}{
\spadpaste{integrate(cosh(a*x)*sinh(a*x), x)}
}
\xtc{
Unfortunately, antiderivatives of most functions cannot be expressed in
terms of elementary functions.
}{
\spadpaste{integrate(log(1 + sqrt(a * x + b)) / x, x)}
}
Given an elementary function to integrate, Axiom returns a formal
integral as above only when it can prove that the integral is not
elementary and not when it cannot determine the integral.
In this rare case it prints a message that it cannot
determine if an elementary integral exists.
%
\xtc{
Similar functions may have antiderivatives
that look quite different because the form of the antiderivative
depends on the sign of a constant that appears in the function.
}{
\spadpaste{integrate(1/(x**2 - 2),x)}
}
\xtc{
}{
\spadpaste{integrate(1/(x**2 + 2),x)}
}
%
If the integrand contains parameters, then there may be several possible
antiderivatives, depending on the signs of expressions of the parameters.
\xtc{
In this case Axiom returns a list of answers that cover all
the possible cases.
Here you
```

```
use the answer involving the square root of \axiom{a} when \axiom{a > 0} and
the answer involving the square root of \axiom{-a} when \axiom{a < 0}.
}{
\spadpaste{integrate(x**2 / (x**4 - a**2), x)}
}

If the parameters and the variables of integration can be complex
numbers rather than real, then the notion of sign is not defined.
In this case all the possible answers can be expressed as one
complex function.
To get that function, rather than a list of real functions, use
\axiomFunFrom{complexIntegrate}{FunctionSpaceComplexIntegration},
which is provided by the package
\axiomType{FunctionSpaceComplexIntegration}.

\xtc{
This operation is used for
integrating complex-valued elementary functions.
}{
\spadpaste{complexIntegrate(x**2 / (x**4 - a**2), x)}
}
\xtc{
As with the real case,
antiderivatives for most complex-valued functions cannot be expressed
in terms of elementary functions.
}{
\spadpaste{complexIntegrate(log(1 + sqrt(a * x + b)) / x, x)}
}

Sometimes \axiomFun{integrate} can involve symbolic algebraic numbers
such as those returned by \axiomFunFrom{rootOf}{Expression}.
To see how to work with these strange generated symbols (such as
\axiom{\%\%a0}), see
\downlink{``Using All Roots of a Polynomial''}{ugxProblemSymRootAllPage}
in Section 8.3.2\ignore{ugxProblemSymRootAll}.

Definite integration is the process of computing the area between the
\axiom{x}-axis and the curve of a function \axiom{f(x)}.  The
fundamental theorem of calculus states that if \axiom{f} is continuous
on an interval \axiom{a..b} and if there exists a function \axiom{g}
that is differentiable on \axiom{a..b} and such that \axiom{D(g, x)}
is equal to \axiom{f}, then the definite integral of \axiom{f} for
\axiom{x} in the interval \axiom{a..b} is equal to \axiom{g(b) -
g(a)}.

\xtc{
The package \axiomType{RationalFunctionDefiniteIntegration} provides
the top-level definite integration operation,
\axiomFunFrom{integrate}{RationalFunctionDefiniteIntegration},
for integrating real-valued rational functions.
```

```
}{
\spadpaste{integrate((x**4 - 3*x**2 + 6)/(x**6-5*x**4+5*x**2+4), x = 1..2)}
}
Axiom checks beforehand that the function you are integrating is
defined on the interval \axiom{a..b}, and prints an error message if it
finds that this is not case, as in the following example:
\begin{verbatim}
integrate(1/(x**2-2), x = 1..2)

    >> Error detected within library code:
       Pole in path of integration
       You are being returned to the top level
       of the interpreter.
\end{verbatim}
When parameters are present in the function, the function may or may not be
defined on the interval of integration.

\xtc{
If this is the case, Axiom issues a warning that a pole might
lie in the path of integration, and does not compute the integral.
}{
\spadpaste{integrate(1/(x**2-a), x = 1..2)}
}

If you know that you are using values of the parameter for which
the function has no pole in the interval of integration, use the
string {\tt "noPole"} as a third argument to
\axiomFunFrom{integrate}{RationalFunctionDefiniteIntegration}:

%
\xtc{
The value here is, of course, incorrect if \axiom{sqrt(a)} is between
\axiom{1} and \axiom{2.}
}{
\spadpaste{integrate(1/(x**2-a), x = 1..2, "noPole")}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugProblemIntegrationPagePatch1}
\begin{paste}{ugProblemIntegrationPageFull1}{ugProblemIntegrationPageEmpty1}
\pastebutton{ugProblemIntegrationPageFull1}{\hidepaste}
\tab{5}\spadcommand{integrate(cosh(a*x)*sinh(a*x), x)}
\indentrel{3}\begin{verbatim}
                 2            2
        sinh(a x)  + cosh(a x)
   (1)  ----------------------
                   4a
```

```
                    Type: Union(Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemIntegrationPageEmpty1}
\begin{paste}{ugProblemIntegrationPageEmpty1}{ugProblemIntegrationPagePatch1}
\pastebutton{ugProblemIntegrationPageEmpty1}{\showpaste}
\tab{5}\spadcommand{integrate(cosh(a*x)*sinh(a*x), x)}
\end{paste}\end{patch}

\begin{patch}{ugProblemIntegrationPagePatch2}
\begin{paste}{ugProblemIntegrationPageFull2}{ugProblemIntegrationPageEmpty2}
\pastebutton{ugProblemIntegrationPageFull2}{\hidepaste}
\tab{5}\spadcommand{integrate(log(1 + sqrt(a * x + b)) / x, x)}
\indentrel{3}\begin{verbatim}
           x
           log(\b + %E a  + 1)
   (2)       d%E
                    %E
                    Type: Union(Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemIntegrationPageEmpty2}
\begin{paste}{ugProblemIntegrationPageEmpty2}{ugProblemIntegrationPagePatch2}
\pastebutton{ugProblemIntegrationPageEmpty2}{\showpaste}
\tab{5}\spadcommand{integrate(log(1 + sqrt(a * x + b)) / x, x)}
\end{paste}\end{patch}

\begin{patch}{ugProblemIntegrationPagePatch3}
\begin{paste}{ugProblemIntegrationPageFull3}{ugProblemIntegrationPageEmpty3}
\pastebutton{ugProblemIntegrationPageFull3}{\hidepaste}
\tab{5}\spadcommand{integrate(1/(x**2 - 2),x)}
\indentrel{3}\begin{verbatim}
              2
            (x  + 2)\2  - 4x
        log()
                   2
                  x  - 2
   (3)

                 2\2
                    Type: Union(Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemIntegrationPageEmpty3}
\begin{paste}{ugProblemIntegrationPageEmpty3}{ugProblemIntegrationPagePatch3}
\pastebutton{ugProblemIntegrationPageEmpty3}{\showpaste}
\tab{5}\spadcommand{integrate(1/(x**2 - 2),x)}
```

```
\end{paste}\end{patch}

\begin{patch}{ugProblemIntegrationPagePatch4}
\begin{paste}{ugProblemIntegrationPageFull4}{ugProblemIntegrationPageEmpty4}
\pastebutton{ugProblemIntegrationPageFull4}{\hidepaste}
\tab{5}\spadcommand{integrate(1/(x**2 + 2),x)}
\indentrel{3}\begin{verbatim}

            x\2
        atan()
              2
   (4)

           \2
                    Type: Union(Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemIntegrationPageEmpty4}
\begin{paste}{ugProblemIntegrationPageEmpty4}{ugProblemIntegrationPagePatch4}
\pastebutton{ugProblemIntegrationPageEmpty4}{\showpaste}
\tab{5}\spadcommand{integrate(1/(x**2 + 2),x)}
\end{paste}\end{patch}

\begin{patch}{ugProblemIntegrationPagePatch5}
\begin{paste}{ugProblemIntegrationPageFull5}{ugProblemIntegrationPageEmpty5}
\pastebutton{ugProblemIntegrationPageFull5}{\hidepaste}
\tab{5}\spadcommand{integrate(x**2 / (x**4 - a**2), x)}
\indentrel{3}\begin{verbatim}
   (5)
         2
        (x  + a)\a  - 2a x          x\a
    log() + 2atan()
               2                      a
              x  - a
   [,

                    4\a
         2
        (x  - a)\- a  + 2a x          x\- a
    log() - 2atan()
                2                        a
               x  + a
    ]

                     4\- a
               Type: Union(List Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugProblemIntegrationPageEmpty5}
\begin{paste}{ugProblemIntegrationPageEmpty5}{ugProblemIntegrationPagePatch5}
\pastebutton{ugProblemIntegrationPageEmpty5}{\showpaste}
\tab{5}\spadcommand{integrate(x**2 / (x**4 - a**2), x)}
\end{paste}\end{patch}

\begin{patch}{ugProblemIntegrationPagePatch6}
\begin{paste}{ugProblemIntegrationPageFull6}{ugProblemIntegrationPageEmpty6}
\pastebutton{ugProblemIntegrationPageFull6}{\hidepaste}
\tab{5}\spadcommand{complexIntegrate(x**2 / (x**4 - a**2), x)}
\indentrel{3}\begin{verbatim}
   (6)

            x\- 4a  + 2a         x\4a  + 2a
       \4a log() - \- 4a log()

                   \- 4a                     \4a
     +

            x\4a  - 2a         x\- 4a  - 2a
       \- 4a log() - \4a log()

                     \4a                    \- 4a
  /

     2\- 4a \4a
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemIntegrationPageEmpty6}
\begin{paste}{ugProblemIntegrationPageEmpty6}{ugProblemIntegrationPagePatch6}
\pastebutton{ugProblemIntegrationPageEmpty6}{\showpaste}
\tab{5}\spadcommand{complexIntegrate(x**2 / (x**4 - a**2), x)}
\end{paste}\end{patch}

\begin{patch}{ugProblemIntegrationPagePatch7}
\begin{paste}{ugProblemIntegrationPageFull7}{ugProblemIntegrationPageEmpty7}
\pastebutton{ugProblemIntegrationPageFull7}{\hidepaste}
\tab{5}\spadcommand{complexIntegrate(log(1 + sqrt(a * x + b)) / x, x)}
\indentrel{3}\begin{verbatim}
           x
           log(\b + %E a  + 1)
   (7)       d%E
                 %E
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemIntegrationPageEmpty7}
```

```
\begin{paste}{ugProblemIntegrationPageEmpty7}{ugProblemIntegrationPagePatch7}
\pastebutton{ugProblemIntegrationPageEmpty7}{\showpaste}
\tab{5}\spadcommand{complexIntegrate(log(1 + sqrt(a * x + b)) / x, x)}
\end{paste}\end{patch}

\begin{patch}{ugProblemIntegrationPagePatch8}
\begin{paste}{ugProblemIntegrationPageFull8}{ugProblemIntegrationPageEmpty8}
\pastebutton{ugProblemIntegrationPageFull8}{\hidepaste}
\tab{5}\spadcommand{integrate((x**4 - 3*x**2 + 6)/(x**6-5*x**4+5*x**2+4), x = 1..2)}
\indentrel{3}\begin{verbatim}
                                              1
        2atan(8) + 2atan(5) + 2atan(2) + 2atan() - %pi
                                              2
   (8)
                               2
Type: Union(f1: OrderedCompletion Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemIntegrationPageEmpty8}
\begin{paste}{ugProblemIntegrationPageEmpty8}{ugProblemIntegrationPagePatch8}
\pastebutton{ugProblemIntegrationPageEmpty8}{\showpaste}
\tab{5}\spadcommand{integrate((x**4 - 3*x**2 + 6)/(x**6-5*x**4+5*x**2+4), x = 1..2)}
\end{paste}\end{patch}

\begin{patch}{ugProblemIntegrationPagePatch9}
\begin{paste}{ugProblemIntegrationPageFull9}{ugProblemIntegrationPageEmpty9}
\pastebutton{ugProblemIntegrationPageFull9}{\hidepaste}
\tab{5}\spadcommand{integrate(1/(x**2-a), x = 1..2)}
\indentrel{3}\begin{verbatim}
   (9)  potentialPole
                   Type: Union(pole: potentialPole,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemIntegrationPageEmpty9}
\begin{paste}{ugProblemIntegrationPageEmpty9}{ugProblemIntegrationPagePatch9}
\pastebutton{ugProblemIntegrationPageEmpty9}{\showpaste}
\tab{5}\spadcommand{integrate(1/(x**2-a), x = 1..2)}
\end{paste}\end{patch}

\begin{patch}{ugProblemIntegrationPagePatch10}
\begin{paste}{ugProblemIntegrationPageFull10}{ugProblemIntegrationPageEmpty10}
\pastebutton{ugProblemIntegrationPageFull10}{\hidepaste}
\tab{5}\spadcommand{integrate(1/(x**2-a), x = 1..2, "noPole")}
\indentrel{3}\begin{verbatim}
   (10)
   [
                    2           3     2
               (- 4a  - 4a)\a  + a  + 6a  + a
```

```
        - log()
                          2
                         a  - 2a + 1
      +
                    2            3      2
            (- 8a  - 32a)\a  + a  + 24a  + 16a
        log()
                          2
                         a  - 8a + 16
   /

      4\a
    ,

          2\- a          \- a
   - atan() + atan()
             a                a
   ]

                \- a
Type: Union(f2: List OrderedCompletion Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemIntegrationPageEmpty10}
\begin{paste}{ugProblemIntegrationPageEmpty10}{ugProblemIntegrationPagePatch10}
\pastebutton{ugProblemIntegrationPageEmpty10}{\showpaste}
\tab{5}\spadcommand{integrate(1/(x**2-a), x = 1..2, "noPole")}
\end{paste}\end{patch}
```

————-

Working with Power Series

⇒ "notitle" (ugxProblemDEQSeriesPage) 12 on page 2230
⇒ "notitle" (ugxProblemSeriesCreatePage) 12 on page 2166
⇒ "notitle" (ugxProblemSeriesCoefficientsPage) 12 on page 2172
⇒ "notitle" (ugxProblemSeriesArithmeticPage) 12 on page 2175
⇒ "notitle" (ugxProblemSeriesFunctionsPage) 12 on page 2178
⇒ "notitle" (ugxProblemSeriesConversionsPage) 12 on page 2186
⇒ "notitle" (ugxProblemSeriesFormulaPage) 12 on page 2194
⇒ "notitle" (ugxProblemSeriesSubstitutePage) 12 on page 2201
⇒ "notitle" (ugxProblemSeriesBernoulliPage) 12 on page 2203

— **ug08.ht** —

```
\begin{page}{ugProblemSeriesPage}{8.9. Working with Power Series}
```

```
\beginscroll
%
Axiom has very sophisticated facilities for working with power
series.
Infinite series are represented by a list of the
coefficients that have already been determined, together with a
function for computing the additional coefficients if needed.
%
The system command that determines how many terms of a series is displayed
is \spadcmd{)set streams calculate}.
For the purposes of this book, we have used this system command to display
fewer than ten terms.
Series can be created from expressions, from functions for the
series coefficients, and from applications of operations on
existing series.
The most general function for creating a series is called
\axiomFun{series}, although you can also use \axiomFun{taylor},
\axiomFun{laurent} and \axiomFun{puiseux} in situations where you
know what kind of exponents are involved.

For information about solving differential equations in terms of
power series, see
\downlink{``Power Series Solutions of Differential Equations''}
{ugxProblemDEQSeriesPage} in
Section 8.10.3\ignore{ugxProblemDEQSeries}.

\beginmenu
  \menudownlink{{8.9.1. Creation of Power Series}}
{ugxProblemSeriesCreatePage}
  \menudownlink{{8.9.2. Coefficients of Power Series}}
{ugxProblemSeriesCoefficientsPage}
  \menudownlink{{8.9.3. Power Series Arithmetic}}
{ugxProblemSeriesArithmeticPage}
  \menudownlink{{8.9.4. Functions on Power Series}}
{ugxProblemSeriesFunctionsPage}
  \menudownlink{{8.9.5. Converting to Power Series}}
{ugxProblemSeriesConversionsPage}
  \menudownlink{{8.9.6. Power Series from Formulas}}
{ugxProblemSeriesFormulaPage}
  \menudownlink{{8.9.7. Substituting Numerical Values in Power Series}}
{ugxProblemSeriesSubstitutePage}
  \menudownlink{{8.9.8. Example: Bernoulli Polynomials and Sums of Powers}}
{ugxProblemSeriesBernoulliPage}
\endmenu
\endscroll
\autobuttons
\end{page}
```

Creation of Power Series

⇒ "notitle" (ugxProblemSeriesConversionsPage) 12 on page 2186
⇒ "notitle" (ugTypesDeclarePage) 7 on page 1641
⇒ "notitle" (ugxProblemSeriesFunctionsPage) 12 on page 2178
⇒ "notitle" (ugxProblemSeriesFormulaPage) 12 on page 2194

— **ug08.ht** —

```
\begin{page}{ugxProblemSeriesCreatePage}{8.9.1. Creation of Power Series}
\beginscroll

\labelSpace{4pc}
\xtc{
This is the easiest way to create a power series.
This tells Axiom that \axiom{x} is to be treated as a power series,
so functions of \axiom{x} are again power series.
}{
\spadpaste{x := series 'x \bound{x}}
}
%
We didn't say anything about the coefficients of the power
series, so the coefficients are general expressions over the integers.
This allows us to introduce denominators, symbolic constants, and other
variables as needed.
\xtc{
Here the coefficients are integers (note that the coefficients
are the Fibonacci
numbers).
}{
\spadpaste{1/(1 - x - x**2) \free{x}}
}
\xtc{
This series has coefficients that are rational numbers.
}{
\spadpaste{sin(x) \free{x}}
}
\xtc{
When you enter this expression
you introduce the symbolic constants \axiom{sin(1)} and \axiom{cos(1).}
}{
\spadpaste{sin(1 + x) \free{x}}
}
\xtc{
When you enter the expression
the variable \axiom{a} appears in the resulting series expansion.
}{
\spadpaste{sin(a * x) \free{x}}
}
```

```
\xtc{
You can also convert an expression into a series expansion.
This expression creates the series expansion of \axiom{1/log(y)}
about \axiom{y = 1}.
For details and more examples, see
\downlink{``Converting to Power Series''}
{ugxProblemSeriesConversionsPage} in Section
8.9.5\ignore{ugxProblemSeriesConversions}.
}{
\spadpaste{series(1/log(y),y = 1)}
}

You can create power series with more general coefficients.
You normally accomplish this via a type declaration (see
\downlink{``Declarations''}{ugTypesDeclarePage} in Section
2.3\ignore{ugTypesDeclare}).
See
\downlink{``Functions on Power Series''}
{ugxProblemSeriesFunctionsPage}
in Section 8.9.4\ignore{ugxProblemSeriesFunctions}
for some warnings about working with declared series.

\xtc{
We declare that \axiom{y} is a one-variable Taylor series
(\axiomType{UTS} is the abbreviation for
\axiomType{UnivariateTaylorSeries})
in the variable \axiom{z} with \axiomType{FLOAT}
(that is, floating-point) coefficients, centered about \axiom{0.}
Then, by assignment, we obtain the Taylor expansion of
\axiom{exp(z)} with floating-point coefficients.
}{
\spadpaste{y : UTS(FLOAT,'z,0) := exp(z) \bound{y}}
}

You can also create a power series by giving an explicit formula
for its \eth{\axiom{n}} coefficient.
For details and more examples, see
\downlink{``Power Series from Formulas''}{ugxProblemSeriesFormulaPage}
in Section 8.9.6\ignore{ugxProblemSeriesFormula}.

\xtc{
To create a series about
\axiom{w = 0} whose \eth{\axiom{n}} Taylor coefficient
is \axiom{1/n!}, you can evaluate this expression.
This is the Taylor expansion of \axiom{exp(w)} at \axiom{w = 0}.
}{
\spadpaste{series(1/factorial(n),n,w = 0)}
}
%
```

```
\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxProblemSeriesCreatePagePatch1}
\begin{paste}{ugxProblemSeriesCreatePageFull1}{ugxProblemSeriesCreatePageEmpty1}
\pastebutton{ugxProblemSeriesCreatePageFull1}{\hidepaste}
\tab{5}\spadcommand{x := series 'x\bound{x }}
\indentrel{3}\begin{verbatim}
   (1)  x
  Type: UnivariatePuiseuxSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesCreatePageEmpty1}
\begin{paste}{ugxProblemSeriesCreatePageEmpty1}{ugxProblemSeriesCreatePagePatch1}
\pastebutton{ugxProblemSeriesCreatePageEmpty1}{\showpaste}
\tab{5}\spadcommand{x := series 'x\bound{x }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesCreatePagePatch2}
\begin{paste}{ugxProblemSeriesCreatePageFull2}{ugxProblemSeriesCreatePageEmpty2}
\pastebutton{ugxProblemSeriesCreatePageFull2}{\hidepaste}
\tab{5}\spadcommand{1/(1 - x - x**2)\free{x }}
\indentrel{3}\begin{verbatim}
   (2)
               2     3     4     5      6      7      8
     1 + x + 2x  + 3x  + 5x  + 8x  + 13x  + 21x  + 34x
   +
        9      10      11
     55x  + 89x   + O(x  )
  Type: UnivariatePuiseuxSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesCreatePageEmpty2}
\begin{paste}{ugxProblemSeriesCreatePageEmpty2}{ugxProblemSeriesCreatePagePatch2}
\pastebutton{ugxProblemSeriesCreatePageEmpty2}{\showpaste}
\tab{5}\spadcommand{1/(1 - x - x**2)\free{x }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesCreatePagePatch3}
\begin{paste}{ugxProblemSeriesCreatePageFull3}{ugxProblemSeriesCreatePageEmpty3}
\pastebutton{ugxProblemSeriesCreatePageFull3}{\hidepaste}
\tab{5}\spadcommand{sin(x)\free{x }}
\indentrel{3}\begin{verbatim}
   (3)
        1  3    1   5     1   7      1    9
     x -  x  +  x  -  x  +  x
```

```
        6      120      5040      362880
   +
          1     11      12
     -  x   + O(x  )
      39916800
  Type: UnivariatePuiseuxSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesCreatePageEmpty3}
\begin{paste}{ugxProblemSeriesCreatePageEmpty3}{ugxProblemSeriesCreatePagePatch3}
\pastebutton{ugxProblemSeriesCreatePageEmpty3}{\showpaste}
\tab{5}\spadcommand{sin(x)\free{x }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesCreatePagePatch4}
\begin{paste}{ugxProblemSeriesCreatePageFull4}{ugxProblemSeriesCreatePageEmpty4}
\pastebutton{ugxProblemSeriesCreatePageFull4}{\hidepaste}
\tab{5}\spadcommand{sin(1 + x)\free{x }}
\indentrel{3}\begin{verbatim}
   (4)
                        sin(1)  2   cos(1)  3   sin(1)  4
     sin(1) + cos(1)x -  x  -  x  +  x
                           2          6          24
   +
     cos(1)  5   sin(1)  6   cos(1)  7   sin(1)  8
      x  -  x  -  x  +  x
       120         720        5040        40320
   +
     cos(1)  9    sin(1)  10      11
      x  -  x   + O(x  )
     362880      3628800
  Type: UnivariatePuiseuxSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesCreatePageEmpty4}
\begin{paste}{ugxProblemSeriesCreatePageEmpty4}{ugxProblemSeriesCreatePagePatch4}
\pastebutton{ugxProblemSeriesCreatePageEmpty4}{\showpaste}
\tab{5}\spadcommand{sin(1 + x)\free{x }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesCreatePagePatch5}
\begin{paste}{ugxProblemSeriesCreatePageFull5}{ugxProblemSeriesCreatePageEmpty5}
\pastebutton{ugxProblemSeriesCreatePageFull5}{\hidepaste}
\tab{5}\spadcommand{sin(a * x)\free{x }}
\indentrel{3}\begin{verbatim}
   (5)
           3       5       7         9
          a   3    a   5    a   7     a     9
```

```
     a x -  x  +  x  -  x  +  x
            6      120      5040      362880
  +
           11
          a      11      12
     -  x   + O(x  )
       39916800
  Type: UnivariatePuiseuxSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesCreatePageEmpty5}
\begin{paste}{ugxProblemSeriesCreatePageEmpty5}{ugxProblemSeriesCreatePagePatch5}
\pastebutton{ugxProblemSeriesCreatePageEmpty5}{\showpaste}
\tab{5}\spadcommand{sin(a * x)\free{x }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesCreatePagePatch6}
\begin{paste}{ugxProblemSeriesCreatePageFull6}{ugxProblemSeriesCreatePageEmpty6}
\pastebutton{ugxProblemSeriesCreatePageFull6}{\hidepaste}
\tab{5}\spadcommand{series(1/log(y),y = 1)}
\indentrel{3}\begin{verbatim}
   (6)
           - 1   1    1            1        2
     (y - 1)    +  -  (y - 1) +  (y - 1)
                 2   12           24
  +
       19        3    3         4    863         5
     -  (y - 1)  +  (y - 1)  -  (y - 1)
       720           160           60480
  +
      275         6    33953         7     8183         8
      (y - 1)  -  (y - 1)  +  (y - 1)
     24192            3628800            1036800
  +
       3250433         9           10
     -  (y - 1)  + O((y - 1)  )
       479001600
  Type: UnivariatePuiseuxSeries(Expression Integer,y,1)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesCreatePageEmpty6}
\begin{paste}{ugxProblemSeriesCreatePageEmpty6}{ugxProblemSeriesCreatePagePatch6}
\pastebutton{ugxProblemSeriesCreatePageEmpty6}{\showpaste}
\tab{5}\spadcommand{series(1/log(y),y = 1)}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesCreatePagePatch7}
\begin{paste}{ugxProblemSeriesCreatePageFull7}{ugxProblemSeriesCreatePageEmpty7}
```

```
\pastebutton{ugxProblemSeriesCreatePageFull7}{\hidepaste}
\tab{5}\spadcommand{y : UTS(FLOAT,'z,0) := exp(z)\bound{y }}
\indentrel{3}\begin{verbatim}
   (7)
                    2                            3
     1.0 + z + 0.5 z  + 0.1666666666 6666666667 z
   +
                               4
     0.0416666666 6666666666 7 z
   +
                                5
     0.0083333333 3333333333 34 z
   +
                                6
     0.0013888888 8888888888 89 z
   +
                                7
     0.0001984126 9841269841 27 z
   +
                                  8
     0.0000248015 8730158730 1587 z
   +
                                   9
     0.0000027557 3192239858 90653 z
   +
                                  10      11
     0.2755731922 3985890653 E -6 z   + O(z  )
              Type: UnivariateTaylorSeries(Float,z,0.0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesCreatePageEmpty7}
\begin{paste}{ugxProblemSeriesCreatePageEmpty7}{ugxProblemSeriesCreatePagePatch7}
\pastebutton{ugxProblemSeriesCreatePageEmpty7}{\showpaste}
\tab{5}\spadcommand{y : UTS(FLOAT,'z,0) := exp(z)\bound{y }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesCreatePagePatch8}
\begin{paste}{ugxProblemSeriesCreatePageFull8}{ugxProblemSeriesCreatePageEmpty8}
\pastebutton{ugxProblemSeriesCreatePageFull8}{\hidepaste}
\tab{5}\spadcommand{series(1/factorial(n),n,w = 0)}
\indentrel{3}\begin{verbatim}
   (8)
            1  2   1  3    1  4    1   5    1   6
     1 + w +  w  +  w  +  w  +  w  +  w
            2      6      24      120      720
   +
     1   7     1    8      1    9      1     10      11
    w  +  w  +  w  +  w   + O(w  )
   5040      40320      362880      3628800
```

```
  Type: UnivariatePuiseuxSeries(Expression Integer,w,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesCreatePageEmpty8}
\begin{paste}{ugxProblemSeriesCreatePageEmpty8}{ugxProblemSeriesCreatePagePatch8}
\pastebutton{ugxProblemSeriesCreatePageEmpty8}{\showpaste}
\tab{5}\spadcommand{series(1/factorial(n),n,w = 0)}
\end{paste}\end{patch}
```

———-

Coefficients of Power Series

— ug08.ht —

```
\begin{page}{ugxProblemSeriesCoefficientsPage}
{8.9.2. Coefficients of Power Series}
\beginscroll

You can extract any coefficient from a power series---even one
that hasn't been computed yet.
This is possible because in Axiom, infinite series are
represented by a list of the coefficients that have already been
determined, together with a function for computing the additional
coefficients.
(This is known as {\it lazy evaluation}.) When you ask for a
coefficient that hasn't yet been computed, Axiom computes
whatever additional coefficients it needs and then stores them in
the representation of the power series.

\xtc{
Here's an example of how to extract the coefficients of a power series.
}{
\spadpaste{x := series(x) \bound{x}}
}
\xtc{
}{
\spadpaste{y := exp(x) * sin(x) \free{x} \bound{y}}
}
\xtc{
This coefficient is readily available.
}{
\spadpaste{coefficient(y,6) \free{y}}
}
\xtc{
But let's get the fifteenth coefficient of \axiom{y}.
```

```
}{
\spadpaste{coefficient(y,15) \free{y} \bound{y15}}
}
\xtc{
If you look at \axiom{y}
then you see that the coefficients up to order \axiom{15}
have all been computed.
}{
\spadpaste{y \free{y15}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxProblemSeriesCoefficientsPagePatch1}
\begin{paste}{ugxProblemSeriesCoefficientsPageFull1}{ugxProblemSeriesCoefficientsPageEmpty1}
\pastebutton{ugxProblemSeriesCoefficientsPageFull1}{\hidepaste}
\tab{5}\spadcommand{x := series(x)\bound{x }}
\indentrel{3}\begin{verbatim}
   (1)  x
  Type: UnivariatePuiseuxSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesCoefficientsPageEmpty1}
\begin{paste}{ugxProblemSeriesCoefficientsPageEmpty1}{ugxProblemSeriesCoefficientsPagePatch1}
\pastebutton{ugxProblemSeriesCoefficientsPageEmpty1}{\showpaste}
\tab{5}\spadcommand{x := series(x)\bound{x }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesCoefficientsPagePatch2}
\begin{paste}{ugxProblemSeriesCoefficientsPageFull2}{ugxProblemSeriesCoefficientsPageEmpty2}
\pastebutton{ugxProblemSeriesCoefficientsPageFull2}{\hidepaste}
\tab{5}\spadcommand{y := exp(x) * sin(x)\free{x }\bound{y }}
\indentrel{3}\begin{verbatim}
   (2)
          2   1  3    1  5    1  6    1   7     1    9
     x + x  +  x  -  x  -  x  -  x  +  x
              3      30      90     630      22680
   +
        1    10      1     11      12
      x   +  x   + O(x  )
     113400       1247400
  Type: UnivariatePuiseuxSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesCoefficientsPageEmpty2}
\begin{paste}{ugxProblemSeriesCoefficientsPageEmpty2}{ugxProblemSeriesCoefficientsPagePatch2}
```

```
\pastebutton{ugxProblemSeriesCoefficientsPageEmpty2}{\showpaste}
\tab{5}\spadcommand{y := exp(x) * sin(x)\free{x }\bound{y }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesCoefficientsPagePatch3}
\begin{paste}{ugxProblemSeriesCoefficientsPageFull3}{ugxProblemSeriesCoefficientsPageEmpty3}
\pastebutton{ugxProblemSeriesCoefficientsPageFull3}{\hidepaste}
\tab{5}\spadcommand{coefficient(y,6)\free{y }}
\indentrel{3}\begin{verbatim}
          1
   (3)  -
         90
                              Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesCoefficientsPageEmpty3}
\begin{paste}{ugxProblemSeriesCoefficientsPageEmpty3}{ugxProblemSeriesCoefficientsPagePatch3}
\pastebutton{ugxProblemSeriesCoefficientsPageEmpty3}{\showpaste}
\tab{5}\spadcommand{coefficient(y,6)\free{y }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesCoefficientsPagePatch4}
\begin{paste}{ugxProblemSeriesCoefficientsPageFull4}{ugxProblemSeriesCoefficientsPageEmpty4}
\pastebutton{ugxProblemSeriesCoefficientsPageFull4}{\hidepaste}
\tab{5}\spadcommand{coefficient(y,15)\free{y }\bound{y15 }}
\indentrel{3}\begin{verbatim}
              1
   (4)  -
         10216206000
                              Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesCoefficientsPageEmpty4}
\begin{paste}{ugxProblemSeriesCoefficientsPageEmpty4}{ugxProblemSeriesCoefficientsPagePatch4}
\pastebutton{ugxProblemSeriesCoefficientsPageEmpty4}{\showpaste}
\tab{5}\spadcommand{coefficient(y,15)\free{y }\bound{y15 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesCoefficientsPagePatch5}
\begin{paste}{ugxProblemSeriesCoefficientsPageFull5}{ugxProblemSeriesCoefficientsPageEmpty5}
\pastebutton{ugxProblemSeriesCoefficientsPageFull5}{\hidepaste}
\tab{5}\spadcommand{y\free{y15 }}
\indentrel{3}\begin{verbatim}
   (5)
         2   1  3    1  5    1  6    1   7     1    9
     x + x  +  x  -  x  -  x  -  x  +  x
               3      30      90      630      22680
   +
```

```
        1    10      1     11       1     13
      x   + x   - x
     113400       1247400       97297200
   +
           1      14        1       15      16
     -  x   -  x   + O(x  )
       681080400       10216206000
  Type: UnivariatePuiseuxSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesCoefficientsPageEmpty5}
\begin{paste}{ugxProblemSeriesCoefficientsPageEmpty5}{ugxProblemSeriesCoefficientsPagePatch5}
\pastebutton{ugxProblemSeriesCoefficientsPageEmpty5}{\showpaste}
\tab{5}\spadcommand{y\free{y15 }}
\end{paste}\end{patch}
```

———-

Power Series Arithmetic

— ug08.ht —

```
\begin{page}{ugxProblemSeriesArithmeticPage}
{8.9.3. Power Series Arithmetic}
\beginscroll

You can manipulate power series using the usual arithmetic operations
\axiomOpFrom{+}{UnivariatePuiseuxSeries},
\axiomOpFrom{-}{UnivariatePuiseuxSeries},
\axiomOpFrom{*}{UnivariatePuiseuxSeries}, and
\axiomOpFrom{/}{UnivariatePuiseuxSeries}.
%

\labelSpace{1pc}
\xtc{
The results of these operations are also power series.
}{
\spadpaste{x := series x \bound{x}}
}
\xtc{
}{
\spadpaste{(3 + x) / (1 + 7*x)}
}
\xtc{
You can also compute \axiom{f(x) ** g(x)}, where
\axiom{f(x)} and \axiom{g(x)}
```

```
are two power series.
}{
\spadpaste{base := 1 / (1 - x) \free{x} \bound{base}}
}
%
\xtc{
}{
\spadpaste{expon := x * base \free{x base} \bound{expon}}
}
%
\xtc{
}{
\spadpaste{base ** expon \free{base expon}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxProblemSeriesArithmeticPagePatch1}
\begin{paste}{ugxProblemSeriesArithmeticPageFull1}{ugxProblemSeriesArithmeticPageEmpty1}
\pastebutton{ugxProblemSeriesArithmeticPageFull1}{\hidepaste}
\tab{5}\spadcommand{x := series x\bound{x }}
\indentrel{3}\begin{verbatim}
   (1)  x
  Type: UnivariatePuiseuxSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesArithmeticPageEmpty1}
\begin{paste}{ugxProblemSeriesArithmeticPageEmpty1}{ugxProblemSeriesArithmeticPagePatch1}
\pastebutton{ugxProblemSeriesArithmeticPageEmpty1}{\showpaste}
\tab{5}\spadcommand{x := series x\bound{x }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesArithmeticPagePatch2}
\begin{paste}{ugxProblemSeriesArithmeticPageFull2}{ugxProblemSeriesArithmeticPageEmpty2}
\pastebutton{ugxProblemSeriesArithmeticPageFull2}{\hidepaste}
\tab{5}\spadcommand{(3 + x) / (1 + 7*x)}
\indentrel{3}\begin{verbatim}
   (2)
                  2       3        4         5          6
     3 - 20x + 140x  - 980x  + 6860x  - 48020x  + 336140x
   +
               7            8             9             10
     - 2352980x  + 16470860x  - 115296020x  + 807072140x
   +
        11
     O(x  )
  Type: UnivariatePuiseuxSeries(Expression Integer,x,0)
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesArithmeticPageEmpty2}
\begin{paste}{ugxProblemSeriesArithmeticPageEmpty2}{ugxProblemSeriesArithmeticPagePatch2}
\pastebutton{ugxProblemSeriesArithmeticPageEmpty2}{\showpaste}
\tab{5}\spadcommand{(3 + x) / (1 + 7*x)}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesArithmeticPagePatch3}
\begin{paste}{ugxProblemSeriesArithmeticPageFull3}{ugxProblemSeriesArithmeticPageEmpty3}
\pastebutton{ugxProblemSeriesArithmeticPageFull3}{\hidepaste}
\tab{5}\spadcommand{base := 1 / (1 - x)\free{x }\bound{base }}
\indentrel{3}\begin{verbatim}
   (3)
             2    3    4    5    6    7    8    9    10
     1 + x + x  + x  + x  + x  + x  + x  + x  + x  + x
   +
        11
     O(x  )
  Type: UnivariatePuiseuxSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesArithmeticPageEmpty3}
\begin{paste}{ugxProblemSeriesArithmeticPageEmpty3}{ugxProblemSeriesArithmeticPagePatch3}
\pastebutton{ugxProblemSeriesArithmeticPageEmpty3}{\showpaste}
\tab{5}\spadcommand{base := 1 / (1 - x)\free{x }\bound{base }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesArithmeticPagePatch4}
\begin{paste}{ugxProblemSeriesArithmeticPageFull4}{ugxProblemSeriesArithmeticPageEmpty4}
\pastebutton{ugxProblemSeriesArithmeticPageFull4}{\hidepaste}
\tab{5}\spadcommand{expon := x * base\free{x base }\bound{expon }}
\indentrel{3}\begin{verbatim}
   (4)
         2    3    4    5    6    7    8    9    10    11
     x + x  + x  + x  + x  + x  + x  + x  + x  + x   + x
   +
        12
     O(x  )
  Type: UnivariatePuiseuxSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesArithmeticPageEmpty4}
\begin{paste}{ugxProblemSeriesArithmeticPageEmpty4}{ugxProblemSeriesArithmeticPagePatch4}
\pastebutton{ugxProblemSeriesArithmeticPageEmpty4}{\showpaste}
\tab{5}\spadcommand{expon := x * base\free{x base }\bound{expon }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugxProblemSeriesArithmeticPagePatch5}
\begin{paste}{ugxProblemSeriesArithmeticPageFull5}{ugxProblemSeriesArithmeticPageEmpty5}
\pastebutton{ugxProblemSeriesArithmeticPageFull5}{\hidepaste}
\tab{5}\spadcommand{base ** expon\free{base expon }}
\indentrel{3}\begin{verbatim}
   (5)
          2   3  3   7  4   43  5   649  6   241  7
     1 + x  +  x  +  x  +  x  +  x  +  x
             2      3      12      120       30
   +
     3706  8   85763  9   245339  10      11
      x  +  x  +  x   + O(x  )
      315       5040       10080
  Type: UnivariatePuiseuxSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesArithmeticPageEmpty5}
\begin{paste}{ugxProblemSeriesArithmeticPageEmpty5}{ugxProblemSeriesArithmeticPagePatch5}
\pastebutton{ugxProblemSeriesArithmeticPageEmpty5}{\showpaste}
\tab{5}\spadcommand{base ** expon\free{base expon }}
\end{paste}\end{patch}
```

————-

Functions on Power Series

— ug08.ht —

```
\begin{page}{ugxProblemSeriesFunctionsPage}
{8.9.4. Functions on Power Series}
\beginscroll

Once you have created a power series, you can apply transcendental
functions
(for example, \axiomFun{exp}, \axiomFun{log},
\axiomFun{sin}, \axiomFun{tan},
\axiomFun{cosh}, etc.) to it.

\labelSpace{1pc}
%
\xtc{
To demonstrate this, we first create the power series
expansion of the rational function
\texht{
${\displaystyle x^2} \over {\displaystyle 1 - 6x + x^2}$
```

```
}{
\axiom{x**2/(1 - 6*x + x**2)}
}
about \axiom{x = 0}.
}{
\spadpaste{x := series 'x \bound{x}}
}
%
\xtc{
}{
\spadpaste{rat := x**2 / (1 - 6*x + x**2) \free{x} \bound{rat}}
}
%
%
\xtc{
If you want to compute the series expansion of
\texht{
$\sin\left({\displaystyle x^2} \over {\displaystyle 1 - 6x + x^2}\right)$
}{
\axiom{sin(x**2/(1 - 6*x + x**2))}
}
you simply compute the sine of \axiom{rat}.
}{
\spadpaste{sin(rat) \free{rat}}
}
%

\beginImportant
\noindent {\bf Warning:}
the type of the coefficients of a power series may
affect the kind of computations that you can do with that series.
This can only happen when you have made a declaration to
specify a series domain with a certain type of coefficient.
\endImportant

\xtc{
If you evaluate
then you have declared that \axiom{y} is a one variable Taylor series
(\axiomType{UTS} is the abbreviation for \axiomType{UnivariateTaylorSeries})
in the variable \axiom{y} with \axiomType{FRAC INT}
(that is, fractions of integer) coefficients, centered about \axiom{0}.
}{
\spadpaste{y : UTS(FRAC INT,y,0) := y \bound{y}}
}
%
\xtc{
You can now compute certain power series in \axiom{y},
{\it provided} that these series have rational coefficients.
}{
\spadpaste{exp(y) \free{y}}
```

```
}
%
\xtc{
You can get examples of such series
by applying transcendental functions to
series in \axiom{y} that have no constant terms.
}{
\spadpaste{tan(y**2) \free{y}}
}
%
\xtc{
}{
\spadpaste{cos(y + y**5) \free{y}}
}
%
%
\xtc{
Similarly, you can compute the logarithm of a power series with rational
coefficients if the constant coefficient is \axiom{1.}
}{
\spadpaste{log(1 + sin(y)) \free{y}}
}
%
If you wanted to apply, say, the operation \axiomFun{exp} to a power
series with a nonzero constant coefficient \texht{$a_0$}{\axiom{a0}},
then the constant coefficient of the result would be
\texht{$e^{a_0}$}{\axiom{exp(a0)}}, which is {\it not} a rational number.
Therefore, evaluating \axiom{exp(2 + tan(y))} would generate an error
message.

If you want to compute the Taylor expansion of \axiom{exp(2 +
tan(y))}, you must ensure that the coefficient domain has an operation
\axiomFun{exp} defined for it.  An example of such a domain is
\axiomType{Expression Integer}, the type of formal functional
expressions over the integers.

\xtc{
When working with coefficients of this type,
}{
\spadpaste{z : UTS(EXPR INT,z,0) := z \bound{z}}
}
\xtc{
this presents no problems.
}{
\spadpaste{exp(2 + tan(z)) \free{z}}
}
%
Another way to create Taylor series whose coefficients are expressions over
the integers is to use \axiomFun{taylor} which works similarly to
\axiomFun{series}.
```

```
%
\xtc{
This is equivalent to the previous computation, except that now we
are using the variable \axiom{w} instead of \axiom{z}.
}{
\spadpaste{w := taylor 'w \bound{w}}
}
\xtc{
}{
\spadpaste{exp(2 + tan(w)) \free{w}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxProblemSeriesFunctionsPagePatch1}
\begin{paste}{ugxProblemSeriesFunctionsPageFull1}{ugxProblemSeriesFunctionsPageEmpty1}
\pastebutton{ugxProblemSeriesFunctionsPageFull1}{\hidepaste}
\tab{5}\spadcommand{x := series 'x\bound{x }}
\indentrel{3}\begin{verbatim}
   (1)  x
  Type: UnivariatePuiseuxSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesFunctionsPageEmpty1}
\begin{paste}{ugxProblemSeriesFunctionsPageEmpty1}{ugxProblemSeriesFunctionsPagePatch1}
\pastebutton{ugxProblemSeriesFunctionsPageEmpty1}{\showpaste}
\tab{5}\spadcommand{x := series 'x\bound{x }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesFunctionsPagePatch2}
\begin{paste}{ugxProblemSeriesFunctionsPageFull2}{ugxProblemSeriesFunctionsPageEmpty2}
\pastebutton{ugxProblemSeriesFunctionsPageFull2}{\hidepaste}
\tab{5}\spadcommand{rat := x**2 / (1 - 6*x + x**2)\free{x }\bound{rat }}
\indentrel{3}\begin{verbatim}
   (2)
     2     3      4       5        6        7         8
    x  + 6x  + 35x  + 204x  + 1189x  + 6930x  + 40391x
  +
           9           10           11            12
    235416x  + 1372105x   + 7997214x   + 46611179x
  +
       13
    O(x  )
  Type: UnivariatePuiseuxSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugxProblemSeriesFunctionsPageEmpty2}
\begin{paste}{ugxProblemSeriesFunctionsPageEmpty2}{ugxProblemSeriesFunctionsPagePatch2}
\pastebutton{ugxProblemSeriesFunctionsPageEmpty2}{\showpaste}
\tab{5}\spadcommand{rat := x**2 / (1 - 6*x + x**2)\free{x }\bound{rat }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesFunctionsPagePatch3}
\begin{paste}{ugxProblemSeriesFunctionsPageFull3}{ugxProblemSeriesFunctionsPageEmpty3}
\pastebutton{ugxProblemSeriesFunctionsPageFull3}{\hidepaste}
\tab{5}\spadcommand{sin(rat)\free{rat }}
\indentrel{3}\begin{verbatim}
   (3)
      2     3      4       5   7133  6        7   80711  8
     x  + 6x  + 35x  + 204x  +  x  + 6927x  +  x
                                6                 2
   +
           9   164285281  10   31888513  11
     235068x  +  x   +  x
                  120             4
   +
     371324777  12      13
      x   + O(x  )
         8
  Type: UnivariatePuiseuxSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesFunctionsPageEmpty3}
\begin{paste}{ugxProblemSeriesFunctionsPageEmpty3}{ugxProblemSeriesFunctionsPagePatch3}
\pastebutton{ugxProblemSeriesFunctionsPageEmpty3}{\showpaste}
\tab{5}\spadcommand{sin(rat)\free{rat }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesFunctionsPagePatch4}
\begin{paste}{ugxProblemSeriesFunctionsPageFull4}{ugxProblemSeriesFunctionsPageEmpty4}
\pastebutton{ugxProblemSeriesFunctionsPageFull4}{\hidepaste}
\tab{5}\spadcommand{y : UTS(FRAC INT,y,0) := y\bound{y }}
\indentrel{3}\begin{verbatim}
   (4)  y
     Type: UnivariateTaylorSeries(Fraction Integer,y,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesFunctionsPageEmpty4}
\begin{paste}{ugxProblemSeriesFunctionsPageEmpty4}{ugxProblemSeriesFunctionsPagePatch4}
\pastebutton{ugxProblemSeriesFunctionsPageEmpty4}{\showpaste}
\tab{5}\spadcommand{y : UTS(FRAC INT,y,0) := y\bound{y }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesFunctionsPagePatch5}
```

```
\begin{paste}{ugxProblemSeriesFunctionsPageFull5}{ugxProblemSeriesFunctionsPageEmpty5}
\pastebutton{ugxProblemSeriesFunctionsPageFull5}{\hidepaste}
\tab{5}\spadcommand{exp(y)\free{y }}
\indentrel{3}\begin{verbatim}
   (5)
            1  2   1  3    1  4    1   5    1   6
     1 + y + y  +  y  +  y  +  y  +  y
            2      6      24      120      720
   +
     1   7     1    8      1    9      1     10      11
    y  +  y  +  y  +  y   + O(y  )
   5040      40320      362880      3628800
     Type: UnivariateTaylorSeries(Fraction Integer,y,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesFunctionsPageEmpty5}
\begin{paste}{ugxProblemSeriesFunctionsPageEmpty5}{ugxProblemSeriesFunctionsPagePatch5}
\pastebutton{ugxProblemSeriesFunctionsPageEmpty5}{\showpaste}
\tab{5}\spadcommand{exp(y)\free{y }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesFunctionsPagePatch6}
\begin{paste}{ugxProblemSeriesFunctionsPageFull6}{ugxProblemSeriesFunctionsPageEmpty6}
\pastebutton{ugxProblemSeriesFunctionsPageFull6}{\hidepaste}
\tab{5}\spadcommand{tan(y**2)\free{y }}
\indentrel{3}\begin{verbatim}
        2   1  6    2  10      11
   (6)  y  +  y  +  y   + O(y  )
            3      15
     Type: UnivariateTaylorSeries(Fraction Integer,y,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesFunctionsPageEmpty6}
\begin{paste}{ugxProblemSeriesFunctionsPageEmpty6}{ugxProblemSeriesFunctionsPagePatch6}
\pastebutton{ugxProblemSeriesFunctionsPageEmpty6}{\showpaste}
\tab{5}\spadcommand{tan(y**2)\free{y }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesFunctionsPagePatch7}
\begin{paste}{ugxProblemSeriesFunctionsPageFull7}{ugxProblemSeriesFunctionsPageEmpty7}
\pastebutton{ugxProblemSeriesFunctionsPageFull7}{\hidepaste}
\tab{5}\spadcommand{cos(y + y**5)\free{y }}
\indentrel{3}\begin{verbatim}
   (7)
        1  2    1  4   721  6    6721  8   1844641  10
     1 -  y  +  y  -  y  +  y  -  y
        2      24      720      40320      3628800
   +
```

```
        11
     O(y  )
     Type: UnivariateTaylorSeries(Fraction Integer,y,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesFunctionsPageEmpty7}
\begin{paste}{ugxProblemSeriesFunctionsPageEmpty7}{ugxProblemSeriesFunctionsPagePatch7}
\pastebutton{ugxProblemSeriesFunctionsPageEmpty7}{\showpaste}
\tab{5}\spadcommand{cos(y + y**5)\free{y }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesFunctionsPagePatch8}
\begin{paste}{ugxProblemSeriesFunctionsPageFull8}{ugxProblemSeriesFunctionsPageEmpty8}
\pastebutton{ugxProblemSeriesFunctionsPageFull8}{\hidepaste}
\tab{5}\spadcommand{log(1 + sin(y))\free{y }}
\indentrel{3}\begin{verbatim}
   (8)
         1  2   1  3    1  4    1  5    1  6    61   7
     y -  y  +  y  -  y  +  y  -  y  +  y
         2      6      12      24      45      5040
   +
        17   8    277   9     31   10      11
     -  y  +  y  -  y   + O(y  )
       2520      72576      14175
     Type: UnivariateTaylorSeries(Fraction Integer,y,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesFunctionsPageEmpty8}
\begin{paste}{ugxProblemSeriesFunctionsPageEmpty8}{ugxProblemSeriesFunctionsPagePatch8}
\pastebutton{ugxProblemSeriesFunctionsPageEmpty8}{\showpaste}
\tab{5}\spadcommand{log(1 + sin(y))\free{y }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesFunctionsPagePatch9}
\begin{paste}{ugxProblemSeriesFunctionsPageFull9}{ugxProblemSeriesFunctionsPageEmpty9}
\pastebutton{ugxProblemSeriesFunctionsPageFull9}{\hidepaste}
\tab{5}\spadcommand{z : UTS(EXPR INT,z,0) := z\bound{z }}
\indentrel{3}\begin{verbatim}
   (9)  z
   Type: UnivariateTaylorSeries(Expression Integer,z,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesFunctionsPageEmpty9}
\begin{paste}{ugxProblemSeriesFunctionsPageEmpty9}{ugxProblemSeriesFunctionsPagePatch9}
\pastebutton{ugxProblemSeriesFunctionsPageEmpty9}{\showpaste}
\tab{5}\spadcommand{z : UTS(EXPR INT,z,0) := z\bound{z }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugxProblemSeriesFunctionsPagePatch10}
\begin{paste}{ugxProblemSeriesFunctionsPageFull10}{ugxProblemSeriesFunctionsPageEmpty10}
\pastebutton{ugxProblemSeriesFunctionsPageFull10}{\hidepaste}
\tab{5}\spadcommand{exp(2 + tan(z))\free{z }}
\indentrel{3}\begin{verbatim}
   (10)
                  2        2         2          2
      2     2    %e   2   %e   3   3%e   4   37%e   5
    %e  + %e z +  z  +  z  +  z  +  z
                 2        2         8        120
   +
       2           2           2             2
    59%e   6   137%e   7   871%e   8   41641%e   9
     z  +  z  +  z  +  z
     240         720        5760        362880
   +
           2
    325249%e   10      11
     z   + O(z  )
     3628800
   Type: UnivariateTaylorSeries(Expression Integer,z,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesFunctionsPageEmpty10}
\begin{paste}{ugxProblemSeriesFunctionsPageEmpty10}{ugxProblemSeriesFunctionsPagePatch10}
\pastebutton{ugxProblemSeriesFunctionsPageEmpty10}{\showpaste}
\tab{5}\spadcommand{exp(2 + tan(z))\free{z }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesFunctionsPagePatch11}
\begin{paste}{ugxProblemSeriesFunctionsPageFull11}{ugxProblemSeriesFunctionsPageEmpty11}
\pastebutton{ugxProblemSeriesFunctionsPageFull11}{\hidepaste}
\tab{5}\spadcommand{w := taylor 'w\bound{w }}
\indentrel{3}\begin{verbatim}
   (11)  w
   Type: UnivariateTaylorSeries(Expression Integer,w,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesFunctionsPageEmpty11}
\begin{paste}{ugxProblemSeriesFunctionsPageEmpty11}{ugxProblemSeriesFunctionsPagePatch11}
\pastebutton{ugxProblemSeriesFunctionsPageEmpty11}{\showpaste}
\tab{5}\spadcommand{w := taylor 'w\bound{w }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesFunctionsPagePatch12}
\begin{paste}{ugxProblemSeriesFunctionsPageFull12}{ugxProblemSeriesFunctionsPageEmpty12}
\pastebutton{ugxProblemSeriesFunctionsPageFull12}{\hidepaste}
```

```
\tab{5}\spadcommand{exp(2 + tan(w))\free{w }}
\indentrel{3}\begin{verbatim}
   (12)
                    2        2         2          2
      2     2     %e   2   %e   3   3%e   4   37%e   5
     %e  + %e w +  w  +  w  +  w  +  w
                   2        2         8        120
   +
        2           2           2             2
     59%e   6   137%e   7   871%e   8   41641%e   9
      w  +  w  +  w  +  w
      240         720        5760        362880
   +
             2
     325249%e   10      11
      w   + O(w  )
      3628800
   Type: UnivariateTaylorSeries(Expression Integer,w,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesFunctionsPageEmpty12}
\begin{paste}{ugxProblemSeriesFunctionsPageEmpty12}{ugxProblemSeriesFunctionsPagePatch12}
\pastebutton{ugxProblemSeriesFunctionsPageEmpty12}{\showpaste}
\tab{5}\spadcommand{exp(2 + tan(w))\free{w }}
\end{paste}\end{patch}
```

———-

Converting to Power Series

— ug08.ht —

```
\begin{page}{ugxProblemSeriesConversionsPage}
{8.9.5. Converting to Power Series}
\beginscroll

The \axiomType{ExpressionToUnivariatePowerSeries} package provides
operations for computing series expansions of functions.

\labelSpace{1pc}
\xtc{
Evaluate this
to compute the Taylor expansion of \axiom{sin x} about
\axiom{x = 0}.
The first argument, \axiom{sin(x)}, specifies the function whose series
expansion is to be computed and the second argument, \axiom{x = 0},
```

```
specifies that the series is to be expanded in power of \axiom{(x - 0)},
that is, in power of \axiom{x}.
}{
\spadpaste{taylor(sin(x),x = 0)}
}
\xtc{
Here is the Taylor expansion of \axiom{sin x} about
\texht{$x = \frac{\pi}{6}$}{\axiom{x = \%pi/6}}:
}{
\spadpaste{taylor(sin(x),x = \%pi/6)}
}

The function to be expanded into a series may have variables other than
the series variable.
%
\xtc{
For example, we may expand \axiom{tan(x*y)} as a Taylor series in
\axiom{x}
}{
\spadpaste{taylor(tan(x*y),x = 0)}
}
%
\xtc{
or as a Taylor series in \axiom{y}.
}{
\spadpaste{taylor(tan(x*y),y = 0)}
}
\xtc{
A more interesting function is
\texht{${\displaystyle t e^{x t}} \over
{\displaystyle e^t - 1}$}{(t * \%e**(x*t))/(\%e**t - 1)}.
When we expand this function as a Taylor series in \axiom{t}
the \eth{\axiom{n}} order coefficient is the \eth{\axiom{n}} Bernoulli
polynomial
divided by \axiom{n!}.
}{
\spadpaste{bern := taylor(t*exp(x*t)/(exp(t) - 1),t = 0) \bound{bern}}
}
\xtc{
Therefore, this and the next expression
produce the same result.
}{
\spadpaste{factorial(6) * coefficient(bern,6) \free{bern}}
}
\xtc{
}{
\spadpaste{bernoulliB(6,x)}
}

Technically, a series with terms of negative degree is not considered to
```

```
be a Taylor series, but, rather, a
{\it Laurent series}.
If you try to compute a Taylor series expansion of
\texht{$\frac{x}{\log x}$}{x/log(x)}
at \axiom{x = 1} via \axiom{taylor(x/log(x),x = 1)}
you get an error message.
The reason is that the function has a {\it pole} at \axiom{x =
1}, meaning that
its series expansion about this point has terms of negative degree.
A series with finitely many terms of negative degree is called a Laurent
series.
\xtc{
You get the desired series expansion by issuing this.
}{
\spadpaste{laurent(x/log(x),x = 1)}
}

Similarly, a series with terms of fractional degree is neither a
Taylor series nor a Laurent series.  Such a series is called a {\it
Puiseux series}.  The expression \axiom{laurent(sqrt(sec(x)),x = 3 *
\%pi/2)} results in an error message because the series expansion
about this point has terms of fractional degree.

\xtc{
However, this command produces what you want.
}{
\spadpaste{puiseux(sqrt(sec(x)),x = 3 * \%pi/2)}
}

Finally, consider the case of functions that do not have Puiseux
expansions about certain points.  An example of this is
\texht{$x^x$}{\axiom{x^x}} about \axiom{x = 0}.
\axiom{puiseux(x**x,x=0)} produces an error message because of the
type of singularity of the function at \axiom{x = 0}.

\xtc{
The general function \axiomFun{series} can be used in this case.
Notice that the series returned is not, strictly speaking, a power series
because of the \axiom{log(x)} in the expansion.
}{
\spadpaste{series(x**x,x=0)}
}

\beginImportant
The operation \axiomFun{series} returns the most general type of
infinite series.
The user who is not interested in distinguishing
between various types of infinite series may wish to use this operation
exclusively.
\endImportant
```

```
\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxProblemSeriesConversionsPagePatch1}
\begin{paste}{ugxProblemSeriesConversionsPageFull1}{ugxProblemSeriesConversionsPageEmpty1}
\pastebutton{ugxProblemSeriesConversionsPageFull1}{\hidepaste}
\tab{5}\spadcommand{taylor(sin(x),x = 0)}
\indentrel{3}\begin{verbatim}
           1  3    1   5     1   7      1    9      11
   (1)  x -  x  +  x  -  x  +  x  + O(x  )
           6      120      5040      362880
   Type: UnivariateTaylorSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesConversionsPageEmpty1}
\begin{paste}{ugxProblemSeriesConversionsPageEmpty1}{ugxProblemSeriesConversionsPagePatch1}
\pastebutton{ugxProblemSeriesConversionsPageEmpty1}{\showpaste}
\tab{5}\spadcommand{taylor(sin(x),x = 0)}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesConversionsPagePatch2}
\begin{paste}{ugxProblemSeriesConversionsPageFull2}{ugxProblemSeriesConversionsPageEmpty2}
\pastebutton{ugxProblemSeriesConversionsPageFull2}{\hidepaste}
\tab{5}\spadcommand{taylor(sin(x),x = \%pi/6)}
\indentrel{3}\begin{verbatim}
   (2)

     1   \3       %pi    1      %pi 2   \3       %pi 3
      +  (x - ) -  (x - )  -  (x - )
     2     2        6     4       6       12        6
   +

      1      %pi 4   \3       %pi 5     1       %pi 6
      (x - )  +  (x - )  -  (x - )
     48       6       240       6      1440       6
   +

        \3       %pi 7     1        %pi 8
     -  (x - )  +  (x - )
       10080       6      80640       6
   +

      \3        %pi 9      1         %pi 10
      (x - )  -  (x - )
     725760       6      7257600       6
   +
            %pi 11
```

```
     O((x - )  )
             6
Type: UnivariateTaylorSeries(Expression Integer,x,pi/6)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesConversionsPageEmpty2}
\begin{paste}{ugxProblemSeriesConversionsPageEmpty2}{ugxProblemSeriesConversionsPagePatch2}
\pastebutton{ugxProblemSeriesConversionsPageEmpty2}{\showpaste}
\tab{5}\spadcommand{taylor(sin(x),x = \%pi/6)}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesConversionsPagePatch3}
\begin{paste}{ugxProblemSeriesConversionsPageFull3}{ugxProblemSeriesConversionsPageEmpty3}
\pastebutton{ugxProblemSeriesConversionsPageFull3}{\hidepaste}
\tab{5}\spadcommand{taylor(tan(x*y),x = 0)}
\indentrel{3}\begin{verbatim}
   (3)
          3         5          7          9
         y   3   2y   5   17y   7   62y   9       11
   y x +  x  +  x  +  x  +  x  + O(x  )
          3        15        315       2835
   Type: UnivariateTaylorSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesConversionsPageEmpty3}
\begin{paste}{ugxProblemSeriesConversionsPageEmpty3}{ugxProblemSeriesConversionsPagePatch3}
\pastebutton{ugxProblemSeriesConversionsPageEmpty3}{\showpaste}
\tab{5}\spadcommand{taylor(tan(x*y),x = 0)}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesConversionsPagePatch4}
\begin{paste}{ugxProblemSeriesConversionsPageFull4}{ugxProblemSeriesConversionsPageEmpty4}
\pastebutton{ugxProblemSeriesConversionsPageFull4}{\hidepaste}
\tab{5}\spadcommand{taylor(tan(x*y),y = 0)}
\indentrel{3}\begin{verbatim}
   (4)
          3         5          7          9
         x   3   2x   5   17x   7   62x   9       11
   x y +  y  +  y  +  y  +  y  + O(y  )
          3        15        315       2835
   Type: UnivariateTaylorSeries(Expression Integer,y,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesConversionsPageEmpty4}
\begin{paste}{ugxProblemSeriesConversionsPageEmpty4}{ugxProblemSeriesConversionsPagePatch4}
\pastebutton{ugxProblemSeriesConversionsPageEmpty4}{\showpaste}
\tab{5}\spadcommand{taylor(tan(x*y),y = 0)}
```

```
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesConversionsPagePatch5}
\begin{paste}{ugxProblemSeriesConversionsPageFull5}{ugxProblemSeriesConversionsPageEmpty5}
\pastebutton{ugxProblemSeriesConversionsPageFull5}{\hidepaste}
\tab{5}\spadcommand{bern := taylor(t*exp(x*t)/(exp(t) - 1),t = 0)\bound{bern }}
\indentrel{3}\begin{verbatim}
   (5)
                    2                  3     2
         2x - 1     6x  - 6x + 1  2   2x  - 3x  + x  3
     1 +  t +  t  +  t
           2           12                  12
   +
        4      3      2             5      4      3
     30x  - 60x  + 30x  - 1  4   6x  - 15x  + 10x  - x  5
      t  +  t
             720                          720
   +
        6       5       4      2
     42x  - 126x  + 105x  - 21x  + 1  6
      t
                30240
   +
       7      6      5     3
     6x  - 21x  + 21x  - 7x  + x  7
      t
              30240
   +
        8       7       6      4      2
     30x  - 120x  + 140x  - 70x  + 20x  - 1  8
      t
                   1209600
   +
        9      8      7      5      3
     10x  - 45x  + 60x  - 42x  + 20x  - 3x  9
      t
                  3628800
   +
        10       9       8       6       4      2
     66x   - 330x  + 495x  - 462x  + 330x  - 99x  + 5  10
      t
                       239500800
   +
        11
     O(t  )
   Type: UnivariateTaylorSeries(Expression Integer,t,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesConversionsPageEmpty5}
```

```
\begin{paste}{ugxProblemSeriesConversionsPageEmpty5}{ugxProblemSeriesConversionsPagePatch5}
\pastebutton{ugxProblemSeriesConversionsPageEmpty5}{\showpaste}
\tab{5}\spadcommand{bern := taylor(t*exp(x*t)/(exp(t) - 1),t = 0)\bound{bern }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesConversionsPagePatch6}
\begin{paste}{ugxProblemSeriesConversionsPageFull6}{ugxProblemSeriesConversionsPageEmpty6}
\pastebutton{ugxProblemSeriesConversionsPageFull6}{\hidepaste}
\tab{5}\spadcommand{factorial(6) * coefficient(bern,6)\free{bern }}
\indentrel{3}\begin{verbatim}
           6       5       4      2
        42x  - 126x  + 105x  - 21x  + 1
   (6)
                      42
                              Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesConversionsPageEmpty6}
\begin{paste}{ugxProblemSeriesConversionsPageEmpty6}{ugxProblemSeriesConversionsPagePatch6}
\pastebutton{ugxProblemSeriesConversionsPageEmpty6}{\showpaste}
\tab{5}\spadcommand{factorial(6) * coefficient(bern,6)\free{bern }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesConversionsPagePatch7}
\begin{paste}{ugxProblemSeriesConversionsPageFull7}{ugxProblemSeriesConversionsPageEmpty7}
\pastebutton{ugxProblemSeriesConversionsPageFull7}{\hidepaste}
\tab{5}\spadcommand{bernoulliB(6,x)}
\indentrel{3}\begin{verbatim}
         6     5   5  4   1  2    1
   (7)  x  - 3x  +  x  -  x  +
                   2      2     42
                     Type: Polynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesConversionsPageEmpty7}
\begin{paste}{ugxProblemSeriesConversionsPageEmpty7}{ugxProblemSeriesConversionsPagePatch7}
\pastebutton{ugxProblemSeriesConversionsPageEmpty7}{\showpaste}
\tab{5}\spadcommand{bernoulliB(6,x)}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesConversionsPagePatch8}
\begin{paste}{ugxProblemSeriesConversionsPageFull8}{ugxProblemSeriesConversionsPageEmpty8}
\pastebutton{ugxProblemSeriesConversionsPageFull8}{\hidepaste}
\tab{5}\spadcommand{laurent(x/log(x),x = 1)}
\indentrel{3}\begin{verbatim}
   (8)
            - 1   3    5            1        2
     (x - 1)    +  +  (x - 1) -  (x - 1)
```

```
              2   12          24
  +
     11        3    11         4    271         5
     (x - 1)  -  (x - 1)  +  (x - 1)
    720           1440          60480
  +
      13         6     7297         7     425         8
    -  (x - 1)  +  (x - 1)  -  (x - 1)
      4480           3628800           290304
  +
      530113         9           10
     (x - 1)  + O((x - 1)  )
    479001600
 Type: UnivariateLaurentSeries(Expression Integer,x,1)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesConversionsPageEmpty8}
\begin{paste}{ugxProblemSeriesConversionsPageEmpty8}{ugxProblemSeriesConversionsPagePatch8}
\pastebutton{ugxProblemSeriesConversionsPageEmpty8}{\showpaste}
\tab{5}\spadcommand{laurent(x/log(x),x = 1)}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesConversionsPagePatch9}
\begin{paste}{ugxProblemSeriesConversionsPageFull9}{ugxProblemSeriesConversionsPageEmpty9}
\pastebutton{ugxProblemSeriesConversionsPageFull9}{\hidepaste}
\tab{5}\spadcommand{puiseux(sqrt(sec(x)),x = 3 * \%pi/2)}
\indentrel{3}\begin{verbatim}
   (9)
             1               3                7
             -
        3%pi   2    1      3%pi 2    1       3%pi 2
    (x - )    +  (x - )  +  (x - )
          2        12        2      160        2
  +
          3%pi 5
    O((x - ) )
            2
Type: UnivariatePuiseuxSeries(Expression Integer,x,(3*pi)/2)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesConversionsPageEmpty9}
\begin{paste}{ugxProblemSeriesConversionsPageEmpty9}{ugxProblemSeriesConversionsPagePatch9}
\pastebutton{ugxProblemSeriesConversionsPageEmpty9}{\showpaste}
\tab{5}\spadcommand{puiseux(sqrt(sec(x)),x = 3 * \%pi/2)}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesConversionsPagePatch10}
\begin{paste}{ugxProblemSeriesConversionsPageFull10}{ugxProblemSeriesConversionsPageEmpty10}
```

```
\pastebutton{ugxProblemSeriesConversionsPageFull10}{\hidepaste}
\tab{5}\spadcommand{series(x**x,x=0)}
\indentrel{3}\begin{verbatim}
   (10)
                         2             3             4
                   log(x)    2   log(x)    3   log(x)    4
     1 + log(x)x +   x   +   x   +   x
                        2             6            24
   +
         5             6             7             8
     log(x)    5   log(x)    6   log(x)    7   log(x)    8
      x  +  x  +  x  +  x
       120          720           5040         40320
   +
         9             10
     log(x)    9   log(x)    10      11
      x  +  x    + O(x  )
      362880       3628800
Type: GeneralUnivariatePowerSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesConversionsPageEmpty10}
\begin{paste}{ugxProblemSeriesConversionsPageEmpty10}{ugxProblemSeriesConversionsPagePatch10}
\pastebutton{ugxProblemSeriesConversionsPageEmpty10}{\showpaste}
\tab{5}\spadcommand{series(x**x,x=0)}
\end{paste}\end{patch}
```

———

Power Series from Formulas

⇒ "notitle" (ugxProblemSeriesConversionsPage) 12 on page 2186
⇒ "notitle" (ugUserAnonPage) 10 on page 1921
⇒ "notitle" (ugxProblemSeriesConversionsPage) 12 on page 2186

— **ug08.ht** —

```
\begin{page}{ugxProblemSeriesFormulaPage}
{8.9.6. Power Series from Formulas}
\beginscroll

The \axiomType{GenerateUnivariatePowerSeries} package enables you to
create power series from explicit formulas for their \eth{\axiom{n}}
coefficients.  In what follows, we construct series expansions for
certain transcendental functions by giving formulas for their
coefficients.  You can also compute such series expansions directly
```

```
simply by specifying the function and the point about which the series
is to be expanded.  See
\downlink{``Converting to Power Series''}
{ugxProblemSeriesConversionsPage} in Section
8.9.5\ignore{ugxProblemSeriesConversions}
for more information.

Consider the Taylor expansion of \texht{$e^x$}{\axiom{\%e**x}}
about \axiom{x = 0}:
\texht{\narrowDisplay{%
\begin{array}{ccl}
e^x &=& \displaystyle 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \cdots \\ \\
    &=& \displaystyle\sum_{n=0}^\infty \frac{x^n}{n!}
\end{array}}%
}{
\begin{verbatim}
%e**x = 1 + x + x**2/2 + x**3/6 + ...
      = sum from n=0 to n=%infty of x**n / n!
\end{verbatim}
}
The \eth{\axiom{n}} Taylor coefficient is \axiom{1/n!}.
%
\xtc{
This is how you create this series in Axiom.
}{
\spadpaste{series(n +-> 1/factorial(n),x = 0)}
}

The first argument specifies a formula for the \eth{\axiom{n}}
coefficient by giving a function that maps \axiom{n} to \axiom{1/n!}.
The second argument specifies that the series is to be expanded in
powers of \axiom{(x - 0)}, that is, in powers of \axiom{x}.  Since we
did not specify an initial degree, the first term in the series was
the term of degree 0 (the constant term).  Note that the formula was
given as an anonymous function.  These are discussed in
\downlink{``Anonymous Functions''}{ugUserAnonPage} in Section
6.17\ignore{ugUserAnon}.

Consider the Taylor expansion of \axiom{log x} about \axiom{x = 1}:
\texht{\narrowDisplay{%
\begin{array}{ccl}
\log(x) &=& \displaystyle (x - 1) - \frac{(x - 1)^2}{2} +
\frac{(x - 1)^3}{3} - \cdots \\ \\
        &=& \displaystyle\sum_{n = 1}^\infty (-1)^{n-1} \frac{(x - 1)^n}{n}
\end{array}}%
}{
\begin{verbatim}
log x = (x - 1) - \(x - 1)^2/2 + (x - 1)^3/3 - ...
      = sum from n=1 to n=%infty of (-1)**(n-1) (x - 1)**n / n
\end{verbatim}
```

```
}
If you were to evaluate the expression
\axiom{series(n +-> (-1)**(n-1) / n, x = 1)}
you would get an error message because Axiom would try to
calculate a term of degree \axiom{0} and therefore divide by \axiom{0.}

\xtc{
Instead, evaluate this.
The third argument, \axiom{1..}, indicates that only terms of degree
\axiom{n = 1, ...} are to be computed.
}{
\spadpaste{series(n +-> (-1)**(n-1)/n,x = 1,1..)}
}
%

Next consider the Taylor expansion of an odd function, say,
\axiom{sin(x)}:
\begin{verbatim}
sin x = x - x**3/3! + x**5/5! - ...
\end{verbatim}
Here every other coefficient is zero and we would like to give an
explicit formula only for the odd Taylor coefficients.
%
\xtc{
This is one way to do it.  The third argument, \axiom{1..}, specifies
that the first term to be computed is the term of degree 1.  The
fourth argument, \axiom{2}, specifies that we increment by \axiom{2}
to find the degrees of subsequent terms, that is, the next term is of
degree \axiom{1 + 2}, the next of degree \axiom{1 + 2 + 2}, etc.
}{
\spadpaste{series(n +-> (-1)**((n-1)/2)/factorial(n),x = 0,1..,2)}
}
%

\xtc{
The initial degree and the increment do not have to be integers.
For example, this expression produces a series expansion of
\texht{$\sin(x^{\frac{1}{3}})$}{\axiom{sin(x**(1/3))}}.
}{
\spadpaste{series(n +-> (-1)**((3*n-1)/2)/factorial(3*n),x = 0,1/3..,2/3)}
}
\xtc{
While the increment must be positive, the initial degree may be negative.
This yields the Laurent expansion of \axiom{csc(x)} at
\axiom{x = 0}.
%
}{
\spadpaste{cscx :=
series(n +-> (-1)**((n-1)/2) * 2 * (2**n-1) * bernoulli(numer(n+1))
/ factorial(n+1), x=0, -1..,2) \bound{cscx}}
```

```
}
\xtc{
Of course, the reciprocal of this power series is the Taylor expansion
of \axiom{sin(x)}.
}{
\spadpaste{1/cscx \free{cscx}}
}
%
\xtc{
As a final example,
here is the Taylor expansion of \axiom{asin(x)} about \axiom{x = 0}.
}{
\spadpaste{asinx := series(n +-> binomial(n-1,(n-1)/2)/(n*2**(n-1)),
x=0,1..,2) \bound{asinx}}
}
\xtc{
When we compute the \axiom{sin} of this series, we get \axiom{x}
(in the sense that all higher terms computed so far are zero).
}{
\spadpaste{sin(asinx) \free{asinx}}
}

As we discussed in
\downlink{``Converting to Power Series''}
{ugxProblemSeriesConversionsPage} in Section
8.9.5\ignore{ugxProblemSeriesConversions},
you can also use the operations \axiomFun{taylor}, \axiomFun{laurent} and
\axiomFun{puiseux} instead of \axiomFun{series} if you know ahead of time
what kind of exponents a series has.
You can't go wrong using \axiomFun{series}, though.

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxProblemSeriesFormulaPagePatch1}
\begin{paste}{ugxProblemSeriesFormulaPageFull1}{ugxProblemSeriesFormulaPageEmpty1}
\pastebutton{ugxProblemSeriesFormulaPageFull1}{\hidepaste}
\tab{5}\spadcommand{series(n +-> 1/factorial(n),x = 0)}
\indentrel{3}\begin{verbatim}
   (1)
             1  2   1  3    1  4    1   5    1   6
     1 + x + x  +  x  +  x  +  x  +  x
             2      6      24      120      720
   +
     1   7     1    8      1    9      1     10      11
    x  +  x  +  x  +  x   + O(x  )
   5040      40320      362880      3628800
  Type: UnivariatePuiseuxSeries(Expression Integer,x,0)
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesFormulaPageEmpty1}
\begin{paste}{ugxProblemSeriesFormulaPageEmpty1}{ugxProblemSeriesFormulaPagePatch1}
\pastebutton{ugxProblemSeriesFormulaPageEmpty1}{\showpaste}
\tab{5}\spadcommand{series(n +-> 1/factorial(n),x = 0)}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesFormulaPagePatch2}
\begin{paste}{ugxProblemSeriesFormulaPageFull2}{ugxProblemSeriesFormulaPageEmpty2}
\pastebutton{ugxProblemSeriesFormulaPageFull2}{\hidepaste}
\tab{5}\spadcommand{series(n +-> (-1)**(n-1)/n,x = 1,1..)}
\indentrel{3}\begin{verbatim}
   (2)
               1        2   1        3   1        4
     (x - 1) -  (x - 1)  +  (x - 1)  -  (x - 1)
               2            3            4
   +
     1        5   1        6   1        7   1        8
      (x - 1)  -  (x - 1)  +  (x - 1)  -  (x - 1)
     5            6            7            8
   +
   1        9    1        10    1        11            12
    (x - 1)  -  (x - 1)   +  (x - 1)   + O((x - 1)  )
   9           10            11
  Type: UnivariatePuiseuxSeries(Expression Integer,x,1)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesFormulaPageEmpty2}
\begin{paste}{ugxProblemSeriesFormulaPageEmpty2}{ugxProblemSeriesFormulaPagePatch2}
\pastebutton{ugxProblemSeriesFormulaPageEmpty2}{\showpaste}
\tab{5}\spadcommand{series(n +-> (-1)**(n-1)/n,x = 1,1..)}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesFormulaPagePatch3}
\begin{paste}{ugxProblemSeriesFormulaPageFull3}{ugxProblemSeriesFormulaPageEmpty3}
\pastebutton{ugxProblemSeriesFormulaPageFull3}{\hidepaste}
\tab{5}\spadcommand{series(n +-> (-1)**((n-1)/2)/factorial(n),x = 0,1..,2)}
\indentrel{3}\begin{verbatim}
   (3)
         1  3    1   5     1   7      1    9
     x -  x  +  x  -  x  +  x
         6      120      5040      362880
   +
           1     11      12
     -  x   + O(x  )
       39916800
  Type: UnivariatePuiseuxSeries(Expression Integer,x,0)
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesFormulaPageEmpty3}
\begin{paste}{ugxProblemSeriesFormulaPageEmpty3}{ugxProblemSeriesFormulaPagePatch3}
\pastebutton{ugxProblemSeriesFormulaPageEmpty3}{\showpaste}
\tab{5}\spadcommand{series(n +-> (-1)**((n-1)/2)/factorial(n),x = 0,1..,2)}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesFormulaPagePatch4}
\begin{paste}{ugxProblemSeriesFormulaPageFull4}{ugxProblemSeriesFormulaPageEmpty4}
\pastebutton{ugxProblemSeriesFormulaPageFull4}{\hidepaste}
\tab{5}\spadcommand{series(n +-> (-1)**((3*n-1)/2)/factorial(3*n),x = 0,1/3..,2/3)}
\indentrel{3}\begin{verbatim}
   (4)
      1              5         7

      3   1      1   3     1   3      1    3
     x  -  x +  x  -  x  +  x
          6     120      5040      362880
   +
                 11

           1      3      4
     -  x   + O(x )
       39916800
  Type: UnivariatePuiseuxSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesFormulaPageEmpty4}
\begin{paste}{ugxProblemSeriesFormulaPageEmpty4}{ugxProblemSeriesFormulaPagePatch4}
\pastebutton{ugxProblemSeriesFormulaPageEmpty4}{\showpaste}
\tab{5}\spadcommand{series(n +-> (-1)**((3*n-1)/2)/factorial(3*n),x = 0,1/3..,2/3)}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesFormulaPagePatch5}
\begin{paste}{ugxProblemSeriesFormulaPageFull5}{ugxProblemSeriesFormulaPageEmpty5}
\pastebutton{ugxProblemSeriesFormulaPageFull5}{\hidepaste}
\tab{5}\spadcommand{cscx := series(n +-> (-1)**((n-1)/2) * 2 * (2**n-1) * bernoulli(numer(n+1)) / factorial
\indentrel{3}\begin{verbatim}
   (5)
      - 1   1      7   3     31   5     127   7
     x    +  x +  x  +  x  +  x
           6     360      15120      604800
   +
       73    9      10
      x  + O(x  )
     3421440
  Type: UnivariatePuiseuxSeries(Expression Integer,x,0)
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesFormulaPageEmpty5}
\begin{paste}{ugxProblemSeriesFormulaPageEmpty5}{ugxProblemSeriesFormulaPagePatch5}
\pastebutton{ugxProblemSeriesFormulaPageEmpty5}{\showpaste}
\tab{5}\spadcommand{cscx := series(n +-> (-1)**((n-1)/2) * 2 * (2**n-1) * bernoulli(numer(n+1))
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesFormulaPagePatch6}
\begin{paste}{ugxProblemSeriesFormulaPageFull6}{ugxProblemSeriesFormulaPageEmpty6}
\pastebutton{ugxProblemSeriesFormulaPageFull6}{\hidepaste}
\tab{5}\spadcommand{1/cscx\free{cscx }}
\indentrel{3}\begin{verbatim}
   (6)
         1  3    1   5     1   7      1    9
     x - x  +  x  -  x  +  x
         6      120      5040      362880
   +
           1     11      12
     -  x   + O(x  )
       39916800
  Type: UnivariatePuiseuxSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesFormulaPageEmpty6}
\begin{paste}{ugxProblemSeriesFormulaPageEmpty6}{ugxProblemSeriesFormulaPagePatch6}
\pastebutton{ugxProblemSeriesFormulaPageEmpty6}{\showpaste}
\tab{5}\spadcommand{1/cscx\free{cscx }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesFormulaPagePatch7}
\begin{paste}{ugxProblemSeriesFormulaPageFull7}{ugxProblemSeriesFormulaPageEmpty7}
\pastebutton{ugxProblemSeriesFormulaPageFull7}{\hidepaste}
\tab{5}\spadcommand{asinx := series(n +-> binomial(n-1,(n-1)/2)/(n*2**(n-1)),x=0,1..,2)\bound{as
\indentrel{3}\begin{verbatim}
   (7)
       1  3    3  5    5   7    35   9    63   11      12
   x +  x  +  x  +  x  +  x  +  x   + O(x  )
       6      40      112      1152      2816
  Type: UnivariatePuiseuxSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesFormulaPageEmpty7}
\begin{paste}{ugxProblemSeriesFormulaPageEmpty7}{ugxProblemSeriesFormulaPagePatch7}
\pastebutton{ugxProblemSeriesFormulaPageEmpty7}{\showpaste}
\tab{5}\spadcommand{asinx := series(n +-> binomial(n-1,(n-1)/2)/(n*2**(n-1)),x=0,1..,2)\bound{as
\end{paste}\end{patch}
```

```
\begin{patch}{ugxProblemSeriesFormulaPagePatch8}
\begin{paste}{ugxProblemSeriesFormulaPageFull8}{ugxProblemSeriesFormulaPageEmpty8}
\pastebutton{ugxProblemSeriesFormulaPageFull8}{\hidepaste}
\tab{5}\spadcommand{sin(asinx)\free{asinx }}
\indentrel{3}\begin{verbatim}
              12
   (8)  x + O(x  )
  Type: UnivariatePuiseuxSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesFormulaPageEmpty8}
\begin{paste}{ugxProblemSeriesFormulaPageEmpty8}{ugxProblemSeriesFormulaPagePatch8}
\pastebutton{ugxProblemSeriesFormulaPageEmpty8}{\showpaste}
\tab{5}\spadcommand{sin(asinx)\free{asinx }}
\end{paste}\end{patch}
```

Substituting Numerical Values in Power Series

— ug08.ht —

```
\begin{page}{ugxProblemSeriesSubstitutePage}
{8.9.7. Substituting Numerical Values in Power Series}
\beginscroll

Use \axiomFunFrom{eval}{UnivariatePowerSeriesCategory}
to substitute a numerical value for a variable in
a power series.
For example, here's a way to obtain numerical approximations of
\axiom{\%e} from the Taylor series
expansion of \axiom{exp(x)}.

\labelSpace{1pc}
\xtc{
First you create the desired Taylor expansion.
}{
\spadpaste{f := taylor(exp(x)) \bound{f}}
}
\xtc{
Then you evaluate the series at the value \axiom{1.0}.
The result is a sequence of the partial sums.
}{
\spadpaste{eval(f,1.0)}
}
```

```
\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxProblemSeriesSubstitutePagePatch1}
\begin{paste}{ugxProblemSeriesSubstitutePageFull1}{ugxProblemSeriesSubstitutePageEmpty1}
\pastebutton{ugxProblemSeriesSubstitutePageFull1}{\hidepaste}
\tab{5}\spadcommand{f := taylor(exp(x))\bound{f }}
\indentrel{3}\begin{verbatim}
   (1)
            1  2   1  3    1  4    1   5    1   6
     1 + x +  x  +  x  +  x  +  x  +  x
            2      6      24      120      720
   +
     1   7     1    8      1    9      1     10      11
    x  +  x  +  x  +  x   + O(x  )
   5040      40320      362880      3628800
   Type: UnivariateTaylorSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesSubstitutePageEmpty1}
\begin{paste}{ugxProblemSeriesSubstitutePageEmpty1}{ugxProblemSeriesSubstitutePagePatch1}
\pastebutton{ugxProblemSeriesSubstitutePageEmpty1}{\showpaste}
\tab{5}\spadcommand{f := taylor(exp(x))\bound{f }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesSubstitutePagePatch2}
\begin{paste}{ugxProblemSeriesSubstitutePageFull2}{ugxProblemSeriesSubstitutePageEmpty2}
\pastebutton{ugxProblemSeriesSubstitutePageFull2}{\hidepaste}
\tab{5}\spadcommand{eval(f,1.0)}
\indentrel{3}\begin{verbatim}
   (2)
   [1.0, 2.0, 2.5, 2.6666666666 666666667,
    2.7083333333 333333333, 2.7166666666 666666667,
    2.7180555555 555555556, 2.7182539682 53968254,
    2.7182787698 412698413, 2.7182815255 731922399, ...]
                          Type: Stream Expression Float
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesSubstitutePageEmpty2}
\begin{paste}{ugxProblemSeriesSubstitutePageEmpty2}{ugxProblemSeriesSubstitutePagePatch2}
\pastebutton{ugxProblemSeriesSubstitutePageEmpty2}{\showpaste}
\tab{5}\spadcommand{eval(f,1.0)}
\end{paste}\end{patch}
```

Example: Bernoulli Polynomials and Sums of Powers

— ug08.ht —

```
\begin{page}{ugxProblemSeriesBernoulliPage}
{8.9.8. Example: Bernoulli Polynomials and Sums of Powers}
\beginscroll

Axiom provides operations for computing definite and
indefinite sums.

\labelSpace{3pc}
\xtc{
You can compute the sum of the first
ten fourth powers by evaluating this.
This creates a list whose entries are
\texht{$m^4$}{\axiom{m**4}} as \texht{$m$}{\axiom{m}} ranges from 1
to 10, and then computes the sum of the entries of that list.
}{
\spadpaste{reduce(+,[m**4 for m in 1..10])}
}
\xtc{
You can also compute a formula for the sum of the first
\texht{$k$}{\axiom{k}} fourth powers, where \texht{$k$}{\axiom{k}} is an
unspecified positive integer.
}{
\spadpaste{sum4 := sum(m**4, m = 1..k) \bound{sum4}}
}
\xtc{
This formula is valid for any positive integer \texht{$k$}{\axiom{k}}.
For instance, if we replace \texht{$k$}{\axiom{k}} by 10,
we obtain the number we computed earlier.
}{
\spadpaste{eval(sum4, k = 10) \free{sum4}}
}

You can compute a formula for the sum of the first
\texht{$k$}{\axiom{k}} \eth{\axiom{n}} powers in a similar fashion.
Just replace the \axiom{4} in the definition of \userfun{sum4} by
any expression not involving \texht{$k$}{\axiom{k}}.
Axiom computes these formulas using Bernoulli polynomials;
we
use the rest of this section to describe this method.

%
\xtc{
First consider this function of \axiom{t} and \axiom{x}.
}{
\spadpaste{f := t*exp(x*t) / (exp(t) - 1) \bound{f}}
```

```
}
\noOutputXtc{
Since the expressions involved get quite large, we tell
Axiom to show us only terms of degree up to \axiom{5.}
}{
\spadpaste{)set streams calculate 5 \bound{set}}
}
%
%
\xtc{
If we look at the Taylor expansion of \axiom{f(x, t)} about \axiom{t = 0,}
we see that the coefficients of the powers of \axiom{t} are polynomials
in \axiom{x}.
}{
\spadpaste{ff := taylor(f,t = 0) \free{f set} \bound{ff}}
}

In fact, the \eth{\axiom{n}} coefficient in this series is essentially
the \eth{\axiom{n}} Bernoulli polynomial: the \eth{\axiom{n}}
coefficient of the series is
\texht{${1 \over {n!}} B_n(x)$}{\axiom{1/n! * Bn(x)}},
where \texht{$B_n(x)$}{\axiom{Bn(x)}}
is the \eth{\axiom{n}} Bernoulli polynomial.  Thus, to obtain the
\eth{\axiom{n}} Bernoulli polynomial, we multiply the \eth{\axiom{n}}
coefficient of the series \axiom{ff} by \axiom{n!}.

\xtc{
For example, the sixth Bernoulli polynomial is this.
}{
\spadpaste{factorial(6) * coefficient(ff,6) \free{ff}}
}
%
\xtc{
We derive some properties of the function \axiom{f(x,t)}.
First we compute \axiom{f(x + 1,t) - f(x,t)}.
}{
\spadpaste{g := eval(f, x = x + 1) - f \bound{g} \free{f}}
}
%
\xtc{
If we normalize \axiom{g}, we see that it has a particularly simple form.
}{
\spadpaste{normalize(g) \free{g}}
}
%
From this it follows that the \eth{\axiom{n}}
coefficient in the Taylor expansion of
\axiom{g(x,t)} at \axiom{t = 0} is
\texht{${1\over{(n-1)\:!}}\:x^{n-1}$}{\axiom{1/(n-1)! * x**(n-1)}}.
\xtc{
```

```
If you want to check this, evaluate the next expression.
}{
\spadpaste{taylor(g,t = 0) \free{g}}
}
%
However, since \axiom{g(x,t) = f(x+1,t)-f(x,t)}, it follows that the
\eth{\axiom{n}} coefficient is
\texht{${1 \over {n!}}\:(B_n(x+1)-B_n(x))$}{\axiom{1/n! * (Bn(x + 1) -
Bn(x))}}.
Equating coefficients, we see that
\texht{${1\over{(n-1)\:!}}\:x^{n-1} = {1\over{n!}}\:(B_n(x + 1) -
B_n(x))$}{\axiom{1/(n-1)! * x**(n-1) = 1/n! * (Bn(x + 1) - Bn(x))}}
and, therefore,
\texht{$x^{n-1} = {1\over{n}}\:(B_n(x + 1) -
B_n(x))$}{\axiom{x**(n-1) = 1/n * (Bn(x + 1) - Bn(x))}}.
%
Let's apply this formula repeatedly, letting \axiom{x} vary between two
integers \axiom{a} and \axiom{b}, with \axiom{a < b}:
%
\begin{verbatim}
  a**(n-1)       = 1/n * (Bn(a + 1) - Bn(a))
  (a + 1)**(n-1) = 1/n * (Bn(a + 2) - Bn(a + 1))
  (a + 2)**(n-1) = 1/n * (Bn(a + 3) - Bn(a + 2))
     .
     .
     .
  (b - 1)**(n-1) = 1/n * (Bn(b) - Bn(b - 1))
  b**(n-1)       = 1/n * (Bn(b + 1) - Bn(b))
\end{verbatim}

When we add these equations we find that
the sum of the left-hand sides is
\texht{$\sum_{m=a}^{b} m^{n-1},$}{\axiom{sum(m = a..b,m ** (n-1))},}%
the sum of the
\texht{$(n-1)^{\hbox{\small\rm st}}$}{\axiom{(n-1)}-st}
powers from \axiom{a} to \axiom{b}.
The sum of the right-hand sides is a ``telescoping series.''
After cancellation, the sum is simply
\texht{${1\over{n}}\:(B_n(b + 1) -
B_n(a))$}{\axiom{1/n * (Bn(b + 1) - Bn(a))}}.

Replacing \axiom{n} by \axiom{n + 1}, we have shown that
\centerline{{\axiom{sum(m = a..b,m ** n) = 1/(n + 1) *
(B<n+1>(b + 1) - B<n+1>(a))}.}}

Let's use this to obtain the formula for the sum of fourth powers.
\xtc{
First we obtain the Bernoulli polynomial \texht{$B_5$}{\axiom{B5}}.
}{
\spadpaste{B5 := factorial(5) * coefficient(ff,5) \free{ff} \bound{B5}}
```

```
}
%
\xtc{
To find the sum of the first \texht{$k$}{\axiom{k}} 4th powers,
we multiply \axiom{1/5} by
\texht{$B_5(k+1) - B_5(1)$}{\axiom{B5(k + 1) - B5(1)}}.
}{
\spadpaste{1/5 * (eval(B5, x = k + 1) - eval(B5, x = 1)) \free{B5}}
}
%
\xtc{
This is the same formula that we obtained via \axiom{sum(m**4, m = 1..k)}.
}{
\spadpaste{sum4 \free{sum4}}
}

At this point you may want to do the same computation, but with an
exponent other than \axiom{4.}
For example, you might try to find a formula for the sum of the
first \texht{$k$}{\axiom{k}} 20th powers.

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxProblemSeriesBernoulliPagePatch1}
\begin{paste}{ugxProblemSeriesBernoulliPageFull1}{ugxProblemSeriesBernoulliPageEmpty1}
\pastebutton{ugxProblemSeriesBernoulliPageFull1}{\hidepaste}
\tab{5}\spadcommand{reduce(+,[m**4 for m in 1..10])}
\indentrel{3}\begin{verbatim}
   (1)  25333
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesBernoulliPageEmpty1}
\begin{paste}{ugxProblemSeriesBernoulliPageEmpty1}{ugxProblemSeriesBernoulliPagePatch1}
\pastebutton{ugxProblemSeriesBernoulliPageEmpty1}{\showpaste}
\tab{5}\spadcommand{reduce(+,[m**4 for m in 1..10])}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesBernoulliPagePatch2}
\begin{paste}{ugxProblemSeriesBernoulliPageFull2}{ugxProblemSeriesBernoulliPageEmpty2}
\pastebutton{ugxProblemSeriesBernoulliPageFull2}{\hidepaste}
\tab{5}\spadcommand{sum4 := sum(m**4, m = 1..k)\bound{sum4 }}
\indentrel{3}\begin{verbatim}
          5      4      3
        6k  + 15k  + 10k  - k
   (2)
                  30
```

```
                      Type: Fraction Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesBernoulliPageEmpty2}
\begin{paste}{ugxProblemSeriesBernoulliPageEmpty2}{ugxProblemSeriesBernoulliPagePatch2}
\pastebutton{ugxProblemSeriesBernoulliPageEmpty2}{\showpaste}
\tab{5}\spadcommand{sum4 := sum(m**4, m = 1..k)\bound{sum4 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesBernoulliPagePatch3}
\begin{paste}{ugxProblemSeriesBernoulliPageFull3}{ugxProblemSeriesBernoulliPageEmpty3}
\pastebutton{ugxProblemSeriesBernoulliPageFull3}{\hidepaste}
\tab{5}\spadcommand{eval(sum4, k = 10)\free{sum4 }}
\indentrel{3}\begin{verbatim}
   (3)  25333
                      Type: Fraction Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesBernoulliPageEmpty3}
\begin{paste}{ugxProblemSeriesBernoulliPageEmpty3}{ugxProblemSeriesBernoulliPagePatch3}
\pastebutton{ugxProblemSeriesBernoulliPageEmpty3}{\showpaste}
\tab{5}\spadcommand{eval(sum4, k = 10)\free{sum4 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesBernoulliPagePatch4}
\begin{paste}{ugxProblemSeriesBernoulliPageFull4}{ugxProblemSeriesBernoulliPageEmpty4}
\pastebutton{ugxProblemSeriesBernoulliPageFull4}{\hidepaste}
\tab{5}\spadcommand{f := t*exp(x*t) / (exp(t) - 1)\bound{f }}
\indentrel{3}\begin{verbatim}
            t x
        t %e
   (4)
          t
        %e  - 1
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesBernoulliPageEmpty4}
\begin{paste}{ugxProblemSeriesBernoulliPageEmpty4}{ugxProblemSeriesBernoulliPagePatch4}
\pastebutton{ugxProblemSeriesBernoulliPageEmpty4}{\showpaste}
\tab{5}\spadcommand{f := t*exp(x*t) / (exp(t) - 1)\bound{f }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesBernoulliPagePatch5}
\begin{paste}{ugxProblemSeriesBernoulliPageFull5}{ugxProblemSeriesBernoulliPageEmpty5}
\pastebutton{ugxProblemSeriesBernoulliPageFull5}{\hidepaste}
\tab{5}\spadcommand{)set streams calculate 5\bound{set }}
```

```
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesBernoulliPageEmpty5}
\begin{paste}{ugxProblemSeriesBernoulliPageEmpty5}{ugxProblemSeriesBernoulliPagePatch5}
\pastebutton{ugxProblemSeriesBernoulliPageEmpty5}{\showpaste}
\tab{5}\spadcommand{)set streams calculate 5\bound{set }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesBernoulliPagePatch6}
\begin{paste}{ugxProblemSeriesBernoulliPageFull6}{ugxProblemSeriesBernoulliPageEmpty6}
\pastebutton{ugxProblemSeriesBernoulliPageFull6}{\hidepaste}
\tab{5}\spadcommand{ff := taylor(f,t = 0)\free{f set }\bound{ff }}
\indentrel{3}\begin{verbatim}
   (5)
                     2                 3     2
         2x - 1     6x  - 6x + 1  2   2x  - 3x  + x  3
     1 +  t +  t  +  t
           2           12                 12
   +
        4      3      2            5      4      3
     30x  - 60x  + 30x  - 1  4   6x  - 15x  + 10x  - x  5
      t  +  t
              720                        720
   +
        6
     O(t )
   Type: UnivariateTaylorSeries(Expression Integer,t,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesBernoulliPageEmpty6}
\begin{paste}{ugxProblemSeriesBernoulliPageEmpty6}{ugxProblemSeriesBernoulliPagePatch6}
\pastebutton{ugxProblemSeriesBernoulliPageEmpty6}{\showpaste}
\tab{5}\spadcommand{ff := taylor(f,t = 0)\free{f set }\bound{ff }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesBernoulliPagePatch7}
\begin{paste}{ugxProblemSeriesBernoulliPageFull7}{ugxProblemSeriesBernoulliPageEmpty7}
\pastebutton{ugxProblemSeriesBernoulliPageFull7}{\hidepaste}
\tab{5}\spadcommand{factorial(6) * coefficient(ff,6)\free{ff }}
\indentrel{3}\begin{verbatim}
           6       5       4      2
        42x  - 126x  + 105x  - 21x  + 1
   (6)
                        42
                                Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugxProblemSeriesBernoulliPageEmpty7}
\begin{paste}{ugxProblemSeriesBernoulliPageEmpty7}{ugxProblemSeriesBernoulliPagePatch7}
\pastebutton{ugxProblemSeriesBernoulliPageEmpty7}{\showpaste}
\tab{5}\spadcommand{factorial(6) * coefficient(ff,6)\free{ff }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesBernoulliPagePatch8}
\begin{paste}{ugxProblemSeriesBernoulliPageFull8}{ugxProblemSeriesBernoulliPageEmpty8}
\pastebutton{ugxProblemSeriesBernoulliPageFull8}{\hidepaste}
\tab{5}\spadcommand{g := eval(f, x = x + 1) - f\bound{g }\free{f }}
\indentrel{3}\begin{verbatim}
            t x + t       t x
        t %e        - t %e
   (7)
                 t
               %e  - 1
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesBernoulliPageEmpty8}
\begin{paste}{ugxProblemSeriesBernoulliPageEmpty8}{ugxProblemSeriesBernoulliPagePatch8}
\pastebutton{ugxProblemSeriesBernoulliPageEmpty8}{\showpaste}
\tab{5}\spadcommand{g := eval(f, x = x + 1) - f\bound{g }\free{f }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesBernoulliPagePatch9}
\begin{paste}{ugxProblemSeriesBernoulliPageFull9}{ugxProblemSeriesBernoulliPageEmpty9}
\pastebutton{ugxProblemSeriesBernoulliPageFull9}{\hidepaste}
\tab{5}\spadcommand{normalize(g)\free{g }}
\indentrel{3}\begin{verbatim}
            t x
   (8)  t %e
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesBernoulliPageEmpty9}
\begin{paste}{ugxProblemSeriesBernoulliPageEmpty9}{ugxProblemSeriesBernoulliPagePatch9}
\pastebutton{ugxProblemSeriesBernoulliPageEmpty9}{\showpaste}
\tab{5}\spadcommand{normalize(g)\free{g }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesBernoulliPagePatch10}
\begin{paste}{ugxProblemSeriesBernoulliPageFull10}{ugxProblemSeriesBernoulliPageEmpty10}
\pastebutton{ugxProblemSeriesBernoulliPageFull10}{\hidepaste}
\tab{5}\spadcommand{taylor(g,t = 0)\free{g }}
\indentrel{3}\begin{verbatim}
                    2       3       4
```

```
              2   x   3   x   4   x   5      6
   (9)  t + x t  +  t  +  t  +  t  + O(t )
                  2       6      24
   Type: UnivariateTaylorSeries(Expression Integer,t,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesBernoulliPageEmpty10}
\begin{paste}{ugxProblemSeriesBernoulliPageEmpty10}{ugxProblemSeriesBernoulliPagePatch10}
\pastebutton{ugxProblemSeriesBernoulliPageEmpty10}{\showpaste}
\tab{5}\spadcommand{taylor(g,t = 0)\free{g }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesBernoulliPagePatch11}
\begin{paste}{ugxProblemSeriesBernoulliPageFull11}{ugxProblemSeriesBernoulliPageEmpty11}
\pastebutton{ugxProblemSeriesBernoulliPageFull11}{\hidepaste}
\tab{5}\spadcommand{B5 := factorial(5) * coefficient(ff,5)\free{ff }\bound{B5 }}
\indentrel{3}\begin{verbatim}
          5      4      3
        6x  - 15x  + 10x  - x
   (10)
                  6
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesBernoulliPageEmpty11}
\begin{paste}{ugxProblemSeriesBernoulliPageEmpty11}{ugxProblemSeriesBernoulliPagePatch11}
\pastebutton{ugxProblemSeriesBernoulliPageEmpty11}{\showpaste}
\tab{5}\spadcommand{B5 := factorial(5) * coefficient(ff,5)\free{ff }\bound{B5 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesBernoulliPagePatch12}
\begin{paste}{ugxProblemSeriesBernoulliPageFull12}{ugxProblemSeriesBernoulliPageEmpty12}
\pastebutton{ugxProblemSeriesBernoulliPageFull12}{\hidepaste}
\tab{5}\spadcommand{1/5 * (eval(B5, x = k + 1) - eval(B5, x = 1))\free{B5 }}
\indentrel{3}\begin{verbatim}
          5      4      3
        6k  + 15k  + 10k  - k
   (11)
                  30
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesBernoulliPageEmpty12}
\begin{paste}{ugxProblemSeriesBernoulliPageEmpty12}{ugxProblemSeriesBernoulliPagePatch12}
\pastebutton{ugxProblemSeriesBernoulliPageEmpty12}{\showpaste}
\tab{5}\spadcommand{1/5 * (eval(B5, x = k + 1) - eval(B5, x = 1))\free{B5 }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugxProblemSeriesBernoulliPagePatch13}
\begin{paste}{ugxProblemSeriesBernoulliPageFull13}{ugxProblemSeriesBernoulliPageEmpty13}
\pastebutton{ugxProblemSeriesBernoulliPageFull13}{\hidepaste}
\tab{5}\spadcommand{sum4\free{sum4 }}
\indentrel{3}\begin{verbatim}
           5      4      3
         6k  + 15k  + 10k  - k
   (12)  ----------------------
                  30
                     Type: Fraction Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemSeriesBernoulliPageEmpty13}
\begin{paste}{ugxProblemSeriesBernoulliPageEmpty13}{ugxProblemSeriesBernoulliPagePatch13}
\pastebutton{ugxProblemSeriesBernoulliPageEmpty13}{\showpaste}
\tab{5}\spadcommand{sum4\free{sum4 }}
\end{paste}\end{patch}
```

———-

Solution of Differential Equations

⇒ "notitle" (ugProblemLinPolEqnPage) 12 on page 2130
⇒ "notitle" (ugxProblemLDEQClosedPage) 12 on page 2212
⇒ "notitle" (ugxProblemNLDEQClosedPage) 12 on page 2220
⇒ "notitle" (ugxProblemDEQSeriesPage) 12 on page 2230

— **ug08.ht** —

```
\begin{page}{ugProblemDEQPage}{8.10. Solution of Differential Equations}
\beginscroll
%
In this section we discuss Axiom's facilities for
solving
differential equations in closed-form and in series.

Axiom provides facilities for closed-form solution of
single differential equations of the following kinds:
\indent{4}
\beginitems
\item[-] linear ordinary differential equations, and
\item[-] non-linear first order ordinary differential equations
when integrating factors can be found just by integration.
\enditems
\indent{0}
```

```
For a discussion of the solution of systems of linear and polynomial
equations, see
\downlink{``Solution of Linear and Polynomial Equations''}
{ugProblemLinPolEqnPage} in Section 8.5\ignore{ugProblemLinPolEqn}.

\beginmenu
    \menudownlink{
{8.10.1. Closed-Form Solutions of Linear Differential Equations}}
{ugxProblemLDEQClosedPage}
    \menudownlink{
{8.10.2. Closed-Form Solutions of Non-Linear Differential Equations}}
{ugxProblemNLDEQClosedPage}
    \menudownlink{
{8.10.3. Power Series Solutions of Differential Equations}}
{ugxProblemDEQSeriesPage}
\endmenu
\endscroll
\autobuttons
\end{page}
```

———–

Closed-Form Solutions of Linear Differential Equations

— ug08.ht —

```
\begin{page}{ugxProblemLDEQClosedPage}
{8.10.1. Closed-Form Solutions of Linear Differential Equations}
\beginscroll

A {\it differential equation} is an equation involving an unknown
{\it function} and one or more of its derivatives.  The equation is
called {\it ordinary} if derivatives with respect to only one dependent
variable appear in the equation (it is called {\it partial}
otherwise).  The package \axiomType{ElementaryFunctionODESolver}
provides the top-level operation \spadfun {solve} for finding
closed-form solutions of ordinary differential equations.

To solve a differential equation, you must first create an operator for
the unknown function.
%
\xtc{
We let \axiom{y} be the unknown function in terms of \axiom{x}.
}{
\spadpaste{y := operator 'y \bound{y}}
}
```

```
%
You then type the equation using \axiomFun{D} to create the
derivatives of the unknown function \axiom{y(x)} where \axiom{x} is any
symbol you choose (the so-called {\it dependent variable}).
%
\xtc{
This is how you enter
the equation \axiom{y'' + y' + y = 0}.
}{
\spadpaste{deq := D(y x, x, 2) + D(y x, x) + y x = 0\bound{e1}\free{y}}
}
%
The simplest way to invoke the \axiomFun{solve} command is with three
arguments.
\begin{items}
\item the differential equation,
\item the operator representing the unknown function,
\item the dependent variable.
\end{items}
%
\xtc{
So, to solve the above equation, we enter this.
}{
\spadpaste{solve(deq, y, x) \free{e1}\free{y}}
}
%
Since linear ordinary differential equations have infinitely many
solutions, \axiomFun{solve} returns a {\it particular solution}
\texht{$f_p$}{\axiom{f_p}}
and a basis
\texht{$f_1,\dots,f_n$}{\axiom{f1,...,fn}}
for the solutions of the corresponding homogenuous equation.
Any expression of the form
\texht{$f_p + c_1 f_1 + \dots c_n f_n$}{\axiom{fp + c1 f1 + ... + cn fn}}
where the \texht{$c_i$}{\axiom{ci}} do not involve
the dependent variable is also a solution.
This is similar to what you get when you solve systems of linear
algebraic equations.

A way to select a unique solution is to specify {\it initial
conditions}: choose a value \axiom{a} for the dependent variable
and specify the values of the unknown function and its derivatives
at \axiom{a}.
If the number of initial conditions is equal to the order of the
equation, then the solution is unique (if it exists in closed
form!) and \axiomFun{solve} tries to find it.
To specify initial conditions to \axiomFun{solve}, use an
\axiomType{Equation} of the form \axiom{x = a} for the third
parameter instead of the dependent variable, and add a fourth
parameter consisting of the list of values \axiom{y(a), y'(a), ...}.
```

```
\xtc{
To find the solution of \axiom{y'' + y = 0} satisfying \axiom{y(0) =
y'(0) = 1}, do this.
}{
\spadpaste{deq := D(y x, x, 2) + y x \bound{e2}\free{y}}
}
\xtc{
You can omit the \axiom{= 0} when you enter the equation to be solved.
}{
\spadpaste{solve(deq, y, x = 0, [1, 1]) \free{e2}\free{y}}
}
%

Axiom is not limited to linear differential equations with constant
coefficients.  It can also find solutions when the coefficients are
rational or algebraic functions of the dependent variable.
Furthermore, Axiom is not limited by the order of the equation.

\xtc{
Axiom can solve the following third order equations with
polynomial coefficients.
}{
\spadpaste{deq := x**3 * D(y x, x, 3) + x**2 * D(y x, x, 2) - 2 *
x * D(y x, x) + 2 * y x = 2 * x**4 \bound{e3}\free{y}}
}
\xtc{
}{
\spadpaste{solve(deq, y, x) \free{e3}\free{y}}
}
%
%
\xtc{
Here we are solving a homogeneous equation.
}{
\spadpaste{deq := (x**9+x**3) * D(y x, x, 3) + 18 * x**8 *
D(y x, x, 2) - 90 * x * D(y x, x) - 30 * (11 * x**6 - 3) * y x
\bound{e4}\free{y}}
}
\xtc{
}{
\spadpaste{solve(deq, y, x) \free{e4}\free{y}}
}

On the other hand, and in contrast with the operation
\axiomFun{integrate}, it can happen that Axiom finds no solution and
that some closed-form solution still exists.  While it is
mathematically complicated to describe exactly when the solutions are
guaranteed to be found, the following statements are correct and form
good guidelines for linear ordinary differential equations:
```

```
\begin{items}
\item If the coefficients are constants, Axiom finds a complete basis
of solutions (i,e, all solutions).
\item If the coefficients are rational functions in the dependent variable,
Axiom at least finds all solutions that do not involve algebraic
functions.
\end{items}
%
Note that this last statement does not mean that Axiom does not find
the solutions that are algebraic functions.
It means that it is not
guaranteed that the algebraic function solutions will be found.
%
\xtc{
This is an example where all the algebraic solutions are found.
}{
\spadpaste{deq := (x**2 + 1) * D(y x, x, 2) + 3 * x *
D(y x, x) + y x = 0 \bound{e5}\free{y}}
}
\xtc{
}{
\spadpaste{solve(deq, y, x) \free{e5}\free{y}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxProblemLDEQClosedPagePatch1}
\begin{paste}{ugxProblemLDEQClosedPageFull1}{ugxProblemLDEQClosedPageEmpty1}
\pastebutton{ugxProblemLDEQClosedPageFull1}{\hidepaste}
\tab{5}\spadcommand{y := operator 'y\bound{y }}
\indentrel{3}\begin{verbatim}
   (1)  y
                                    Type: BasicOperator
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemLDEQClosedPageEmpty1}
\begin{paste}{ugxProblemLDEQClosedPageEmpty1}{ugxProblemLDEQClosedPagePatch1}
\pastebutton{ugxProblemLDEQClosedPageEmpty1}{\showpaste}
\tab{5}\spadcommand{y := operator 'y\bound{y }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemLDEQClosedPagePatch2}
\begin{paste}{ugxProblemLDEQClosedPageFull2}{ugxProblemLDEQClosedPageEmpty2}
\pastebutton{ugxProblemLDEQClosedPageFull2}{\hidepaste}
\tab{5}\spadcommand{deq := D(y x, x, 2) + D(y x, x) + y x = 0\bound{e1 }\free{y }}
\indentrel{3}\begin{verbatim}
```

```
        ,,        ,
   (2)  y  (x) + y (x) + y(x)= 0

                     Type: Equation Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemLDEQClosedPageEmpty2}
\begin{paste}{ugxProblemLDEQClosedPageEmpty2}{ugxProblemLDEQClosedPagePatch2}
\pastebutton{ugxProblemLDEQClosedPageEmpty2}{\showpaste}
\tab{5}\spadcommand{deq := D(y x, x, 2) + D(y x, x) + y x = 0\bound{e1 }\free{y }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemLDEQClosedPagePatch3}
\begin{paste}{ugxProblemLDEQClosedPageFull3}{ugxProblemLDEQClosedPageEmpty3}
\pastebutton{ugxProblemLDEQClosedPageFull3}{\hidepaste}
\tab{5}\spadcommand{solve(deq, y, x)\free{e1 }\free{y }}
\indentrel{3}\begin{verbatim}
   (3)
   [particular= 0,
                          x     x
                        -    -
                x\3      2     2    x\3
    basis= [cos()%e   ,%e   sin()]]
                 2                    2
Type: Union(Record(particular: Expression Integer,basis: List Expression Integer),...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemLDEQClosedPageEmpty3}
\begin{paste}{ugxProblemLDEQClosedPageEmpty3}{ugxProblemLDEQClosedPagePatch3}
\pastebutton{ugxProblemLDEQClosedPageEmpty3}{\showpaste}
\tab{5}\spadcommand{solve(deq, y, x)\free{e1 }\free{y }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemLDEQClosedPagePatch4}
\begin{paste}{ugxProblemLDEQClosedPageFull4}{ugxProblemLDEQClosedPageEmpty4}
\pastebutton{ugxProblemLDEQClosedPageFull4}{\hidepaste}
\tab{5}\spadcommand{deq := D(y x, x, 2) + y x\bound{e2 }\free{y }}
\indentrel{3}\begin{verbatim}
        ,,
   (4)  y  (x) + y(x)

                              Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemLDEQClosedPageEmpty4}
\begin{paste}{ugxProblemLDEQClosedPageEmpty4}{ugxProblemLDEQClosedPagePatch4}
\pastebutton{ugxProblemLDEQClosedPageEmpty4}{\showpaste}
```

```
\tab{5}\spadcommand{deq := D(y x, x, 2) + y x\bound{e2 }\free{y }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemLDEQClosedPagePatch5}
\begin{paste}{ugxProblemLDEQClosedPageFull5}{ugxProblemLDEQClosedPageEmpty5}
\pastebutton{ugxProblemLDEQClosedPageFull5}{\hidepaste}
\tab{5}\spadcommand{solve(deq, y, x = 0, [1, 1])\free{e2 }\free{y }}
\indentrel{3}\begin{verbatim}
   (5)  sin(x) + cos(x)
                    Type: Union(Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemLDEQClosedPageEmpty5}
\begin{paste}{ugxProblemLDEQClosedPageEmpty5}{ugxProblemLDEQClosedPagePatch5}
\pastebutton{ugxProblemLDEQClosedPageEmpty5}{\showpaste}
\tab{5}\spadcommand{solve(deq, y, x = 0, [1, 1])\free{e2 }\free{y }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemLDEQClosedPagePatch6}
\begin{paste}{ugxProblemLDEQClosedPageFull6}{ugxProblemLDEQClosedPageEmpty6}
\pastebutton{ugxProblemLDEQClosedPageFull6}{\hidepaste}
\tab{5}\spadcommand{deq := x**3 * D(y x, x, 3) + x**2 * D(y x, x, 2) - 2 * x * D(y x, x) + 2 * y x = 2 * x*
\indentrel{3}\begin{verbatim}
         3 ,,,       2 ,,          ,               4
   (6)  x y   (x) + x y  (x) - 2xy (x) + 2y(x)= 2x

                      Type: Equation Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemLDEQClosedPageEmpty6}
\begin{paste}{ugxProblemLDEQClosedPageEmpty6}{ugxProblemLDEQClosedPagePatch6}
\pastebutton{ugxProblemLDEQClosedPageEmpty6}{\showpaste}
\tab{5}\spadcommand{deq := x**3 * D(y x, x, 3) + x**2 * D(y x, x, 2) - 2 * x * D(y x, x) + 2 * y x = 2 * x*
\end{paste}\end{patch}

\begin{patch}{ugxProblemLDEQClosedPagePatch7}
\begin{paste}{ugxProblemLDEQClosedPageFull7}{ugxProblemLDEQClosedPageEmpty7}
\pastebutton{ugxProblemLDEQClosedPageFull7}{\hidepaste}
\tab{5}\spadcommand{solve(deq, y, x)\free{e3 }\free{y }}
\indentrel{3}\begin{verbatim}
   (7)
                 5      3      2
                x  - 10x  + 20x  + 4
   [particular= ,
                        15x
              3     2      3      3     2
            2x  - 3x  + 1 x  - 1 x  - 3x  - 1
    basis= [,,]]
```

```
                  x          x         x
Type: Union(Record(particular: Expression Integer,basis: List Expression Integer),...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemLDEQClosedPageEmpty7}
\begin{paste}{ugxProblemLDEQClosedPageEmpty7}{ugxProblemLDEQClosedPagePatch7}
\pastebutton{ugxProblemLDEQClosedPageEmpty7}{\showpaste}
\tab{5}\spadcommand{solve(deq, y, x)\free{e3 }\free{y }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemLDEQClosedPagePatch8}
\begin{paste}{ugxProblemLDEQClosedPageFull8}{ugxProblemLDEQClosedPageEmpty8}
\pastebutton{ugxProblemLDEQClosedPageFull8}{\hidepaste}
\tab{5}\spadcommand{deq := (x**9+x**3) * D(y x, x, 3) + 18 * x**8 * D(y x, x, 2) - 90 * x * D(y
\indentrel{3}\begin{verbatim}
   (8)
       9    3  ,,,         8 ,,           ,
     (x  + x )y   (x) + 18x y  (x) - 90xy (x)

   +
            6
     (- 330x  + 90)y(x)
                              Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemLDEQClosedPageEmpty8}
\begin{paste}{ugxProblemLDEQClosedPageEmpty8}{ugxProblemLDEQClosedPagePatch8}
\pastebutton{ugxProblemLDEQClosedPageEmpty8}{\showpaste}
\tab{5}\spadcommand{deq := (x**9+x**3) * D(y x, x, 3) + 18 * x**8 * D(y x, x, 2) - 90 * x * D(y
\end{paste}\end{patch}

\begin{patch}{ugxProblemLDEQClosedPagePatch9}
\begin{paste}{ugxProblemLDEQClosedPageFull9}{ugxProblemLDEQClosedPageEmpty9}
\pastebutton{ugxProblemLDEQClosedPageFull9}{\hidepaste}
\tab{5}\spadcommand{solve(deq, y, x)\free{e4 }\free{y }}
\indentrel{3}\begin{verbatim}
   (9)
   [particular= 0,

                       - \91 log(x)     \91 log(x)
               x   x %e              x %e
    basis= [,,]]
             6            6                6
            x  + 1       x  + 1           x  + 1
Type: Union(Record(particular: Expression Integer,basis: List Expression Integer),...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugxProblemLDEQClosedPageEmpty9}
\begin{paste}{ugxProblemLDEQClosedPageEmpty9}{ugxProblemLDEQClosedPagePatch9}
\pastebutton{ugxProblemLDEQClosedPageEmpty9}{\showpaste}
\tab{5}\spadcommand{solve(deq, y, x)\free{e4 }\free{y }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemLDEQClosedPagePatch10}
\begin{paste}{ugxProblemLDEQClosedPageFull10}{ugxProblemLDEQClosedPageEmpty10}
\pastebutton{ugxProblemLDEQClosedPageFull10}{\hidepaste}
\tab{5}\spadcommand{deq := (x**2 + 1) * D(y x, x, 2) + 3 * x * D(y x, x) + y x = 0\bound{e5 }\free{y }}
\indentrel{3}\begin{verbatim}
           2      ,,          ,
   (10)  (x  + 1)y  (x) + 3xy (x) + y(x)= 0

                      Type: Equation Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemLDEQClosedPageEmpty10}
\begin{paste}{ugxProblemLDEQClosedPageEmpty10}{ugxProblemLDEQClosedPagePatch10}
\pastebutton{ugxProblemLDEQClosedPageEmpty10}{\showpaste}
\tab{5}\spadcommand{deq := (x**2 + 1) * D(y x, x, 2) + 3 * x * D(y x, x) + y x = 0\bound{e5 }\free{y }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemLDEQClosedPagePatch11}
\begin{paste}{ugxProblemLDEQClosedPageFull11}{ugxProblemLDEQClosedPageEmpty11}
\pastebutton{ugxProblemLDEQClosedPageFull11}{\hidepaste}
\tab{5}\spadcommand{solve(deq, y, x)\free{e5 }\free{y }}
\indentrel{3}\begin{verbatim}
   (11)

                                             2
                              1      log(\x  + 1  - x)
   [particular= 0,basis= [,]]

                            2             2
                          \x  + 1       \x  + 1
Type: Union(Record(particular: Expression Integer,basis: List Expression Integer),...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemLDEQClosedPageEmpty11}
\begin{paste}{ugxProblemLDEQClosedPageEmpty11}{ugxProblemLDEQClosedPagePatch11}
\pastebutton{ugxProblemLDEQClosedPageEmpty11}{\showpaste}
\tab{5}\spadcommand{solve(deq, y, x)\free{e5 }\free{y }}
\end{paste}\end{patch}
```

Closed-Form Solutions of Non-Linear DEs

— ug08.ht —

```
\begin{page}{ugxProblemNLDEQClosedPage}
{8.10.2. Closed-Form Solutions of Non-Linear DEs}
\beginscroll

This is an example that shows how to solve a non-linear
first order ordinary differential equation manually when an integrating
factor can be found just by integration.
At the end, we show you how to solve it directly.

Let's solve the differential equation \axiom{y' = y / (x + y log y)}.
%
\xtc{
Using the notation
\axiom{m(x, y) + n(x, y) y' = 0},
we have \axiom{m = -y} and \axiom{n = x + y log y}.
}{
\spadpaste{m := -y \bound{m}}
}
\xtc{
}{
\spadpaste{n := x + y * log y \bound{n}}
}
%
\xtc{
We first check for exactness, that is, does \axiom{dm/dy = dn/dx}?
}{
\spadpaste{D(m, y) - D(n, x) \free{m n}}
}
%
This is not zero, so the equation is not exact.
Therefore we must look for
an integrating factor: a function \axiom{mu(x,y)} such that
\axiom{d(mu m)/dy = d(mu n)/dx}.
Normally, we first search for \axiom{mu(x,y)} depending only on
\axiom{x} or only on \axiom{y}.
%
\xtc{
Let's search for such a \axiom{mu(x)} first.
}{
\spadpaste{mu := operator 'mu \bound{mu}}
}
\xtc{
}{
\spadpaste{a := D(mu(x) * m, y) - D(mu(x) * n, x) \bound{a}\free{m n mu}}
}
```

```
%
%
\xtc{
If the above is zero for a function
\axiom{mu} that does {\it not} depend on \axiom{y}, then
\axiom{mu(x)} is an integrating factor.
}{
\spadpaste{solve(a = 0, mu, x) \free{mu a}}
}
%
The solution depends on \axiom{y}, so there is no integrating
factor that depends on \axiom{x} only.
%
\xtc{
Let's look for one that depends on \axiom{y} only.
}{
\spadpaste{b := D(mu(y) * m, y) - D(mu(y) * n, x) \bound{b}\free{mu m}}
}
\xtc{
}{
\spadpaste{sb := solve(b = 0, mu, y) \free{mu b}\bound{sb}}
}
\noindent
We've found one!
%
\xtc{
The above \axiom{mu(y)} is an integrating factor.
We must multiply our initial equation
(that is, \axiom{m} and \axiom{n}) by the integrating factor.
}{
\spadpaste{intFactor := sb.basis.1 \bound{intFactor}\free{sb}}
}
\xtc{
}{
\spadpaste{m := intFactor * m \bound{m1}\free{m intFactor}}
}
\xtc{
}{
\spadpaste{n := intFactor * n \bound{n1}\free{n intFactor}}
}
%
\xtc{
Let's check for exactness.
}{
\spadpaste{D(m, y) - D(n, x) \free{m1 n1}}
}
%
We must solve the exact equation, that is, find a function
\axiom{s(x,y)} such that
\axiom{ds/dx = m}  and \axiom{ds/dy = n}.
```

```
%
\xtc{
We start by writing \axiom{s(x, y) = h(y) + integrate(m, x)}
where \axiom{h(y)} is an unknown function of \axiom{y}.
This guarantees that \axiom{ds/dx = m}.
}{
\spadpaste{h := operator 'h \bound{h}}
}
\xtc{
}{
\spadpaste{sol := h y + integrate(m, x) \bound{sol}\free{h m1}}
}
%
%
\xtc{
All we want is to find \axiom{h(y)} such that
\axiom{ds/dy = n}.
}{
\spadpaste{dsol := D(sol, y) \free{sol}\bound{dsol}}
}
\xtc{
}{
\spadpaste{nsol := solve(dsol = n, h, y) \free{dsol n1 h}\bound{nsol}}
}
%
\xtc{
The above particular solution is the \axiom{h(y)} we want, so we just
replace \axiom{h(y)} by it in the implicit solution.
}{
\spadpaste{eval(sol, h y = nsol.particular) \free{sol h nsol}}
}
%
A first integral of the initial equation is obtained by setting
this result equal to an arbitrary constant.

Now that we've seen how to solve the equation ``by hand,''
we show you how to do it with the \axiomFun{solve} operation.
\xtc{
First define \axiom{y} to be an operator.
}{
\spadpaste{y := operator 'y \bound{y}}
}
\xtc{
Next we create the differential equation.
}{
\spadpaste{deq := D(y x, x) = y(x) / (x + y(x) * log y x)
\bound{deqi}\free{y}}
}
\xtc{
Finally, we solve it.
```

```
}{
\spadpaste{solve(deq, y, x) \free{deqi y}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxProblemNLDEQClosedPagePatch1}
\begin{paste}{ugxProblemNLDEQClosedPageFull1}{ugxProblemNLDEQClosedPageEmpty1}
\pastebutton{ugxProblemNLDEQClosedPageFull1}{\hidepaste}
\tab{5}\spadcommand{m := -y\bound{m }}
\indentrel{3}\begin{verbatim}
   (1)  - y
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemNLDEQClosedPageEmpty1}
\begin{paste}{ugxProblemNLDEQClosedPageEmpty1}{ugxProblemNLDEQClosedPagePatch1}
\pastebutton{ugxProblemNLDEQClosedPageEmpty1}{\showpaste}
\tab{5}\spadcommand{m := -y\bound{m }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemNLDEQClosedPagePatch2}
\begin{paste}{ugxProblemNLDEQClosedPageFull2}{ugxProblemNLDEQClosedPageEmpty2}
\pastebutton{ugxProblemNLDEQClosedPageFull2}{\hidepaste}
\tab{5}\spadcommand{n := x + y * log y\bound{n }}
\indentrel{3}\begin{verbatim}
   (2)  y log(y) + x
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemNLDEQClosedPageEmpty2}
\begin{paste}{ugxProblemNLDEQClosedPageEmpty2}{ugxProblemNLDEQClosedPagePatch2}
\pastebutton{ugxProblemNLDEQClosedPageEmpty2}{\showpaste}
\tab{5}\spadcommand{n := x + y * log y\bound{n }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemNLDEQClosedPagePatch3}
\begin{paste}{ugxProblemNLDEQClosedPageFull3}{ugxProblemNLDEQClosedPageEmpty3}
\pastebutton{ugxProblemNLDEQClosedPageFull3}{\hidepaste}
\tab{5}\spadcommand{D(m, y) - D(n, x)\free{m n }}
\indentrel{3}\begin{verbatim}
   (3)  - 2
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugxProblemNLDEQClosedPageEmpty3}
\begin{paste}{ugxProblemNLDEQClosedPageEmpty3}{ugxProblemNLDEQClosedPagePatch3}
\pastebutton{ugxProblemNLDEQClosedPageEmpty3}{\showpaste}
\tab{5}\spadcommand{D(m, y) - D(n, x)\free{m n }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemNLDEQClosedPagePatch4}
\begin{paste}{ugxProblemNLDEQClosedPageFull4}{ugxProblemNLDEQClosedPageEmpty4}
\pastebutton{ugxProblemNLDEQClosedPageFull4}{\hidepaste}
\tab{5}\spadcommand{mu := operator 'mu\bound{mu }}
\indentrel{3}\begin{verbatim}
   (4)  mu
                                    Type: BasicOperator
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemNLDEQClosedPageEmpty4}
\begin{paste}{ugxProblemNLDEQClosedPageEmpty4}{ugxProblemNLDEQClosedPagePatch4}
\pastebutton{ugxProblemNLDEQClosedPageEmpty4}{\showpaste}
\tab{5}\spadcommand{mu := operator 'mu\bound{mu }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemNLDEQClosedPagePatch5}
\begin{paste}{ugxProblemNLDEQClosedPageFull5}{ugxProblemNLDEQClosedPageEmpty5}
\pastebutton{ugxProblemNLDEQClosedPageFull5}{\hidepaste}
\tab{5}\spadcommand{a := D(mu(x) * m, y) - D(mu(x) * n, x)\bound{a }\free{m n mu }}
\indentrel{3}\begin{verbatim}
                           ,
   (5)  (- y log(y) - x)mu (x) - 2mu(x)

                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemNLDEQClosedPageEmpty5}
\begin{paste}{ugxProblemNLDEQClosedPageEmpty5}{ugxProblemNLDEQClosedPagePatch5}
\pastebutton{ugxProblemNLDEQClosedPageEmpty5}{\showpaste}
\tab{5}\spadcommand{a := D(mu(x) * m, y) - D(mu(x) * n, x)\bound{a }\free{m n mu }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemNLDEQClosedPagePatch6}
\begin{paste}{ugxProblemNLDEQClosedPageFull6}{ugxProblemNLDEQClosedPageEmpty6}
\pastebutton{ugxProblemNLDEQClosedPageFull6}{\hidepaste}
\tab{5}\spadcommand{solve(a = 0, mu, x)\free{mu a }}
\indentrel{3}\begin{verbatim}
   (6)
                                        1
   [particular= 0,basis= []]
                          2      2                   2
                         y log(y)  + 2x y log(y) + x
```

```
Type: Union(Record(particular: Expression Integer,basis: List Expression Integer),...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemNLDEQClosedPageEmpty6}
\begin{paste}{ugxProblemNLDEQClosedPageEmpty6}{ugxProblemNLDEQClosedPagePatch6}
\pastebutton{ugxProblemNLDEQClosedPageEmpty6}{\showpaste}
\tab{5}\spadcommand{solve(a = 0, mu, x)\free{mu a }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemNLDEQClosedPagePatch7}
\begin{paste}{ugxProblemNLDEQClosedPageFull7}{ugxProblemNLDEQClosedPageEmpty7}
\pastebutton{ugxProblemNLDEQClosedPageFull7}{\hidepaste}
\tab{5}\spadcommand{b := D(mu(y) * m, y) - D(mu(y) * n, x)\bound{b }\free{mu m }}
\indentrel{3}\begin{verbatim}
              ,
   (7)  - ymu (y) - 2mu(y)

                              Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemNLDEQClosedPageEmpty7}
\begin{paste}{ugxProblemNLDEQClosedPageEmpty7}{ugxProblemNLDEQClosedPagePatch7}
\pastebutton{ugxProblemNLDEQClosedPageEmpty7}{\showpaste}
\tab{5}\spadcommand{b := D(mu(y) * m, y) - D(mu(y) * n, x)\bound{b }\free{mu m }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemNLDEQClosedPagePatch8}
\begin{paste}{ugxProblemNLDEQClosedPageFull8}{ugxProblemNLDEQClosedPageEmpty8}
\pastebutton{ugxProblemNLDEQClosedPageFull8}{\hidepaste}
\tab{5}\spadcommand{sb := solve(b = 0, mu, y)\free{mu b }\bound{sb }}
\indentrel{3}\begin{verbatim}
                                1
   (8)  [particular= 0,basis= []]
                                2
                               y
Type: Union(Record(particular: Expression Integer,basis: List Expression Integer),...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemNLDEQClosedPageEmpty8}
\begin{paste}{ugxProblemNLDEQClosedPageEmpty8}{ugxProblemNLDEQClosedPagePatch8}
\pastebutton{ugxProblemNLDEQClosedPageEmpty8}{\showpaste}
\tab{5}\spadcommand{sb := solve(b = 0, mu, y)\free{mu b }\bound{sb }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemNLDEQClosedPagePatch9}
\begin{paste}{ugxProblemNLDEQClosedPageFull9}{ugxProblemNLDEQClosedPageEmpty9}
\pastebutton{ugxProblemNLDEQClosedPageFull9}{\hidepaste}
```

```
\tab{5}\spadcommand{intFactor := sb.basis.1\bound{intFactor }\free{sb }}
\indentrel{3}\begin{verbatim}
        1
   (9)
        2
       y
                              Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemNLDEQClosedPageEmpty9}
\begin{paste}{ugxProblemNLDEQClosedPageEmpty9}{ugxProblemNLDEQClosedPagePatch9}
\pastebutton{ugxProblemNLDEQClosedPageEmpty9}{\showpaste}
\tab{5}\spadcommand{intFactor := sb.basis.1\bound{intFactor }\free{sb }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemNLDEQClosedPagePatch10}
\begin{paste}{ugxProblemNLDEQClosedPageFull10}{ugxProblemNLDEQClosedPageEmpty10}
\pastebutton{ugxProblemNLDEQClosedPageFull10}{\hidepaste}
\tab{5}\spadcommand{m := intFactor * m\bound{m1 }\free{m intFactor }}
\indentrel{3}\begin{verbatim}
          1
   (10)  -
          y
                              Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemNLDEQClosedPageEmpty10}
\begin{paste}{ugxProblemNLDEQClosedPageEmpty10}{ugxProblemNLDEQClosedPagePatch10}
\pastebutton{ugxProblemNLDEQClosedPageEmpty10}{\showpaste}
\tab{5}\spadcommand{m := intFactor * m\bound{m1 }\free{m intFactor }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemNLDEQClosedPagePatch11}
\begin{paste}{ugxProblemNLDEQClosedPageFull11}{ugxProblemNLDEQClosedPageEmpty11}
\pastebutton{ugxProblemNLDEQClosedPageFull11}{\hidepaste}
\tab{5}\spadcommand{n := intFactor * n\bound{n1 }\free{n intFactor }}
\indentrel{3}\begin{verbatim}
         y log(y) + x
   (11)
              2
             y
                              Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemNLDEQClosedPageEmpty11}
\begin{paste}{ugxProblemNLDEQClosedPageEmpty11}{ugxProblemNLDEQClosedPagePatch11}
\pastebutton{ugxProblemNLDEQClosedPageEmpty11}{\showpaste}
```

```
\tab{5}\spadcommand{n := intFactor * n\bound{n1 }\free{n intFactor }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemNLDEQClosedPagePatch12}
\begin{paste}{ugxProblemNLDEQClosedPageFull12}{ugxProblemNLDEQClosedPageEmpty12}
\pastebutton{ugxProblemNLDEQClosedPageFull12}{\hidepaste}
\tab{5}\spadcommand{D(m, y) - D(n, x)\free{m1 n1 }}
\indentrel{3}\begin{verbatim}
   (12)  0
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemNLDEQClosedPageEmpty12}
\begin{paste}{ugxProblemNLDEQClosedPageEmpty12}{ugxProblemNLDEQClosedPagePatch12}
\pastebutton{ugxProblemNLDEQClosedPageEmpty12}{\showpaste}
\tab{5}\spadcommand{D(m, y) - D(n, x)\free{m1 n1 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemNLDEQClosedPagePatch13}
\begin{paste}{ugxProblemNLDEQClosedPageFull13}{ugxProblemNLDEQClosedPageEmpty13}
\pastebutton{ugxProblemNLDEQClosedPageFull13}{\hidepaste}
\tab{5}\spadcommand{h := operator 'h\bound{h }}
\indentrel{3}\begin{verbatim}
   (13)  h
                                    Type: BasicOperator
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemNLDEQClosedPageEmpty13}
\begin{paste}{ugxProblemNLDEQClosedPageEmpty13}{ugxProblemNLDEQClosedPagePatch13}
\pastebutton{ugxProblemNLDEQClosedPageEmpty13}{\showpaste}
\tab{5}\spadcommand{h := operator 'h\bound{h }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemNLDEQClosedPagePatch14}
\begin{paste}{ugxProblemNLDEQClosedPageFull14}{ugxProblemNLDEQClosedPageEmpty14}
\pastebutton{ugxProblemNLDEQClosedPageFull14}{\hidepaste}
\tab{5}\spadcommand{sol := h y + integrate(m, x)\bound{sol }\free{h m1 }}
\indentrel{3}\begin{verbatim}
         y h(y) - x
   (14)  ----------
              y
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemNLDEQClosedPageEmpty14}
\begin{paste}{ugxProblemNLDEQClosedPageEmpty14}{ugxProblemNLDEQClosedPagePatch14}
\pastebutton{ugxProblemNLDEQClosedPageEmpty14}{\showpaste}
```

```
\tab{5}\spadcommand{sol := h y + integrate(m, x)\bound{sol }\free{h m1 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemNLDEQClosedPagePatch15}
\begin{paste}{ugxProblemNLDEQClosedPageFull15}{ugxProblemNLDEQClosedPageEmpty15}
\pastebutton{ugxProblemNLDEQClosedPageFull15}{\hidepaste}
\tab{5}\spadcommand{dsol := D(sol, y)\free{sol }\bound{dsol }}
\indentrel{3}\begin{verbatim}
          2 ,
         y h (y) + x

   (15)
              2
             y
                                Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemNLDEQClosedPageEmpty15}
\begin{paste}{ugxProblemNLDEQClosedPageEmpty15}{ugxProblemNLDEQClosedPagePatch15}
\pastebutton{ugxProblemNLDEQClosedPageEmpty15}{\showpaste}
\tab{5}\spadcommand{dsol := D(sol, y)\free{sol }\bound{dsol }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemNLDEQClosedPagePatch16}
\begin{paste}{ugxProblemNLDEQClosedPageFull16}{ugxProblemNLDEQClosedPageEmpty16}
\pastebutton{ugxProblemNLDEQClosedPageFull16}{\hidepaste}
\tab{5}\spadcommand{nsol := solve(dsol = n, h, y)\free{dsol n1 h }\bound{nsol }}
\indentrel{3}\begin{verbatim}
                              2
                      log(y)
   (16)  [particular= ,basis= [1]]
                           2
Type: Union(Record(particular: Expression Integer,basis: List Expression Integer),...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemNLDEQClosedPageEmpty16}
\begin{paste}{ugxProblemNLDEQClosedPageEmpty16}{ugxProblemNLDEQClosedPagePatch16}
\pastebutton{ugxProblemNLDEQClosedPageEmpty16}{\showpaste}
\tab{5}\spadcommand{nsol := solve(dsol = n, h, y)\free{dsol n1 h }\bound{nsol }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemNLDEQClosedPagePatch17}
\begin{paste}{ugxProblemNLDEQClosedPageFull17}{ugxProblemNLDEQClosedPageEmpty17}
\pastebutton{ugxProblemNLDEQClosedPageFull17}{\hidepaste}
\tab{5}\spadcommand{eval(sol, h y = nsol.particular)\free{sol h nsol }}
\indentrel{3}\begin{verbatim}
                2
         y log(y)  - 2x
```

```
   (17)
              2y
                                Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemNLDEQClosedPageEmpty17}
\begin{paste}{ugxProblemNLDEQClosedPageEmpty17}{ugxProblemNLDEQClosedPagePatch17}
\pastebutton{ugxProblemNLDEQClosedPageEmpty17}{\showpaste}
\tab{5}\spadcommand{eval(sol, h y = nsol.particular)\free{sol h nsol }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemNLDEQClosedPagePatch18}
\begin{paste}{ugxProblemNLDEQClosedPageFull18}{ugxProblemNLDEQClosedPageEmpty18}
\pastebutton{ugxProblemNLDEQClosedPageFull18}{\hidepaste}
\tab{5}\spadcommand{y := operator 'y\bound{y }}
\indentrel{3}\begin{verbatim}
   (18)  y
                                     Type: BasicOperator
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemNLDEQClosedPageEmpty18}
\begin{paste}{ugxProblemNLDEQClosedPageEmpty18}{ugxProblemNLDEQClosedPagePatch18}
\pastebutton{ugxProblemNLDEQClosedPageEmpty18}{\showpaste}
\tab{5}\spadcommand{y := operator 'y\bound{y }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemNLDEQClosedPagePatch19}
\begin{paste}{ugxProblemNLDEQClosedPageFull19}{ugxProblemNLDEQClosedPageEmpty19}
\pastebutton{ugxProblemNLDEQClosedPageFull19}{\hidepaste}
\tab{5}\spadcommand{deq := D(y x, x) = y(x) / (x + y(x) * log y x)\bound{deqi }\free{y }}
\indentrel{3}\begin{verbatim}
          ,            y(x)
   (19)  y (x)=
               y(x)log(y(x)) + x
                     Type: Equation Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemNLDEQClosedPageEmpty19}
\begin{paste}{ugxProblemNLDEQClosedPageEmpty19}{ugxProblemNLDEQClosedPagePatch19}
\pastebutton{ugxProblemNLDEQClosedPageEmpty19}{\showpaste}
\tab{5}\spadcommand{deq := D(y x, x) = y(x) / (x + y(x) * log y x)\bound{deqi }\free{y }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemNLDEQClosedPagePatch20}
\begin{paste}{ugxProblemNLDEQClosedPageFull20}{ugxProblemNLDEQClosedPageEmpty20}
\pastebutton{ugxProblemNLDEQClosedPageFull20}{\hidepaste}
\tab{5}\spadcommand{solve(deq, y, x)\free{deqi y }}
```

```
\indentrel{3}\begin{verbatim}
                      2
         y(x)log(y(x))  - 2x
   (20)
                2y(x)
                    Type: Union(Expression Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemNLDEQClosedPageEmpty20}
\begin{paste}{ugxProblemNLDEQClosedPageEmpty20}{ugxProblemNLDEQClosedPagePatch20}
\pastebutton{ugxProblemNLDEQClosedPageEmpty20}{\showpaste}
\tab{5}\spadcommand{solve(deq, y, x)\free{deqi y }}
\end{paste}\end{patch}
```

———-

Power Series Solutions of Differential Equations

— ug08.ht —

```
\begin{page}{ugxProblemDEQSeriesPage}
{8.10.3. Power Series Solutions of Differential Equations}
\beginscroll

The command to solve differential equations in power
series
around
a particular initial point with specific initial conditions is called
\axiomFun{seriesSolve}.
It can take a variety of parameters, so we illustrate
its use with some examples.

\labelSpace{1pc}
\noOutputXtc{
Since the coefficients of some solutions
are quite large, we reset the default to compute only seven terms.
}{
\spadpaste{)set streams calculate 7 \bound{c7}}
}

You can solve a single nonlinear equation of any order. For example,
we solve  \axiom{y''' = sin(y'') * exp(y) + cos(x)}
subject to \axiom{y(0) = 1, y'(0) = 0, y''(0) = 0}.

\xtc{
We first tell Axiom
```

```
that the symbol \axiom{'y} denotes a new operator.
}{
\spadpaste{y := operator 'y \bound{y}}
}
\xtc{
Enter the differential equation using \axiom{y} like any system
function.
}{
\spadpaste{eq := D(y(x), x, 3) - sin(D(y(x), x, 2))*exp(y(x)) =
cos(x)\bound{eq}\free{y}}
}
%
\xtc{
Solve it around \axiom{x = 0} with the initial conditions
\axiom{y(0) = 1, y'(0) = y''(0) = 0}.
}{
\spadpaste{seriesSolve(eq, y, x = 0, [1, 0, 0])\free{y}\free{eq}\free{c7}}
}

You can also solve a system of nonlinear first order equations.
For example, we solve a system that has \axiom{tan(t)} and
\axiom{sec(t)} as solutions.

\xtc{
We tell Axiom that \axiom{x} is also an operator.
}{
\spadpaste{x := operator 'x\bound{x}}
}
\xtc{
Enter the two equations forming our system.
}{
\spadpaste{eq1 := D(x(t), t) = 1 + x(t)**2\free{x}\free{y}\bound{eq1}}
}
%
\xtc{
}{
\spadpaste{eq2 := D(y(t), t) = x(t) * y(t)\free{x}\free{y}\bound{eq2}}
}
%
\xtc{
Solve the system around \axiom{t = 0} with the initial conditions
\axiom{x(0) = 0} and \axiom{y(0) = 1}.
Notice that since we give the unknowns in the
order \axiom{[x, y]}, the answer is a list of two series in the order
\axiom{[series for x(t), series for y(t)]}.
}{
\spadpaste{seriesSolve([eq2, eq1], [x, y], t = 0,
[y(0) = 1, x(0) = 0])\free{x}\free{y}\free{eq1}\free{eq2}\free{c7}}
}
\noindent
```

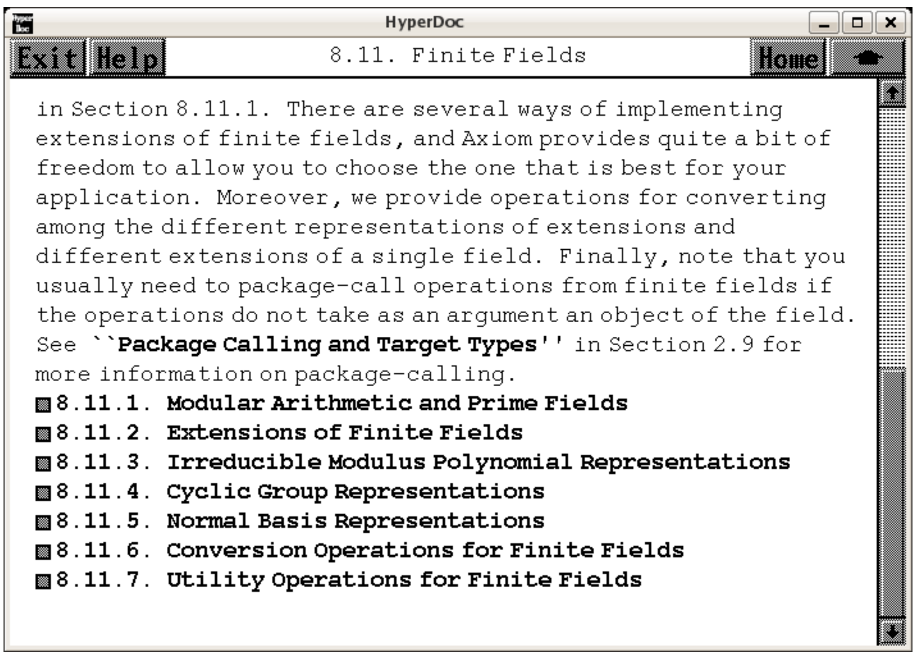
```
The order in which we give the
equations and the initial conditions has no effect on the order of
the solution.

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxProblemDEQSeriesPagePatch1}
\begin{paste}{ugxProblemDEQSeriesPageFull1}{ugxProblemDEQSeriesPageEmpty1}
\pastebutton{ugxProblemDEQSeriesPageFull1}{\hidepaste}
\tab{5}\spadcommand{)set streams calculate 7\bound{c7 }}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemDEQSeriesPageEmpty1}
\begin{paste}{ugxProblemDEQSeriesPageEmpty1}{ugxProblemDEQSeriesPagePatch1}
\pastebutton{ugxProblemDEQSeriesPageEmpty1}{\showpaste}
\tab{5}\spadcommand{)set streams calculate 7\bound{c7 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemDEQSeriesPagePatch2}
\begin{paste}{ugxProblemDEQSeriesPageFull2}{ugxProblemDEQSeriesPageEmpty2}
\pastebutton{ugxProblemDEQSeriesPageFull2}{\hidepaste}
\tab{5}\spadcommand{y := operator 'y\bound{y }}
\indentrel{3}\begin{verbatim}
   (1)  y
                                    Type: BasicOperator
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemDEQSeriesPageEmpty2}
\begin{paste}{ugxProblemDEQSeriesPageEmpty2}{ugxProblemDEQSeriesPagePatch2}
\pastebutton{ugxProblemDEQSeriesPageEmpty2}{\showpaste}
\tab{5}\spadcommand{y := operator 'y\bound{y }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemDEQSeriesPagePatch3}
\begin{paste}{ugxProblemDEQSeriesPageFull3}{ugxProblemDEQSeriesPageEmpty3}
\pastebutton{ugxProblemDEQSeriesPageFull3}{\hidepaste}
\tab{5}\spadcommand{eq := D(y(x), x, 3) - sin(D(y(x), x, 2))*exp(y(x)) = cos(x)\bound{eq }\free{
\indentrel{3}\begin{verbatim}
         ,,,        y(x)     ,,
   (2)  y   (x) - %e    sin(y  (x))= cos(x)

                      Type: Equation Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugxProblemDEQSeriesPageEmpty3}
\begin{paste}{ugxProblemDEQSeriesPageEmpty3}{ugxProblemDEQSeriesPagePatch3}
\pastebutton{ugxProblemDEQSeriesPageEmpty3}{\showpaste}
\tab{5}\spadcommand{eq := D(y(x), x, 3) - sin(D(y(x), x, 2))*exp(y(x)) = cos(x)\bound{eq }\free{y }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemDEQSeriesPagePatch4}
\begin{paste}{ugxProblemDEQSeriesPageFull4}{ugxProblemDEQSeriesPageEmpty4}
\pastebutton{ugxProblemDEQSeriesPageFull4}{\hidepaste}
\tab{5}\spadcommand{seriesSolve(eq, y, x = 0, [1, 0, 0])\free{y }\free{eq }\free{c7 }}
\indentrel{3}\begin{verbatim}
   (3)
                          2            3
         1  3   %e  4   %e  - 1  5   %e  - 2%e  6
     1 +  x  +  x  +  x  +  x
         6      24       120           720
   +
       4      2
     %e  - 8%e  + 4%e + 1  7      8
      x  + O(x )
             5040
   Type: UnivariateTaylorSeries(Expression Integer,x,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemDEQSeriesPageEmpty4}
\begin{paste}{ugxProblemDEQSeriesPageEmpty4}{ugxProblemDEQSeriesPagePatch4}
\pastebutton{ugxProblemDEQSeriesPageEmpty4}{\showpaste}
\tab{5}\spadcommand{seriesSolve(eq, y, x = 0, [1, 0, 0])\free{y }\free{eq }\free{c7 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemDEQSeriesPagePatch5}
\begin{paste}{ugxProblemDEQSeriesPageFull5}{ugxProblemDEQSeriesPageEmpty5}
\pastebutton{ugxProblemDEQSeriesPageFull5}{\hidepaste}
\tab{5}\spadcommand{x := operator 'x\bound{x }}
\indentrel{3}\begin{verbatim}
   (4)  x
                                    Type: BasicOperator
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemDEQSeriesPageEmpty5}
\begin{paste}{ugxProblemDEQSeriesPageEmpty5}{ugxProblemDEQSeriesPagePatch5}
\pastebutton{ugxProblemDEQSeriesPageEmpty5}{\showpaste}
\tab{5}\spadcommand{x := operator 'x\bound{x }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemDEQSeriesPagePatch6}
\begin{paste}{ugxProblemDEQSeriesPageFull6}{ugxProblemDEQSeriesPageEmpty6}
\pastebutton{ugxProblemDEQSeriesPageFull6}{\hidepaste}
```

```
\tab{5}\spadcommand{eq1 := D(x(t), t) = 1 + x(t)**2\free{x }\free{y }\bound{eq1 }}
\indentrel{3}\begin{verbatim}
         ,         2
   (5)  x (t)= x(t)  + 1

                      Type: Equation Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemDEQSeriesPageEmpty6}
\begin{paste}{ugxProblemDEQSeriesPageEmpty6}{ugxProblemDEQSeriesPagePatch6}
\pastebutton{ugxProblemDEQSeriesPageEmpty6}{\showpaste}
\tab{5}\spadcommand{eq1 := D(x(t), t) = 1 + x(t)**2\free{x }\free{y }\bound{eq1 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemDEQSeriesPagePatch7}
\begin{paste}{ugxProblemDEQSeriesPageFull7}{ugxProblemDEQSeriesPageEmpty7}
\pastebutton{ugxProblemDEQSeriesPageFull7}{\hidepaste}
\tab{5}\spadcommand{eq2 := D(y(t), t) = x(t) * y(t)\free{x }\free{y }\bound{eq2 }}
\indentrel{3}\begin{verbatim}
         ,
   (6)  y (t)= x(t)y(t)

                      Type: Equation Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemDEQSeriesPageEmpty7}
\begin{paste}{ugxProblemDEQSeriesPageEmpty7}{ugxProblemDEQSeriesPagePatch7}
\pastebutton{ugxProblemDEQSeriesPageEmpty7}{\showpaste}
\tab{5}\spadcommand{eq2 := D(y(t), t) = x(t) * y(t)\free{x }\free{y }\bound{eq2 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemDEQSeriesPagePatch8}
\begin{paste}{ugxProblemDEQSeriesPageFull8}{ugxProblemDEQSeriesPageEmpty8}
\pastebutton{ugxProblemDEQSeriesPageFull8}{\hidepaste}
\tab{5}\spadcommand{seriesSolve([eq2, eq1], [x, y], t = 0, [y(0) = 1, x(0) = 0])\free{x }\free{y
\indentrel{3}\begin{verbatim}
   (7)
        1  3    2  5    17  7      8
   [t + - t  + -- t  + --- t  + O(t ),
        3      15      315
        1  2    5  4    61  6      8
    1 + - t  + -- t  + --- t  + O(t )]
        2      24      720
Type: List UnivariateTaylorSeries(Expression Integer,t,0)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemDEQSeriesPageEmpty8}
```

```
\begin{paste}{ugxProblemDEQSeriesPageEmpty8}{ugxProblemDEQSeriesPagePatch8}
\pastebutton{ugxProblemDEQSeriesPageEmpty8}{\showpaste}
\tab{5}\spadcommand{seriesSolve([eq2, eq1], [x, y], t = 0, [y(0) = 1, x(0) = 0])\free{x }\free{y }\free{eq1
\end{paste}\end{patch}
```

Finite Fields



⇐ "Axiom Number Types" (NumberPage) 3.80 on page 1021
⇒ "Modular Arithmetic and Prime Fields" (ugxProblemFinitePrimePage) 12 on page 2237
⇒ "Extensions of Finite Fields" (ugxProblemFiniteExtensionFinitePage) 12 on page 2246
⇒ "Irreducible Modulus Polynomial Representations" (ugxProblemFiniteModulusPage) 12 on page 2249
⇒ "Cyclic Group Representations" (ugxProblemFiniteCyclicPage) 12 on page 2258
⇒ "Normal Basis Representations" (ugxProblemFiniteNormalPage) 12 on page 2264
⇒ "Conversion Operations for Finite Fields" (ugxProblemFiniteConversionPage) 12 on page 2272

⇒ "Utility Operations for Finite Fields" (ugxProblemFiniteUtilityPage) 12 on page 2280

— **ug08.ht** —

```
\begin{page}{ugProblemFinitePage}{8.11. Finite Fields}
\beginscroll
A {\it finite field} (also called a {\it Galois field}) is a
```

```
finite algebraic structure where one can add, multiply and divide
under the same laws (for example, commutativity, associativity or
distributivity) as apply to the rational, real or complex numbers.
Unlike those three fields, for any finite field there exists a
positive prime integer \smath{p}, called the
\axiomFun{characteristic}, such that
\texht{$p \: x = 0$}{\axiom{p * x = 0}}
for any element \smath{x} in the finite field.
In fact, the number of elements in a finite field is a power of
the characteristic and for each prime \smath{p} and positive
integer \smath{n} there exists exactly one finite field with
\texht{$p^n$}{\axiom{p**n}} elements, up to
isomorphism.\footnote{For more information about the algebraic
structure and properties of finite fields, see, for example, S.
Lang, {\it Algebra}, Second Edition, New York: Addison-Wesley
Publishing Company, Inc., 1984, ISBN 0 201 05487 6; or R.
Lidl, H.
Niederreiter, {\it Finite Fields}, Encyclopedia of Mathematics and
Its Applications, Vol.
20, Cambridge: Cambridge Univ.
Press, 1983, ISBN 0 521 30240 4.}

When \axiom{n = 1,} the field has \smath{p} elements and is
called a {\it prime field}, discussed in \texht{the next
section}{
\downlink{``Modular Arithmetic and Prime Fields''}
{ugxProblemFinitePrimePage}
in Section 8.11.1\ignore{ugxProblemFinitePrime}}.
There are several ways of implementing extensions of finite
fields, and Axiom provides quite a bit of freedom to allow
you to choose the one that is best for your application.
Moreover, we provide operations for converting among the different
representations of extensions and different extensions of a single
field.
Finally, note that you usually need to package-call operations
from finite fields if the operations do not take as an argument an
object of the field.
See \downlink{``Package Calling and Target Types''}
{ugTypesPkgCallPage} in Section 2.9\ignore{ugTypesPkgCall} for more
information on package-calling.

\beginmenu
\menudownlink{{8.11.1. Modular Arithmetic and Prime Fields}}
{ugxProblemFinitePrimePage}
\menudownlink{{8.11.2. Extensions of Finite Fields}}
{ugxProblemFiniteExtensionFinitePage}
\menudownlink{{8.11.3. Irreducible Modulus Polynomial Representations}}
{ugxProblemFiniteModulusPage}
\menudownlink{{8.11.4. Cyclic Group Representations}}
{ugxProblemFiniteCyclicPage}
```

```
\menudownlink{{8.11.5. Normal Basis Representations}}
{ugxProblemFiniteNormalPage}
\menudownlink{{8.11.6. Conversion Operations for Finite Fields}}
{ugxProblemFiniteConversionPage}
\menudownlink{{8.11.7. Utility Operations for Finite Fields}}
{ugxProblemFiniteUtilityPage}
\endmenu
\endscroll
\autobuttons
\end{page}
```

———-

Modular Arithmetic and Prime Fields

— ug08.ht —

```
\begin{page}{ugxProblemFinitePrimePage}
{8.11.1. Modular Arithmetic and Prime Fields}
\beginscroll

Let \smath{n} be a positive integer.
It is well known that you can get the same result if you perform
addition, subtraction or multiplication of integers and then take
the remainder on dividing by \smath{n} as if
you had first done such remaindering on the operands, performed the
arithmetic and then (if necessary) done remaindering again.
This allows us to speak of arithmetic
{\it modulo} \smath{n} or, more simply
{\it mod} \smath{n}.
\xtc{
In Axiom, you use \axiomType{IntegerMod} to do such arithmetic.
}{
\spadpaste{(a,b) : IntegerMod 12 \bound{abdec}}
}
\xtc{
}{
\spadpaste{(a, b) := (16, 7) \free{abdec}\bound{a b}}
}
\xtc{
}{
\spadpaste{[a - b, a * b] \free{a b}}
}
\xtc{
If \smath{n} is not prime, there is only a limited notion of
reciprocals and division.
}{
```

```
\spadpaste{a / b \free{a b}}
}
\xtc{
}{
\spadpaste{recip a \free{a}}
}
\xtc{
Here \axiom{7} and \axiom{12} are relatively prime, so \axiom{7}
has a multiplicative inverse modulo \axiom{12}.
}{
\spadpaste{recip b \free{b}}
}

If we take \smath{n} to be a prime number \smath{p},
then taking inverses and, therefore, division are generally defined.
\xtc{
Use \axiomType{PrimeField} instead of \axiomType{IntegerMod}
for \smath{n} prime.
}{
\spadpaste{c : PrimeField 11 := 8 \bound{c}}
}
\xtc{
}{
\spadpaste{inv c \free{c}}
}
\xtc{
You can also use \axiom{1/c} and \axiom{c**(-1)} for the inverse of
\smath{c}.
}{
\spadpaste{9/c \free{c}}
}

\axiomType{PrimeField} (abbreviation \axiomType{PF}) checks if its
argument is prime when you try to use an operation from it.
If you know the argument is prime (particularly if it is large),
\axiomType{InnerPrimeField} (abbreviation \axiomType{IPF}) assumes
the argument has already been verified to be prime.
If you do use a number that is not prime, you will eventually get
an error message, most likely a division by zero message.
For computer science applications, the most important finite fields
are \axiomType{PrimeField 2} and its extensions.

\xtc{
In the following examples, we work with the finite field with
\smath{p = 101} elements.
}{
\spadpaste{GF101 := PF 101 \bound{GF101} }
}
\xtc{
Like many domains in Axiom, finite fields provide an operation
```

```
for returning a random element of the domain.
}{
\spadpaste{x := random()\$GF101 \bound{x}\free{GF101}}
}
\xtc{
}{
\spadpaste{y : GF101 := 37 \bound{y}\free{GF101}}
}
\xtc{
}{
\spadpaste{z := x/y \bound{z}\free{x y}}
}
\xtc{
}{
\spadpaste{z * y - x \free{x y z}}
}
%
\xtc{
The element \axiom{2} is a {\it primitive element} of this field,
}{
\spadpaste{pe := primitiveElement()\$GF101 \bound{pe}\free{GF101}}
}
%
\xtc{
in the sense that its powers enumerate all nonzero elements.
}{
\spadpaste{[pe**i for i in 0..99] \free{pe}}
}
%
%
\xtc{
If every nonzero element is a power of a primitive element, how do you
determine what the exponent is?
Use
\axiomFun{discreteLog}.
}{
\spadpaste{ex := discreteLog(y) \bound{ex}\free{y}}
}
\xtc{
}{
\spadpaste{pe ** ex \free{ex pe}}
}
%
\xtc{
The \axiomFun{order} of a nonzero element \smath{x} is the
smallest positive integer \smath{t} such
\texht{$x^t = 1$}{\axiom{x**t = 1}}.
}{
\spadpaste{order y \free{y}}
}
```

```
\xtc{
The order of a primitive element is the defining \smath{p-1}.
}{
\spadpaste{order pe \free{pe}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxProblemFinitePrimePagePatch1}
\begin{paste}{ugxProblemFinitePrimePageFull1}{ugxProblemFinitePrimePageEmpty1}
\pastebutton{ugxProblemFinitePrimePageFull1}{\hidepaste}
\tab{5}\spadcommand{(a,b) : IntegerMod 12\bound{abdec }}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFinitePrimePageEmpty1}
\begin{paste}{ugxProblemFinitePrimePageEmpty1}{ugxProblemFinitePrimePagePatch1}
\pastebutton{ugxProblemFinitePrimePageEmpty1}{\showpaste}
\tab{5}\spadcommand{(a,b) : IntegerMod 12\bound{abdec }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFinitePrimePagePatch2}
\begin{paste}{ugxProblemFinitePrimePageFull2}{ugxProblemFinitePrimePageEmpty2}
\pastebutton{ugxProblemFinitePrimePageFull2}{\hidepaste}
\tab{5}\spadcommand{(a, b) := (16, 7)\free{abdec }\bound{a b }}
\indentrel{3}\begin{verbatim}
   (2)  7
                                   Type: IntegerMod 12
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFinitePrimePageEmpty2}
\begin{paste}{ugxProblemFinitePrimePageEmpty2}{ugxProblemFinitePrimePagePatch2}
\pastebutton{ugxProblemFinitePrimePageEmpty2}{\showpaste}
\tab{5}\spadcommand{(a, b) := (16, 7)\free{abdec }\bound{a b }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFinitePrimePagePatch3}
\begin{paste}{ugxProblemFinitePrimePageFull3}{ugxProblemFinitePrimePageEmpty3}
\pastebutton{ugxProblemFinitePrimePageFull3}{\hidepaste}
\tab{5}\spadcommand{[a - b, a * b]\free{a b }}
\indentrel{3}\begin{verbatim}
   (3)  [9,4]
                              Type: List IntegerMod 12
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugxProblemFinitePrimePageEmpty3}
\begin{paste}{ugxProblemFinitePrimePageEmpty3}{ugxProblemFinitePrimePagePatch3}
\pastebutton{ugxProblemFinitePrimePageEmpty3}{\showpaste}
\tab{5}\spadcommand{[a - b, a * b]\free{a b }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFinitePrimePagePatch4}
\begin{paste}{ugxProblemFinitePrimePageFull4}{ugxProblemFinitePrimePageEmpty4}
\pastebutton{ugxProblemFinitePrimePageFull4}{\hidepaste}
\tab{5}\spadcommand{a / b\free{a b }}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFinitePrimePageEmpty4}
\begin{paste}{ugxProblemFinitePrimePageEmpty4}{ugxProblemFinitePrimePagePatch4}
\pastebutton{ugxProblemFinitePrimePageEmpty4}{\showpaste}
\tab{5}\spadcommand{a / b\free{a b }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFinitePrimePagePatch5}
\begin{paste}{ugxProblemFinitePrimePageFull5}{ugxProblemFinitePrimePageEmpty5}
\pastebutton{ugxProblemFinitePrimePageFull5}{\hidepaste}
\tab{5}\spadcommand{recip a\free{a }}
\indentrel{3}\begin{verbatim}
   (4)  "failed"
                             Type: Union("failed",...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFinitePrimePageEmpty5}
\begin{paste}{ugxProblemFinitePrimePageEmpty5}{ugxProblemFinitePrimePagePatch5}
\pastebutton{ugxProblemFinitePrimePageEmpty5}{\showpaste}
\tab{5}\spadcommand{recip a\free{a }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFinitePrimePagePatch6}
\begin{paste}{ugxProblemFinitePrimePageFull6}{ugxProblemFinitePrimePageEmpty6}
\pastebutton{ugxProblemFinitePrimePageFull6}{\hidepaste}
\tab{5}\spadcommand{recip b\free{b }}
\indentrel{3}\begin{verbatim}
   (5)  7
                        Type: Union(IntegerMod 12,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFinitePrimePageEmpty6}
\begin{paste}{ugxProblemFinitePrimePageEmpty6}{ugxProblemFinitePrimePagePatch6}
\pastebutton{ugxProblemFinitePrimePageEmpty6}{\showpaste}
```

```
\tab{5}\spadcommand{recip b\free{b }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFinitePrimePagePatch7}
\begin{paste}{ugxProblemFinitePrimePageFull7}{ugxProblemFinitePrimePageEmpty7}
\pastebutton{ugxProblemFinitePrimePageFull7}{\hidepaste}
\tab{5}\spadcommand{c : PrimeField 11 := 8\bound{c }}
\indentrel{3}\begin{verbatim}
   (6)  8
                                    Type: PrimeField 11
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFinitePrimePageEmpty7}
\begin{paste}{ugxProblemFinitePrimePageEmpty7}{ugxProblemFinitePrimePagePatch7}
\pastebutton{ugxProblemFinitePrimePageEmpty7}{\showpaste}
\tab{5}\spadcommand{c : PrimeField 11 := 8\bound{c }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFinitePrimePagePatch8}
\begin{paste}{ugxProblemFinitePrimePageFull8}{ugxProblemFinitePrimePageEmpty8}
\pastebutton{ugxProblemFinitePrimePageFull8}{\hidepaste}
\tab{5}\spadcommand{inv c\free{c }}
\indentrel{3}\begin{verbatim}
   (7)  7
                                    Type: PrimeField 11
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFinitePrimePageEmpty8}
\begin{paste}{ugxProblemFinitePrimePageEmpty8}{ugxProblemFinitePrimePagePatch8}
\pastebutton{ugxProblemFinitePrimePageEmpty8}{\showpaste}
\tab{5}\spadcommand{inv c\free{c }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFinitePrimePagePatch9}
\begin{paste}{ugxProblemFinitePrimePageFull9}{ugxProblemFinitePrimePageEmpty9}
\pastebutton{ugxProblemFinitePrimePageFull9}{\hidepaste}
\tab{5}\spadcommand{9/c\free{c }}
\indentrel{3}\begin{verbatim}
   (8)  8
                                    Type: PrimeField 11
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFinitePrimePageEmpty9}
\begin{paste}{ugxProblemFinitePrimePageEmpty9}{ugxProblemFinitePrimePagePatch9}
\pastebutton{ugxProblemFinitePrimePageEmpty9}{\showpaste}
\tab{5}\spadcommand{9/c\free{c }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugxProblemFinitePrimePagePatch10}
\begin{paste}{ugxProblemFinitePrimePageFull10}{ugxProblemFinitePrimePageEmpty10}
\pastebutton{ugxProblemFinitePrimePageFull10}{\hidepaste}
\tab{5}\spadcommand{GF101 := PF 101\bound{GF101 }}
\indentrel{3}\begin{verbatim}
   (9)  PrimeField 101
                                           Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFinitePrimePageEmpty10}
\begin{paste}{ugxProblemFinitePrimePageEmpty10}{ugxProblemFinitePrimePagePatch10}
\pastebutton{ugxProblemFinitePrimePageEmpty10}{\showpaste}
\tab{5}\spadcommand{GF101 := PF 101\bound{GF101 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFinitePrimePagePatch11}
\begin{paste}{ugxProblemFinitePrimePageFull11}{ugxProblemFinitePrimePageEmpty11}
\pastebutton{ugxProblemFinitePrimePageFull11}{\hidepaste}
\tab{5}\spadcommand{x := random()$GF101\bound{x }\free{GF101 }}
\indentrel{3}\begin{verbatim}
   (10)  78
                                   Type: PrimeField 101
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFinitePrimePageEmpty11}
\begin{paste}{ugxProblemFinitePrimePageEmpty11}{ugxProblemFinitePrimePagePatch11}
\pastebutton{ugxProblemFinitePrimePageEmpty11}{\showpaste}
\tab{5}\spadcommand{x := random()$GF101\bound{x }\free{GF101 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFinitePrimePagePatch12}
\begin{paste}{ugxProblemFinitePrimePageFull12}{ugxProblemFinitePrimePageEmpty12}
\pastebutton{ugxProblemFinitePrimePageFull12}{\hidepaste}
\tab{5}\spadcommand{y : GF101 := 37\bound{y }\free{GF101 }}
\indentrel{3}\begin{verbatim}
   (11)  37
                                   Type: PrimeField 101
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFinitePrimePageEmpty12}
\begin{paste}{ugxProblemFinitePrimePageEmpty12}{ugxProblemFinitePrimePagePatch12}
\pastebutton{ugxProblemFinitePrimePageEmpty12}{\showpaste}
\tab{5}\spadcommand{y : GF101 := 37\bound{y }\free{GF101 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFinitePrimePagePatch13}
```

```
\begin{paste}{ugxProblemFinitePrimePageFull13}{ugxProblemFinitePrimePageEmpty13}
\pastebutton{ugxProblemFinitePrimePageFull13}{\hidepaste}
\tab{5}\spadcommand{z := x/y\bound{z }\free{x y }}
\indentrel{3}\begin{verbatim}
   (12)  84
                                   Type: PrimeField 101
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFinitePrimePageEmpty13}
\begin{paste}{ugxProblemFinitePrimePageEmpty13}{ugxProblemFinitePrimePagePatch13}
\pastebutton{ugxProblemFinitePrimePageEmpty13}{\showpaste}
\tab{5}\spadcommand{z := x/y\bound{z }\free{x y }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFinitePrimePagePatch14}
\begin{paste}{ugxProblemFinitePrimePageFull14}{ugxProblemFinitePrimePageEmpty14}
\pastebutton{ugxProblemFinitePrimePageFull14}{\hidepaste}
\tab{5}\spadcommand{z * y - x\free{x y z }}
\indentrel{3}\begin{verbatim}
   (13)  0
                                   Type: PrimeField 101
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFinitePrimePageEmpty14}
\begin{paste}{ugxProblemFinitePrimePageEmpty14}{ugxProblemFinitePrimePagePatch14}
\pastebutton{ugxProblemFinitePrimePageEmpty14}{\showpaste}
\tab{5}\spadcommand{z * y - x\free{x y z }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFinitePrimePagePatch15}
\begin{paste}{ugxProblemFinitePrimePageFull15}{ugxProblemFinitePrimePageEmpty15}
\pastebutton{ugxProblemFinitePrimePageFull15}{\hidepaste}
\tab{5}\spadcommand{pe := primitiveElement()$GF101\bound{pe }\free{GF101 }}
\indentrel{3}\begin{verbatim}
   (14)  2
                                   Type: PrimeField 101
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFinitePrimePageEmpty15}
\begin{paste}{ugxProblemFinitePrimePageEmpty15}{ugxProblemFinitePrimePagePatch15}
\pastebutton{ugxProblemFinitePrimePageEmpty15}{\showpaste}
\tab{5}\spadcommand{pe := primitiveElement()$GF101\bound{pe }\free{GF101 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFinitePrimePagePatch16}
\begin{paste}{ugxProblemFinitePrimePageFull16}{ugxProblemFinitePrimePageEmpty16}
\pastebutton{ugxProblemFinitePrimePageFull16}{\hidepaste}
```

```
\tab{5}\spadcommand{[pe**i for i in 0..99]\free{pe }}
\indentrel{3}\begin{verbatim}
   (15)
   [1, 2, 4, 8, 16, 32, 64, 27, 54, 7, 14, 28, 56, 11,
    22, 44, 88, 75, 49, 98, 95, 89, 77, 53, 5, 10, 20,
    40, 80, 59, 17, 34, 68, 35, 70, 39, 78, 55, 9, 18,
    36, 72, 43, 86, 71, 41, 82, 63, 25, 50, 100, 99, 97,
    93, 85, 69, 37, 74, 47, 94, 87, 73, 45, 90, 79, 57,
    13, 26, 52, 3, 6, 12, 24, 48, 96, 91, 81, 61, 21, 42,
    84, 67, 33, 66, 31, 62, 23, 46, 92, 83, 65, 29, 58,
    15, 30, 60, 19, 38, 76, 51]
                              Type: List PrimeField 101
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFinitePrimePageEmpty16}
\begin{paste}{ugxProblemFinitePrimePageEmpty16}{ugxProblemFinitePrimePagePatch16}
\pastebutton{ugxProblemFinitePrimePageEmpty16}{\showpaste}
\tab{5}\spadcommand{[pe**i for i in 0..99]\free{pe }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFinitePrimePagePatch17}
\begin{paste}{ugxProblemFinitePrimePageFull17}{ugxProblemFinitePrimePageEmpty17}
\pastebutton{ugxProblemFinitePrimePageFull17}{\hidepaste}
\tab{5}\spadcommand{ex := discreteLog(y)\bound{ex }\free{y }}
\indentrel{3}\begin{verbatim}
   (16)  56
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFinitePrimePageEmpty17}
\begin{paste}{ugxProblemFinitePrimePageEmpty17}{ugxProblemFinitePrimePagePatch17}
\pastebutton{ugxProblemFinitePrimePageEmpty17}{\showpaste}
\tab{5}\spadcommand{ex := discreteLog(y)\bound{ex }\free{y }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFinitePrimePagePatch18}
\begin{paste}{ugxProblemFinitePrimePageFull18}{ugxProblemFinitePrimePageEmpty18}
\pastebutton{ugxProblemFinitePrimePageFull18}{\hidepaste}
\tab{5}\spadcommand{pe ** ex\free{ex pe }}
\indentrel{3}\begin{verbatim}
   (17)  37
                                   Type: PrimeField 101
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFinitePrimePageEmpty18}
\begin{paste}{ugxProblemFinitePrimePageEmpty18}{ugxProblemFinitePrimePagePatch18}
\pastebutton{ugxProblemFinitePrimePageEmpty18}{\showpaste}
```

```
\tab{5}\spadcommand{pe ** ex\free{ex pe }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFinitePrimePagePatch19}
\begin{paste}{ugxProblemFinitePrimePageFull19}{ugxProblemFinitePrimePageEmpty19}
\pastebutton{ugxProblemFinitePrimePageFull19}{\hidepaste}
\tab{5}\spadcommand{order y\free{y }}
\indentrel{3}\begin{verbatim}
   (18)  25
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFinitePrimePageEmpty19}
\begin{paste}{ugxProblemFinitePrimePageEmpty19}{ugxProblemFinitePrimePagePatch19}
\pastebutton{ugxProblemFinitePrimePageEmpty19}{\showpaste}
\tab{5}\spadcommand{order y\free{y }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFinitePrimePagePatch20}
\begin{paste}{ugxProblemFinitePrimePageFull20}{ugxProblemFinitePrimePageEmpty20}
\pastebutton{ugxProblemFinitePrimePageFull20}{\hidepaste}
\tab{5}\spadcommand{order pe\free{pe }}
\indentrel{3}\begin{verbatim}
   (19)  100
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFinitePrimePageEmpty20}
\begin{paste}{ugxProblemFinitePrimePageEmpty20}{ugxProblemFinitePrimePagePatch20}
\pastebutton{ugxProblemFinitePrimePageEmpty20}{\showpaste}
\tab{5}\spadcommand{order pe\free{pe }}
\end{paste}\end{patch}
```

———-

Extensions of Finite Fields

— ug08.ht —

```
\begin{page}{ugxProblemFiniteExtensionFinitePage}
{8.11.2. Extensions of Finite Fields}
\beginscroll

When you want to work with an extension of a finite field in Axiom,
you have three choices to make:
```

```
\indent{4}
\beginitems
\item[1. ] Do you want to generate an extension of the prime field
(for example, \axiomType{PrimeField 2}) or an extension of a given field?
%
\item[2. ] Do you want to use a representation that is particularly
efficient for multiplication, exponentiation and addition but
uses a lot of computer memory (a representation that models the cyclic
group structure of the multiplicative group of the field extension
and uses a Zech logarithm table),  one that
uses a normal basis for the vector space structure of the field
extension, or one that performs arithmetic modulo an irreducible
polynomial?
The cyclic group representation is only usable up to ``medium''
(relative to your machine's performance)
sized fields.
If the field is large and the normal basis is relatively simple,
the normal basis representation is more efficient for exponentiation than
the irreducible polynomial representation.
%
\item[3. ] Do you want to provide a polynomial explicitly, a root of which
``generates'' the extension in one of the three senses in (2),
or do you wish to have the polynomial generated for you?
\enditems
\indent{0}

This illustrates one of the most important features of Axiom:
you can choose exactly the right data-type and representation to
suit your application best.

We first tell you what domain constructors to use for each case
above, and then give some examples.

\texht{\hangafter=1\hangindent=2pc}{\noindent}
Constructors that automatically generate extensions of the prime field:
\newline
\axiomType{FiniteField} \newline
\axiomType{FiniteFieldCyclicGroup} \newline
\axiomType{FiniteFieldNormalBasis}

\texht{\hangafter=1\hangindent=2pc}{\noindent}
Constructors that generate extensions of an arbitrary field:
\newline
\axiomType{FiniteFieldExtension} \newline
\axiomType{FiniteFieldExtensionByPolynomial} \newline
\axiomType{FiniteFieldCyclicGroupExtension} \newline
\axiomType{FiniteFieldCyclicGroupExtensionByPolynomial} \newline
\axiomType{FiniteFieldNormalBasisExtension} \newline
\axiomType{FiniteFieldNormalBasisExtensionByPolynomial}
```

```
\texht{\hangafter=1\hangindent=2pc}{\noindent}
Constructors that use a cyclic group representation:
\newline
\axiomType{FiniteFieldCyclicGroup} \newline
\axiomType{FiniteFieldCyclicGroupExtension} \newline
\axiomType{FiniteFieldCyclicGroupExtensionByPolynomial}

\texht{\hangafter=1\hangindent=2pc}{\noindent}
Constructors that use a normal basis representation:
\newline
\axiomType{FiniteFieldNormalBasis} \newline
\axiomType{FiniteFieldNormalBasisExtension} \newline
\axiomType{FiniteFieldNormalBasisExtensionByPolynomial}

\texht{\hangafter=1\hangindent=2pc}{\noindent}
Constructors that use an irreducible modulus polynomial representation:
\newline
\axiomType{FiniteField} \newline
\axiomType{FiniteFieldExtension} \newline
\axiomType{FiniteFieldExtensionByPolynomial}

\texht{\hangafter=1\hangindent=2pc}{\noindent}
Constructors that generate a polynomial for you:
\newline
\axiomType{FiniteField} \newline
\axiomType{FiniteFieldExtension} \newline
\axiomType{FiniteFieldCyclicGroup} \newline
\axiomType{FiniteFieldCyclicGroupExtension} \newline
\axiomType{FiniteFieldNormalBasis} \newline
\axiomType{FiniteFieldNormalBasisExtension}

\texht{\hangafter=1\hangindent=2pc}{\noindent}
Constructors for which you provide a polynomial:
\newline
\axiomType{FiniteFieldExtensionByPolynomial} \newline
\axiomType{FiniteFieldCyclicGroupExtensionByPolynomial} \newline
\axiomType{FiniteFieldNormalBasisExtensionByPolynomial}

These constructors are discussed in the following sections where
we collect together descriptions of extension fields that have the
same underlying representation.\footnote{For
more information on the implementation aspects of finite
fields, see
J. Grabmeier, A. Scheerhorn, {\it Finite Fields in AXIOM,}
Technical Report, IBM Heidelberg Scientific Center, 1992.}

If you don't really care about all this detail, just use
\axiomType{FiniteField}.
As your knowledge of your application and its Axiom implementation
grows, you can come back and choose an alternative constructor that
```

```
may improve the efficiency of your code.
Note that the exported operations are almost the same for all constructors
of finite field extensions and include the operations exported by
\axiomType{PrimeField}.

\endscroll
\autobuttons
\end{page}
```

———-

Irreducible Mod Polynomial Representations

⇒ “notitle” (ugxProblemFiniteExtensionFinitePage) 12 on page 2246

— **ug08.ht** —

```
\begin{page}{ugxProblemFiniteModulusPage}
{8.11.3. Irreducible Mod Polynomial Representations}
\beginscroll

All finite field extension constructors discussed in this
section
use a representation that performs arithmetic with univariate
(one-variable) polynomials modulo an irreducible polynomial.
This polynomial may be given explicitly by you or automatically
generated.
The ground field may be the prime field or one you specify.
See
\downlink{‘‘Extensions of Finite Fields’’}
{ugxProblemFiniteExtensionFinitePage} in Section 8.11.2
\ignore{ugxProblemFiniteExtensionFinite}
for general information about finite field extensions.

For \axiomType{FiniteField} (abbreviation \axiomType{FF}) you provide a
prime number \smath{p} and an extension degree \smath{n}.
This degree can be 1.
%
\xtc{
Axiom uses the prime field \axiomType{PrimeField(p)},
here \axiomType{PrimeField 2},
and it chooses an irreducible polynomial of degree \smath{n},
here 12, over the ground field.
}{
\spadpaste{GF4096 := FF(2,12); \bound{GF4096}}
}
%
```

```
The objects in the generated field extension are polynomials
of degree at most \smath{n-1} with coefficients in the
prime field.
The polynomial indeterminate is automatically chosen by Axiom and
is typically something like \axiom{\%A} or \axiom{\%D}.
These (strange) variables are {\it only} for output display;
there are several ways to construct elements of this field.

The operation \axiomFun{index} enumerates the elements of the field
extension and accepts as argument the integers from 1 to
\smath{p \texht{^}{**} n}.
%
\xtc{
The expression
\axiom{index(p)} always gives the indeterminate.
}{
\spadpaste{a := index(2)\$GF4096 \bound{a}\free{GF4096}}
}
%
%
\xtc{
You can build polynomials in \smath{a} and calculate in
\axiom{GF4096}.
}{
\spadpaste{b := a**12 - a**5 + a \bound{b}\free{a}}
}
\xtc{
}{
\spadpaste{b ** 1000 \free{b}}
}
\xtc{
}{
\spadpaste{c := a/b \free{a b}\bound{c}}
}
%
\xtc{
Among the available operations are \axiomFun{norm} and \axiomFun{trace}.
}{
\spadpaste{norm c \free{c}}
}
\xtc{
}{
\spadpaste{trace c \free{c}}
}
%
%

Since any nonzero element is a power of a primitive element, how
do we discover what the exponent is?
%
```

```
\xtc{
The operation \axiomFun{discreteLog} calculates
the exponent and,
if it is called with only one argument, always refers to the primitive
element returned by \axiomFun{primitiveElement}.
}{
\spadpaste{dL := discreteLog a \free{a}\bound{dL}}
}
\xtc{
}{
\spadpaste{g ** dL \free{dL g}}
}

\axiomType{FiniteFieldExtension} (abbreviation \axiomType{FFX}) is
similar to \axiomType{FiniteField} except that the
ground-field for \axiomType{FiniteFieldExtension} is arbitrary and
chosen by you.
%
\xtc{
In case you select the prime field as ground field, there is
essentially no difference between the constructed two finite field
extensions.
}{
\spadpaste{GF16 := FF(2,4); \bound{GF16}}
}
\xtc{
}{
\spadpaste{GF4096 := FFX(GF16,3); \bound{GF4096x}\free{GF16}}
}
\xtc{
}{
\spadpaste{r := (random()\$GF4096) ** 20 \free{GF4096x}\bound{r}}
}
\xtc{
}{
\spadpaste{norm(r) \free{r}}
}
%

\axiomType{FiniteFieldExtensionByPolynomial} (abbreviation \axiomType{FFP})
is similar to \axiomType{FiniteField} and \axiomType{FiniteFieldExtension}
but is more general.
%
\xtc{
}{
\spadpaste{GF4 := FF(2,2); \bound{GF4}}
}
\xtc{
}{
\spadpaste{f := nextIrreduciblePoly(random(6)\$FFPOLY(GF4))\$FFPOLY(GF4)
```

```
\free{GF4}\bound{f}}
}
\xtc{
For \axiomType{FFP} you choose both the
ground field and the irreducible polynomial used in the representation.
The degree of the extension is the degree of the polynomial.
}{
\spadpaste{GF4096 := FFP(GF4,f); \bound{GF4096y}\free{f GF4}}
}
\xtc{
}{
\spadpaste{discreteLog random()\$GF4096 \free{GF4096y}}
}
%

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxProblemFiniteModulusPagePatch1}
\begin{paste}{ugxProblemFiniteModulusPageFull1}{ugxProblemFiniteModulusPageEmpty1}
\pastebutton{ugxProblemFiniteModulusPageFull1}{\hidepaste}
\tab{5}\spadcommand{GF4096 := FF(2,12);\bound{GF4096 }}
\indentrel{3}\begin{verbatim}
                                           Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteModulusPageEmpty1}
\begin{paste}{ugxProblemFiniteModulusPageEmpty1}{ugxProblemFiniteModulusPagePatch1}
\pastebutton{ugxProblemFiniteModulusPageEmpty1}{\showpaste}
\tab{5}\spadcommand{GF4096 := FF(2,12);\bound{GF4096 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteModulusPagePatch2}
\begin{paste}{ugxProblemFiniteModulusPageFull2}{ugxProblemFiniteModulusPageEmpty2}
\pastebutton{ugxProblemFiniteModulusPageFull2}{\hidepaste}
\tab{5}\spadcommand{a := index(2)$GF4096\bound{a }\free{GF4096 }}
\indentrel{3}\begin{verbatim}
   (2)  %CS
                                Type: FiniteField(2,12)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteModulusPageEmpty2}
\begin{paste}{ugxProblemFiniteModulusPageEmpty2}{ugxProblemFiniteModulusPagePatch2}
\pastebutton{ugxProblemFiniteModulusPageEmpty2}{\showpaste}
\tab{5}\spadcommand{a := index(2)$GF4096\bound{a }\free{GF4096 }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugxProblemFiniteModulusPagePatch3}
\begin{paste}{ugxProblemFiniteModulusPageFull3}{ugxProblemFiniteModulusPageEmpty3}
\pastebutton{ugxProblemFiniteModulusPageFull3}{\hidepaste}
\tab{5}\spadcommand{b := a**12 - a**5 + a\bound{b }\free{a }}
\indentrel{3}\begin{verbatim}
           5      3
   (3)  %CS  + %CS  + %CS + 1
                                Type: FiniteField(2,12)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteModulusPageEmpty3}
\begin{paste}{ugxProblemFiniteModulusPageEmpty3}{ugxProblemFiniteModulusPagePatch3}
\pastebutton{ugxProblemFiniteModulusPageEmpty3}{\showpaste}
\tab{5}\spadcommand{b := a**12 - a**5 + a\bound{b }\free{a }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteModulusPagePatch4}
\begin{paste}{ugxProblemFiniteModulusPageFull4}{ugxProblemFiniteModulusPageEmpty4}
\pastebutton{ugxProblemFiniteModulusPageFull4}{\hidepaste}
\tab{5}\spadcommand{b ** 1000\free{b }}
\indentrel{3}\begin{verbatim}
           10      9      7      5      4      3
   (4)  %CS   + %CS  + %CS  + %CS  + %CS  + %CS  + %CS
                                Type: FiniteField(2,12)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteModulusPageEmpty4}
\begin{paste}{ugxProblemFiniteModulusPageEmpty4}{ugxProblemFiniteModulusPagePatch4}
\pastebutton{ugxProblemFiniteModulusPageEmpty4}{\showpaste}
\tab{5}\spadcommand{b ** 1000\free{b }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteModulusPagePatch5}
\begin{paste}{ugxProblemFiniteModulusPageFull5}{ugxProblemFiniteModulusPageEmpty5}
\pastebutton{ugxProblemFiniteModulusPageFull5}{\hidepaste}
\tab{5}\spadcommand{c := a/b\free{a b }\bound{c }}
\indentrel{3}\begin{verbatim}
           11      8      7      5      4      3      2
   (5)  %CS   + %CS  + %CS  + %CS  + %CS  + %CS  + %CS
                                Type: FiniteField(2,12)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteModulusPageEmpty5}
\begin{paste}{ugxProblemFiniteModulusPageEmpty5}{ugxProblemFiniteModulusPagePatch5}
\pastebutton{ugxProblemFiniteModulusPageEmpty5}{\showpaste}
\tab{5}\spadcommand{c := a/b\free{a b }\bound{c }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugxProblemFiniteModulusPagePatch6}
\begin{paste}{ugxProblemFiniteModulusPageFull6}{ugxProblemFiniteModulusPageEmpty6}
\pastebutton{ugxProblemFiniteModulusPageFull6}{\hidepaste}
\tab{5}\spadcommand{norm c\free{c }}
\indentrel{3}\begin{verbatim}
   (6)  1
                                     Type: PrimeField 2
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteModulusPageEmpty6}
\begin{paste}{ugxProblemFiniteModulusPageEmpty6}{ugxProblemFiniteModulusPagePatch6}
\pastebutton{ugxProblemFiniteModulusPageEmpty6}{\showpaste}
\tab{5}\spadcommand{norm c\free{c }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteModulusPagePatch7}
\begin{paste}{ugxProblemFiniteModulusPageFull7}{ugxProblemFiniteModulusPageEmpty7}
\pastebutton{ugxProblemFiniteModulusPageFull7}{\hidepaste}
\tab{5}\spadcommand{trace c\free{c }}
\indentrel{3}\begin{verbatim}
   (7)  0
                                     Type: PrimeField 2
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteModulusPageEmpty7}
\begin{paste}{ugxProblemFiniteModulusPageEmpty7}{ugxProblemFiniteModulusPagePatch7}
\pastebutton{ugxProblemFiniteModulusPageEmpty7}{\showpaste}
\tab{5}\spadcommand{trace c\free{c }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteModulusPagePatch8}
\begin{paste}{ugxProblemFiniteModulusPageFull8}{ugxProblemFiniteModulusPageEmpty8}
\pastebutton{ugxProblemFiniteModulusPageFull8}{\hidepaste}
\tab{5}\spadcommand{dL := discreteLog a\free{a }\bound{dL }}
\indentrel{3}\begin{verbatim}
   (8)  1729
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteModulusPageEmpty8}
\begin{paste}{ugxProblemFiniteModulusPageEmpty8}{ugxProblemFiniteModulusPagePatch8}
\pastebutton{ugxProblemFiniteModulusPageEmpty8}{\showpaste}
\tab{5}\spadcommand{dL := discreteLog a\free{a }\bound{dL }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteModulusPagePatch9}
```

```
\begin{paste}{ugxProblemFiniteModulusPageFull9}{ugxProblemFiniteModulusPageEmpty9}
\pastebutton{ugxProblemFiniteModulusPageFull9}{\hidepaste}
\tab{5}\spadcommand{g ** dL\free{dL g }}
\indentrel{3}\begin{verbatim}
         1729
   (9)  g
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteModulusPageEmpty9}
\begin{paste}{ugxProblemFiniteModulusPageEmpty9}{ugxProblemFiniteModulusPagePatch9}
\pastebutton{ugxProblemFiniteModulusPageEmpty9}{\showpaste}
\tab{5}\spadcommand{g ** dL\free{dL g }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteModulusPagePatch10}
\begin{paste}{ugxProblemFiniteModulusPageFull10}{ugxProblemFiniteModulusPageEmpty10}
\pastebutton{ugxProblemFiniteModulusPageFull10}{\hidepaste}
\tab{5}\spadcommand{GF16 := FF(2,4);\bound{GF16 }}
\indentrel{3}\begin{verbatim}
                                           Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteModulusPageEmpty10}
\begin{paste}{ugxProblemFiniteModulusPageEmpty10}{ugxProblemFiniteModulusPagePatch10}
\pastebutton{ugxProblemFiniteModulusPageEmpty10}{\showpaste}
\tab{5}\spadcommand{GF16 := FF(2,4);\bound{GF16 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteModulusPagePatch11}
\begin{paste}{ugxProblemFiniteModulusPageFull11}{ugxProblemFiniteModulusPageEmpty11}
\pastebutton{ugxProblemFiniteModulusPageFull11}{\hidepaste}
\tab{5}\spadcommand{GF4096 := FFX(GF16,3);\bound{GF4096x }\free{GF16 }}
\indentrel{3}\begin{verbatim}
                                           Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteModulusPageEmpty11}
\begin{paste}{ugxProblemFiniteModulusPageEmpty11}{ugxProblemFiniteModulusPagePatch11}
\pastebutton{ugxProblemFiniteModulusPageEmpty11}{\showpaste}
\tab{5}\spadcommand{GF4096 := FFX(GF16,3);\bound{GF4096x }\free{GF16 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteModulusPagePatch12}
\begin{paste}{ugxProblemFiniteModulusPageFull12}{ugxProblemFiniteModulusPageEmpty12}
\pastebutton{ugxProblemFiniteModulusPageFull12}{\hidepaste}
\tab{5}\spadcommand{r := (random()$GF4096) ** 20\free{GF4096x }\bound{r }}
```

```
\indentrel{3}\begin{verbatim}
             3              2
   (12)  (%CT  + %CT + 1)%CU  + %CT %CU
         Type: FiniteFieldExtension(FiniteField(2,4),3)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteModulusPageEmpty12}
\begin{paste}{ugxProblemFiniteModulusPageEmpty12}{ugxProblemFiniteModulusPagePatch12}
\pastebutton{ugxProblemFiniteModulusPageEmpty12}{\showpaste}
\tab{5}\spadcommand{r := (random()$GF4096) ** 20\free{GF4096x }\bound{r }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteModulusPagePatch13}
\begin{paste}{ugxProblemFiniteModulusPageFull13}{ugxProblemFiniteModulusPageEmpty13}
\pastebutton{ugxProblemFiniteModulusPageFull13}{\hidepaste}
\tab{5}\spadcommand{norm(r)\free{r }}
\indentrel{3}\begin{verbatim}
            2
   (13)  %CT  + %CT
                                 Type: FiniteField(2,4)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteModulusPageEmpty13}
\begin{paste}{ugxProblemFiniteModulusPageEmpty13}{ugxProblemFiniteModulusPagePatch13}
\pastebutton{ugxProblemFiniteModulusPageEmpty13}{\showpaste}
\tab{5}\spadcommand{norm(r)\free{r }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteModulusPagePatch14}
\begin{paste}{ugxProblemFiniteModulusPageFull14}{ugxProblemFiniteModulusPageEmpty14}
\pastebutton{ugxProblemFiniteModulusPageFull14}{\hidepaste}
\tab{5}\spadcommand{GF4 := FF(2,2);\bound{GF4 }}
\indentrel{3}\begin{verbatim}
                                           Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteModulusPageEmpty14}
\begin{paste}{ugxProblemFiniteModulusPageEmpty14}{ugxProblemFiniteModulusPagePatch14}
\pastebutton{ugxProblemFiniteModulusPageEmpty14}{\showpaste}
\tab{5}\spadcommand{GF4 := FF(2,2);\bound{GF4 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteModulusPagePatch15}
\begin{paste}{ugxProblemFiniteModulusPageFull15}{ugxProblemFiniteModulusPageEmpty15}
\pastebutton{ugxProblemFiniteModulusPageFull15}{\hidepaste}
\tab{5}\spadcommand{f := nextIrreduciblePoly(random(6)$FFPOLY(GF4))$FFPOLY(GF4)\free{GF4 }\bound
\indentrel{3}\begin{verbatim}
```

```
         6    5             4             2
   (15)  ?  + ?  + (%CV + 1)?  + (%CV + 1)?  + ? + %CV
Type: Union(SparseUnivariatePolynomial FiniteField(2,2),...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteModulusPageEmpty15}
\begin{paste}{ugxProblemFiniteModulusPageEmpty15}{ugxProblemFiniteModulusPagePatch15}
\pastebutton{ugxProblemFiniteModulusPageEmpty15}{\showpaste}
\tab{5}\spadcommand{f := nextIrreduciblePoly(random(6)$FFPOLY(GF4))$FFPOLY(GF4)\free{GF4 }\bound{f }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteModulusPagePatch16}
\begin{paste}{ugxProblemFiniteModulusPageFull16}{ugxProblemFiniteModulusPageEmpty16}
\pastebutton{ugxProblemFiniteModulusPageFull16}{\hidepaste}
\tab{5}\spadcommand{GF4096 := FFP(GF4,f);\bound{GF4096y }\free{f GF4 }}
\indentrel{3}\begin{verbatim}
                                             Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteModulusPageEmpty16}
\begin{paste}{ugxProblemFiniteModulusPageEmpty16}{ugxProblemFiniteModulusPagePatch16}
\pastebutton{ugxProblemFiniteModulusPageEmpty16}{\showpaste}
\tab{5}\spadcommand{GF4096 := FFP(GF4,f);\bound{GF4096y }\free{f GF4 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteModulusPagePatch17}
\begin{paste}{ugxProblemFiniteModulusPageFull17}{ugxProblemFiniteModulusPageEmpty17}
\pastebutton{ugxProblemFiniteModulusPageFull17}{\hidepaste}
\tab{5}\spadcommand{discreteLog random()$GF4096\free{GF4096y }}
\indentrel{3}\begin{verbatim}
   (17)  3370
                                    Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteModulusPageEmpty17}
\begin{paste}{ugxProblemFiniteModulusPageEmpty17}{ugxProblemFiniteModulusPagePatch17}
\pastebutton{ugxProblemFiniteModulusPageEmpty17}{\showpaste}
\tab{5}\spadcommand{discreteLog random()$GF4096\free{GF4096y }}
\end{paste}\end{patch}
```

Cyclic Group Representations

⇒ "notitle" (ugxProblemFiniteUtilityPage) 12 on page 2280

— ug08.ht —

```
\begin{page}{ugxProblemFiniteCyclicPage}
{8.11.4. Cyclic Group Representations}
\beginscroll

In every finite field there exist elements whose powers are all the
nonzero elements of the field.
Such an element is called a {\it primitive element}.

In \axiomType{FiniteFieldCyclicGroup} (abbreviation \axiomType{FFCG})
the nonzero elements are represented by the powers of a fixed
primitive element of the field (that is, a generator of its cyclic
multiplicative group).  Multiplication (and hence exponentiation)
using this representation is easy.  To do addition, we consider our
primitive element as the root of a primitive polynomial (an
irreducible polynomial whose roots are all primitive).  See
\downlink{``Utility Operations for Finite Fields''}
{ugxProblemFiniteUtilityPage} in Section
8.11.7\ignore{ugxProblemFiniteUtility} for
examples of how to compute such a polynomial.

%
\xtc{
To use \axiomType{FiniteFieldCyclicGroup} you provide a prime number and an
extension degree.
}{
\spadpaste{GF81 := FFCG(3,4); \bound{GF81}}
}
%
%
\xtc{
Axiom uses the prime field, here \axiomType{PrimeField 3}, as the
ground field and it chooses a primitive polynomial of degree
\smath{n}, here 4, over the prime field.
}{
\spadpaste{a := primitiveElement()\$GF81 \bound{a}\free{GF81}}
}
%
%
\xtc{
You can calculate in \axiom{GF81}.
}{
\spadpaste{b  := a**12 - a**5 + a \bound{b}\free{a}}
}
%
```

```
\xtc{
In this representation of finite fields the discrete logarithm
of an element can be seen directly in its output form.
}{
\spadpaste{b \free{b}}
}
\xtc{
}{
\spadpaste{discreteLog b \free{b}}
}
%

\axiomType{FiniteFieldCyclicGroupExtension} (abbreviation
\axiomType{FFCGX}) is similar to \axiomType{FiniteFieldCyclicGroup}
except that the ground field for
\axiomType{FiniteFieldCyclicGroupExtension} is arbitrary and chosen
by you.
In case you select the prime field as ground field, there is
essentially no difference between the constructed two finite field
extensions.
%
\xtc{
}{
\spadpaste{GF9 := FF(3,2); \bound{GF9}}
}
\xtc{
}{
\spadpaste{GF729 := FFCGX(GF9,3); \bound{GF729}\free{GF9}}
}
\xtc{
}{
\spadpaste{r := (random()\$GF729) ** 20 \free{GF729}\bound{r}}
}
\xtc{
}{
\spadpaste{trace(r) \free{r}}
}
%

\axiomType{FiniteFieldCyclicGroupExtensionByPolynomial} (abbreviation
\axiomType{FFCGP}) is similar to \axiomType{FiniteFieldCyclicGroup}
and \axiomType{FiniteFieldCyclicGroupExtension} but is more general.
For \axiomType{FiniteFieldCyclicGroupExtensionByPolynomial} you choose
both the ground field and the irreducible polynomial used in the
representation.  The degree of the extension is the degree of the
polynomial.

\xtc{
}{
\spadpaste{GF3  := PrimeField 3; \bound{GF3}}
```

```
}
\xtc{
We use a utility operation to generate an irreducible primitive
polynomial (see
\downlink{``Utility Operations for Finite Fields''}
{ugxProblemFiniteUtilityPage}
in Section 8.11.7\ignore{ugxProblemFiniteUtility}).
The polynomial has one variable that is ``anonymous'': it displays
as a question mark.
}{
\spadpaste{f := createPrimitivePoly(4)\$FFPOLY(GF3) \bound{f}\free{GF3}}
}
\xtc{
}{
\spadpaste{GF81 := FFCGP(GF3,f); \bound{GF81x}\free{f GF3}}
}
\xtc{
Let's look at a random element from this field.
}{
\spadpaste{random()\$GF81 \free{GF81x}}
}
%

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxProblemFiniteCyclicPagePatch1}
\begin{paste}{ugxProblemFiniteCyclicPageFull1}{ugxProblemFiniteCyclicPageEmpty1}
\pastebutton{ugxProblemFiniteCyclicPageFull1}{\hidepaste}
\tab{5}\spadcommand{GF81 := FFCG(3,4);\bound{GF81 }}
\indentrel{3}\begin{verbatim}
                                            Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteCyclicPageEmpty1}
\begin{paste}{ugxProblemFiniteCyclicPageEmpty1}{ugxProblemFiniteCyclicPagePatch1}
\pastebutton{ugxProblemFiniteCyclicPageEmpty1}{\showpaste}
\tab{5}\spadcommand{GF81 := FFCG(3,4);\bound{GF81 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteCyclicPagePatch2}
\begin{paste}{ugxProblemFiniteCyclicPageFull2}{ugxProblemFiniteCyclicPageEmpty2}
\pastebutton{ugxProblemFiniteCyclicPageFull2}{\hidepaste}
\tab{5}\spadcommand{a := primitiveElement()$GF81\bound{a }\free{GF81 }}
\indentrel{3}\begin{verbatim}
           1
   (2)  %BF
                       Type: FiniteFieldCyclicGroup(3,4)
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteCyclicPageEmpty2}
\begin{paste}{ugxProblemFiniteCyclicPageEmpty2}{ugxProblemFiniteCyclicPagePatch2}
\pastebutton{ugxProblemFiniteCyclicPageEmpty2}{\showpaste}
\tab{5}\spadcommand{a := primitiveElement()$GF81\bound{a }\free{GF81 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteCyclicPagePatch3}
\begin{paste}{ugxProblemFiniteCyclicPageFull3}{ugxProblemFiniteCyclicPageEmpty3}
\pastebutton{ugxProblemFiniteCyclicPageFull3}{\hidepaste}
\tab{5}\spadcommand{b := a**12 - a**5 + a\bound{b }\free{a }}
\indentrel{3}\begin{verbatim}
          72
   (3)  %BF
                     Type: FiniteFieldCyclicGroup(3,4)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteCyclicPageEmpty3}
\begin{paste}{ugxProblemFiniteCyclicPageEmpty3}{ugxProblemFiniteCyclicPagePatch3}
\pastebutton{ugxProblemFiniteCyclicPageEmpty3}{\showpaste}
\tab{5}\spadcommand{b := a**12 - a**5 + a\bound{b }\free{a }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteCyclicPagePatch4}
\begin{paste}{ugxProblemFiniteCyclicPageFull4}{ugxProblemFiniteCyclicPageEmpty4}
\pastebutton{ugxProblemFiniteCyclicPageFull4}{\hidepaste}
\tab{5}\spadcommand{b\free{b }}
\indentrel{3}\begin{verbatim}
          72
   (4)  %BF
                     Type: FiniteFieldCyclicGroup(3,4)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteCyclicPageEmpty4}
\begin{paste}{ugxProblemFiniteCyclicPageEmpty4}{ugxProblemFiniteCyclicPagePatch4}
\pastebutton{ugxProblemFiniteCyclicPageEmpty4}{\showpaste}
\tab{5}\spadcommand{b\free{b }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteCyclicPagePatch5}
\begin{paste}{ugxProblemFiniteCyclicPageFull5}{ugxProblemFiniteCyclicPageEmpty5}
\pastebutton{ugxProblemFiniteCyclicPageFull5}{\hidepaste}
\tab{5}\spadcommand{discreteLog b\free{b }}
\indentrel{3}\begin{verbatim}
   (5)  72
                                 Type: PositiveInteger
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteCyclicPageEmpty5}
\begin{paste}{ugxProblemFiniteCyclicPageEmpty5}{ugxProblemFiniteCyclicPagePatch5}
\pastebutton{ugxProblemFiniteCyclicPageEmpty5}{\showpaste}
\tab{5}\spadcommand{discreteLog b\free{b }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteCyclicPagePatch6}
\begin{paste}{ugxProblemFiniteCyclicPageFull6}{ugxProblemFiniteCyclicPageEmpty6}
\pastebutton{ugxProblemFiniteCyclicPageFull6}{\hidepaste}
\tab{5}\spadcommand{GF9 := FF(3,2);\bound{GF9 }}
\indentrel{3}\begin{verbatim}
                                             Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteCyclicPageEmpty6}
\begin{paste}{ugxProblemFiniteCyclicPageEmpty6}{ugxProblemFiniteCyclicPagePatch6}
\pastebutton{ugxProblemFiniteCyclicPageEmpty6}{\showpaste}
\tab{5}\spadcommand{GF9 := FF(3,2);\bound{GF9 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteCyclicPagePatch7}
\begin{paste}{ugxProblemFiniteCyclicPageFull7}{ugxProblemFiniteCyclicPageEmpty7}
\pastebutton{ugxProblemFiniteCyclicPageFull7}{\hidepaste}
\tab{5}\spadcommand{GF729 := FFCGX(GF9,3);\bound{GF729 }\free{GF9 }}
\indentrel{3}\begin{verbatim}
                                             Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteCyclicPageEmpty7}
\begin{paste}{ugxProblemFiniteCyclicPageEmpty7}{ugxProblemFiniteCyclicPagePatch7}
\pastebutton{ugxProblemFiniteCyclicPageEmpty7}{\showpaste}
\tab{5}\spadcommand{GF729 := FFCGX(GF9,3);\bound{GF729 }\free{GF9 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteCyclicPagePatch8}
\begin{paste}{ugxProblemFiniteCyclicPageFull8}{ugxProblemFiniteCyclicPageEmpty8}
\pastebutton{ugxProblemFiniteCyclicPageFull8}{\hidepaste}
\tab{5}\spadcommand{r := (random()$GF729) ** 20\free{GF729 }\bound{r }}
\indentrel{3}\begin{verbatim}
           68
   (8)  %CN
Type: FiniteFieldCyclicGroupExtension(FiniteField(3,2),3)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugxProblemFiniteCyclicPageEmpty8}
\begin{paste}{ugxProblemFiniteCyclicPageEmpty8}{ugxProblemFiniteCyclicPagePatch8}
\pastebutton{ugxProblemFiniteCyclicPageEmpty8}{\showpaste}
\tab{5}\spadcommand{r := (random()$GF729) ** 20\free{GF729 }\bound{r }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteCyclicPagePatch9}
\begin{paste}{ugxProblemFiniteCyclicPageFull9}{ugxProblemFiniteCyclicPageEmpty9}
\pastebutton{ugxProblemFiniteCyclicPageFull9}{\hidepaste}
\tab{5}\spadcommand{trace(r)\free{r }}
\indentrel{3}\begin{verbatim}
   (9)  2
                                 Type: FiniteField(3,2)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteCyclicPageEmpty9}
\begin{paste}{ugxProblemFiniteCyclicPageEmpty9}{ugxProblemFiniteCyclicPagePatch9}
\pastebutton{ugxProblemFiniteCyclicPageEmpty9}{\showpaste}
\tab{5}\spadcommand{trace(r)\free{r }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteCyclicPagePatch10}
\begin{paste}{ugxProblemFiniteCyclicPageFull10}{ugxProblemFiniteCyclicPageEmpty10}
\pastebutton{ugxProblemFiniteCyclicPageFull10}{\hidepaste}
\tab{5}\spadcommand{GF3 := PrimeField 3;\bound{GF3 }}
\indentrel{3}\begin{verbatim}
                                           Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteCyclicPageEmpty10}
\begin{paste}{ugxProblemFiniteCyclicPageEmpty10}{ugxProblemFiniteCyclicPagePatch10}
\pastebutton{ugxProblemFiniteCyclicPageEmpty10}{\showpaste}
\tab{5}\spadcommand{GF3 := PrimeField 3;\bound{GF3 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteCyclicPagePatch11}
\begin{paste}{ugxProblemFiniteCyclicPageFull11}{ugxProblemFiniteCyclicPageEmpty11}
\pastebutton{ugxProblemFiniteCyclicPageFull11}{\hidepaste}
\tab{5}\spadcommand{f := createPrimitivePoly(4)$FFPOLY(GF3)\bound{f }\free{GF3 }}
\indentrel{3}\begin{verbatim}
          4
   (11)  ?  + ? + 2
          Type: SparseUnivariatePolynomial PrimeField 3
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteCyclicPageEmpty11}
\begin{paste}{ugxProblemFiniteCyclicPageEmpty11}{ugxProblemFiniteCyclicPagePatch11}
```

```
\pastebutton{ugxProblemFiniteCyclicPageEmpty11}{\showpaste}
\tab{5}\spadcommand{f := createPrimitivePoly(4)$FFPOLY(GF3)\bound{f }\free{GF3 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteCyclicPagePatch12}
\begin{paste}{ugxProblemFiniteCyclicPageFull12}{ugxProblemFiniteCyclicPageEmpty12}
\pastebutton{ugxProblemFiniteCyclicPageFull12}{\hidepaste}
\tab{5}\spadcommand{GF81 := FFCGP(GF3,f);\bound{GF81x }\free{f GF3 }}
\indentrel{3}\begin{verbatim}
                                            Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteCyclicPageEmpty12}
\begin{paste}{ugxProblemFiniteCyclicPageEmpty12}{ugxProblemFiniteCyclicPagePatch12}
\pastebutton{ugxProblemFiniteCyclicPageEmpty12}{\showpaste}
\tab{5}\spadcommand{GF81 := FFCGP(GF3,f);\bound{GF81x }\free{f GF3 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteCyclicPagePatch13}
\begin{paste}{ugxProblemFiniteCyclicPageFull13}{ugxProblemFiniteCyclicPageEmpty13}
\pastebutton{ugxProblemFiniteCyclicPageFull13}{\hidepaste}
\tab{5}\spadcommand{random()$GF81\free{GF81x }}
\indentrel{3}\begin{verbatim}
            8
   (13)  %BF
Type: FiniteFieldCyclicGroupExtensionByPolynomial(PrimeField 3,?**4+?+2)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteCyclicPageEmpty13}
\begin{paste}{ugxProblemFiniteCyclicPageEmpty13}{ugxProblemFiniteCyclicPagePatch13}
\pastebutton{ugxProblemFiniteCyclicPageEmpty13}{\showpaste}
\tab{5}\spadcommand{random()$GF81\free{GF81x }}
\end{paste}\end{patch}
```

———-

Normal Basis Representations

⇒ “notitle” (ugxProblemFiniteUtilityPage) 12 on page 2280

— **ug08.ht** —

```
\begin{page}{ugxProblemFiniteNormalPage}
{8.11.5. Normal Basis Representations}
\beginscroll
```

```
Let \smath{K} be a finite extension of degree \smath{n} of the
finite field \smath{F} and let \smath{F} have \smath{q}
elements.
An element \smath{x} of \smath{K} is said to be
{\it normal} over \smath{F} if the elements
\centerline{{\axiom{1, x**q, x**(q**2), ..., x**(q**(n-1))}}}
form a basis of \smath{K} as a vector space over \smath{F}.
Such a basis is called a {\it normal basis}.\footnote{This
agrees with the general definition of a normal basis because the
\smath{n} distinct powers of the automorphism
\texht{$x \mapsto x^q$}{\axiom{x +-> x**q}}
constitute the Galois group of \smath{K/F}.}

If \smath{x} is normal over \smath{F}, its minimal
polynomial is also said to be {\it normal} over \smath{F}.
There exist normal bases for all finite extensions of arbitrary
finite fields.

In \axiomType{FiniteFieldNormalBasis} (abbreviation
\axiomType{FFNB}), the elements of the finite field are represented
by coordinate vectors with respect to a normal basis.

\xtc{
You provide a prime \smath{p} and an extension degree
\smath{n}.
}{
\spadpaste{K := FFNB(3,8) \bound{K}}
}

Axiom uses the prime field \axiomType{PrimeField(p)}, here
\axiomType{PrimeField 3}, and it chooses a normal polynomial of degree
\smath{n}, here 8, over the ground field.  The remainder class of the
indeterminate is used as the normal element.  The polynomial
indeterminate is automatically chosen by Axiom and is typically
something like \axiom{\%A} or \axiom{\%D}.  These (strange) variables
are only for output display; there are several ways to construct
elements of this field.  The output of the basis elements is something
like
\texht{$\%A^{q^i}.$}{
\begin{verbatim}
   i
  q
%A  .
\end{verbatim}
}
%
\xtc{
}{
\spadpaste{a := normalElement()\$K \bound{a}\free{K}}
}
```

```
%
%
\xtc{
You can calculate in \smath{K} using \smath{a}.
}{
\spadpaste{b  := a**12 - a**5 + a \bound{b}\free{a}}
}

\axiomType{FiniteFieldNormalBasisExtension}
(abbreviation \axiomType{FFNBX}) is similar to
\axiomType{FiniteFieldNormalBasis}
except that the groundfield for
\axiomType{FiniteFieldNormalBasisExtension} is arbitrary and chosen
by you.  In case you select the prime field as ground field, there
is essentially no difference between the constructed two finite field
extensions.

\xtc{
}{
\spadpaste{GF9 := FFNB(3,2); \bound{GF9}}
}
\xtc{
}{
\spadpaste{GF729 := FFNBX(GF9,3); \bound{GF729}\free{GF9}}
}
\xtc{
}{
\spadpaste{r := random()\$GF729 \bound{r}\free{GF729}}
}
\xtc{
}{
\spadpaste{r + r**3 + r**9 + r**27 \free{r}}
}

\axiomType{FiniteFieldNormalBasisExtensionByPolynomial}
(abbreviation \axiomType{FFNBP}) is similar to
\axiomType{FiniteFieldNormalBasis} and
\axiomType{FiniteFieldNormalBasisExtension} but is more general.
For \axiomType{FiniteFieldNormalBasisExtensionByPolynomial} you
choose both the ground field and the irreducible polynomial used
in the representation.
The degree of the extension is the degree of the polynomial.

%
\xtc{
}{
\spadpaste{GF3 := PrimeField 3; \bound{GF3}}
}
\xtc{
We use a utility operation to generate an irreducible normal
```

```
polynomial (see
\downlink{``Utility Operations for Finite Fields''}
{ugxProblemFiniteUtilityPage}
in Section 8.11.7\ignore{ugxProblemFiniteUtility}).
The polynomial has one variable that is ``anonymous'': it displays
as a question mark.
}{
\spadpaste{f := createNormalPoly(4)\$FFPOLY(GF3) \free{GF3}\bound{f}}
}
\xtc{
}{
\spadpaste{GF81 := FFNBP(GF3,f); \bound{GF81}\free{f GF3}}
}
\xtc{
Let's look at a random element from this field.
}{
\spadpaste{r := random()\$GF81 \free{GF81}\bound{r1}}
}
\xtc{
}{
\spadpaste{r * r**3 * r**9 * r**27 \free{r1}}
}
\xtc{
}{
\spadpaste{norm r \free{r1}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxProblemFiniteNormalPagePatch1}
\begin{paste}{ugxProblemFiniteNormalPageFull1}{ugxProblemFiniteNormalPageEmpty1}
\pastebutton{ugxProblemFiniteNormalPageFull1}{\hidepaste}
\tab{5}\spadcommand{K := FFNB(3,8)\bound{K }}
\indentrel{3}\begin{verbatim}
   (1)  FiniteFieldNormalBasis(3,8)
                                          Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteNormalPageEmpty1}
\begin{paste}{ugxProblemFiniteNormalPageEmpty1}{ugxProblemFiniteNormalPagePatch1}
\pastebutton{ugxProblemFiniteNormalPageEmpty1}{\showpaste}
\tab{5}\spadcommand{K := FFNB(3,8)\bound{K }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteNormalPagePatch2}
\begin{paste}{ugxProblemFiniteNormalPageFull2}{ugxProblemFiniteNormalPageEmpty2}
\pastebutton{ugxProblemFiniteNormalPageFull2}{\hidepaste}
```

```
\tab{5}\spadcommand{a := normalElement()$K\bound{a }\free{K }}
\indentrel{3}\begin{verbatim}
   (2)  %CO
                      Type: FiniteFieldNormalBasis(3,8)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteNormalPageEmpty2}
\begin{paste}{ugxProblemFiniteNormalPageEmpty2}{ugxProblemFiniteNormalPagePatch2}
\pastebutton{ugxProblemFiniteNormalPageEmpty2}{\showpaste}
\tab{5}\spadcommand{a := normalElement()$K\bound{a }\free{K }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteNormalPagePatch3}
\begin{paste}{ugxProblemFiniteNormalPageFull3}{ugxProblemFiniteNormalPageEmpty3}
\pastebutton{ugxProblemFiniteNormalPageFull3}{\hidepaste}
\tab{5}\spadcommand{b := a**12 - a**5 + a\bound{b }\free{a }}
\indentrel{3}\begin{verbatim}
             7       5
            q       q       q
   (3)  2%CO   + %CO   + %CO
                      Type: FiniteFieldNormalBasis(3,8)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteNormalPageEmpty3}
\begin{paste}{ugxProblemFiniteNormalPageEmpty3}{ugxProblemFiniteNormalPagePatch3}
\pastebutton{ugxProblemFiniteNormalPageEmpty3}{\showpaste}
\tab{5}\spadcommand{b := a**12 - a**5 + a\bound{b }\free{a }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteNormalPagePatch4}
\begin{paste}{ugxProblemFiniteNormalPageFull4}{ugxProblemFiniteNormalPageEmpty4}
\pastebutton{ugxProblemFiniteNormalPageFull4}{\hidepaste}
\tab{5}\spadcommand{GF9 := FFNB(3,2);\bound{GF9 }}
\indentrel{3}\begin{verbatim}
                                          Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteNormalPageEmpty4}
\begin{paste}{ugxProblemFiniteNormalPageEmpty4}{ugxProblemFiniteNormalPagePatch4}
\pastebutton{ugxProblemFiniteNormalPageEmpty4}{\showpaste}
\tab{5}\spadcommand{GF9 := FFNB(3,2);\bound{GF9 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteNormalPagePatch5}
\begin{paste}{ugxProblemFiniteNormalPageFull5}{ugxProblemFiniteNormalPageEmpty5}
\pastebutton{ugxProblemFiniteNormalPageFull5}{\hidepaste}
\tab{5}\spadcommand{GF729 := FFNBX(GF9,3);\bound{GF729 }\free{GF9 }}
```

```
\indentrel{3}\begin{verbatim}
                                         Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteNormalPageEmpty5}
\begin{paste}{ugxProblemFiniteNormalPageEmpty5}{ugxProblemFiniteNormalPagePatch5}
\pastebutton{ugxProblemFiniteNormalPageEmpty5}{\showpaste}
\tab{5}\spadcommand{GF729 := FFNBX(GF9,3);\bound{GF729 }\free{GF9 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteNormalPagePatch6}
\begin{paste}{ugxProblemFiniteNormalPageFull6}{ugxProblemFiniteNormalPageEmpty6}
\pastebutton{ugxProblemFiniteNormalPageFull6}{\hidepaste}
\tab{5}\spadcommand{r := random()$GF729\bound{r }\free{GF729 }}
\indentrel{3}\begin{verbatim}
                2
           q   q
   (6)  2%CP %CQ
Type: FiniteFieldNormalBasisExtension(FiniteFieldNormalBasis(3,2),3)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteNormalPageEmpty6}
\begin{paste}{ugxProblemFiniteNormalPageEmpty6}{ugxProblemFiniteNormalPagePatch6}
\pastebutton{ugxProblemFiniteNormalPageEmpty6}{\showpaste}
\tab{5}\spadcommand{r := random()$GF729\bound{r }\free{GF729 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteNormalPagePatch7}
\begin{paste}{ugxProblemFiniteNormalPageFull7}{ugxProblemFiniteNormalPageEmpty7}
\pastebutton{ugxProblemFiniteNormalPageFull7}{\hidepaste}
\tab{5}\spadcommand{r + r**3 + r**9 + r**27\free{r }}
\indentrel{3}\begin{verbatim}
                         2
            q           q            q       q
   (7)  (2%CP  + 2%CP)%CQ   + 2%CP %CQ  + 2%CP %CQ
Type: FiniteFieldNormalBasisExtension(FiniteFieldNormalBasis(3,2),3)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteNormalPageEmpty7}
\begin{paste}{ugxProblemFiniteNormalPageEmpty7}{ugxProblemFiniteNormalPagePatch7}
\pastebutton{ugxProblemFiniteNormalPageEmpty7}{\showpaste}
\tab{5}\spadcommand{r + r**3 + r**9 + r**27\free{r }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteNormalPagePatch8}
\begin{paste}{ugxProblemFiniteNormalPageFull8}{ugxProblemFiniteNormalPageEmpty8}
\pastebutton{ugxProblemFiniteNormalPageFull8}{\hidepaste}
```

```
\tab{5}\spadcommand{GF3 := PrimeField 3;\bound{GF3 }}
\indentrel{3}\begin{verbatim}
                                            Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteNormalPageEmpty8}
\begin{paste}{ugxProblemFiniteNormalPageEmpty8}{ugxProblemFiniteNormalPagePatch8}
\pastebutton{ugxProblemFiniteNormalPageEmpty8}{\showpaste}
\tab{5}\spadcommand{GF3 := PrimeField 3;\bound{GF3 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteNormalPagePatch9}
\begin{paste}{ugxProblemFiniteNormalPageFull9}{ugxProblemFiniteNormalPageEmpty9}
\pastebutton{ugxProblemFiniteNormalPageFull9}{\hidepaste}
\tab{5}\spadcommand{f := createNormalPoly(4)$FFPOLY(GF3)\free{GF3 }\bound{f }}
\indentrel{3}\begin{verbatim}
         4     3
   (9)  ?  + 2?  + 2
          Type: SparseUnivariatePolynomial PrimeField 3
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteNormalPageEmpty9}
\begin{paste}{ugxProblemFiniteNormalPageEmpty9}{ugxProblemFiniteNormalPagePatch9}
\pastebutton{ugxProblemFiniteNormalPageEmpty9}{\showpaste}
\tab{5}\spadcommand{f := createNormalPoly(4)$FFPOLY(GF3)\free{GF3 }\bound{f }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteNormalPagePatch10}
\begin{paste}{ugxProblemFiniteNormalPageFull10}{ugxProblemFiniteNormalPageEmpty10}
\pastebutton{ugxProblemFiniteNormalPageFull10}{\hidepaste}
\tab{5}\spadcommand{GF81 := FFNBP(GF3,f);\bound{GF81 }\free{f GF3 }}
\indentrel{3}\begin{verbatim}
                                            Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteNormalPageEmpty10}
\begin{paste}{ugxProblemFiniteNormalPageEmpty10}{ugxProblemFiniteNormalPagePatch10}
\pastebutton{ugxProblemFiniteNormalPageEmpty10}{\showpaste}
\tab{5}\spadcommand{GF81 := FFNBP(GF3,f);\bound{GF81 }\free{f GF3 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteNormalPagePatch11}
\begin{paste}{ugxProblemFiniteNormalPageFull11}{ugxProblemFiniteNormalPageEmpty11}
\pastebutton{ugxProblemFiniteNormalPageFull11}{\hidepaste}
\tab{5}\spadcommand{r := random()$GF81\free{GF81 }\bound{r1 }}
\indentrel{3}\begin{verbatim}
             3        2
```

```
             q        q        q
   (11)  %CR   + %CR   + %CR
Type: FiniteFieldNormalBasisExtensionByPolynomial(PrimeField 3,?**4+2*?**3+2)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteNormalPageEmpty11}
\begin{paste}{ugxProblemFiniteNormalPageEmpty11}{ugxProblemFiniteNormalPagePatch11}
\pastebutton{ugxProblemFiniteNormalPageEmpty11}{\showpaste}
\tab{5}\spadcommand{r := random()$GF81\free{GF81 }\bound{r1 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteNormalPagePatch12}
\begin{paste}{ugxProblemFiniteNormalPageFull12}{ugxProblemFiniteNormalPageEmpty12}
\pastebutton{ugxProblemFiniteNormalPageFull12}{\hidepaste}
\tab{5}\spadcommand{r * r**3 * r**9 * r**27\free{r1 }}
\indentrel{3}\begin{verbatim}
               3         2
              q         q         q
   (12)  2%CR   + 2%CR   + 2%CR  + 2%CR
Type: FiniteFieldNormalBasisExtensionByPolynomial(PrimeField 3,?**4+2*?**3+2)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteNormalPageEmpty12}
\begin{paste}{ugxProblemFiniteNormalPageEmpty12}{ugxProblemFiniteNormalPagePatch12}
\pastebutton{ugxProblemFiniteNormalPageEmpty12}{\showpaste}
\tab{5}\spadcommand{r * r**3 * r**9 * r**27\free{r1 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteNormalPagePatch13}
\begin{paste}{ugxProblemFiniteNormalPageFull13}{ugxProblemFiniteNormalPageEmpty13}
\pastebutton{ugxProblemFiniteNormalPageFull13}{\hidepaste}
\tab{5}\spadcommand{norm r\free{r1 }}
\indentrel{3}\begin{verbatim}
   (13)  2
                                     Type: PrimeField 3
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteNormalPageEmpty13}
\begin{paste}{ugxProblemFiniteNormalPageEmpty13}{ugxProblemFiniteNormalPagePatch13}
\pastebutton{ugxProblemFiniteNormalPageEmpty13}{\showpaste}
\tab{5}\spadcommand{norm r\free{r1 }}
\end{paste}\end{patch}
```

Conversion Operations for Finite Fields

⇒ "notitle" (ugxProblemFiniteExtensionFinitePage) 12 on page 2246

— **ug08.ht** —

```
\begin{page}{ugxProblemFiniteConversionPage}
{8.11.6. Conversion Operations for Finite Fields}
\beginscroll

\labelSpace{5pc}
%
\xtc{
Let \texht{$K$}{\axiom{K}} be a finite field.
}{
\spadpaste{K := PrimeField 3 \bound{K}}
}
%
An extension field \texht{$K_m$}{\axiom{Km}} of degree
\smath{m} over \texht{$K$}{\axiom{K}} is a subfield of an
extension field \texht{$K_n$}{\axiom{Kn}} of degree \smath{n}
over \texht{$K$}{\axiom{K}} if and only if \smath{m} divides
\smath{n}.
\texht{
\centerline{{\begin{tabular}{ccc}}}
\centerline{{$K_n$ }}
\centerline{{$|$ }}
\centerline{{$K_m$ & $\Longleftrightarrow$ & $m | n$ }}
\centerline{{$|$ }}
\centerline{{K}}
\centerline{{\end{tabular}}}
}{
\begin{verbatim}
Kn
|
Km   <==>  m | n
|
K
\end{verbatim}
}
\axiomType{FiniteFieldHomomorphisms} provides conversion operations
between different extensions of one
fixed finite ground field and between different representations of
these finite fields.
\xtc{
Let's choose \smath{m} and \smath{n},
}{
\spadpaste{(m,n) := (4,8) \bound{m n}}
}
\xtc{
```

```
build the field extensions,
}{
\spadpaste{Km := FiniteFieldExtension(K,m) \bound{Km}\free{K m}}
}
\xtc{
and pick two random elements from the smaller field.
}{
\spadpaste{Kn := FiniteFieldExtension(K,n) \bound{Kn}\free{K n}}
}
\xtc{
}{
\spadpaste{a1 := random()\$Km \bound{a1}\free{Km}}
}
\xtc{
}{
\spadpaste{b1 := random()\$Km \bound{b1}\free{Km}}
}
%
\xtc{
Since \smath{m} divides \smath{n},
\texht{$K_m$}{\axiom{Km}} is a subfield of \texht{$K_n$}{\axiom{Kn}}.
}{
\spadpaste{a2 := a1 :: Kn \bound{a2}\free{a1 Kn}}
}
\xtc{
Therefore we can convert the elements of \texht{$K_m$}{\axiom{Km}}
into elements of \texht{$K_n$}{\axiom{Kn}}.
}{
\spadpaste{b2 := b1 :: Kn \bound{b2}\free{b1 Kn}}
}
%
%
\xtc{
To check this, let's do some arithmetic.
}{
\spadpaste{a1+b1 - ((a2+b2) :: Km) \free{a1 a2 b1 b2 Km Kn}}
}
\xtc{
}{
\spadpaste{a1*b1 - ((a2*b2) :: Km) \free{a1 a2 b1 b2 Km Kn}}
}
%
There are also conversions available for the
situation, when \texht{$K_m$}{\axiom{Km}} and \texht{$K_n$}{\axiom{Kn}}
are represented in different ways (see
\downlink{``Extensions of Finite Fields''}
{ugxProblemFiniteExtensionFinitePage} in Section 8.11.2
\ignore{ugxProblemFiniteExtensionFinite}).
For example let's choose \texht{$K_m$}{\axiom{Km}} where the
representation is 0 plus the cyclic multiplicative group and
```

```
\texht{$K_n$}{\axiom{Kn}} with a normal basis representation.
\xtc{
}{
\spadpaste{Km := FFCGX(K,m) \bound{Km2}\free{K m}}
}
\xtc{
}{
\spadpaste{Kn := FFNBX(K,n) \bound{Kn2}\free{K n}}
}
\xtc{
}{
\spadpaste{(a1,b1) := (random()\$Km,random()\$Km)
\bound{a12 b12}\free{Km2}}
}
\xtc{
}{
\spadpaste{a2 := a1 :: Kn \bound{a22}\free{a12 Kn2}}
}
\xtc{
}{
\spadpaste{b2 := b1 :: Kn \bound{b22}\free{b12 Kn2}}
}
%
\xtc{
Check the arithmetic again.
}{
\spadpaste{a1+b1 - ((a2+b2) :: Km) \free{a12 a22 b12 b22 Km2}}
}
\xtc{
}{
\spadpaste{a1*b1 - ((a2*b2) :: Km) \free{a12 a22 b12 b22 Km2}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxProblemFiniteConversionPagePatch1}
\begin{paste}{ugxProblemFiniteConversionPageFull1}{ugxProblemFiniteConversionPageEmpty1}
\pastebutton{ugxProblemFiniteConversionPageFull1}{\hidepaste}
\tab{5}\spadcommand{K := PrimeField 3\bound{K }}
\indentrel{3}\begin{verbatim}
   (1)  PrimeField 3
                                            Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteConversionPageEmpty1}
\begin{paste}{ugxProblemFiniteConversionPageEmpty1}{ugxProblemFiniteConversionPagePatch1}
\pastebutton{ugxProblemFiniteConversionPageEmpty1}{\showpaste}
```

```
\tab{5}\spadcommand{K := PrimeField 3\bound{K }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteConversionPagePatch2}
\begin{paste}{ugxProblemFiniteConversionPageFull2}{ugxProblemFiniteConversionPageEmpty2}
\pastebutton{ugxProblemFiniteConversionPageFull2}{\hidepaste}
\tab{5}\spadcommand{(m,n) := (4,8)\bound{m n }}
\indentrel{3}\begin{verbatim}
   (2)  8
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteConversionPageEmpty2}
\begin{paste}{ugxProblemFiniteConversionPageEmpty2}{ugxProblemFiniteConversionPagePatch2}
\pastebutton{ugxProblemFiniteConversionPageEmpty2}{\showpaste}
\tab{5}\spadcommand{(m,n) := (4,8)\bound{m n }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteConversionPagePatch3}
\begin{paste}{ugxProblemFiniteConversionPageFull3}{ugxProblemFiniteConversionPageEmpty3}
\pastebutton{ugxProblemFiniteConversionPageFull3}{\hidepaste}
\tab{5}\spadcommand{Km := FiniteFieldExtension(K,m)\bound{Km }\free{K m }}
\indentrel{3}\begin{verbatim}
   (3)  FiniteFieldExtension(PrimeField 3,4)
                                           Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteConversionPageEmpty3}
\begin{paste}{ugxProblemFiniteConversionPageEmpty3}{ugxProblemFiniteConversionPagePatch3}
\pastebutton{ugxProblemFiniteConversionPageEmpty3}{\showpaste}
\tab{5}\spadcommand{Km := FiniteFieldExtension(K,m)\bound{Km }\free{K m }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteConversionPagePatch4}
\begin{paste}{ugxProblemFiniteConversionPageFull4}{ugxProblemFiniteConversionPageEmpty4}
\pastebutton{ugxProblemFiniteConversionPageFull4}{\hidepaste}
\tab{5}\spadcommand{Kn := FiniteFieldExtension(K,n)\bound{Kn }\free{K n }}
\indentrel{3}\begin{verbatim}
   (4)  FiniteFieldExtension(PrimeField 3,8)
                                           Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteConversionPageEmpty4}
\begin{paste}{ugxProblemFiniteConversionPageEmpty4}{ugxProblemFiniteConversionPagePatch4}
\pastebutton{ugxProblemFiniteConversionPageEmpty4}{\showpaste}
\tab{5}\spadcommand{Kn := FiniteFieldExtension(K,n)\bound{Kn }\free{K n }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugxProblemFiniteConversionPagePatch5}
\begin{paste}{ugxProblemFiniteConversionPageFull5}{ugxProblemFiniteConversionPageEmpty5}
\pastebutton{ugxProblemFiniteConversionPageFull5}{\hidepaste}
\tab{5}\spadcommand{a1 := random()$Km\bound{a1 }\free{Km }}
\indentrel{3}\begin{verbatim}
            3       2
   (5)  2%BD  + 2%BD  + 2%BD + 1
             Type: FiniteFieldExtension(PrimeField 3,4)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteConversionPageEmpty5}
\begin{paste}{ugxProblemFiniteConversionPageEmpty5}{ugxProblemFiniteConversionPagePatch5}
\pastebutton{ugxProblemFiniteConversionPageEmpty5}{\showpaste}
\tab{5}\spadcommand{a1 := random()$Km\bound{a1 }\free{Km }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteConversionPagePatch6}
\begin{paste}{ugxProblemFiniteConversionPageFull6}{ugxProblemFiniteConversionPageEmpty6}
\pastebutton{ugxProblemFiniteConversionPageFull6}{\hidepaste}
\tab{5}\spadcommand{b1 := random()$Km\bound{b1 }\free{Km }}
\indentrel{3}\begin{verbatim}
           3      2
   (6)  %BD  + %BD  + 2%BD + 2
             Type: FiniteFieldExtension(PrimeField 3,4)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteConversionPageEmpty6}
\begin{paste}{ugxProblemFiniteConversionPageEmpty6}{ugxProblemFiniteConversionPagePatch6}
\pastebutton{ugxProblemFiniteConversionPageEmpty6}{\showpaste}
\tab{5}\spadcommand{b1 := random()$Km\bound{b1 }\free{Km }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteConversionPagePatch7}
\begin{paste}{ugxProblemFiniteConversionPageFull7}{ugxProblemFiniteConversionPageEmpty7}
\pastebutton{ugxProblemFiniteConversionPageFull7}{\hidepaste}
\tab{5}\spadcommand{a2 := a1 :: Kn\bound{a2 }\free{a1 Kn }}
\indentrel{3}\begin{verbatim}
            6
   (7)  2%BE  + 1
             Type: FiniteFieldExtension(PrimeField 3,8)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteConversionPageEmpty7}
\begin{paste}{ugxProblemFiniteConversionPageEmpty7}{ugxProblemFiniteConversionPagePatch7}
\pastebutton{ugxProblemFiniteConversionPageEmpty7}{\showpaste}
\tab{5}\spadcommand{a2 := a1 :: Kn\bound{a2 }\free{a1 Kn }}
```

```
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteConversionPagePatch8}
\begin{paste}{ugxProblemFiniteConversionPageFull8}{ugxProblemFiniteConversionPageEmpty8}
\pastebutton{ugxProblemFiniteConversionPageFull8}{\hidepaste}
\tab{5}\spadcommand{b2 := b1 :: Kn\bound{b2 }\free{b1 Kn }}
\indentrel{3}\begin{verbatim}
           6       4      2
   (8)  2%BE  + 2%BE  + %BE  + 2
            Type: FiniteFieldExtension(PrimeField 3,8)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteConversionPageEmpty8}
\begin{paste}{ugxProblemFiniteConversionPageEmpty8}{ugxProblemFiniteConversionPagePatch8}
\pastebutton{ugxProblemFiniteConversionPageEmpty8}{\showpaste}
\tab{5}\spadcommand{b2 := b1 :: Kn\bound{b2 }\free{b1 Kn }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteConversionPagePatch9}
\begin{paste}{ugxProblemFiniteConversionPageFull9}{ugxProblemFiniteConversionPageEmpty9}
\pastebutton{ugxProblemFiniteConversionPageFull9}{\hidepaste}
\tab{5}\spadcommand{a1+b1 - ((a2+b2) :: Km)\free{a1 a2 b1 b2 Km Kn }}
\indentrel{3}\begin{verbatim}
   (9)  0
            Type: FiniteFieldExtension(PrimeField 3,4)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteConversionPageEmpty9}
\begin{paste}{ugxProblemFiniteConversionPageEmpty9}{ugxProblemFiniteConversionPagePatch9}
\pastebutton{ugxProblemFiniteConversionPageEmpty9}{\showpaste}
\tab{5}\spadcommand{a1+b1 - ((a2+b2) :: Km)\free{a1 a2 b1 b2 Km Kn }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteConversionPagePatch10}
\begin{paste}{ugxProblemFiniteConversionPageFull10}{ugxProblemFiniteConversionPageEmpty10}
\pastebutton{ugxProblemFiniteConversionPageFull10}{\hidepaste}
\tab{5}\spadcommand{a1*b1 - ((a2*b2) :: Km)\free{a1 a2 b1 b2 Km Kn }}
\indentrel{3}\begin{verbatim}
   (10)  0
            Type: FiniteFieldExtension(PrimeField 3,4)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteConversionPageEmpty10}
\begin{paste}{ugxProblemFiniteConversionPageEmpty10}{ugxProblemFiniteConversionPagePatch10}
\pastebutton{ugxProblemFiniteConversionPageEmpty10}{\showpaste}
\tab{5}\spadcommand{a1*b1 - ((a2*b2) :: Km)\free{a1 a2 b1 b2 Km Kn }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugxProblemFiniteConversionPagePatch11}
\begin{paste}{ugxProblemFiniteConversionPageFull11}{ugxProblemFiniteConversionPageEmpty11}
\pastebutton{ugxProblemFiniteConversionPageFull11}{\hidepaste}
\tab{5}\spadcommand{Km := FFCGX(K,m)\bound{Km2 }\free{K m }}
\indentrel{3}\begin{verbatim}
   (11)  FiniteFieldCyclicGroupExtension(PrimeField 3,4)
                                          Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteConversionPageEmpty11}
\begin{paste}{ugxProblemFiniteConversionPageEmpty11}{ugxProblemFiniteConversionPagePatch11}
\pastebutton{ugxProblemFiniteConversionPageEmpty11}{\showpaste}
\tab{5}\spadcommand{Km := FFCGX(K,m)\bound{Km2 }\free{K m }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteConversionPagePatch12}
\begin{paste}{ugxProblemFiniteConversionPageFull12}{ugxProblemFiniteConversionPageEmpty12}
\pastebutton{ugxProblemFiniteConversionPageFull12}{\hidepaste}
\tab{5}\spadcommand{Kn := FFNBX(K,n)\bound{Kn2 }\free{K n }}
\indentrel{3}\begin{verbatim}
   (12)  FiniteFieldNormalBasisExtension(PrimeField 3,8)
                                          Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteConversionPageEmpty12}
\begin{paste}{ugxProblemFiniteConversionPageEmpty12}{ugxProblemFiniteConversionPagePatch12}
\pastebutton{ugxProblemFiniteConversionPageEmpty12}{\showpaste}
\tab{5}\spadcommand{Kn := FFNBX(K,n)\bound{Kn2 }\free{K n }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteConversionPagePatch13}
\begin{paste}{ugxProblemFiniteConversionPageFull13}{ugxProblemFiniteConversionPageEmpty13}
\pastebutton{ugxProblemFiniteConversionPageFull13}{\hidepaste}
\tab{5}\spadcommand{(a1,b1) := (random()$Km,random()$Km)\bound{a12 b12 }\free{Km2 }}
\indentrel{3}\begin{verbatim}
           13
   (13)  %BF
  Type: FiniteFieldCyclicGroupExtension(PrimeField 3,4)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteConversionPageEmpty13}
\begin{paste}{ugxProblemFiniteConversionPageEmpty13}{ugxProblemFiniteConversionPagePatch13}
\pastebutton{ugxProblemFiniteConversionPageEmpty13}{\showpaste}
\tab{5}\spadcommand{(a1,b1) := (random()$Km,random()$Km)\bound{a12 b12 }\free{Km2 }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugxProblemFiniteConversionPagePatch14}
\begin{paste}{ugxProblemFiniteConversionPageFull14}{ugxProblemFiniteConversionPageEmpty14}
\pastebutton{ugxProblemFiniteConversionPageFull14}{\hidepaste}
\tab{5}\spadcommand{a2 := a1 :: Kn\bound{a22 }\free{a12 Kn2 }}
\indentrel{3}\begin{verbatim}
   (14)
        6        5        4        2
       q        q        q        q        q
   2%BG   + 2%BG   + 2%BG   + 2%BG   + 2%BG  + 2%BG
  Type: FiniteFieldNormalBasisExtension(PrimeField 3,8)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteConversionPageEmpty14}
\begin{paste}{ugxProblemFiniteConversionPageEmpty14}{ugxProblemFiniteConversionPagePatch14}
\pastebutton{ugxProblemFiniteConversionPageEmpty14}{\showpaste}
\tab{5}\spadcommand{a2 := a1 :: Kn\bound{a22 }\free{a12 Kn2 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteConversionPagePatch15}
\begin{paste}{ugxProblemFiniteConversionPageFull15}{ugxProblemFiniteConversionPageEmpty15}
\pastebutton{ugxProblemFiniteConversionPageFull15}{\hidepaste}
\tab{5}\spadcommand{b2 := b1 :: Kn\bound{b22 }\free{b12 Kn2 }}
\indentrel{3}\begin{verbatim}
   (15)
          7       6       5       4        3       2
         q       q       q       q        q       q
     2%BG   + %BG   + %BG   + %BG   + 2%BG   + %BG
   +
        q
     %BG  + %BG
  Type: FiniteFieldNormalBasisExtension(PrimeField 3,8)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteConversionPageEmpty15}
\begin{paste}{ugxProblemFiniteConversionPageEmpty15}{ugxProblemFiniteConversionPagePatch15}
\pastebutton{ugxProblemFiniteConversionPageEmpty15}{\showpaste}
\tab{5}\spadcommand{b2 := b1 :: Kn\bound{b22 }\free{b12 Kn2 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteConversionPagePatch16}
\begin{paste}{ugxProblemFiniteConversionPageFull16}{ugxProblemFiniteConversionPageEmpty16}
\pastebutton{ugxProblemFiniteConversionPageFull16}{\hidepaste}
\tab{5}\spadcommand{a1+b1 - ((a2+b2) :: Km)\free{a12 a22 b12 b22 Km2 }}
\indentrel{3}\begin{verbatim}
   (16)  0
  Type: FiniteFieldCyclicGroupExtension(PrimeField 3,4)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugxProblemFiniteConversionPageEmpty16}
\begin{paste}{ugxProblemFiniteConversionPageEmpty16}{ugxProblemFiniteConversionPagePatch16}
\pastebutton{ugxProblemFiniteConversionPageEmpty16}{\showpaste}
\tab{5}\spadcommand{a1+b1 - ((a2+b2) :: Km)\free{a12 a22 b12 b22 Km2 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteConversionPagePatch17}
\begin{paste}{ugxProblemFiniteConversionPageFull17}{ugxProblemFiniteConversionPageEmpty17}
\pastebutton{ugxProblemFiniteConversionPageFull17}{\hidepaste}
\tab{5}\spadcommand{a1*b1 - ((a2*b2) :: Km)\free{a12 a22 b12 b22 Km2 }}
\indentrel{3}\begin{verbatim}
   (17)  0
  Type: FiniteFieldCyclicGroupExtension(PrimeField 3,4)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteConversionPageEmpty17}
\begin{paste}{ugxProblemFiniteConversionPageEmpty17}{ugxProblemFiniteConversionPagePatch17}
\pastebutton{ugxProblemFiniteConversionPageEmpty17}{\showpaste}
\tab{5}\spadcommand{a1*b1 - ((a2*b2) :: Km)\free{a12 a22 b12 b22 Km2 }}
\end{paste}\end{patch}
```

———-

Utility Operations for Finite Fields

— **ug08.ht** —

```
\begin{page}{ugxProblemFiniteUtilityPage}
{8.11.7. Utility Operations for Finite Fields}
\beginscroll

\axiomType{FiniteFieldPolynomialPackage} (abbreviation
\axiomType{FFPOLY})
provides operations for generating, counting and testing polynomials
over finite fields. Let’s start with a couple of definitions:
\indent{4}
\beginitems
\item[-] A polynomial is {\it primitive} if its roots are primitive
elements in an extension of the coefficient field of degree equal
to the degree of the polynomial.
\item[-] A polynomial is {\it normal} over its coefficient field
if its roots are linearly independent
elements in an extension of the coefficient field of degree equal
to the degree of the polynomial.
\enditems
```

```
\indent{0}
In what follows, many of the generated polynomials have one
‘‘anonymous’’ variable.
This indeterminate is displayed as a question mark (\spadSyntax{?}).

\xtc{
To fix ideas, let’s use the field with five elements for the first
few examples.
}{
\spadpaste{GF5 := PF 5; \bound{GF5}}
}
%
%
\xtc{
You can generate irreducible polynomials of any (positive) degree
(within the storage capabilities of the computer and your ability
to wait) by using
\axiomFunFrom{createIrreduciblePoly}{FiniteFieldPolynomialPackage}.
}{
\spadpaste{f := createIrreduciblePoly(8)\$FFPOLY(GF5) \bound{f}\free{GF5}}
}
%
\xtc{
Does this polynomial have other important properties? Use
\axiomFun{primitive?} to test whether it is a primitive polynomial.
}{
\spadpaste{primitive?(f)\$FFPOLY(GF5) \free{f}}
}
\xtc{
Use \axiomFun{normal?} to test whether it is a normal polynomial.
}{
\spadpaste{normal?(f)\$FFPOLY(GF5) \free{f}}
}
\noindent
Note that this is actually a trivial case,
because a normal polynomial of degree \smath{n}
must have a nonzero term of degree \smath{n-1}.
We will refer back to this later.

\xtc{
To get a primitive polynomial of degree 8 just issue this.
}{
\spadpaste{p := createPrimitivePoly(8)\$FFPOLY(GF5) \bound{p}\free{GF5}}
}
\xtc{
}{
\spadpaste{primitive?(p)\$FFPOLY(GF5) \free{p}}
}
\xtc{
This polynomial is not normal,
```

```
}{
\spadpaste{normal?(p)\$FFPOLY(GF5) \free{p}}
}
\xtc{
but if you want a normal one simply write this.
}{
\spadpaste{n := createNormalPoly(8)\$FFPOLY(GF5) \bound{n} \free{GF5}}
}
\xtc{
This polynomial is not primitive!
}{
\spadpaste{primitive?(n)\$FFPOLY(GF5) \free{n}}
}
This could have been seen directly, as
the constant term is 1 here, which is not a primitive
element up to the factor (\axiom{-1}) raised to the degree of the
polynomial.\footnote{Cf. Lidl, R. \& Niederreiter,
H., {\it Finite Fields,} Encycl. of Math. 20, (Addison-Wesley, 1983),
p.90, Th. 3.18.}

What about
polynomials that are both primitive and normal?
The existence of such a polynomial is by no means obvious.
\footnote{The existence of such polynomials is proved in
Lenstra, H. W. \& Schoof, R. J., {\it Primitive
Normal Bases for Finite Fields,} Math. Comp. 48, 1987, pp. 217-231.}
%
\xtc{
If you really need one use either
\axiomFunFrom{createPrimitiveNormalPoly}{FiniteFieldPolynomialPackage} or
\axiomFunFrom{createNormalPrimitivePoly}{FiniteFieldPolynomialPackage}.
}{
\spadpaste{createPrimitiveNormalPoly(8)\$FFPOLY(GF5) \free{GF5}}
}
%

If you want to obtain additional polynomials of the various types above
as given by the {\bf create...} operations above, you can use the {\bf
next...} operations.
For instance,
\axiomFunFrom{nextIrreduciblePoly}{FiniteFieldPolynomialPackage} yields
the next monic irreducible polynomial with the same degree as the input
polynomial.
By ``next'' we mean ``next in a natural order using the terms and
coefficients.''
This will become more clear in the following examples.

\xtc{
This is the field with five elements.
}{
```

```
\spadpaste{GF5 := PF 5; \bound{GF5}}
}
%
\xtc{
Our first example irreducible polynomial, say
of degree 3, must be ``greater'' than this.
}{
\spadpaste{h := monomial(1,8)\$SUP(GF5) \bound{h}\free{GF5}}
}
\xtc{
You can generate it by doing this.
}{
\spadpaste{nh := nextIrreduciblePoly(h)\$FFPOLY(GF5) \bound{nh}\free{h}}
}
%
\xtc{
Notice that this polynomial is not the same as the one
\axiomFunFrom{createIrreduciblePoly}{FiniteFieldPolynomialPackage}.
}{
\spadpaste{createIrreduciblePoly(3)\$FFPOLY(GF5) \free{GF5}}
}
\xtc{
You can step through all irreducible polynomials of degree 8 over
the field with 5 elements by repeatedly issuing this.
}{
\spadpaste{nh := nextIrreduciblePoly(nh)\$FFPOLY(GF5) \free{nh}}
}
\xtc{
You could also ask for the total number of these.
}{
\spadpaste{numberOfIrreduciblePoly(5)\$FFPOLY(GF5) \free{GF5}}
}

We hope that ``natural order'' on polynomials is now clear:
first we compare the number of monomials of
two polynomials (``more'' is ``greater'');
then, if necessary, the degrees of these monomials (lexicographically),
and lastly their coefficients (also
lexicographically, and using the operation \axiomFun{lookup} if
our field is not a prime field).
Also note that we make both polynomials monic before looking at the
coefficients:
multiplying either polynomial  by a nonzero constant
produces the same result.

%
\xtc{
The package
\axiomType{FiniteFieldPolynomialPackage} also provides similar
operations for primitive and normal polynomials. With
```

```
the exception of the number of primitive normal polynomials;
we're not aware of any known formula for this.
}{
\spadpaste{numberOfPrimitivePoly(3)\$FFPOLY(GF5) \free{GF5}}
}
%
%
\xtc{
Take these,
}{
\spadpaste{m := monomial(1,1)\$SUP(GF5) \bound{m}\free{GF5}}
}
\xtc{
}{
\spadpaste{f := m**3 + 4*m**2 + m + 2 \bound{fx}\free{m}}
}
%
%
\xtc{
and then we have:
}{
\spadpaste{f1 := nextPrimitivePoly(f)\$FFPOLY(GF5) \free{fx}\bound{f1}}
}
\xtc{
What happened?
}{
\spadpaste{nextPrimitivePoly(f1)\$FFPOLY(GF5) \free{f1}}
}
%
Well, for the ordering used in
\axiomFunFrom{nextPrimitivePoly}{FiniteFieldPolynomialPackage} we
use as first criterion a comparison of the constant terms of the
polynomials.
Analogously, in
\axiomFunFrom{nextNormalPoly}{FiniteFieldPolynomialPackage} we first
compare the monomials of degree 1 less than the degree of the
polynomials (which is nonzero, by an earlier remark).
%
\xtc{
}{
\spadpaste{f := m**3 + m**2 + 4*m + 1 \bound{fy} \free{m}}
}
\xtc{
}{
\spadpaste{f1 := nextNormalPoly(f)\$FFPOLY(GF5) \free{fy}\bound{f1y}}
}
\xtc{
}{
\spadpaste{nextNormalPoly(f1)\$FFPOLY(GF5) \free{f1y}}
}
```

```
%
\noindent
We don't have to restrict ourselves to prime fields.
%
\xtc{
Let's consider, say, a field with 16 elements.
}{
\spadpaste{GF16 := FFX(FFX(PF 2,2),2); \bound{GF16} }
}
%
%
\xtc{
We can apply any of the operations described above.
}{
\spadpaste{createIrreduciblePoly(5)\$FFPOLY(GF16) \free{GF16}}
}

\xtc{
Axiom also provides operations
for producing random polynomials of a given degree
}{
\spadpaste{random(5)\$FFPOLY(GF16) \free{GF16}}
}
\xtc{
or with degree between two given bounds.
}{
\spadpaste{random(3,9)\$FFPOLY(GF16) \free{GF16}}
}

\axiomType{\axiom{FiniteFieldPolynomialPackage2}} (abbreviation
\axiomType{FFPOLY2})
exports an operation \axiomFun{rootOfIrreduciblePoly}
for finding one root of an irreducible polynomial \axiom{f}
in an extension field of the coefficient field.
The degree of the extension has to be a multiple of the degree of \axiom{f}.
It is not checked whether \axiom{f} actually is irreducible.

%
\xtc{
To illustrate this operation, we fix a ground field \axiom{GF}
}{
\spadpaste{GF2 := PrimeField 2; \bound{GF2}}
}
%
%
\xtc{
and then an extension field.
}{
\spadpaste{F := FFX(GF2,12) \bound{F}\free{GF2}}
}
```

```
%
%
\xtc{
We construct an irreducible polynomial over \axiom{GF2}.
}{
\spadpaste{f := createIrreduciblePoly(6)\$FFPOLY(GF2)
\bound{fz}\free{GF2}}
}
%
%
\xtc{
We compute a root of \axiom{f}.
}{
\spadpaste{root := rootOfIrreduciblePoly(f)\$FFPOLY2(F,GF2)
\free{F GF2 fz}\bound{root}}
}
%
%and check the result
%\spadcommand{eval(f, monomial(1,1)\$SUP(F) = root) \free{fz F root}}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugxProblemFiniteUtilityPagePatch1}
\begin{paste}{ugxProblemFiniteUtilityPageFull1}{ugxProblemFiniteUtilityPageEmpty1}
\pastebutton{ugxProblemFiniteUtilityPageFull1}{\hidepaste}
\tab{5}\spadcommand{GF5 := PF 5;\bound{GF5 }}
\indentrel{3}\begin{verbatim}
                                        Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPageEmpty1}
\begin{paste}{ugxProblemFiniteUtilityPageEmpty1}{ugxProblemFiniteUtilityPagePatch1}
\pastebutton{ugxProblemFiniteUtilityPageEmpty1}{\showpaste}
\tab{5}\spadcommand{GF5 := PF 5;\bound{GF5 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPagePatch2}
\begin{paste}{ugxProblemFiniteUtilityPageFull2}{ugxProblemFiniteUtilityPageEmpty2}
\pastebutton{ugxProblemFiniteUtilityPageFull2}{\hidepaste}
\tab{5}\spadcommand{f := createIrreduciblePoly(8)$FFPOLY(GF5)\bound{f }\free{GF5 }}
\indentrel{3}\begin{verbatim}
        8    4
   (2)  ?  + ?  + 2
          Type: SparseUnivariatePolynomial PrimeField 5
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugxProblemFiniteUtilityPageEmpty2}
\begin{paste}{ugxProblemFiniteUtilityPageEmpty2}{ugxProblemFiniteUtilityPagePatch2}
\pastebutton{ugxProblemFiniteUtilityPageEmpty2}{\showpaste}
\tab{5}\spadcommand{f := createIrreduciblePoly(8)$FFPOLY(GF5)\bound{f }\free{GF5 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPagePatch3}
\begin{paste}{ugxProblemFiniteUtilityPageFull3}{ugxProblemFiniteUtilityPageEmpty3}
\pastebutton{ugxProblemFiniteUtilityPageFull3}{\hidepaste}
\tab{5}\spadcommand{primitive?(f)$FFPOLY(GF5)\free{f }}
\indentrel{3}\begin{verbatim}
   (3)  false
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPageEmpty3}
\begin{paste}{ugxProblemFiniteUtilityPageEmpty3}{ugxProblemFiniteUtilityPagePatch3}
\pastebutton{ugxProblemFiniteUtilityPageEmpty3}{\showpaste}
\tab{5}\spadcommand{primitive?(f)$FFPOLY(GF5)\free{f }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPagePatch4}
\begin{paste}{ugxProblemFiniteUtilityPageFull4}{ugxProblemFiniteUtilityPageEmpty4}
\pastebutton{ugxProblemFiniteUtilityPageFull4}{\hidepaste}
\tab{5}\spadcommand{normal?(f)$FFPOLY(GF5)\free{f }}
\indentrel{3}\begin{verbatim}
   (4)  false
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPageEmpty4}
\begin{paste}{ugxProblemFiniteUtilityPageEmpty4}{ugxProblemFiniteUtilityPagePatch4}
\pastebutton{ugxProblemFiniteUtilityPageEmpty4}{\showpaste}
\tab{5}\spadcommand{normal?(f)$FFPOLY(GF5)\free{f }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPagePatch5}
\begin{paste}{ugxProblemFiniteUtilityPageFull5}{ugxProblemFiniteUtilityPageEmpty5}
\pastebutton{ugxProblemFiniteUtilityPageFull5}{\hidepaste}
\tab{5}\spadcommand{p := createPrimitivePoly(8)$FFPOLY(GF5)\bound{p }\free{GF5 }}
\indentrel{3}\begin{verbatim}
         8    3    2
   (5)  ?  + ?  + ?  + ? + 2
          Type: SparseUnivariatePolynomial PrimeField 5
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPageEmpty5}
```

```
\begin{paste}{ugxProblemFiniteUtilityPageEmpty5}{ugxProblemFiniteUtilityPagePatch5}
\pastebutton{ugxProblemFiniteUtilityPageEmpty5}{\showpaste}
\tab{5}\spadcommand{p := createPrimitivePoly(8)$FFPOLY(GF5)\bound{p }\free{GF5 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPagePatch6}
\begin{paste}{ugxProblemFiniteUtilityPageFull6}{ugxProblemFiniteUtilityPageEmpty6}
\pastebutton{ugxProblemFiniteUtilityPageFull6}{\hidepaste}
\tab{5}\spadcommand{primitive?(p)$FFPOLY(GF5)\free{p }}
\indentrel{3}\begin{verbatim}
   (6)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPageEmpty6}
\begin{paste}{ugxProblemFiniteUtilityPageEmpty6}{ugxProblemFiniteUtilityPagePatch6}
\pastebutton{ugxProblemFiniteUtilityPageEmpty6}{\showpaste}
\tab{5}\spadcommand{primitive?(p)$FFPOLY(GF5)\free{p }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPagePatch7}
\begin{paste}{ugxProblemFiniteUtilityPageFull7}{ugxProblemFiniteUtilityPageEmpty7}
\pastebutton{ugxProblemFiniteUtilityPageFull7}{\hidepaste}
\tab{5}\spadcommand{normal?(p)$FFPOLY(GF5)\free{p }}
\indentrel{3}\begin{verbatim}
   (7)  false
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPageEmpty7}
\begin{paste}{ugxProblemFiniteUtilityPageEmpty7}{ugxProblemFiniteUtilityPagePatch7}
\pastebutton{ugxProblemFiniteUtilityPageEmpty7}{\showpaste}
\tab{5}\spadcommand{normal?(p)$FFPOLY(GF5)\free{p }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPagePatch8}
\begin{paste}{ugxProblemFiniteUtilityPageFull8}{ugxProblemFiniteUtilityPageEmpty8}
\pastebutton{ugxProblemFiniteUtilityPageFull8}{\hidepaste}
\tab{5}\spadcommand{n := createNormalPoly(8)$FFPOLY(GF5)\bound{n }\free{GF5 }}
\indentrel{3}\begin{verbatim}
         8     7    3
   (8)  ?  + 4?  + ?  + 1
          Type: SparseUnivariatePolynomial PrimeField 5
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPageEmpty8}
\begin{paste}{ugxProblemFiniteUtilityPageEmpty8}{ugxProblemFiniteUtilityPagePatch8}
```

```
\pastebutton{ugxProblemFiniteUtilityPageEmpty8}{\showpaste}
\tab{5}\spadcommand{n := createNormalPoly(8)$FFPOLY(GF5)\bound{n }\free{GF5 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPagePatch9}
\begin{paste}{ugxProblemFiniteUtilityPageFull9}{ugxProblemFiniteUtilityPageEmpty9}
\pastebutton{ugxProblemFiniteUtilityPageFull9}{\hidepaste}
\tab{5}\spadcommand{primitive?(n)$FFPOLY(GF5)\free{n }}
\indentrel{3}\begin{verbatim}
   (9)  false
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPageEmpty9}
\begin{paste}{ugxProblemFiniteUtilityPageEmpty9}{ugxProblemFiniteUtilityPagePatch9}
\pastebutton{ugxProblemFiniteUtilityPageEmpty9}{\showpaste}
\tab{5}\spadcommand{primitive?(n)$FFPOLY(GF5)\free{n }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPagePatch10}
\begin{paste}{ugxProblemFiniteUtilityPageFull10}{ugxProblemFiniteUtilityPageEmpty10}
\pastebutton{ugxProblemFiniteUtilityPageFull10}{\hidepaste}
\tab{5}\spadcommand{createPrimitiveNormalPoly(8)$FFPOLY(GF5)\free{GF5 }}
\indentrel{3}\begin{verbatim}
          8     7     5
   (10)  ?  + 4?  + 2?  + 2
          Type: SparseUnivariatePolynomial PrimeField 5
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPageEmpty10}
\begin{paste}{ugxProblemFiniteUtilityPageEmpty10}{ugxProblemFiniteUtilityPagePatch10}
\pastebutton{ugxProblemFiniteUtilityPageEmpty10}{\showpaste}
\tab{5}\spadcommand{createPrimitiveNormalPoly(8)$FFPOLY(GF5)\free{GF5 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPagePatch11}
\begin{paste}{ugxProblemFiniteUtilityPageFull11}{ugxProblemFiniteUtilityPageEmpty11}
\pastebutton{ugxProblemFiniteUtilityPageFull11}{\hidepaste}
\tab{5}\spadcommand{GF5 := PF 5;\bound{GF5 }}
\indentrel{3}\begin{verbatim}
                                           Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPageEmpty11}
\begin{paste}{ugxProblemFiniteUtilityPageEmpty11}{ugxProblemFiniteUtilityPagePatch11}
\pastebutton{ugxProblemFiniteUtilityPageEmpty11}{\showpaste}
\tab{5}\spadcommand{GF5 := PF 5;\bound{GF5 }}
```

```
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPagePatch12}
\begin{paste}{ugxProblemFiniteUtilityPageFull12}{ugxProblemFiniteUtilityPageEmpty12}
\pastebutton{ugxProblemFiniteUtilityPageFull12}{\hidepaste}
\tab{5}\spadcommand{h := monomial(1,8)$SUP(GF5)\bound{h }\free{GF5 }}
\indentrel{3}\begin{verbatim}
          8
   (12)  ?
          Type: SparseUnivariatePolynomial PrimeField 5
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPageEmpty12}
\begin{paste}{ugxProblemFiniteUtilityPageEmpty12}{ugxProblemFiniteUtilityPagePatch12}
\pastebutton{ugxProblemFiniteUtilityPageEmpty12}{\showpaste}
\tab{5}\spadcommand{h := monomial(1,8)$SUP(GF5)\bound{h }\free{GF5 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPagePatch13}
\begin{paste}{ugxProblemFiniteUtilityPageFull13}{ugxProblemFiniteUtilityPageEmpty13}
\pastebutton{ugxProblemFiniteUtilityPageFull13}{\hidepaste}
\tab{5}\spadcommand{nh := nextIrreduciblePoly(h)$FFPOLY(GF5)\bound{nh }\free{h }}
\indentrel{3}\begin{verbatim}
          8
   (13)  ?  + 2
Type: Union(SparseUnivariatePolynomial PrimeField 5,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPageEmpty13}
\begin{paste}{ugxProblemFiniteUtilityPageEmpty13}{ugxProblemFiniteUtilityPagePatch13}
\pastebutton{ugxProblemFiniteUtilityPageEmpty13}{\showpaste}
\tab{5}\spadcommand{nh := nextIrreduciblePoly(h)$FFPOLY(GF5)\bound{nh }\free{h }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPagePatch14}
\begin{paste}{ugxProblemFiniteUtilityPageFull14}{ugxProblemFiniteUtilityPageEmpty14}
\pastebutton{ugxProblemFiniteUtilityPageFull14}{\hidepaste}
\tab{5}\spadcommand{createIrreduciblePoly(3)$FFPOLY(GF5)\free{GF5 }}
\indentrel{3}\begin{verbatim}
          3
   (14)  ?  + ? + 1
          Type: SparseUnivariatePolynomial PrimeField 5
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPageEmpty14}
\begin{paste}{ugxProblemFiniteUtilityPageEmpty14}{ugxProblemFiniteUtilityPagePatch14}
\pastebutton{ugxProblemFiniteUtilityPageEmpty14}{\showpaste}
```

```
\tab{5}\spadcommand{createIrreduciblePoly(3)$FFPOLY(GF5)\free{GF5 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPagePatch15}
\begin{paste}{ugxProblemFiniteUtilityPageFull15}{ugxProblemFiniteUtilityPageEmpty15}
\pastebutton{ugxProblemFiniteUtilityPageFull15}{\hidepaste}
\tab{5}\spadcommand{nh := nextIrreduciblePoly(nh)$FFPOLY(GF5)\free{nh }}
\indentrel{3}\begin{verbatim}
          8
   (15)  ?  + 3
Type: Union(SparseUnivariatePolynomial PrimeField 5,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPageEmpty15}
\begin{paste}{ugxProblemFiniteUtilityPageEmpty15}{ugxProblemFiniteUtilityPagePatch15}
\pastebutton{ugxProblemFiniteUtilityPageEmpty15}{\showpaste}
\tab{5}\spadcommand{nh := nextIrreduciblePoly(nh)$FFPOLY(GF5)\free{nh }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPagePatch16}
\begin{paste}{ugxProblemFiniteUtilityPageFull16}{ugxProblemFiniteUtilityPageEmpty16}
\pastebutton{ugxProblemFiniteUtilityPageFull16}{\hidepaste}
\tab{5}\spadcommand{numberOfIrreduciblePoly(5)$FFPOLY(GF5)\free{GF5 }}
\indentrel{3}\begin{verbatim}
   (16)  624
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPageEmpty16}
\begin{paste}{ugxProblemFiniteUtilityPageEmpty16}{ugxProblemFiniteUtilityPagePatch16}
\pastebutton{ugxProblemFiniteUtilityPageEmpty16}{\showpaste}
\tab{5}\spadcommand{numberOfIrreduciblePoly(5)$FFPOLY(GF5)\free{GF5 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPagePatch17}
\begin{paste}{ugxProblemFiniteUtilityPageFull17}{ugxProblemFiniteUtilityPageEmpty17}
\pastebutton{ugxProblemFiniteUtilityPageFull17}{\hidepaste}
\tab{5}\spadcommand{numberOfPrimitivePoly(3)$FFPOLY(GF5)\free{GF5 }}
\indentrel{3}\begin{verbatim}
   (17)  20
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPageEmpty17}
\begin{paste}{ugxProblemFiniteUtilityPageEmpty17}{ugxProblemFiniteUtilityPagePatch17}
\pastebutton{ugxProblemFiniteUtilityPageEmpty17}{\showpaste}
\tab{5}\spadcommand{numberOfPrimitivePoly(3)$FFPOLY(GF5)\free{GF5 }}
```

```
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPagePatch18}
\begin{paste}{ugxProblemFiniteUtilityPageFull18}{ugxProblemFiniteUtilityPageEmpty18}
\pastebutton{ugxProblemFiniteUtilityPageFull18}{\hidepaste}
\tab{5}\spadcommand{m := monomial(1,1)$SUP(GF5)\bound{m }\free{GF5 }}
\indentrel{3}\begin{verbatim}
   (18)  ?
         Type: SparseUnivariatePolynomial PrimeField 5
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPageEmpty18}
\begin{paste}{ugxProblemFiniteUtilityPageEmpty18}{ugxProblemFiniteUtilityPagePatch18}
\pastebutton{ugxProblemFiniteUtilityPageEmpty18}{\showpaste}
\tab{5}\spadcommand{m := monomial(1,1)$SUP(GF5)\bound{m }\free{GF5 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPagePatch19}
\begin{paste}{ugxProblemFiniteUtilityPageFull19}{ugxProblemFiniteUtilityPageEmpty19}
\pastebutton{ugxProblemFiniteUtilityPageFull19}{\hidepaste}
\tab{5}\spadcommand{f := m**3 + 4*m**2 + m + 2\bound{fx }\free{m }}
\indentrel{3}\begin{verbatim}
          3     2
   (19)  ?  + 4?  + ? + 2
         Type: SparseUnivariatePolynomial PrimeField 5
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPageEmpty19}
\begin{paste}{ugxProblemFiniteUtilityPageEmpty19}{ugxProblemFiniteUtilityPagePatch19}
\pastebutton{ugxProblemFiniteUtilityPageEmpty19}{\showpaste}
\tab{5}\spadcommand{f := m**3 + 4*m**2 + m + 2\bound{fx }\free{m }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPagePatch20}
\begin{paste}{ugxProblemFiniteUtilityPageFull20}{ugxProblemFiniteUtilityPageEmpty20}
\pastebutton{ugxProblemFiniteUtilityPageFull20}{\hidepaste}
\tab{5}\spadcommand{f1 := nextPrimitivePoly(f)$FFPOLY(GF5)\free{fx }\bound{f1 }}
\indentrel{3}\begin{verbatim}
          3     2
   (20)  ?  + 4?  + 4? + 2
Type: Union(SparseUnivariatePolynomial PrimeField 5,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPageEmpty20}
\begin{paste}{ugxProblemFiniteUtilityPageEmpty20}{ugxProblemFiniteUtilityPagePatch20}
\pastebutton{ugxProblemFiniteUtilityPageEmpty20}{\showpaste}
\tab{5}\spadcommand{f1 := nextPrimitivePoly(f)$FFPOLY(GF5)\free{fx }\bound{f1 }}
```

```
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPagePatch21}
\begin{paste}{ugxProblemFiniteUtilityPageFull21}{ugxProblemFiniteUtilityPageEmpty21}
\pastebutton{ugxProblemFiniteUtilityPageFull21}{\hidepaste}
\tab{5}\spadcommand{nextPrimitivePoly(f1)$FFPOLY(GF5)\free{f1 }}
\indentrel{3}\begin{verbatim}
          3     2
   (21)  ?  + 2?  + 3
Type: Union(SparseUnivariatePolynomial PrimeField 5,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPageEmpty21}
\begin{paste}{ugxProblemFiniteUtilityPageEmpty21}{ugxProblemFiniteUtilityPagePatch21}
\pastebutton{ugxProblemFiniteUtilityPageEmpty21}{\showpaste}
\tab{5}\spadcommand{nextPrimitivePoly(f1)$FFPOLY(GF5)\free{f1 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPagePatch22}
\begin{paste}{ugxProblemFiniteUtilityPageFull22}{ugxProblemFiniteUtilityPageEmpty22}
\pastebutton{ugxProblemFiniteUtilityPageFull22}{\hidepaste}
\tab{5}\spadcommand{f := m**3 + m**2 + 4*m + 1\bound{fy }\free{m }}
\indentrel{3}\begin{verbatim}
          3    2
   (22)  ?  + ?  + 4? + 1
          Type: SparseUnivariatePolynomial PrimeField 5
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPageEmpty22}
\begin{paste}{ugxProblemFiniteUtilityPageEmpty22}{ugxProblemFiniteUtilityPagePatch22}
\pastebutton{ugxProblemFiniteUtilityPageEmpty22}{\showpaste}
\tab{5}\spadcommand{f := m**3 + m**2 + 4*m + 1\bound{fy }\free{m }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPagePatch23}
\begin{paste}{ugxProblemFiniteUtilityPageFull23}{ugxProblemFiniteUtilityPageEmpty23}
\pastebutton{ugxProblemFiniteUtilityPageFull23}{\hidepaste}
\tab{5}\spadcommand{f1 := nextNormalPoly(f)$FFPOLY(GF5)\free{fy }\bound{f1y }}
\indentrel{3}\begin{verbatim}
          3    2
   (23)  ?  + ?  + 4? + 3
Type: Union(SparseUnivariatePolynomial PrimeField 5,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPageEmpty23}
\begin{paste}{ugxProblemFiniteUtilityPageEmpty23}{ugxProblemFiniteUtilityPagePatch23}
\pastebutton{ugxProblemFiniteUtilityPageEmpty23}{\showpaste}
```

```
\tab{5}\spadcommand{f1 := nextNormalPoly(f)$FFPOLY(GF5)\free{fy }\bound{f1y }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPagePatch24}
\begin{paste}{ugxProblemFiniteUtilityPageFull24}{ugxProblemFiniteUtilityPageEmpty24}
\pastebutton{ugxProblemFiniteUtilityPageFull24}{\hidepaste}
\tab{5}\spadcommand{nextNormalPoly(f1)$FFPOLY(GF5)\free{f1y }}
\indentrel{3}\begin{verbatim}
          3     2
   (24)  ?  + 2?  + 1
Type: Union(SparseUnivariatePolynomial PrimeField 5,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPageEmpty24}
\begin{paste}{ugxProblemFiniteUtilityPageEmpty24}{ugxProblemFiniteUtilityPagePatch24}
\pastebutton{ugxProblemFiniteUtilityPageEmpty24}{\showpaste}
\tab{5}\spadcommand{nextNormalPoly(f1)$FFPOLY(GF5)\free{f1y }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPagePatch25}
\begin{paste}{ugxProblemFiniteUtilityPageFull25}{ugxProblemFiniteUtilityPageEmpty25}
\pastebutton{ugxProblemFiniteUtilityPageFull25}{\hidepaste}
\tab{5}\spadcommand{GF16 := FFX(FFX(PF 2,2),2);\bound{GF16 }}
\indentrel{3}\begin{verbatim}
                                            Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPageEmpty25}
\begin{paste}{ugxProblemFiniteUtilityPageEmpty25}{ugxProblemFiniteUtilityPagePatch25}
\pastebutton{ugxProblemFiniteUtilityPageEmpty25}{\showpaste}
\tab{5}\spadcommand{GF16 := FFX(FFX(PF 2,2),2);\bound{GF16 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPagePatch26}
\begin{paste}{ugxProblemFiniteUtilityPageFull26}{ugxProblemFiniteUtilityPageEmpty26}
\pastebutton{ugxProblemFiniteUtilityPageFull26}{\hidepaste}
\tab{5}\spadcommand{createIrreduciblePoly(5)$FFPOLY(GF16)\free{GF16 }}
\indentrel{3}\begin{verbatim}
          5
   (26)  ?  + %CZ
Type: SparseUnivariatePolynomial FiniteFieldExtension(FiniteFieldExtension(PrimeField 2,2),2)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPageEmpty26}
\begin{paste}{ugxProblemFiniteUtilityPageEmpty26}{ugxProblemFiniteUtilityPagePatch26}
\pastebutton{ugxProblemFiniteUtilityPageEmpty26}{\showpaste}
\tab{5}\spadcommand{createIrreduciblePoly(5)$FFPOLY(GF16)\free{GF16 }}
```

```
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPagePatch27}
\begin{paste}{ugxProblemFiniteUtilityPageFull27}{ugxProblemFiniteUtilityPageEmpty27}
\pastebutton{ugxProblemFiniteUtilityPageFull27}{\hidepaste}
\tab{5}\spadcommand{random(5)$FFPOLY(GF16)\free{GF16 }}
\indentrel{3}\begin{verbatim}
   (27)
     5             4                             3
    ?  + (%CV + 1)?  + ((%CV + 1)%CZ + %CV + 1)?
   +
                2
    (%CV + 1)?  + ((%CV + 1)%CZ + %CV + 1)?
   +
    (%CV + 1)%CZ + 1
Type: SparseUnivariatePolynomial FiniteFieldExtension(FiniteFieldExtension(PrimeField 2,2),2)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPageEmpty27}
\begin{paste}{ugxProblemFiniteUtilityPageEmpty27}{ugxProblemFiniteUtilityPagePatch27}
\pastebutton{ugxProblemFiniteUtilityPageEmpty27}{\showpaste}
\tab{5}\spadcommand{random(5)$FFPOLY(GF16)\free{GF16 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPagePatch28}
\begin{paste}{ugxProblemFiniteUtilityPageFull28}{ugxProblemFiniteUtilityPageEmpty28}
\pastebutton{ugxProblemFiniteUtilityPageFull28}{\hidepaste}
\tab{5}\spadcommand{random(3,9)$FFPOLY(GF16)\free{GF16 }}
\indentrel{3}\begin{verbatim}
   (28)
     4                   3                         2
    ?  + (%CZ + %CV + 1)?  + (%CV %CZ + %CV + 1)?
   +
    (%CZ + %CV)? + (%CV + 1)%CZ
Type: SparseUnivariatePolynomial FiniteFieldExtension(FiniteFieldExtension(PrimeField 2,2),2)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPageEmpty28}
\begin{paste}{ugxProblemFiniteUtilityPageEmpty28}{ugxProblemFiniteUtilityPagePatch28}
\pastebutton{ugxProblemFiniteUtilityPageEmpty28}{\showpaste}
\tab{5}\spadcommand{random(3,9)$FFPOLY(GF16)\free{GF16 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPagePatch29}
\begin{paste}{ugxProblemFiniteUtilityPageFull29}{ugxProblemFiniteUtilityPageEmpty29}
\pastebutton{ugxProblemFiniteUtilityPageFull29}{\hidepaste}
\tab{5}\spadcommand{GF2 := PrimeField 2;\bound{GF2 }}
\indentrel{3}\begin{verbatim}
```

```
                                         Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPageEmpty29}
\begin{paste}{ugxProblemFiniteUtilityPageEmpty29}{ugxProblemFiniteUtilityPagePatch29}
\pastebutton{ugxProblemFiniteUtilityPageEmpty29}{\showpaste}
\tab{5}\spadcommand{GF2 := PrimeField 2;\bound{GF2 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPagePatch30}
\begin{paste}{ugxProblemFiniteUtilityPageFull30}{ugxProblemFiniteUtilityPageEmpty30}
\pastebutton{ugxProblemFiniteUtilityPageFull30}{\hidepaste}
\tab{5}\spadcommand{F := FFX(GF2,12)\bound{F }\free{GF2 }}
\indentrel{3}\begin{verbatim}
   (30)  FiniteFieldExtension(PrimeField 2,12)
                                         Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPageEmpty30}
\begin{paste}{ugxProblemFiniteUtilityPageEmpty30}{ugxProblemFiniteUtilityPagePatch30}
\pastebutton{ugxProblemFiniteUtilityPageEmpty30}{\showpaste}
\tab{5}\spadcommand{F := FFX(GF2,12)\bound{F }\free{GF2 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPagePatch31}
\begin{paste}{ugxProblemFiniteUtilityPageFull31}{ugxProblemFiniteUtilityPageEmpty31}
\pastebutton{ugxProblemFiniteUtilityPageFull31}{\hidepaste}
\tab{5}\spadcommand{f := createIrreduciblePoly(6)$FFPOLY(GF2)\bound{fz }\free{GF2 }}
\indentrel{3}\begin{verbatim}
          6
   (31)  ?  + ? + 1
          Type: SparseUnivariatePolynomial PrimeField 2
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPageEmpty31}
\begin{paste}{ugxProblemFiniteUtilityPageEmpty31}{ugxProblemFiniteUtilityPagePatch31}
\pastebutton{ugxProblemFiniteUtilityPageEmpty31}{\showpaste}
\tab{5}\spadcommand{f := createIrreduciblePoly(6)$FFPOLY(GF2)\bound{fz }\free{GF2 }}
\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPagePatch32}
\begin{paste}{ugxProblemFiniteUtilityPageFull32}{ugxProblemFiniteUtilityPageEmpty32}
\pastebutton{ugxProblemFiniteUtilityPageFull32}{\hidepaste}
\tab{5}\spadcommand{root := rootOfIrreduciblePoly(f)$FFPOLY2(F,GF2)\free{F GF2 fz }\bound{root }
\indentrel{3}\begin{verbatim}
            11      8      7      5
   (32)  %CS   + %CS  + %CS  + %CS  + %CS + 1
```

```
          Type: FiniteFieldExtension(PrimeField 2,12)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugxProblemFiniteUtilityPageEmpty32}
\begin{paste}{ugxProblemFiniteUtilityPageEmpty32}{ugxProblemFiniteUtilityPagePatch32}
\pastebutton{ugxProblemFiniteUtilityPageEmpty32}{\showpaste}
\tab{5}\spadcommand{root := rootOfIrreduciblePoly(f)$FFPOLY2(F,GF2)\free{F GF2 fz }\bound{root }}
\end{paste}\end{patch}
```

———-

Primary Decomposition of Ideals

— **ug08.ht** —

```
\begin{page}{ugProblemIdealPage}{8.12. Primary Decomposition of Ideals}
\beginscroll

Axiom provides a facility for
the primary decomposition of polynomial ideals over fields of
characteristic zero.  The algorithm
and works in essentially two steps:
\indent{4}
\beginitems
\item[1. ] the problem is solved for 0-dimensional ideals by ``generic''
projection on the last coordinate
\item[2. ] a ``reduction process''
uses localization and ideal quotients to reduce the general case to
the 0-dimensional one.
\enditems
\indent{0}

The Axiom constructor
\axiomType{PolynomialIdeals} represents ideals with coefficients in
any field and supports the basic ideal operations, including
intersection, sum and quotient.  \axiomType{IdealDecompositionPackage}
contains the specific operations for the primary decomposition and the
computation of the radical of an ideal with polynomial coefficients in
a field of characteristic 0 with an effective algorithm for factoring
polynomials.

The following examples illustrate the capabilities of this facility.
%
\xtc{
First consider the ideal generated by
\texht{$x^2 + y^2 - 1$}{\axiom{x**2 + y**2 - 1}}
```

```
(which defines a circle in the \axiom{(x,y)}-plane) and the ideal
generated by \texht{$x^2 - y^2$}{\axiom{x**2 - y**2}} (corresponding to the
straight lines \axiom{x = y} and \axiom{x = -y}.
}{
\spadpaste{(n,m) : List DMP([x,y],FRAC INT) \bound{nm}}
}
\xtc{
}{
\spadpaste{m := [x**2+y**2-1] \free{nm} \bound{m}}
}
\xtc{
}{
\spadpaste{n := [x**2-y**2] \free{nm} \bound{n}}
}
%
%
\xtc{
We find the equations defining the intersection of the two loci.
This correspond to the sum of the associated ideals.
}{
\spadpaste{id := ideal m  + ideal n \free{n m} \bound{id}}
}
%
%
\xtc{
We can check if the locus contains only a finite number of points,
that is, if the ideal is zero-dimensional.
}{
\spadpaste{zeroDim? id \free{id}}
}
\xtc{
}{
\spadpaste{zeroDim?(ideal m) \free{m}}
}
\xtc{
}{
\spadpaste{dimension ideal m \free{m}}
}
\xtc{
We can find polynomial relations among the generators
(\axiom{f} and \axiom{g} are the parametric equations of the knot).
}{
\spadpaste{(f,g):DMP([x,y],FRAC INT) \bound{fg}}
}
\xtc{
}{
\spadpaste{f := x**2-1 \free{fg} \bound{f}}
}
\xtc{
}{
```

```
\spadpaste{g := x*(x**2-1) \free{fg} \bound{g}}
}
\xtc{
}{
\spadpaste{relationsIdeal [f,g] \free{f g}}
}

\xtc{
We can compute the primary decomposition of an ideal.
}{
\spadpaste{l: List DMP([x,y,z],FRAC INT) \bound{ll}}
}
\xtc{
}{
\spadpaste{l:=[x**2+2*y**2,x*z**2-y*z,z**2-4] \free{ll} \bound{l}}
}
\xtc{
}{
\spadpaste{ld:=primaryDecomp ideal l \free{l} \bound{ld}}
}
\xtc{
We can intersect back.
}{
\spadpaste{reduce(intersect,ld) \free{ld}}
}

\xtc{
We can compute the radical of every primary component.
}{
\spadpaste{reduce(intersect,[radical ld.i for i in 1..2]) \free{ld}}
}
\xtc{
Their intersection is equal to the radical of the ideal of \axiom{l}.
}{
\spadpaste{radical ideal l \free{l}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugProblemIdealPagePatch1}
\begin{paste}{ugProblemIdealPageFull1}{ugProblemIdealPageEmpty1}
\pastebutton{ugProblemIdealPageFull1}{\hidepaste}
\tab{5}\spadcommand{(n,m) : List DMP([x,y],FRAC INT)\bound{nm }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugProblemIdealPageEmpty1}
\begin{paste}{ugProblemIdealPageEmpty1}{ugProblemIdealPagePatch1}
\pastebutton{ugProblemIdealPageEmpty1}{\showpaste}
\tab{5}\spadcommand{(n,m) : List DMP([x,y],FRAC INT)\bound{nm }}
\end{paste}\end{patch}

\begin{patch}{ugProblemIdealPagePatch2}
\begin{paste}{ugProblemIdealPageFull2}{ugProblemIdealPageEmpty2}
\pastebutton{ugProblemIdealPageFull2}{\hidepaste}
\tab{5}\spadcommand{m := [x**2+y**2-1]\free{nm }\bound{m }}
\indentrel{3}\begin{verbatim}
          2    2
   (2)  [x  + y  - 1]
Type: List DistributedMultivariatePolynomial([x,y],Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemIdealPageEmpty2}
\begin{paste}{ugProblemIdealPageEmpty2}{ugProblemIdealPagePatch2}
\pastebutton{ugProblemIdealPageEmpty2}{\showpaste}
\tab{5}\spadcommand{m := [x**2+y**2-1]\free{nm }\bound{m }}
\end{paste}\end{patch}

\begin{patch}{ugProblemIdealPagePatch3}
\begin{paste}{ugProblemIdealPageFull3}{ugProblemIdealPageEmpty3}
\pastebutton{ugProblemIdealPageFull3}{\hidepaste}
\tab{5}\spadcommand{n := [x**2-y**2]\free{nm }\bound{n }}
\indentrel{3}\begin{verbatim}
          2    2
   (3)  [x  - y ]
Type: List DistributedMultivariatePolynomial([x,y],Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemIdealPageEmpty3}
\begin{paste}{ugProblemIdealPageEmpty3}{ugProblemIdealPagePatch3}
\pastebutton{ugProblemIdealPageEmpty3}{\showpaste}
\tab{5}\spadcommand{n := [x**2-y**2]\free{nm }\bound{n }}
\end{paste}\end{patch}

\begin{patch}{ugProblemIdealPagePatch4}
\begin{paste}{ugProblemIdealPageFull4}{ugProblemIdealPageEmpty4}
\pastebutton{ugProblemIdealPageFull4}{\hidepaste}
\tab{5}\spadcommand{id := ideal m + ideal n\free{n m }\bound{id }}
\indentrel{3}\begin{verbatim}
          2   1  2   1
   (4)  [x  - ,y  - ]
              2      2
Type: PolynomialIdeals(Fraction Integer,DirectProduct(2,NonNegativeInteger),OrderedVariableList
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemIdealPageEmpty4}
\begin{paste}{ugProblemIdealPageEmpty4}{ugProblemIdealPagePatch4}
\pastebutton{ugProblemIdealPageEmpty4}{\showpaste}
\tab{5}\spadcommand{id := ideal m + ideal n\free{n m }\bound{id }}
\end{paste}\end{patch}

\begin{patch}{ugProblemIdealPagePatch5}
\begin{paste}{ugProblemIdealPageFull5}{ugProblemIdealPageEmpty5}
\pastebutton{ugProblemIdealPageFull5}{\hidepaste}
\tab{5}\spadcommand{zeroDim? id\free{id }}
\indentrel{3}\begin{verbatim}
   (5)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemIdealPageEmpty5}
\begin{paste}{ugProblemIdealPageEmpty5}{ugProblemIdealPagePatch5}
\pastebutton{ugProblemIdealPageEmpty5}{\showpaste}
\tab{5}\spadcommand{zeroDim? id\free{id }}
\end{paste}\end{patch}

\begin{patch}{ugProblemIdealPagePatch6}
\begin{paste}{ugProblemIdealPageFull6}{ugProblemIdealPageEmpty6}
\pastebutton{ugProblemIdealPageFull6}{\hidepaste}
\tab{5}\spadcommand{zeroDim?(ideal m)\free{m }}
\indentrel{3}\begin{verbatim}
   (6)  false
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemIdealPageEmpty6}
\begin{paste}{ugProblemIdealPageEmpty6}{ugProblemIdealPagePatch6}
\pastebutton{ugProblemIdealPageEmpty6}{\showpaste}
\tab{5}\spadcommand{zeroDim?(ideal m)\free{m }}
\end{paste}\end{patch}

\begin{patch}{ugProblemIdealPagePatch7}
\begin{paste}{ugProblemIdealPageFull7}{ugProblemIdealPageEmpty7}
\pastebutton{ugProblemIdealPageFull7}{\hidepaste}
\tab{5}\spadcommand{dimension ideal m\free{m }}
\indentrel{3}\begin{verbatim}
   (7)  1
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

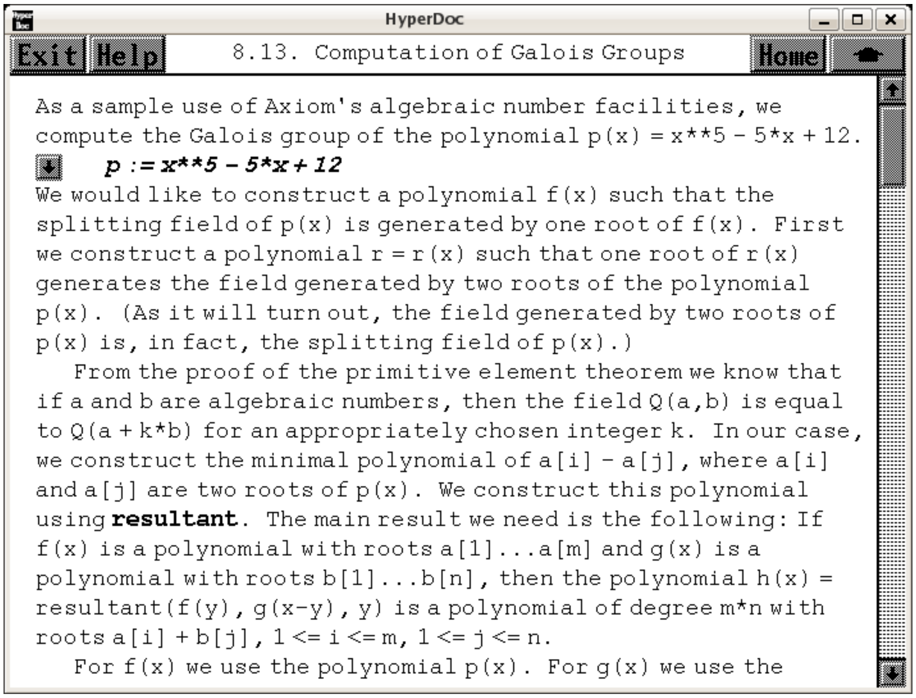
```
\begin{patch}{ugProblemIdealPageEmpty7}
\begin{paste}{ugProblemIdealPageEmpty7}{ugProblemIdealPagePatch7}
\pastebutton{ugProblemIdealPageEmpty7}{\showpaste}
\tab{5}\spadcommand{dimension ideal m\free{m }}
\end{paste}\end{patch}

\begin{patch}{ugProblemIdealPagePatch8}
\begin{paste}{ugProblemIdealPageFull8}{ugProblemIdealPageEmpty8}
\pastebutton{ugProblemIdealPageFull8}{\hidepaste}
\tab{5}\spadcommand{(f,g):DMP([x,y],FRAC INT)\bound{fg }}
\indentrel{3}\begin{verbatim}
                                              Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemIdealPageEmpty8}
\begin{paste}{ugProblemIdealPageEmpty8}{ugProblemIdealPagePatch8}
\pastebutton{ugProblemIdealPageEmpty8}{\showpaste}
\tab{5}\spadcommand{(f,g):DMP([x,y],FRAC INT)\bound{fg }}
\end{paste}\end{patch}

\begin{patch}{ugProblemIdealPagePatch9}
\begin{paste}{ugProblemIdealPageFull9}{ugProblemIdealPageEmpty9}
\pastebutton{ugProblemIdealPageFull9}{\hidepaste}
\tab{5}\spadcommand{f := x**2-1\free{fg }\bound{f }}
\indentrel{3}\begin{verbatim}
         2
   (9)  x  - 1
Type: DistributedMultivariatePolynomial([x,y],Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemIdealPageEmpty9}
\begin{paste}{ugProblemIdealPageEmpty9}{ugProblemIdealPagePatch9}
\pastebutton{ugProblemIdealPageEmpty9}{\showpaste}
\tab{5}\spadcommand{f := x**2-1\free{fg }\bound{f }}
\end{paste}\end{patch}

\begin{patch}{ugProblemIdealPagePatch10}
\begin{paste}{ugProblemIdealPageFull10}{ugProblemIdealPageEmpty10}
\pastebutton{ugProblemIdealPageFull10}{\hidepaste}
\tab{5}\spadcommand{g := x*(x**2-1)\free{fg }\bound{g }}
\indentrel{3}\begin{verbatim}
          3
   (10)  x  - x
Type: DistributedMultivariatePolynomial([x,y],Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemIdealPageEmpty10}
```

```
\begin{paste}{ugProblemIdealPageEmpty10}{ugProblemIdealPagePatch10}
\pastebutton{ugProblemIdealPageEmpty10}{\showpaste}
\tab{5}\spadcommand{g := x*(x**2-1)\free{fg }\bound{g }}
\end{paste}\end{patch}

\begin{patch}{ugProblemIdealPagePatch11}
\begin{paste}{ugProblemIdealPageFull11}{ugProblemIdealPageEmpty11}
\pastebutton{ugProblemIdealPageFull11}{\hidepaste}
\tab{5}\spadcommand{relationsIdeal [f,g]\free{f g }}
\indentrel{3}\begin{verbatim}
   (11)
         2      3      2           2           3
   [- %CY  + %CX  + %CX ] | [%CX= x  - 1,%CY= x  - x]
Type: SuchThat(List Polynomial Fraction Integer,List Equation Polynomial Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemIdealPageEmpty11}
\begin{paste}{ugProblemIdealPageEmpty11}{ugProblemIdealPagePatch11}
\pastebutton{ugProblemIdealPageEmpty11}{\showpaste}
\tab{5}\spadcommand{relationsIdeal [f,g]\free{f g }}
\end{paste}\end{patch}

\begin{patch}{ugProblemIdealPagePatch12}
\begin{paste}{ugProblemIdealPageFull12}{ugProblemIdealPageEmpty12}
\pastebutton{ugProblemIdealPageFull12}{\hidepaste}
\tab{5}\spadcommand{l: List DMP([x,y,z],FRAC INT)\bound{ll }}
\indentrel{3}\begin{verbatim}
                                              Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemIdealPageEmpty12}
\begin{paste}{ugProblemIdealPageEmpty12}{ugProblemIdealPagePatch12}
\pastebutton{ugProblemIdealPageEmpty12}{\showpaste}
\tab{5}\spadcommand{l: List DMP([x,y,z],FRAC INT)\bound{ll }}
\end{paste}\end{patch}

\begin{patch}{ugProblemIdealPagePatch13}
\begin{paste}{ugProblemIdealPageFull13}{ugProblemIdealPageEmpty13}
\pastebutton{ugProblemIdealPageFull13}{\hidepaste}
\tab{5}\spadcommand{l:=[x**2+2*y**2,x*z**2-y*z,z**2-4]\free{ll }\bound{l }}
\indentrel{3}\begin{verbatim}
          2     2    2        2
   (13)  [x  + 2y ,x z  - y z,z  - 4]
Type: List DistributedMultivariatePolynomial([x,y,z],Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemIdealPageEmpty13}
```

```
\begin{paste}{ugProblemIdealPageEmpty13}{ugProblemIdealPagePatch13}
\pastebutton{ugProblemIdealPageEmpty13}{\showpaste}
\tab{5}\spadcommand{l:=[x**2+2*y**2,x*z**2-y*z,z**2-4]\free{ll }\bound{l }}
\end{paste}\end{patch}

\begin{patch}{ugProblemIdealPagePatch14}
\begin{paste}{ugProblemIdealPageFull14}{ugProblemIdealPageEmpty14}
\pastebutton{ugProblemIdealPageFull14}{\hidepaste}
\tab{5}\spadcommand{ld:=primaryDecomp ideal l\free{l }\bound{ld }}
\indentrel{3}\begin{verbatim}
              1    2             1    2
   (14)  [[x +  y,y ,z + 2],[x -  y,y ,z - 2]]
              2                  2
Type: List PolynomialIdeals(Fraction Integer,DirectProduct(3,NonNegativeInteger),OrderedVariable
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemIdealPageEmpty14}
\begin{paste}{ugProblemIdealPageEmpty14}{ugProblemIdealPagePatch14}
\pastebutton{ugProblemIdealPageEmpty14}{\showpaste}
\tab{5}\spadcommand{ld:=primaryDecomp ideal l\free{l }\bound{ld }}
\end{paste}\end{patch}

\begin{patch}{ugProblemIdealPagePatch15}
\begin{paste}{ugProblemIdealPageFull15}{ugProblemIdealPageEmpty15}
\pastebutton{ugProblemIdealPageFull15}{\hidepaste}
\tab{5}\spadcommand{reduce(intersect,ld)\free{ld }}
\indentrel{3}\begin{verbatim}
             1      2  2
   (15)  [x -  y z,y ,z  - 4]
             4
Type: PolynomialIdeals(Fraction Integer,DirectProduct(3,NonNegativeInteger),OrderedVariableList
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemIdealPageEmpty15}
\begin{paste}{ugProblemIdealPageEmpty15}{ugProblemIdealPagePatch15}
\pastebutton{ugProblemIdealPageEmpty15}{\showpaste}
\tab{5}\spadcommand{reduce(intersect,ld)\free{ld }}
\end{paste}\end{patch}

\begin{patch}{ugProblemIdealPagePatch16}
\begin{paste}{ugProblemIdealPageFull16}{ugProblemIdealPageEmpty16}
\pastebutton{ugProblemIdealPageFull16}{\hidepaste}
\tab{5}\spadcommand{reduce(intersect,[radical ld.i for i in 1..2])\free{ld }}
\indentrel{3}\begin{verbatim}
              2
   (16)  [x,y,z  - 4]
Type: PolynomialIdeals(Fraction Integer,DirectProduct(3,NonNegativeInteger),OrderedVariableList
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemIdealPageEmpty16}
\begin{paste}{ugProblemIdealPageEmpty16}{ugProblemIdealPagePatch16}
\pastebutton{ugProblemIdealPageEmpty16}{\showpaste}
\tab{5}\spadcommand{reduce(intersect,[radical ld.i for i in 1..2])\free{ld }}
\end{paste}\end{patch}

\begin{patch}{ugProblemIdealPagePatch17}
\begin{paste}{ugProblemIdealPageFull17}{ugProblemIdealPageEmpty17}
\pastebutton{ugProblemIdealPageFull17}{\hidepaste}
\tab{5}\spadcommand{radical ideal l\free{l }}
\indentrel{3}\begin{verbatim}
               2
   (17)  [x,y,z  - 4]
Type: PolynomialIdeals(Fraction Integer,DirectProduct(3,NonNegativeInteger),OrderedVariableList [x,y,z],Dist
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemIdealPageEmpty17}
\begin{paste}{ugProblemIdealPageEmpty17}{ugProblemIdealPagePatch17}
\pastebutton{ugProblemIdealPageEmpty17}{\showpaste}
\tab{5}\spadcommand{radical ideal l\free{l }}
\end{paste}\end{patch}
```

Computation of Galois Groups



⇐ "FactoredFunctions2" (FactoredFnsTwoXmpPage) 3.44 on page 539
⇒ "Factored" (FactoredXmpPage) 3.43 on page 516
⇒ "Clef" (ugAvailCLEFPage) 6 on page 1487

— ug08.ht —

```
\begin{page}{ugProblemGaloisPage}{8.13. Computation of Galois Groups}
\beginscroll
%
As a sample use of Axiom's algebraic number facilities,
we compute
the Galois group of the polynomial
\texht{$p(x) = x^5 - 5 x + 12$}{\axiom{p(x) = x**5 - 5*x + 12}}.
%
\xtc{
}{
\spadpaste{p := x**5 - 5*x + 12 \bound{p}}
}
%
We would like to construct a polynomial \smath{f(x)}
such that the splitting
field
of \smath{p(x)} is generated by one root of \smath{f(x)}.
First we construct a polynomial \smath{r = r(x)} such that one
```

```
root of \smath{r(x)} generates the field generated by two roots of
the polynomial \smath{p(x)}.
(As it will turn out, the field generated by two roots of
\smath{p(x)} is, in fact, the splitting field of
\smath{p(x)}.)

From the proof of the primitive element theorem we know that
if \smath{a} and \smath{b} are
algebraic numbers, then the field
\texht{${\bf Q}(a,b)$}{\axiom{Q(a,b)}} is equal to
\texht{${\bf Q}(a+kb)$}{\axiom{Q(a + k*b)}} for an
appropriately chosen integer \smath{k}.
In our case, we construct the minimal polynomial of
\texht{$a_i - a_j$}{\axiom{a[i] - a[j]}}, where
\texht{$a_i$}{\axiom{a[i]}} and
\texht{$a_j$}{\axiom{a[j]}} are two roots of \smath{p(x)}.
We construct this polynomial using \axiomFun{resultant}.
The main result we  need is the following:
If \smath{f(x)} is a polynomial with roots
\texht{$a_i \ldots a_m$}{\axiom{a[1]...a[m]}} and
\smath{g(x)} is a polynomial
with roots
\texht{$b_i \ldots b_n$}{\axiom{b[1]...b[n]}}, then the polynomial
\axiom{h(x) = resultant(f(y), g(x-y), y)}
is a polynomial of degree \smath{m*n} with
roots
\texht{$a_i + b_j, i = 1 \ldots m, j = 1 \ldots n$}
{\axiom{a[i] + b[j], 1 <= i <= m, 1 <= j <= n}}.

\xtc{
For \smath{f(x)} we use the polynomial \smath{p(x)}.
For \smath{g(x)} we use the polynomial \smath{-p(-x)}.
Thus, the polynomial we  first construct is
\axiom{resultant(p(y), -p(y-x), y)}.
}{
\spadpaste{q := resultant(eval(p,x,y),-eval(p,x,y-x),y) \free{p} \bound{q}}
}
%
The roots of \smath{q(x)} are
\texht{$a_i - a_j, i \leq 1, j \leq 5$}
{\axiom{a[i] - a[j], 1 <= i,j <= 5}}.
Of course, there are five pairs \smath{(i,j)} with \smath{i = j},
so \axiom{0} is a 5-fold root of \smath{q(x)}.
%
\xtc{
Let's get rid of this factor.
}{
\spadpaste{q1 := exquo(q, x**5) \free{q} \bound{q1}}
}
\xtc{
```

```
Factor the polynomial \axiom{q1}.
}{
\spadpaste{factoredQ := factor q1 \free{q1} \bound{factoredQ}}
}

We see that \axiom{q1} has two irreducible factors, each of degree
\axiom{10}.  (The fact that the polynomial \axiom{q1} has two factors
of degree \axiom{10} is enough to show that the Galois group of
\smath{p(x)} is the dihedral group of order \axiom{10}.\footnote{See
McKay, Soicher, Computing Galois Groups over the Rationals, Journal of
Number Theory 20, 273-281 (1983).  We do not assume the results of
this paper, however, and we continue with the computation.}  Note that
the type of \axiom{factoredQ} is \axiomType{FR POLY INT}, that is,
\axiomType{Factored Polynomial Integer}.  This is a special data type
for recording factorizations of polynomials with integer coefficients
(see \downlink{`Factored'}{FactoredXmpPage} \ignore{Factored}).

\xtc{
We can access the individual factors using the operation
\axiomFunFrom{nthFactor}{Factored}.
}{
\spadpaste{r := nthFactor(factoredQ,1) \free{factoredQ} \bound{r}}
}
%

Consider the polynomial \smath{r = r(x)}.
This is the minimal polynomial of the difference of two roots of
\smath{p(x)}.
Thus, the splitting field of \smath{p(x)} contains a subfield of
degree \axiom{10}.
We show that this subfield is, in fact, the splitting field of
\smath{p(x)} by showing that \smath{p(x)} factors completely
over this field.
%
\xtc{
First we create a symbolic root of the polynomial \smath{r(x)}.
(We replaced \axiom{x} by \axiom{b} in the
polynomial \axiom{r} so that our symbolic root would be
printed as \axiom{b}.)
}{
\spadpaste{beta:AN := rootOf(eval(r,x,b)) \free{r} \bound{beta}}
}
\xtc{
We next tell Axiom to view \smath{p(x)} as a univariate polynomial
in \axiom{x}
with algebraic number coefficients.
This is accomplished with this type declaration.
}{
\spadpaste{p := p::UP(x,INT)::UP(x,AN) \free{p} \bound{declareP}}
}
```

```
%
%
\xtc{
Factor \smath{p(x)} over the field
\texht{${\bf Q}(\beta)$}{\axiom{Q(beta)}}.
(This computation will take some time!)
}{
\spadpaste{algFactors := factor(p,[beta]) \free{declareP beta}
\bound{algFactors}}
}
%
When factoring over number fields, it is important to specify the
field over which the polynomial is to be factored, as polynomials
have different factorizations over different fields.
When you use the operation \axiomFun{factor}, the field over which
the polynomial is factored is the field generated by
\indent{4}
\beginitems
\item[1. ] the algebraic numbers that appear
in the coefficients of the polynomial, and
\item[2. ] the algebraic numbers that
appear in a list passed as an optional second argument of the operation.
\enditems
\indent{0}
In our case, the coefficients of \axiom{p}
are all rational integers and only \axiom{beta}
appears in the list, so the field is simply
\texht{${\bf Q}(\beta)$}{\axiom{Q(beta)}}.
%
\xtc{
It was necessary to give the list \axiom{[beta]}
as a second argument of the operation
because otherwise the polynomial would have been factored over the field
generated by its coefficients, namely the rational numbers.
}{
\spadpaste{factor(p) \free{declareP}}
}
%
We have shown that the splitting field of \smath{p(x)} has degree
\axiom{10}.
Since the symmetric group of degree 5 has only one transitive subgroup
of order \axiom{10}, we know that the Galois group of \smath{p(x)} must be
this group, the dihedral group
of order \axiom{10}.
Rather than stop here, we explicitly compute the action of the Galois
group on the roots of \smath{p(x)}.

First we assign the roots of \smath{p(x)} as the values of five
variables.
\xtc{
```

```
We can obtain an individual root by negating the constant coefficient of
one of the factors of \smath{p(x)}.
}{
\spadpaste{factor1 := nthFactor(algFactors,1) \free{algFactors}
\bound{factor1}}
}
\xtc{
}{
\spadpaste{root1 := -coefficient(factor1,0) \free{factor1} \bound{root1}}
}
%
%
\xtc{
We can obtain a list of all the roots in this way.
}{
\spadpaste{
roots := [-coefficient(nthFactor(algFactors,i),0) for i in 1..5]
\free{algFactors} \bound{roots}}
}

The expression
\begin{verbatim}
- coefficient(nthFactor(algFactors, i), 0)}
\end{verbatim}
is the \eth{\axiom{i }} root
of \smath{p(x)} and the elements of \axiom{roots} are the \eth{\axiom{i }}
roots of \smath{p(x)} as \axiom{i} ranges from \axiom{1} to \axiom{5}.

\xtc{
Assign the roots as the values of the variables \axiom{a1,...,a5}.
}{
\spadpaste{
(a1,a2,a3,a4,a5) := (roots.1,roots.2,roots.3,roots.4,roots.5)
\free{roots} \bound{ais}}
}
%

Next we express the roots of \smath{r(x)} as polynomials in
\axiom{beta}.  We could obtain these roots by calling the operation
\axiomFun{factor}: \axiom{factor(r, [beta])} factors \axiom{r(x)} over
\texht{${\bf Q}(\beta)$}{\axiom{Q(beta)}}.  However, this is a lengthy
computation and we can obtain the roots of \smath{r(x)} as differences
of the roots \axiom{a1,...,a5} of \smath{p(x)}.  Only ten of these
differences are roots of \smath{r(x)} and the other ten are roots of
the other irreducible factor of \axiom{q1}.  We can determine if a
given value is a root of \smath{r(x)} by evaluating \smath{r(x)} at
that particular value.  (Of course, the order in which factors are
returned by the operation \axiomFun{factor} is unimportant and may
change with different implementations of the operation.  Therefore, we
cannot predict in advance which differences are roots of \smath{r(x)}
```

```
and which are not.)

\xtc{
Let's look at four examples (two are roots of \smath{r(x)} and
two are not).
}{
\spadpaste{eval(r,x,a1 - a2) \free{ais}}
}
\xtc{
}{
\spadpaste{eval(r,x,a1 - a3) \free{ais}}
}
\xtc{
}{
\spadpaste{eval(r,x,a1 - a4) \free{ais}}
}
\xtc{
}{
\spadpaste{eval(r,x,a1 - a5) \free{ais}}
}
%

Take one of the differences that was a root of \smath{r(x)}
and assign it to the variable \axiom{bb}.
\xtc{
For example, if \axiom{eval(r,x,a1 - a4)} returned \axiom{0}, you would
enter this.
}{
\spadpaste{bb := a1 - a4 \free{ais} \bound{bb}}
}
Of course, if the difference is, in fact, equal to the root \axiom{beta},
you should choose another root of \smath{r(x)}.

Automorphisms of the splitting field are given by mapping a generator of
the field, namely \axiom{beta}, to other roots of its minimal polynomial.
Let's see what happens when \axiom{beta} is mapped to \axiom{bb}.
%
\xtc{
We compute the images of the roots \axiom{a1,...,a5}
under this automorphism:
}{
\spadpaste{aa1 := subst(a1,beta = bb) \free{beta bb ais} \bound{aa1}}
}
\xtc{
}{
\spadpaste{aa2 := subst(a2,beta = bb) \free{beta bb ais} \bound{aa2}}
}
\xtc{
}{
\spadpaste{aa3 := subst(a3,beta = bb) \free{beta bb ais} \bound{aa3}}
```

```
}
\xtc{
}{
\spadpaste{aa4 := subst(a4,beta = bb) \free{beta bb ais} \bound{aa4}}
}
\xtc{
}{
\spadpaste{aa5 := subst(a5,beta = bb) \free{beta bb ais} \bound{aa5}}
}
%
Of course, the values \axiom{aa1,...,aa5} are simply a permutation
of the values \axiom{a1,...,a5}.
%
\xtc{
Let's find the value of \axiom{aa1} (execute as many of the
following five commands as necessary).
}{
\spadpaste{(aa1 = a1) :: Boolean \free{aa1}}
}
\xtc{
}{
\spadpaste{(aa1 = a2) :: Boolean \free{aa1}}
}
\xtc{
}{
\spadpaste{(aa1 = a3) :: Boolean \free{aa1}}
}
\xtc{
}{
\spadpaste{(aa1 = a4) :: Boolean \free{aa1}}
}
\xtc{
}{
\spadpaste{(aa1 = a5) :: Boolean \free{aa1}}
}

Proceeding in this fashion, you can find the values of
\axiom{aa2,...aa5}.\footnote{Here you should use the \Clef{} line
editor.  See \downlink{``Clef''}{ugAvailCLEFPage} in
Section 1.1.1\ignore{ugAvailCLEF} for more information
about \Clef{}.}  You have represented the automorphism \axiom{beta ->
bb} as a permutation of the roots \axiom{a1,...,a5}.  If you wish, you
can repeat this computation for all the roots of \smath{r(x)} and
represent the Galois group of \smath{p(x)} as a subgroup of the
symmetric group on five letters.

Here are two other problems that you may attack in a similar fashion:
\indent{4}
\beginitems
\item[1. ] Show that the Galois group of
```

```
\texht{$p(x) = x^4 + 2 x^3 - 2 x^2 - 3 x + 1$}{\axiom{
p(x) = x**4 + 2*x**3 - 2*x**2 - 3*x + 1}}
is the dihedral group of order eight.
(The splitting field of this polynomial is the Hilbert class field
of
the quadratic field
\texht{${\bf Q}(\sqrt{145})$}{\axiom{Q(sqrt(145))}}.)
\item[2. ] Show that the Galois group of
\texht{$p(x) = x^6 + 108$}{\axiom{p(x) = x**6 + 108}}
has order 6 and is
isomorphic to \texht{$S_3,$}{} the symmetric group on three letters.
(The splitting field of this polynomial is the splitting field of
\texht{$x^3 - 2$}{\axiom{x**3 - 2}}.)
\enditems
\indent{0}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugProblemGaloisPagePatch1}
\begin{paste}{ugProblemGaloisPageFull1}{ugProblemGaloisPageEmpty1}
\pastebutton{ugProblemGaloisPageFull1}{\hidepaste}
\tab{5}\spadcommand{p := x**5 - 5*x + 12\bound{p }}
\indentrel{3}\begin{verbatim}
         5
   (1)  x  - 5x + 12
                              Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPageEmpty1}
\begin{paste}{ugProblemGaloisPageEmpty1}{ugProblemGaloisPagePatch1}
\pastebutton{ugProblemGaloisPageEmpty1}{\showpaste}
\tab{5}\spadcommand{p := x**5 - 5*x + 12\bound{p }}
\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPagePatch2}
\begin{paste}{ugProblemGaloisPageFull2}{ugProblemGaloisPageEmpty2}
\pastebutton{ugProblemGaloisPageFull2}{\hidepaste}
\tab{5}\spadcommand{q := resultant(eval(p,x,y),-eval(p,x,y-x),y)\free{p }\bound{q }}
\indentrel{3}\begin{verbatim}
   (2)
      25      21        17          15        13
     x   - 50x   - 2375x   + 90000x   - 5000x
   +
            11          9           7            5
     2700000x   + 250000x  + 18000000x  + 64000000x
                              Type: Polynomial Integer
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPageEmpty2}
\begin{paste}{ugProblemGaloisPageEmpty2}{ugProblemGaloisPagePatch2}
\pastebutton{ugProblemGaloisPageEmpty2}{\showpaste}
\tab{5}\spadcommand{q := resultant(eval(p,x,y),-eval(p,x,y-x),y)\free{p }\bound{q }}
\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPagePatch3}
\begin{paste}{ugProblemGaloisPageFull3}{ugProblemGaloisPageEmpty3}
\pastebutton{ugProblemGaloisPageFull3}{\hidepaste}
\tab{5}\spadcommand{q1 := exquo(q, x**5)\free{q }\bound{q1 }}
\indentrel{3}\begin{verbatim}
   (3)
      20      16        12         10        8           6
     x   - 50x   - 2375x   + 90000x   - 5000x  + 2700000x
   +
            4             2
     250000x  + 18000000x  + 64000000
                    Type: Union(Polynomial Integer,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPageEmpty3}
\begin{paste}{ugProblemGaloisPageEmpty3}{ugProblemGaloisPagePatch3}
\pastebutton{ugProblemGaloisPageEmpty3}{\showpaste}
\tab{5}\spadcommand{q1 := exquo(q, x**5)\free{q }\bound{q1 }}
\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPagePatch4}
\begin{paste}{ugProblemGaloisPageFull4}{ugProblemGaloisPageEmpty4}
\pastebutton{ugProblemGaloisPageFull4}{\hidepaste}
\tab{5}\spadcommand{factoredQ := factor q1\free{q1 }\bound{factoredQ }}
\indentrel{3}\begin{verbatim}
   (4)
       10      8      6        4        2
     (x   - 10x  - 75x  + 1500x  - 5500x  + 16000)
  *
       10      8       6       4        2
     (x   + 10x  + 125x  + 500x  + 2500x  + 4000)
                      Type: Factored Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPageEmpty4}
\begin{paste}{ugProblemGaloisPageEmpty4}{ugProblemGaloisPagePatch4}
\pastebutton{ugProblemGaloisPageEmpty4}{\showpaste}
\tab{5}\spadcommand{factoredQ := factor q1\free{q1 }\bound{factoredQ }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugProblemGaloisPagePatch5}
\begin{paste}{ugProblemGaloisPageFull5}{ugProblemGaloisPageEmpty5}
\pastebutton{ugProblemGaloisPageFull5}{\hidepaste}
\tab{5}\spadcommand{r := nthFactor(factoredQ,1)\free{factoredQ }\bound{r }}
\indentrel{3}\begin{verbatim}
         10      8      6        4         2
   (5)  x   - 10x  - 75x  + 1500x  - 5500x  + 16000
                               Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPageEmpty5}
\begin{paste}{ugProblemGaloisPageEmpty5}{ugProblemGaloisPagePatch5}
\pastebutton{ugProblemGaloisPageEmpty5}{\showpaste}
\tab{5}\spadcommand{r := nthFactor(factoredQ,1)\free{factoredQ }\bound{r }}
\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPagePatch6}
\begin{paste}{ugProblemGaloisPageFull6}{ugProblemGaloisPageEmpty6}
\pastebutton{ugProblemGaloisPageFull6}{\hidepaste}
\tab{5}\spadcommand{beta:AN := rootOf(eval(r,x,b))\free{r }\bound{beta }}
\indentrel{3}\begin{verbatim}
   (6)  b
                                  Type: AlgebraicNumber
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPageEmpty6}
\begin{paste}{ugProblemGaloisPageEmpty6}{ugProblemGaloisPagePatch6}
\pastebutton{ugProblemGaloisPageEmpty6}{\showpaste}
\tab{5}\spadcommand{beta:AN := rootOf(eval(r,x,b))\free{r }\bound{beta }}
\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPagePatch7}
\begin{paste}{ugProblemGaloisPageFull7}{ugProblemGaloisPageEmpty7}
\pastebutton{ugProblemGaloisPageFull7}{\hidepaste}
\tab{5}\spadcommand{p := p::UP(x,INT)::UP(x,AN)\free{p }\bound{declareP }}
\indentrel{3}\begin{verbatim}
         5
   (7)  x  - 5x + 12
          Type: UnivariatePolynomial(x,AlgebraicNumber)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPageEmpty7}
\begin{paste}{ugProblemGaloisPageEmpty7}{ugProblemGaloisPagePatch7}
\pastebutton{ugProblemGaloisPageEmpty7}{\showpaste}
\tab{5}\spadcommand{p := p::UP(x,INT)::UP(x,AN)\free{p }\bound{declareP }}
\end{paste}\end{patch}
```

```
\begin{patch}{ugProblemGaloisPagePatch8}
\begin{paste}{ugProblemGaloisPageFull8}{ugProblemGaloisPageEmpty8}
\pastebutton{ugProblemGaloisPageFull8}{\hidepaste}
\tab{5}\spadcommand{algFactors := factor(p,[beta])\free{declareP beta }\bound{algFactors }}
\indentrel{3}\begin{verbatim}
   (8)
       x
     +
                 9       8       7         6          5
           - 85b  - 116b  + 780b  + 2640b  + 14895b
         +
                   4          3          2
           - 8820b  - 127050b  - 327000b  - 405200b
         +
           2062400
      /
         1339200
  *
                8       6        4          2
          - 17b  + 156b  + 2979b  - 25410b  - 14080
     (x + )
                               66960
  *
       x
     +
                8        6          4           2
           143b  - 2100b  - 10485b  + 290550b  - 334800b
         +
           - 960800
      /
         669600
  *
       x
     +
                8        6          4           2
           143b  - 2100b  - 10485b  + 290550b  + 334800b
         +
           - 960800
      /
         669600
  *
       x
     +
               9       8       7         6          5
           85b  - 116b  - 780b  + 2640b  - 14895b
         +
                   4          3          2
           - 8820b  + 127050b  - 327000b  + 405200b
         +
           2062400
```

```
     /
        1339200
 Type: Factored UnivariatePolynomial(x,AlgebraicNumber)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPageEmpty8}
\begin{paste}{ugProblemGaloisPageEmpty8}{ugProblemGaloisPagePatch8}
\pastebutton{ugProblemGaloisPageEmpty8}{\showpaste}
\tab{5}\spadcommand{algFactors := factor(p,[beta])\free{declareP beta }\bound{algFactors }}
\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPagePatch9}
\begin{paste}{ugProblemGaloisPageFull9}{ugProblemGaloisPageEmpty9}
\pastebutton{ugProblemGaloisPageFull9}{\hidepaste}
\tab{5}\spadcommand{factor(p)\free{declareP }}
\indentrel{3}\begin{verbatim}
        5
   (9)  x  - 5x + 12
 Type: Factored UnivariatePolynomial(x,AlgebraicNumber)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPageEmpty9}
\begin{paste}{ugProblemGaloisPageEmpty9}{ugProblemGaloisPagePatch9}
\pastebutton{ugProblemGaloisPageEmpty9}{\showpaste}
\tab{5}\spadcommand{factor(p)\free{declareP }}
\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPagePatch10}
\begin{paste}{ugProblemGaloisPageFull10}{ugProblemGaloisPageEmpty10}
\pastebutton{ugProblemGaloisPageFull10}{\hidepaste}
\tab{5}\spadcommand{factor1 := nthFactor(algFactors,1)\free{algFactors }\bound{factor1 }}
\indentrel{3}\begin{verbatim}
   (10)
     x
   +
             9       8       7        6         5
        - 85b  - 116b  + 780b  + 2640b  + 14895b
      +
             4          3          2
      - 8820b  - 127050b  - 327000b  - 405200b + 2062400
    /
      1339200
         Type: UnivariatePolynomial(x,AlgebraicNumber)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPageEmpty10}
\begin{paste}{ugProblemGaloisPageEmpty10}{ugProblemGaloisPagePatch10}
```

```
\pastebutton{ugProblemGaloisPageEmpty10}{\showpaste}
\tab{5}\spadcommand{factor1 := nthFactor(algFactors,1)\free{algFactors }\bound{factor1 }}
\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPagePatch11}
\begin{paste}{ugProblemGaloisPageFull11}{ugProblemGaloisPageEmpty11}
\pastebutton{ugProblemGaloisPageFull11}{\hidepaste}
\tab{5}\spadcommand{root1 := -coefficient(factor1,0)\free{factor1 }\bound{root1 }}
\indentrel{3}\begin{verbatim}
   (11)
          9       8       7        6         5        4
       85b  + 116b  - 780b  - 2640b  - 14895b  + 8820b
     +
              3          2
       127050b  + 327000b  + 405200b - 2062400
  /
     1339200
                                  Type: AlgebraicNumber
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPageEmpty11}
\begin{paste}{ugProblemGaloisPageEmpty11}{ugProblemGaloisPagePatch11}
\pastebutton{ugProblemGaloisPageEmpty11}{\showpaste}
\tab{5}\spadcommand{root1 := -coefficient(factor1,0)\free{factor1 }\bound{root1 }}
\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPagePatch12}
\begin{paste}{ugProblemGaloisPageFull12}{ugProblemGaloisPageEmpty12}
\pastebutton{ugProblemGaloisPageFull12}{\hidepaste}
\tab{5}\spadcommand{roots := [-coefficient(nthFactor(algFactors,i),0) for i in 1..5]\free{algFac
\indentrel{3}\begin{verbatim}
   (12)
   [
            9       8       7        6         5        4
         85b  + 116b  - 780b  - 2640b  - 14895b  + 8820b
       +
                3          2
         127050b  + 327000b  + 405200b - 2062400
    /
       1339200
      ,
       8       6        4         2
    17b  - 156b  - 2979b  + 25410b  + 14080
    ,
                    66960

               8        6         4          2
         - 143b  + 2100b  + 10485b  - 290550b  + 334800b
       +
```

```
          960800
      /
        669600
       ,

                8        6         4          2
          - 143b  + 2100b  + 10485b  - 290550b  - 334800b
        +
          960800
      /
        669600
       ,

               9       8       7        6         5
          - 85b  + 116b  + 780b  - 2640b  + 14895b
        +
               4          3          2
          8820b  - 127050b  + 327000b  - 405200b - 2062400
      /
        1339200
       ]
                              Type: List AlgebraicNumber
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPageEmpty12}
\begin{paste}{ugProblemGaloisPageEmpty12}{ugProblemGaloisPagePatch12}
\pastebutton{ugProblemGaloisPageEmpty12}{\showpaste}
\tab{5}\spadcommand{roots := [-coefficient(nthFactor(algFactors,i),0) for i in 1..5]\free{algFactors }\bound
\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPagePatch13}
\begin{paste}{ugProblemGaloisPageFull13}{ugProblemGaloisPageEmpty13}
\pastebutton{ugProblemGaloisPageFull13}{\hidepaste}
\tab{5}\spadcommand{(a1,a2,a3,a4,a5) := (roots.1,roots.2,roots.3,roots.4,roots.5)\free{roots }\bound{ais }}
\indentrel{3}\begin{verbatim}
   (13)
             9       8       7        6         5        4
        - 85b  + 116b  + 780b  - 2640b  + 14895b  + 8820b
      +
                  3          2
        - 127050b  + 327000b  - 405200b - 2062400
    /
      1339200
                                   Type: AlgebraicNumber
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPageEmpty13}
\begin{paste}{ugProblemGaloisPageEmpty13}{ugProblemGaloisPagePatch13}
```

```
\pastebutton{ugProblemGaloisPageEmpty13}{\showpaste}
\tab{5}\spadcommand{(a1,a2,a3,a4,a5) := (roots.1,roots.2,roots.3,roots.4,roots.5)\free{roots }\b
\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPagePatch14}
\begin{paste}{ugProblemGaloisPageFull14}{ugProblemGaloisPageEmpty14}
\pastebutton{ugProblemGaloisPageFull14}{\hidepaste}
\tab{5}\spadcommand{eval(r,x,a1 - a2)\free{ais }}
\indentrel{3}\begin{verbatim}
   (14)  0
                       Type: Polynomial AlgebraicNumber
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPageEmpty14}
\begin{paste}{ugProblemGaloisPageEmpty14}{ugProblemGaloisPagePatch14}
\pastebutton{ugProblemGaloisPageEmpty14}{\showpaste}
\tab{5}\spadcommand{eval(r,x,a1 - a2)\free{ais }}
\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPagePatch15}
\begin{paste}{ugProblemGaloisPageFull15}{ugProblemGaloisPageEmpty15}
\pastebutton{ugProblemGaloisPageFull15}{\hidepaste}
\tab{5}\spadcommand{eval(r,x,a1 - a3)\free{ais }}
\indentrel{3}\begin{verbatim}
   (15)
             9         8          7          6           5
       47905b  + 66920b  - 536100b  - 980400b  - 3345075b
     +
                 4           3             2
       - 5787000b  + 75572250b  + 161688000b  - 184600000b
     +
       - 710912000
  /
     4464
                       Type: Polynomial AlgebraicNumber
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPageEmpty15}
\begin{paste}{ugProblemGaloisPageEmpty15}{ugProblemGaloisPagePatch15}
\pastebutton{ugProblemGaloisPageEmpty15}{\showpaste}
\tab{5}\spadcommand{eval(r,x,a1 - a3)\free{ais }}
\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPagePatch16}
\begin{paste}{ugProblemGaloisPageFull16}{ugProblemGaloisPageEmpty16}
\pastebutton{ugProblemGaloisPageFull16}{\hidepaste}
\tab{5}\spadcommand{eval(r,x,a1 - a4)\free{ais }}
\indentrel{3}\begin{verbatim}
```

```
   (16)  0
                       Type: Polynomial AlgebraicNumber
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPageEmpty16}
\begin{paste}{ugProblemGaloisPageEmpty16}{ugProblemGaloisPagePatch16}
\pastebutton{ugProblemGaloisPageEmpty16}{\showpaste}
\tab{5}\spadcommand{eval(r,x,a1 - a4)\free{ais }}
\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPagePatch17}
\begin{paste}{ugProblemGaloisPageFull17}{ugProblemGaloisPageEmpty17}
\pastebutton{ugProblemGaloisPageFull17}{\hidepaste}
\tab{5}\spadcommand{eval(r,x,a1 - a5)\free{ais }}
\indentrel{3}\begin{verbatim}
             8        6         4          2
         405b  + 3450b  - 19875b  - 198000b  - 588000
   (17)  
                              31
                       Type: Polynomial AlgebraicNumber
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPageEmpty17}
\begin{paste}{ugProblemGaloisPageEmpty17}{ugProblemGaloisPagePatch17}
\pastebutton{ugProblemGaloisPageEmpty17}{\showpaste}
\tab{5}\spadcommand{eval(r,x,a1 - a5)\free{ais }}
\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPagePatch18}
\begin{paste}{ugProblemGaloisPageFull18}{ugProblemGaloisPageEmpty18}
\pastebutton{ugProblemGaloisPageFull18}{\hidepaste}
\tab{5}\spadcommand{bb := a1 - a4\free{ais }\bound{bb }}
\indentrel{3}\begin{verbatim}
   (18)
          9       8       7        6         5         4
       85b  + 402b  - 780b  - 6840b  - 14895b  - 12150b
     +
             3          2
       127050b  + 908100b  + 1074800b - 3984000
  /
     1339200
                                  Type: AlgebraicNumber
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPageEmpty18}
\begin{paste}{ugProblemGaloisPageEmpty18}{ugProblemGaloisPagePatch18}
\pastebutton{ugProblemGaloisPageEmpty18}{\showpaste}
```

```
\tab{5}\spadcommand{bb := a1 - a4\free{ais }\bound{bb }}
\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPagePatch19}
\begin{paste}{ugProblemGaloisPageFull19}{ugProblemGaloisPageEmpty19}
\pastebutton{ugProblemGaloisPageFull19}{\hidepaste}
\tab{5}\spadcommand{aa1 := subst(a1,beta = bb)\free{beta bb ais }\bound{aa1 }}
\indentrel{3}\begin{verbatim}
   (19)
              8        6         4          2
       - 143b  + 2100b  + 10485b  - 290550b  + 334800b
     +
       960800
  /
     669600
                                  Type: AlgebraicNumber
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPageEmpty19}
\begin{paste}{ugProblemGaloisPageEmpty19}{ugProblemGaloisPagePatch19}
\pastebutton{ugProblemGaloisPageEmpty19}{\showpaste}
\tab{5}\spadcommand{aa1 := subst(a1,beta = bb)\free{beta bb ais }\bound{aa1 }}
\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPagePatch20}
\begin{paste}{ugProblemGaloisPageFull20}{ugProblemGaloisPageEmpty20}
\pastebutton{ugProblemGaloisPageFull20}{\hidepaste}
\tab{5}\spadcommand{aa2 := subst(a2,beta = bb)\free{beta bb ais }\bound{aa2 }}
\indentrel{3}\begin{verbatim}
   (20)
             9       8       7        6         5        4
       - 85b  + 116b  + 780b  - 2640b  + 14895b  + 8820b
     +
                 3          2
       - 127050b  + 327000b  - 405200b - 2062400
  /
     1339200
                                  Type: AlgebraicNumber
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPageEmpty20}
\begin{paste}{ugProblemGaloisPageEmpty20}{ugProblemGaloisPagePatch20}
\pastebutton{ugProblemGaloisPageEmpty20}{\showpaste}
\tab{5}\spadcommand{aa2 := subst(a2,beta = bb)\free{beta bb ais }\bound{aa2 }}
\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPagePatch21}
\begin{paste}{ugProblemGaloisPageFull21}{ugProblemGaloisPageEmpty21}
```

```
\pastebutton{ugProblemGaloisPageFull21}{\hidepaste}
\tab{5}\spadcommand{aa3 := subst(a3,beta = bb)\free{beta bb ais }\bound{aa3 }}
\indentrel{3}\begin{verbatim}
   (21)
         9       8       7        6         5        4
       85b  + 116b  - 780b  - 2640b  - 14895b  + 8820b
     +
              3          2
       127050b  + 327000b  + 405200b - 2062400
  /
     1339200
                                  Type: AlgebraicNumber
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPageEmpty21}
\begin{paste}{ugProblemGaloisPageEmpty21}{ugProblemGaloisPagePatch21}
\pastebutton{ugProblemGaloisPageEmpty21}{\showpaste}
\tab{5}\spadcommand{aa3 := subst(a3,beta = bb)\free{beta bb ais }\bound{aa3 }}
\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPagePatch22}
\begin{paste}{ugProblemGaloisPageFull22}{ugProblemGaloisPageEmpty22}
\pastebutton{ugProblemGaloisPageFull22}{\hidepaste}
\tab{5}\spadcommand{aa4 := subst(a4,beta = bb)\free{beta bb ais }\bound{aa4 }}
\indentrel{3}\begin{verbatim}
   (22)
            8        6         4          2
       - 143b  + 2100b  + 10485b  - 290550b  - 334800b
     +
       960800
  /
     669600
                                  Type: AlgebraicNumber
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPageEmpty22}
\begin{paste}{ugProblemGaloisPageEmpty22}{ugProblemGaloisPagePatch22}
\pastebutton{ugProblemGaloisPageEmpty22}{\showpaste}
\tab{5}\spadcommand{aa4 := subst(a4,beta = bb)\free{beta bb ais }\bound{aa4 }}
\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPagePatch23}
\begin{paste}{ugProblemGaloisPageFull23}{ugProblemGaloisPageEmpty23}
\pastebutton{ugProblemGaloisPageFull23}{\hidepaste}
\tab{5}\spadcommand{aa5 := subst(a5,beta = bb)\free{beta bb ais }\bound{aa5 }}
\indentrel{3}\begin{verbatim}
           8       6        4         2
         17b  - 156b  - 2979b  + 25410b  + 14080
```

```
   (23)
                          66960
                                  Type: AlgebraicNumber
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPageEmpty23}
\begin{paste}{ugProblemGaloisPageEmpty23}{ugProblemGaloisPagePatch23}
\pastebutton{ugProblemGaloisPageEmpty23}{\showpaste}
\tab{5}\spadcommand{aa5 := subst(a5,beta = bb)\free{beta bb ais }\bound{aa5 }}
\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPagePatch24}
\begin{paste}{ugProblemGaloisPageFull24}{ugProblemGaloisPageEmpty24}
\pastebutton{ugProblemGaloisPageFull24}{\hidepaste}
\tab{5}\spadcommand{(aa1 = a1) :: Boolean\free{aa1 }}
\indentrel{3}\begin{verbatim}
   (24)  false
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPageEmpty24}
\begin{paste}{ugProblemGaloisPageEmpty24}{ugProblemGaloisPagePatch24}
\pastebutton{ugProblemGaloisPageEmpty24}{\showpaste}
\tab{5}\spadcommand{(aa1 = a1) :: Boolean\free{aa1 }}
\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPagePatch25}
\begin{paste}{ugProblemGaloisPageFull25}{ugProblemGaloisPageEmpty25}
\pastebutton{ugProblemGaloisPageFull25}{\hidepaste}
\tab{5}\spadcommand{(aa1 = a2) :: Boolean\free{aa1 }}
\indentrel{3}\begin{verbatim}
   (25)  false
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPageEmpty25}
\begin{paste}{ugProblemGaloisPageEmpty25}{ugProblemGaloisPagePatch25}
\pastebutton{ugProblemGaloisPageEmpty25}{\showpaste}
\tab{5}\spadcommand{(aa1 = a2) :: Boolean\free{aa1 }}
\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPagePatch26}
\begin{paste}{ugProblemGaloisPageFull26}{ugProblemGaloisPageEmpty26}
\pastebutton{ugProblemGaloisPageFull26}{\hidepaste}
\tab{5}\spadcommand{(aa1 = a3) :: Boolean\free{aa1 }}
\indentrel{3}\begin{verbatim}
   (26)  true
```

```
                                            Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPageEmpty26}
\begin{paste}{ugProblemGaloisPageEmpty26}{ugProblemGaloisPagePatch26}
\pastebutton{ugProblemGaloisPageEmpty26}{\showpaste}
\tab{5}\spadcommand{(aa1 = a3) :: Boolean\free{aa1 }}
\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPagePatch27}
\begin{paste}{ugProblemGaloisPageFull27}{ugProblemGaloisPageEmpty27}
\pastebutton{ugProblemGaloisPageFull27}{\hidepaste}
\tab{5}\spadcommand{(aa1 = a4) :: Boolean\free{aa1 }}
\indentrel{3}\begin{verbatim}
   (27)  false
                                            Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPageEmpty27}
\begin{paste}{ugProblemGaloisPageEmpty27}{ugProblemGaloisPagePatch27}
\pastebutton{ugProblemGaloisPageEmpty27}{\showpaste}
\tab{5}\spadcommand{(aa1 = a4) :: Boolean\free{aa1 }}
\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPagePatch28}
\begin{paste}{ugProblemGaloisPageFull28}{ugProblemGaloisPageEmpty28}
\pastebutton{ugProblemGaloisPageFull28}{\hidepaste}
\tab{5}\spadcommand{(aa1 = a5) :: Boolean\free{aa1 }}
\indentrel{3}\begin{verbatim}
   (28)  false
                                            Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemGaloisPageEmpty28}
\begin{paste}{ugProblemGaloisPageEmpty28}{ugProblemGaloisPagePatch28}
\pastebutton{ugProblemGaloisPageEmpty28}{\showpaste}
\tab{5}\spadcommand{(aa1 = a5) :: Boolean\free{aa1 }}
\end{paste}\end{patch}
```

Non-Associative Algebras and Genetic Laws

⇒ "notitle" (OctonionXmpPage) 3.81 on page 1044

— ug08.ht —

```
\begin{page}{ugProblemGeneticPage}
{8.14. Non-Associative Algebras and Genetic Laws}
\beginscroll

Many algebraic structures of mathematics and Axiom
have a multiplication operation \axiomOp{*} that satisfies
the associativity law
\texht{$a*(b*c) = (a*b)*c$}{\spad{a*(b*c) = (a*b)*c}}
for all \smath{a}, \smath{b} and \smath{c}.
The octonions (see
\downlink{`Octonion'}{OctonionXmpPage}\ignore{Octonion})
are a well known exception.
There are many other interesting non-associative structures,
such as the class of
Lie algebras.\footnote{Two Axiom implementations of Lie algebras
are \spadtype{LieSquareMatrix} and \spadtype{FreeNilpotentLie}.}
Lie algebras can be used, for example, to analyse Lie symmetry
algebras of partial differential equations.
In this section we show a different application of non-associative
algebras, the modelling of genetic laws.

The Axiom library contains several constructors for
creating non-assoc\-i\-a\-tive structures,
ranging from the categories \spadtype{Monad},
\spadtype{NonAssociativeRng}, and
\spadtype{FramedNonAssociativeAlgebra}, to the domains
\spadtype{AlgebraGivenByStructuralConstants} and
\spadtype{GenericNonAssociativeAlgebra}.
Furthermore, the package \spadtype{AlgebraPackage} provides
operations for analysing the structure of such algebras.\footnote{%
The interested reader can learn more about these aspects of the
Axiom library from the paper
``Computations in Algebras of Finite Rank,''
by Johannes Grabmeier and Robert Wisbauer,
Technical Report, IBM Heidelberg Scientific Center, 1992.}

Mendel's genetic laws are often written in a form like
\texht{
\narrowDisplay{
Aa \times Aa = {1\over 4}AA + {1\over 2}Aa + {1\over 4}aa.}}
{\spad{Aa * Aa = (1/4)*AA + (1/2)*Aa + (1/4)*aa.}
}
The implementation of general algebras in Axiom allows us to
use this as the definition for multiplication in an algebra.
Hence, it is possible to study
questions of genetic inheritance using Axiom.
To demonstrate this more precisely,  we discuss one example from a
monograph of \texht{A. W\"orz-Busekros}{A. Woerz-Busekros},
```

```
where you can also find a general setting of this theory.\footnote{%
\texht{W\"{o}rz-Busekros}{Woerz-Busekros}, A.,
{\it Algebras in Genetics},
Springer Lectures Notes in Biomathematics 36, Berlin e.a. (1980).
In particular, see example 1.3.}

We assume that there is an infinitely large random mating population.
Random mating of two gametes \texht{$a_i$}{\spad{ai}}  and
\texht{$a_j$}{\spad{aj}} gives zygotes
\texht{$a_ia_j$}{\spad{ai aj}}, which produce new gametes.
In classical Mendelian segregation we have
\texht{$a_ia_j = {1 \over 2}a_i+{1 \over 2}a_j$}
{\spad{ai aj = (1/2)*ai+(1/2)*aj}}.
In general, we have
\texht{\narrowDisplay{a_ia_j = \sum_{k=1}^n \gamma_{i,j}^k\ a_k.}}%
{\spad{ai aj = gammaij1 a1 + gammaij2 a2 + ... + gammaijn an}}
The segregation rates \texht{$\gamma_{i,j}$}{\spad{gammaij}}
are  the structural constants of an
\smath{n}-dimensional algebra.
This is provided in Axiom by
the constructor \spadtype{AlgebraGivenByStructuralConstants}
(abbreviation \spadtype{ALGSC}).

Consider two coupled autosomal loci with alleles
\smath{A},\smath{a}, \smath{B}, and \smath{b}, building four
different gametes
\texht{$a_1 =  AB, a_2 =  Ab, a_3 =  aB,$ and $a_4 =  ab$}%
{\spad{a1 :=  AB, a2 :=  Ab, a3 :=  aB,} and \spad{a4 :=  ab}}.
The zygotes \texht{$a_ia_j$}{\spad{ai aj}}
produce gametes \texht{$a_i$}{\spad{ai}} and
\texht{$a_j$}{\spad{aj}}
with classical Mendelian segregation.
Zygote \texht{$a_1a_4$}{a1 a4} undergoes transition
to \texht{$a_2a_3$}{\spad{a2 a3}}
and vice versa with probability
\texht{$0 \le \theta \le {1\over 2}$}{0 <= theta <= 1/2}.

\xtc{
Define a list
\texht{$[(\gamma_{i,j}^k) 1 \le k \le 4]$}{\spad{[(gammaijk) 1 <= k <= 4]}}
of four four-by-four matrices giving the segregation
rates.
We use the value \smath{1/10} for \smath{\theta}.
}{
\spadpaste{segregationRates : List SquareMatrix(4,FRAC INT) :=
[matrix [ [1, 1/2, 1/2, 9/20], [1/2, 0, 1/20, 0], [1/2, 1/20, 0, 0],
[9/20, 0, 0, 0] ], matrix [ [0, 1/2, 0, 1/20], [1/2, 1, 9/20, 1/2],
[0, 9/20, 0, 0], [1/20, 1/2, 0, 0] ], matrix [ [0, 0, 1/2, 1/20],
[0, 0, 9/20, 0], [1/2, 9/20, 1, 1/2], [1/20, 0, 1/2, 0] ],
matrix [ [0, 0, 0, 9/20], [0, 0, 1/20, 1/2], [0, 1/20, 0, 1/2],
```

```
[9/20, 1/2, 1/2, 1] ] ] \bound{segregationRates}}
}
\xtc{
Choose the appropriate symbols for the basis of gametes,
}{
\spadpaste{gametes := ['AB,'Ab,'aB,'ab]  \bound{gametes}}
}
\xtc{
Define the algebra.
}{
\spadpaste{A := ALGSC(FRAC INT, 4, gametes, segregationRates);
\bound{A}\free{gametes, segregationRates}}
}

\xtc{
What are the probabilities for zygote
\texht{$a_1a_4$}{a1 a4} to produce the different gametes?
}{
\spadpaste{a := basis()\$A;  a.1*a.4}
}

Elements in this algebra whose coefficients sum to one play a
distinguished role.
They represent a population with the distribution of gametes
reflected by the coefficients with respect to the basis of
gametes.

Random mating of different populations \smath{x} and \smath{y}
is described by their product \smath{x*y}.

\xtc{
This product is commutative only
if the gametes are not sex-dependent, as in our example.
}{
\spadpaste{commutative?()\$A \free{A}}
}
\xtc{
In general, it is not associative.
}{
\spadpaste{associative?()\$A \free{A}}
}

Random mating within a population \smath{x} is described by
\smath{x*x.}
The next generation is \smath{(x*x)*(x*x).}

\xtc{
Use decimal numbers to compare the distributions more easily.
}{
\spadpaste{x : ALGSC(DECIMAL, 4, gametes, segregationRates) :=
```

```
convert [3/10, 1/5, 1/10, 2/5]\bound{x}\free{gametes, segregationRates}}
}
\xtc{
To compute directly the gametic distribution in the fifth
generation, we use \spadfun{plenaryPower}.
}{
\spadpaste{plenaryPower(x,5) \free{x}}
}

We now ask two questions:
Does this distribution converge to an equilibrium state?
What are the distributions that are stable?

\xtc{
This is an invariant of the algebra and it is used to answer
the first question.
The new indeterminates describe a symbolic distribution.
}{
\spadpaste{q := leftRankPolynomial()
\$GCNAALG(FRAC INT, 4, gametes, segregationRates) ::
UP(Y, POLY FRAC INT)\bound{q}\free{gametes, segregationRates}}
}
\xtc{
Because the coefficient \texht{${9 \over 20}$}{\axiom{9/20}} has absolute
value less than 1, all distributions do converge,
by a theorem of this theory.
}{
\spadpaste{factor(q :: POLY FRAC INT) \free{q}}
}
\xtc{
The second question is answered by searching for idempotents in the
algebra.
}{
\spadpaste{cI := conditionsForIdempotents()
\$GCNAALG(FRAC INT, 4, gametes, segregationRates)
\bound{cI} \free{gametes, segregationRates}}
}
\xtc{
Solve these equations and look at the first solution.
}{
\spadpaste{gbs:= groebnerFactorize cI; gbs.1\free{cI}\bound{gbs}}
}

Further analysis using the package \spadtype{PolynomialIdeals}
shows that there is a two-dimensional variety of equilibrium states and all
other solutions are contained in it.

\xtc{
Choose one equilibrium state by setting two indeterminates to
concrete values.
```

```
}{
\spadpaste{sol := solve concat(gbs.1,[\%x1-1/10,\%x2-1/10])
\bound{sol} \free{gbs}}
}
\xtc{
}{
\spadpaste{e : A := represents reverse (map(rhs, sol.1)
:: List FRAC INT)\bound{e} \free{A, sol}}
}
\xtc{
Verify the result.
}{
\spadpaste{e*e-e \free{e}}
}
\endscroll
\autobuttons
\end{page}

\begin{patch}{ugProblemGeneticPagePatch1}
\begin{paste}{ugProblemGeneticPageFull1}{ugProblemGeneticPageEmpty1}
\pastebutton{ugProblemGeneticPageFull1}{\hidepaste}
\tab{5}\spadcommand{segregationRates : List SquareMatrix(4,FRAC INT) := [matrix [ [1, 1/2, 1/2,
\indentrel{3}\begin{verbatim}
   (1)
        1   1    9      1        1          1    1
    1           0      0     0   0
        2   2   20      2       20          2   20

    1        1      1        9  1            9
       0     0      1        0   0     0
    2       20      2       20  2           20
   [              ,               ,               ,
    1    1             9          1    9      1
         0    0    0      0   0        1
    2   20            20          2   20      2

     9              1  1            1      1
      0    0    0       0   0     0      0
    20             20  2           20      2
                9
    0    0    0
               20

             1   1
    0    0
            20   2
                  ]
         1       1
    0      0
        20       2
```

```
     9  1   1
            1
    20  2   2
            Type: List SquareMatrix(4,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemGeneticPageEmpty1}
\begin{paste}{ugProblemGeneticPageEmpty1}{ugProblemGeneticPagePatch1}
\pastebutton{ugProblemGeneticPageEmpty1}{\showpaste}
\tab{5}\spadcommand{segregationRates : List SquareMatrix(4,FRAC INT) := [matrix [ [1, 1/2, 1/2, 9/20], [1/2
\end{paste}\end{patch}

\begin{patch}{ugProblemGeneticPagePatch2}
\begin{paste}{ugProblemGeneticPageFull2}{ugProblemGeneticPageEmpty2}
\pastebutton{ugProblemGeneticPageFull2}{\hidepaste}
\tab{5}\spadcommand{gametes := ['AB,'Ab,'aB,'ab]\bound{gametes }}
\indentrel{3}\begin{verbatim}
   (2)  [AB,Ab,aB,ab]
          Type: List OrderedVariableList [AB,Ab,aB,ab]
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemGeneticPageEmpty2}
\begin{paste}{ugProblemGeneticPageEmpty2}{ugProblemGeneticPagePatch2}
\pastebutton{ugProblemGeneticPageEmpty2}{\showpaste}
\tab{5}\spadcommand{gametes := ['AB,'Ab,'aB,'ab]\bound{gametes }}
\end{paste}\end{patch}

\begin{patch}{ugProblemGeneticPagePatch3}
\begin{paste}{ugProblemGeneticPageFull3}{ugProblemGeneticPageEmpty3}
\pastebutton{ugProblemGeneticPageFull3}{\hidepaste}
\tab{5}\spadcommand{A := ALGSC(FRAC INT, 4, gametes, segregationRates);\bound{A }\free{gametes segregationRa
\indentrel{3}\begin{verbatim}
                                          Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemGeneticPageEmpty3}
\begin{paste}{ugProblemGeneticPageEmpty3}{ugProblemGeneticPagePatch3}
\pastebutton{ugProblemGeneticPageEmpty3}{\showpaste}
\tab{5}\spadcommand{A := ALGSC(FRAC INT, 4, gametes, segregationRates);\bound{A }\free{gametes segregationRa
\end{paste}\end{patch}

\begin{patch}{ugProblemGeneticPagePatch4}
\begin{paste}{ugProblemGeneticPageFull4}{ugProblemGeneticPageEmpty4}
\pastebutton{ugProblemGeneticPageFull4}{\hidepaste}
\tab{5}\spadcommand{a := basis()$A; a.1*a.4}
\indentrel{3}\begin{verbatim}
```

```
         9       1       1       9
   (4)   ab +  aB +  Ab +  AB
        20      20      20      20
Type: AlgebraGivenByStructuralConstants(Fraction Integer,4,[AB,Ab,aB,ab],[MATRIX,MATRIX,MATRIX,M
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemGeneticPageEmpty4}
\begin{paste}{ugProblemGeneticPageEmpty4}{ugProblemGeneticPagePatch4}
\pastebutton{ugProblemGeneticPageEmpty4}{\showpaste}
\tab{5}\spadcommand{a := basis()$A; a.1*a.4}
\end{paste}\end{patch}

\begin{patch}{ugProblemGeneticPagePatch5}
\begin{paste}{ugProblemGeneticPageFull5}{ugProblemGeneticPageEmpty5}
\pastebutton{ugProblemGeneticPageFull5}{\hidepaste}
\tab{5}\spadcommand{commutative?()$A\free{A }}
\indentrel{3}\begin{verbatim}
   algebra is commutative
   (5)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemGeneticPageEmpty5}
\begin{paste}{ugProblemGeneticPageEmpty5}{ugProblemGeneticPagePatch5}
\pastebutton{ugProblemGeneticPageEmpty5}{\showpaste}
\tab{5}\spadcommand{commutative?()$A\free{A }}
\end{paste}\end{patch}

\begin{patch}{ugProblemGeneticPagePatch6}
\begin{paste}{ugProblemGeneticPageFull6}{ugProblemGeneticPageEmpty6}
\pastebutton{ugProblemGeneticPageFull6}{\hidepaste}
\tab{5}\spadcommand{associative?()$A\free{A }}
\indentrel{3}\begin{verbatim}
   algebra is not associative
   (6)  false
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemGeneticPageEmpty6}
\begin{paste}{ugProblemGeneticPageEmpty6}{ugProblemGeneticPagePatch6}
\pastebutton{ugProblemGeneticPageEmpty6}{\showpaste}
\tab{5}\spadcommand{associative?()$A\free{A }}
\end{paste}\end{patch}

\begin{patch}{ugProblemGeneticPagePatch7}
\begin{paste}{ugProblemGeneticPageFull7}{ugProblemGeneticPageEmpty7}
\pastebutton{ugProblemGeneticPageFull7}{\hidepaste}
```

```
\tab{5}\spadcommand{x : ALGSC(DECIMAL, 4, gametes, segregationRates) := convert [3/10, 1/5, 1/10, 2/5]\bound
\indentrel{3}\begin{verbatim}
   (7)  0.4ab + 0.1aB + 0.2Ab + 0.3AB
Type: AlgebraGivenByStructuralConstants(DecimalExpansion,4,[AB,Ab,aB,ab],[MATRIX,MATRIX,MATRIX,MATRIX])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemGeneticPageEmpty7}
\begin{paste}{ugProblemGeneticPageEmpty7}{ugProblemGeneticPagePatch7}
\pastebutton{ugProblemGeneticPageEmpty7}{\showpaste}
\tab{5}\spadcommand{x : ALGSC(DECIMAL, 4, gametes, segregationRates) := convert [3/10, 1/5, 1/10, 2/5]\bound
\end{paste}\end{patch}

\begin{patch}{ugProblemGeneticPagePatch8}
\begin{paste}{ugProblemGeneticPageFull8}{ugProblemGeneticPageEmpty8}
\pastebutton{ugProblemGeneticPageFull8}{\hidepaste}
\tab{5}\spadcommand{plenaryPower(x,5)\free{x }}
\indentrel{3}\begin{verbatim}
   (8)  0.36561ab + 0.13439aB + 0.23439Ab + 0.26561AB
Type: AlgebraGivenByStructuralConstants(DecimalExpansion,4,[AB,Ab,aB,ab],[MATRIX,MATRIX,MATRIX,MATRIX])
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemGeneticPageEmpty8}
\begin{paste}{ugProblemGeneticPageEmpty8}{ugProblemGeneticPagePatch8}
\pastebutton{ugProblemGeneticPageEmpty8}{\showpaste}
\tab{5}\spadcommand{plenaryPower(x,5)\free{x }}
\end{paste}\end{patch}

\begin{patch}{ugProblemGeneticPagePatch9}
\begin{paste}{ugProblemGeneticPageFull9}{ugProblemGeneticPageEmpty9}
\pastebutton{ugProblemGeneticPageFull9}{\hidepaste}
\tab{5}\spadcommand{q := leftRankPolynomial()$GCNAALG(FRAC INT, 4, gametes, segregationRates) :: UP(Y, POLY
\indentrel{3}\begin{verbatim}
   (9)
      3      29       29       29       29      2
     Y  + (-  %x4 -  %x3 -  %x2 -  %x1)Y
             20       20       20       20
   +
         9    2     9        9        9            9    2
         %x4  + ( %x3 +  %x2 +  %x1)%x4 +  %x3
        20          10       10       10           20
      +
          9        9             9    2    9
        ( %x2 +  %x1)%x3 +  %x2  +  %x1 %x2
         10       10            20        10
      +
         9    2
         %x1
        20
```

```
    *
       Y
Type: UnivariatePolynomial(Y,Polynomial Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemGeneticPageEmpty9}
\begin{paste}{ugProblemGeneticPageEmpty9}{ugProblemGeneticPagePatch9}
\pastebutton{ugProblemGeneticPageEmpty9}{\showpaste}
\tab{5}\spadcommand{q := leftRankPolynomial()$GCNAALG(FRAC INT, 4, gametes, segregationRates) ::
\end{paste}\end{patch}

\begin{patch}{ugProblemGeneticPagePatch10}
\begin{paste}{ugProblemGeneticPageFull10}{ugProblemGeneticPageEmpty10}
\pastebutton{ugProblemGeneticPageFull10}{\hidepaste}
\tab{5}\spadcommand{factor(q :: POLY FRAC INT)\free{q }}
\indentrel{3}\begin{verbatim}
   (10)
     (Y - %x4 - %x3 - %x2 - %x1)
  *
           9        9        9        9
     (Y -  %x4 -  %x3 -  %x2 -  %x1)Y
          20       20       20       20
             Type: Factored Polynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemGeneticPageEmpty10}
\begin{paste}{ugProblemGeneticPageEmpty10}{ugProblemGeneticPagePatch10}
\pastebutton{ugProblemGeneticPageEmpty10}{\showpaste}
\tab{5}\spadcommand{factor(q :: POLY FRAC INT)\free{q }}
\end{paste}\end{patch}

\begin{patch}{ugProblemGeneticPagePatch11}
\begin{paste}{ugProblemGeneticPageFull11}{ugProblemGeneticPageEmpty11}
\pastebutton{ugProblemGeneticPageFull11}{\hidepaste}
\tab{5}\spadcommand{cI := conditionsForIdempotents()$GCNAALG(FRAC INT, 4, gametes, segregationRa
\indentrel{3}\begin{verbatim}
   (11)
    9            1                              2
   [ %x1 %x4 + ( %x2 + %x1)%x3 + %x1 %x2 + %x1  - %x1,
    10           10
          1            9              2
    (%x2 +  %x1)%x4 +  %x2 %x3 + %x2  + (%x1 - 1)%x2,
          10           10
          1              2     9
    (%x3 +  %x1)%x4 + %x3  + ( %x2 + %x1 - 1)%x3,
          10                   10
       2                9              1
    %x4  + (%x3 + %x2 +  %x1 - 1)%x4 +  %x2 %x3]
```

```
                        10                10
                 Type: List Polynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemGeneticPageEmpty11}
\begin{paste}{ugProblemGeneticPageEmpty11}{ugProblemGeneticPagePatch11}
\pastebutton{ugProblemGeneticPageEmpty11}{\showpaste}
\tab{5}\spadcommand{cI := conditionsForIdempotents()$GCNAALG(FRAC INT, 4, gametes, segregationRates)\bound{c
\end{paste}\end{patch}

\begin{patch}{ugProblemGeneticPagePatch12}
\begin{paste}{ugProblemGeneticPageFull12}{ugProblemGeneticPageEmpty12}
\pastebutton{ugProblemGeneticPageFull12}{\hidepaste}
\tab{5}\spadcommand{gbs:= groebnerFactorize cI; gbs.1\free{cI }\bound{gbs }}
\indentrel{3}\begin{verbatim}
   (12)
   [%x4 + %x3 + %x2 + %x1 - 1,
                                   2
    (%x2 + %x1)%x3 + %x1 %x2 + %x1  - %x1]
                 Type: List Polynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemGeneticPageEmpty12}
\begin{paste}{ugProblemGeneticPageEmpty12}{ugProblemGeneticPagePatch12}
\pastebutton{ugProblemGeneticPageEmpty12}{\showpaste}
\tab{5}\spadcommand{gbs:= groebnerFactorize cI; gbs.1\free{cI }\bound{gbs }}
\end{paste}\end{patch}

\begin{patch}{ugProblemGeneticPagePatch13}
\begin{paste}{ugProblemGeneticPageFull13}{ugProblemGeneticPageEmpty13}
\pastebutton{ugProblemGeneticPageFull13}{\hidepaste}
\tab{5}\spadcommand{sol := solve concat(gbs.1,[\%x1-1/10,\%x2-1/10])\bound{sol }\free{gbs }}
\indentrel{3}\begin{verbatim}
                2      2       1       1
   (13)  [[%x4= ,%x3= ,%x2= ,%x1= ]]
                5      5      10      10
   Type: List List Equation Fraction Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemGeneticPageEmpty13}
\begin{paste}{ugProblemGeneticPageEmpty13}{ugProblemGeneticPagePatch13}
\pastebutton{ugProblemGeneticPageEmpty13}{\showpaste}
\tab{5}\spadcommand{sol := solve concat(gbs.1,[\%x1-1/10,\%x2-1/10])\bound{sol }\free{gbs }}
\end{paste}\end{patch}

\begin{patch}{ugProblemGeneticPagePatch14}
\begin{paste}{ugProblemGeneticPageFull14}{ugProblemGeneticPageEmpty14}
```

```
\pastebutton{ugProblemGeneticPageFull14}{\hidepaste}
\tab{5}\spadcommand{e : A := represents reverse (map(rhs, sol.1) :: List FRAC INT)\bound{e }\fre
\indentrel{3}\begin{verbatim}
         2      2       1       1
   (14)   ab +  aB +  Ab +  AB
         5      5      10      10
Type: AlgebraGivenByStructuralConstants(Fraction Integer,4,[AB,Ab,aB,ab],[MATRIX,MATRIX,MATRIX,M
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemGeneticPageEmpty14}
\begin{paste}{ugProblemGeneticPageEmpty14}{ugProblemGeneticPagePatch14}
\pastebutton{ugProblemGeneticPageEmpty14}{\showpaste}
\tab{5}\spadcommand{e : A := represents reverse (map(rhs, sol.1) :: List FRAC INT)\bound{e }\fre
\end{paste}\end{patch}

\begin{patch}{ugProblemGeneticPagePatch15}
\begin{paste}{ugProblemGeneticPageFull15}{ugProblemGeneticPageEmpty15}
\pastebutton{ugProblemGeneticPageFull15}{\hidepaste}
\tab{5}\spadcommand{e*e-e\free{e }}
\indentrel{3}\begin{verbatim}
   (15)  0
Type: AlgebraGivenByStructuralConstants(Fraction Integer,4,[AB,Ab,aB,ab],[MATRIX,MATRIX,MATRIX,M
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugProblemGeneticPageEmpty15}
\begin{paste}{ugProblemGeneticPageEmpty15}{ugProblemGeneticPagePatch15}
\pastebutton{ugProblemGeneticPageEmpty15}{\showpaste}
\tab{5}\spadcommand{e*e-e\free{e }}
\end{paste}\end{patch}
```

Chapter 13

Users Guide Chapter 10 (ug10.ht)

Interactive Programming

⇒ "notitle" (ugAppGraphicsPage) 20 on page 2589
⇒ "notitle" (ugIntProgDrawingPage) 13 on page 2338
⇒ "notitle" (ugIntProgRibbonPage) 13 on page 2344
⇒ "notitle" (ugIntProgColorPage) 13 on page 2347
⇒ "notitle" (ugIntProgPLCPage) 13 on page 2348
⇒ "notitle" (ugIntProgColorArrPage) 13 on page 2355
⇒ "notitle" (ugIntProgVecFieldsPage) 13 on page 2357
⇒ "notitle" (ugIntProgCompFunsPage) 13 on page 2361
⇒ "notitle" (ugIntProgFunctionsPage) 13 on page 2364
⇒ "notitle" (ugIntProgNewtonPage) 13 on page 2366

— **ug10.ht** —

```
\begin{page}{ugIntProgPage}{10. Interactive Programming}
\beginscroll

Programming in the interpreter is easy.
So is the use of Axiom's graphics facility.
Both are rather flexible and allow you to use them for many
interesting applications.
However, both require learning some basic ideas and skills.

All graphics examples in the \Gallery{} section are either
produced directly by interactive commands or by interpreter
programs.
Four of these programs are introduced here.
```

```
By the end of this chapter you will know enough about graphics and
programming in the interpreter to not only understand all these
examples, but to tackle interesting and difficult problems on your
own.
\downlink{``Programs for Axiom Images''}{ugAppGraphicsPage} in
Appendix G\ignore{ugAppGraphics} lists
all the remaining commands and
programs used to create these images.

\beginmenu
\menudownlink{{10.1. Drawing Ribbons Interactively}}
{ugIntProgDrawingPage}
\menudownlink{{10.2. A Ribbon Program}}
{ugIntProgRibbonPage}
\menudownlink{{10.3. Coloring and Positioning Ribbons}}
{ugIntProgColorPage}
\menudownlink{{10.4. Points, Lines, and Curves}}
{ugIntProgPLCPage}
\menudownlink{{10.5. A Bouquet of Arrows}}
{ugIntProgColorArrPage}
\menudownlink{{10.6. Drawing Complex Vector Fields}}
{ugIntProgVecFieldsPage}
\menudownlink{{10.7. Drawing Complex Functions}}
{ugIntProgCompFunsPage}
\menudownlink{{10.8. Functions Producing Functions}}
{ugIntProgFunctionsPage}
\menudownlink{{10.9. Automatic Newton Iteration Formulas}}
{ugIntProgNewtonPage}
\endmenu
\endscroll
\autobuttons
\end{page}
```

Drawing Ribbons Interactively

⇒ “notitle” (ugGraphPage) 11 on page 1967

— ug10.ht —

```
\begin{page}{ugIntProgDrawingPage}{10.1. Drawing Ribbons Interactively}
\beginscroll
%

We begin our discussion of interactive graphics with the creation
of a useful facility: plotting ribbons of two-graphs in
three-space.
```

```
Suppose you want to draw the \twodim{} graphs of \smath{n}
functions \texht{$f_i(x), 1 \leq i \leq n,$}{\spad{f_i(x), 1 <= i
<= 5,}} all over some fixed range of \smath{x}.
One approach is to create a \twodim{} graph for each one, then
superpose one on top of the other.
What you will more than likely get is a jumbled mess.
Even if you make each function a different color, the result is
likely to be confusing.

A better approach is to display each of the \smath{f_i(x)} in three
dimensions as a ‘‘ribbon’’ of some appropriate width along the
\smath{y}-direction, laying down each  ribbon next to the
previous one.
A ribbon is simply a function of \smath{x} and \smath{y} depending
only on \smath{x.}

We illustrate this for \smath{f_i(x)} defined as simple powers of
\smath{x} for \smath{x} ranging between \smath{-1} and \smath{1}.

\psXtc{
Draw the ribbon for \texht{$z = x^2$}{\spad{z=x ** 2}}.
}{
\graphpaste{draw(x**2,x=-1..1,y=0..1)}
}{
\epsffile[0 0 295 295]{../ps/ribbon1.ps}
}

Now that was easy!
What you get is a ‘‘wire-mesh’’ rendition of the ribbon.
That’s fine for now.
Notice that the mesh-size is small in both the \smath{x} and the
\smath{y} directions.
Axiom normally computes points in both these directions.
This is unnecessary.
One step is all we need in the \smath{y}-direction.
To have Axiom economize on \spad{y}-points, we re-draw the
ribbon with option \spad{var2Steps == 1}.

\psXtc{
Re-draw the ribbon, but with option \spad{var2Steps == 1}
so that only \spad{1} step is computed in the
\smath{y} direction.
}{
\graphpaste{vp := draw(x**2,x=-1..1,y=0..1,var2Steps==1) \bound{d1}}
}{
\epsffile[0 0 295 295]{../ps/ribbon2.ps}
}

The operation has created a viewport, that is, a graphics window
on your screen.
```

```
We assigned the viewport to \spad{vp} and now we manipulate
its contents.


Graphs are objects, like numbers and algebraic expressions.
You may want to do some experimenting with graphs.
For example, say
\begin{verbatim}
showRegion(vp, "on")
\end{verbatim}
to put a bounding box around the ribbon.
Try it!
Issue \spad{rotate(vp, -45, 90)} to rotate the
figure \smath{-45} longitudinal degrees and \smath{90} latitudinal
degrees.

\psXtc{
Here is a different rotation.
This turns the graph so you can view it along the \smath{y}-axis.
}{
\spadpaste{rotate(vp, 0, -90)\bound{d3}\free{d1}}
}{
\epsffile[0 0 295 295]{../ps/ribbon2r.ps}
}

There are many other things you can do.
In fact, most everything you can do interactively using the
\threedim{} control panel (such as translating, zooming, resizing,
coloring, perspective and lighting selections) can also be done
directly by operations (see \downlink{‘‘Graphics’’}{ugGraphPage}
in Chapter 7\ignore{ugGraph} for more details).

When you are done experimenting, say \spad{reset(vp)} to restore the
picture to its original position and settings.


Let’s add another ribbon to our picture---one
for \texht{$x^3$}{\spad{x**3}}.
Since \smath{y} ranges from \smath{0} to \smath{1} for the
first ribbon, now let \smath{y} range from \smath{1} to
\smath{2.}
This puts the second ribbon next to the first one.

How do you add a second ribbon to the viewport?
One method is
to extract the ‘‘space’’ component from the
viewport using the operation
\spadfunFrom{subspace}{ThreeDimensionalViewport}.
You can think of the space component as the object inside the
window (here, the ribbon).
```

```
Let's call it \spad{sp}.
To add the second ribbon, you draw the second ribbon using the
option \spad{space == sp}.

\xtc{
Extract the space component of \spad{vp}.
}{
\spadpaste{sp := subspace(vp)\bound{d5}\free{d1}}
}

\psXtc{
Add the ribbon for
\texht{$x^3$}{\spad{x**3}} alongside that for
\texht{$x^2$}{\spad{x**2}}.
}{
\graphpaste{vp := draw(x**3,x=-1..1,y=1..2,var2Steps==1, space==sp)
\bound{d6}\free{d5}}
}{
\epsffile[0 0 295 295]{../ps/ribbons.ps}
}

Unless you moved the original viewport, the new viewport covers
the old one.
You might want to check that the old object is still there by
moving the top window.

Let's show quadrilateral polygon outlines on the ribbons and then
enclose the ribbons in a box.

\psXtc{
Show quadrilateral polygon outlines.
}{
\spadpaste{drawStyle(vp,"shade");outlineRender(vp,"on")
\bound{d10}\free{d6}}
}{
\epsffile[0 0 295 295]{../ps/ribbons2.ps}
}
\psXtc{
Enclose the ribbons in a box.
}{
\spadpaste{rotate(vp,20,-60); showRegion(vp,"on")\bound{d11}\free{d10}}
}{
\epsffile[0 0 295 295]{../ps/ribbons2b.ps}
}

This process has become tedious!
If we had to add two or three more ribbons, we would have to
repeat the above steps several more times.
It is time to write an interpreter program to help us take care of
the details.
```

```
\endscroll
\autobuttons
\end{page}

\begin{patch}{ugIntProgDrawingPagePatch1}
\begin{paste}{ugIntProgDrawingPageFull1}{ugIntProgDrawingPageEmpty1}
\pastebutton{ugIntProgDrawingPageFull1}{\hidepaste}
\tab{5}\spadgraph{draw(x**2,x=-1..1,y=0..1)}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/ugintprogdrawingpage1.view/image}}{vi
\end{paste}\end{patch}

\begin{patch}{ugIntProgDrawingPageEmpty1}
\begin{paste}{ugIntProgDrawingPageEmpty1}{ugIntProgDrawingPagePatch1}
\pastebutton{ugIntProgDrawingPageEmpty1}{\showpaste}
\tab{5}\spadgraph{draw(x**2,x=-1..1,y=0..1)}
\end{paste}\end{patch}

\begin{patch}{ugIntProgDrawingPagePatch2}
\begin{paste}{ugIntProgDrawingPageFull2}{ugIntProgDrawingPageEmpty2}
\pastebutton{ugIntProgDrawingPageFull2}{\hidepaste}
\tab{5}\spadgraph{vp := draw(x**2,x=-1..1,y=0..1,var2Steps==1)\bound{d1 }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/ugintprogdrawingpage2.view/image}}{vi
\end{paste}\end{patch}

\begin{patch}{ugIntProgDrawingPageEmpty2}
\begin{paste}{ugIntProgDrawingPageEmpty2}{ugIntProgDrawingPagePatch2}
\pastebutton{ugIntProgDrawingPageEmpty2}{\showpaste}
\tab{5}\spadgraph{vp := draw(x**2,x=-1..1,y=0..1,var2Steps==1)\bound{d1 }}
\end{paste}\end{patch}

\begin{patch}{ugIntProgDrawingPagePatch3}
\begin{paste}{ugIntProgDrawingPageFull3}{ugIntProgDrawingPageEmpty3}
\pastebutton{ugIntProgDrawingPageFull3}{\hidepaste}
\tab{5}\spadcommand{rotate(vp, 0, -90)\bound{d3 }\free{d1 }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntProgDrawingPageEmpty3}
\begin{paste}{ugIntProgDrawingPageEmpty3}{ugIntProgDrawingPagePatch3}
\pastebutton{ugIntProgDrawingPageEmpty3}{\showpaste}
\tab{5}\spadcommand{rotate(vp, 0, -90)\bound{d3 }\free{d1 }}
\end{paste}\end{patch}

\begin{patch}{ugIntProgDrawingPagePatch4}
\begin{paste}{ugIntProgDrawingPageFull4}{ugIntProgDrawingPageEmpty4}
\pastebutton{ugIntProgDrawingPageFull4}{\hidepaste}
\tab{5}\spadcommand{sp := subspace(vp)\bound{d5 }\free{d1 }}
```

```
\indentrel{3}\begin{verbatim}
   (4)  3-Space with 1 component
                              Type: ThreeSpace DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntProgDrawingPageEmpty4}
\begin{paste}{ugIntProgDrawingPageEmpty4}{ugIntProgDrawingPagePatch4}
\pastebutton{ugIntProgDrawingPageEmpty4}{\showpaste}
\tab{5}\spadcommand{sp := subspace(vp)\bound{d5 }\free{d1 }}
\end{paste}\end{patch}

\begin{patch}{ugIntProgDrawingPagePatch5}
\begin{paste}{ugIntProgDrawingPageFull5}{ugIntProgDrawingPageEmpty5}
\pastebutton{ugIntProgDrawingPageFull5}{\hidepaste}
\tab{5}\spadgraph{vp := draw(x**3,x=-1..1,y=1..2,var2Steps==1, space==sp)\bound{d6 }\free{d5 }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/ugintprogdrawingpage5.view/image}}{viewalone\spa
\end{paste}\end{patch}

\begin{patch}{ugIntProgDrawingPageEmpty5}
\begin{paste}{ugIntProgDrawingPageEmpty5}{ugIntProgDrawingPagePatch5}
\pastebutton{ugIntProgDrawingPageEmpty5}{\showpaste}
\tab{5}\spadgraph{vp := draw(x**3,x=-1..1,y=1..2,var2Steps==1, space==sp)\bound{d6 }\free{d5 }}
\end{paste}\end{patch}

\begin{patch}{ugIntProgDrawingPagePatch6}
\begin{paste}{ugIntProgDrawingPageFull6}{ugIntProgDrawingPageEmpty6}
\pastebutton{ugIntProgDrawingPageFull6}{\hidepaste}
\tab{5}\spadcommand{drawStyle(vp,"shade");outlineRender(vp,"on")\bound{d10 }\free{d6 }}
\indentrel{3}\begin{verbatim}
                                              Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntProgDrawingPageEmpty6}
\begin{paste}{ugIntProgDrawingPageEmpty6}{ugIntProgDrawingPagePatch6}
\pastebutton{ugIntProgDrawingPageEmpty6}{\showpaste}
\tab{5}\spadcommand{drawStyle(vp,"shade");outlineRender(vp,"on")\bound{d10 }\free{d6 }}
\end{paste}\end{patch}

\begin{patch}{ugIntProgDrawingPagePatch7}
\begin{paste}{ugIntProgDrawingPageFull7}{ugIntProgDrawingPageEmpty7}
\pastebutton{ugIntProgDrawingPageFull7}{\hidepaste}
\tab{5}\spadcommand{rotate(vp,20,-60); showRegion(vp,"on")\bound{d11 }\free{d10 }}
\indentrel{3}\begin{verbatim}
                                              Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntProgDrawingPageEmpty7}
```

```
\begin{paste}{ugIntProgDrawingPageEmpty7}{ugIntProgDrawingPagePatch7}
\pastebutton{ugIntProgDrawingPageEmpty7}{\showpaste}
\tab{5}\spadcommand{rotate(vp,20,-60); showRegion(vp,"on")\bound{d11 }\free{d10 }}
\end{paste}\end{patch}
```

———-

A Ribbon Program

⇒ "notitle" (ugUserPage) 10 on page 1823
⇒ "notitle" (ugLangBlocksPage) 9 on page 1753

— **ug10.ht** —

```
\begin{page}{ugIntProgRibbonPage}{10.2. A Ribbon Program}
\beginscroll
%

The above approach creates a new viewport for each additional
ribbon.
A better approach is to build one object composed of all ribbons
before creating a viewport.
To do this, use \spadfun{makeObject} rather than \spadfun{draw}.
The operations have similar formats, but
\spadfun{draw} returns a viewport and
\spadfun{makeObject} returns a space object.

We now create a function \userfun{drawRibbons} of two arguments:
\spad{flist}, a list of formulas for the ribbons you want to draw,
and \spad{xrange}, the range over which you want them drawn.
Using this function, you can just say
\begin{verbatim}
drawRibbons([x**2, x**3], x=-1..1)
\end{verbatim}
to do all of the work required in the last section.
Here is the \userfun{drawRibbons} program.
Invoke your favorite editor and create a file called {\bf ribbon.input}
containing the following program.

\beginImportant

\noindent
{\tt 1.\ \ \ drawRibbons(flist,\ xrange)\ ==}\newline
{\tt 2.\ \ \ \ \ sp\ :=\ createThreeSpace()}\newline
{\tt 3.\ \ \ \ \ y0\ :=\ 0}\newline
{\tt 4.\ \ \ \ \ for\ f\ in\ flist\ repeat}\newline
{\tt 5.\ \ \ \ \ \ \ makeObject(f,\ xrange,\ y=y0..y0+1,\ }\newline
```

```
{\tt 6.\ \ \ \ \ \ \ \ \ \ \ space==sp,\ var2Steps\ ==\ 1)}\newline
{\tt 7.\ \ \ \ \ \ \ \ y0\ :=\ y0\ +\ 1}\newline
{\tt 8.\ \ \ \ \ \ vp\ :=\ makeViewport3D(sp,\ "Ribbons")}\newline
{\tt 9.\ \ \ \ \ \ drawStyle(vp,\ "shade")}\newline
{\tt 10.\ \ \ \ outlineRender(vp,\ "on")}\newline
{\tt 11.\ \ \ \ showRegion(vp,"on")}\newline
{\tt 12.\ \ \ \ n\ :=\ \#\ flist}\newline
{\tt 13.\ \ \ \ zoom(vp,n,1,n)}\newline
{\tt 14.\ \ \ \ rotate(vp,0,75)}\newline
{\tt 15.\ \ \ \ vp}\newline
\caption{The first \protect\pspadfun{drawRibbons} function.}
\label{fig-ribdraw1}
\endImportant
```

Here are some remarks on the syntax used in the \pspadfun{drawRibbons} function (consult \downlink{‘‘User-Defined Functions, Macros and Rules’’}{ugUserPage} in Chapter 6\ignore{ugUser} for more details). Unlike most other programming languages which use semicolons, parentheses, or {\it begin}--{\it end} brackets to delineate the structure of programs, the structure of an Axiom program is determined by indentation. The first line of the function definition always begins in column 1. All other lines of the function are indented with respect to the first line and form a \spadgloss{pile} (see \downlink{‘‘Blocks’’}{ugLangBlocksPage} in Section 5.2\ignore{ugLangBlocks}).

The definition of \userfun{drawRibbons} consists of a pile of expressions to be executed one after another. Each expression of the pile is indented at the same level. Lines 4-7 designate one single expression: since lines 5-7 are indented with respect to the others, these lines are treated as a continuation of line 4. Also since lines 5 and 7 have the same indentation level, these lines designate a pile within the outer pile.

The last line of a pile usually gives the value returned by the pile. Here it is also the value returned by the function. Axiom knows this is the last line of the function because it is the last line of the file. In other cases, a new expression beginning in column one signals the end of a function.

The line \spad{drawStyle(vp,"shade")} is given after the viewport has been created to select the draw style. We have also used the \spadfunFrom{zoom}{ThreeDimensionalViewport} option. Without the zoom, the viewport region would be scaled equally in all three coordinate directions.

Let’s try the function \userfun{drawRibbons}.
First you must read the file to give Axiom the function definition.

```
\xtc{
Read the input file.
}{
\spadpaste{)read ribbon \bound{s0}}
}
\psXtc{
Draw ribbons for \texht{$x, x^2,\dots, x^5$}{x, x**2,...,x**5}
for \texht{$-1 \leq x \leq 1$}{-1 <= x <= 1}
}{
\graphpaste{drawRibbons([x**i for i in 1..5],x=-1..1) \free{s0}}
}{
\epsffile[0 0 295 295]{../ps/ribbons5.ps}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugIntProgRibbonPagePatch1}
\begin{paste}{ugIntProgRibbonPageFull1}{ugIntProgRibbonPageEmpty1}
\pastebutton{ugIntProgRibbonPageFull1}{\hidepaste}
\tab{5}\spadcommand{)read ribbon\bound{s0 }}
\indentrel{3}\begin{verbatim}
                                              Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntProgRibbonPageEmpty1}
\begin{paste}{ugIntProgRibbonPageEmpty1}{ugIntProgRibbonPagePatch1}
\pastebutton{ugIntProgRibbonPageEmpty1}{\showpaste}
\tab{5}\spadcommand{)read ribbon\bound{s0 }}
\end{paste}\end{patch}

\begin{patch}{ugIntProgRibbonPagePatch2}
\begin{paste}{ugIntProgRibbonPageFull2}{ugIntProgRibbonPageEmpty2}
\pastebutton{ugIntProgRibbonPageFull2}{\hidepaste}
\tab{5}\spadgraph{drawRibbons([x**i for i in 1..5],x=-1..1)\free{s0 }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/ugintprogribbonpage2.view/image}}{vie
\end{paste}\end{patch}

\begin{patch}{ugIntProgRibbonPageEmpty2}
\begin{paste}{ugIntProgRibbonPageEmpty2}{ugIntProgRibbonPagePatch2}
\pastebutton{ugIntProgRibbonPageEmpty2}{\showpaste}
\tab{5}\spadgraph{drawRibbons([x**i for i in 1..5],x=-1..1)\free{s0 }}
\end{paste}\end{patch}
```

Coloring and Positioning Ribbons

— **ug10.ht** —

```
\begin{page}{ugIntProgColorPage}{10.3. Coloring and Positioning Ribbons}
\beginscroll
%

Before leaving the ribbon example, we  make two improvements.
Normally, the color given to each point in the space is a
function of its height within a bounding box.
The points at the bottom of the
box are red, those at the top are purple.

To change the normal coloring, you can give
an option \spad{colorFunction == {\it function}}.
When Axiom goes about displaying the data, it
determines the range of colors used for all points within the box.
Axiom then distributes these numbers uniformly over the number of hues.
Here we use the simple color function
\texht{$(x,y) \mapsto i$}{(x,y) +-> i} for the
\eth{\smath{i}} ribbon.

Also, we add an argument \spad{yrange} so you can give the range of
\spad{y} occupied by the ribbons.
For example, if the \spad{yrange} is given as
\spad{y=0..1} and there are \smath{5} ribbons to be displayed, each
ribbon would have width \smath{0.2} and would appear in the
range \texht{$0 \leq y \leq 1$}{\spad{0 <= y <= 1}}.

Refer to lines 4-9.
Line 4 assigns to \spad{yVar} the variable part of the
\spad{yrange} (after all, it need not be \spad{y}).
Suppose that \spad{yrange} is given as \spad{t = a..b} where \spad{a} and
\spad{b} have numerical values.
Then line 5 assigns the value of \spad{a} to the variable \spad{y0}.
Line 6 computes the width of the ribbon by dividing the difference of
\spad{a} and \spad{b} by the number, \spad{num}, of ribbons.
The result is assigned to the variable \spad{width}.
Note that in the for-loop in line 7, we are iterating in parallel; it is
not a nested loop.

\beginImportant

\noindent
{\tt 1.\ \ \ drawRibbons(flist,\ xrange,\ yrange)\ ==}\newline
{\tt 2.\ \ \ \ \ sp\ :=\ createThreeSpace()}\newline
{\tt 3.\ \ \ \ \ num\ :=\ \#\ flist}\newline
{\tt 4.\ \ \ \ \ yVar\ :=\ variable\ yrange}\newline
```

```
{\tt 5.\ \ \ \ \ y0:Float\ \ \ \ :=\ lo\ segment\ yrange}\newline
{\tt 6.\ \ \ \ \ width:Float\ :=\ (hi\ segment\ yrange\ -\ y0)/num}
\newline
{\tt 7.\ \ \ \ \ for\ f\ in\ flist\ for\ color\ in\ 1..num\ repeat}
\newline
{\tt 8.\ \ \ \ \ \ \ makeObject(f,\ xrange,\ yVar\ =\ y0..y0+width,}
\newline
{\tt 9.\ \ \ \ \ \ \ \ \ var2Steps\ ==\ 1,\ colorFunction\ ==\
(x,y)\ +->\ color,\ \_}\newline
{\tt 10.\ \ \ \ \ \ \ \ space\ ==\ sp)}\newline
{\tt 11.\ \ \ \ \ \ y0\ :=\ y0\ +\ width}\newline
{\tt 12.\ \ \ \ vp\ :=\ makeViewport3D(sp,\ "Ribbons")}\newline
{\tt 13.\ \ \ \ drawStyle(vp,\ "shade")}\newline
{\tt 14.\ \ \ \ outlineRender(vp,\ "on")}\newline
{\tt 15.\ \ \ \ showRegion(vp,\ "on")}\newline
{\tt 16.\ \ \ \ vp}\newline
\caption{The final \protect\pspadfun{drawRibbons} function.}
\label{fig-ribdraw2}
\endImportant


\endscroll
\autobuttons
\end{page}
```

———-

Points, Lines, and Curves

— ug10.ht —

```
\begin{page}{ugIntProgPLCPage}{10.4. Points, Lines, and Curves}
\beginscroll
%
What you have seen so far is a high-level program using the
graphics facility.
We now turn to the more basic notions of points, lines, and curves
in \threedim{} graphs.
These facilities use small floats (objects
of type \spadtype{DoubleFloat}) for data.
Let us first give names to the small float values \smath{0} and
\smath{1}.
\xtc{
The small float 0.
}{
\spadpaste{zero := 0.0@DFLOAT \bound{d1}}
}
```

```
\xtc{
The small float 1.
}{
\spadpaste{one  := 1.0@DFLOAT \bound{d2}}
}
The \spadSyntax{@} sign means ‘‘of the type.’’ Thus \spad{zero} is
\smath{0.0} of the type \spadtype{DoubleFloat}.
You can also say \spad{0.0::DFLOAT}.

Points can have four small float components: \smath{x, y, z}
coordinates and an optional color.  A ‘‘curve’’ is simply a list of
points connected by straight line segments.
\xtc{
Create the point \spad{origin} with color zero, that is, the lowest color
on the color map.
}{
\spadpaste{origin := point [zero,zero,zero,zero] \free{d1}\bound{d3}}
}
\xtc{
Create the point \spad{unit} with color zero.
}{
\spadpaste{unit := point [one,one,one,zero] \free{d1 d2}\bound{d4}}
}
\xtc{
Create the curve (well, here, a line) from
\spad{origin} to \spad{unit}.
}{
\spadpaste{line := [origin, unit] \free{d3 d4} \bound{d5}}
}

We make this line segment into an arrow by adding an arrowhead.
The arrowhead extends to,
say, \spad{p3} on the left, and to, say, \spad{p4} on the right.
To describe an arrow, you tell Axiom to draw the two curves
\spad{[p1, p2, p3]} and \spad{[p2, p4].}
We also decide through experimentation on
values for \spad{arrowScale}, the ratio of the size of
the arrowhead to the stem of the arrow, and \spad{arrowAngle},
the angle between the arrowhead and the arrow.

Invoke your favorite editor and create
an input file called {\bf arrows.input}.
This input file first defines the values of
%\spad{origin},\spad{unit},
\spad{arrowAngle} and \spad{arrowScale}, then
defines the function \userfun{makeArrow}\texht{$(p_1, p_2)$}{(p1, p2)} to
draw an arrow from point \texht{$p_1$}{p1} to \texht{$p_2$}{p2}.

\beginImportant
```

```
\noindent
{\tt 1.\ \ \ arrowAngle\ :=\ \%pi-\%pi/10.0@DFLOAT}\newline
{\tt 2.\ \ \ arrowScale\ :=\ 0.2@DFLOAT}\newline
{\tt 3.\ \ \ }\newline
{\tt 4.\ \ \ makeArrow(p1,\ p2)\ ==}\newline
{\tt 5.\ \ \ \ \ delta\ :=\ p2\ -\ p1}\newline
{\tt 6.\ \ \ \ \ len\ :=\ arrowScale\ *\ length\ delta}\newline
{\tt 7.\ \ \ \ \ theta\ :=\ atan(delta.1,\ delta.2)}\newline
{\tt 8.\ \ \ \ \ c1\ :=\ len*cos(theta\ +\ arrowAngle)}\newline
{\tt 9.\ \ \ \ \ s1\ :=\ len*sin(theta\ +\ arrowAngle)}\newline
{\tt 10.\ \ \ \ c2\ :=\ len*cos(theta\ -\ arrowAngle)}\newline
{\tt 11.\ \ \ \ s2\ :=\ len*sin(theta\ -\ arrowAngle)}\newline
{\tt 12.\ \ \ \ z\ \ :=\ p2.3*(1\ -\ arrowScale)}\newline
{\tt 13.\ \ \ \ p3\ :=\ point\ [p2.1\ +\ c1,\ p2.2\ +\ s1,\ z,\ p2.4]}
\newline
{\tt 14.\ \ \ \ p4\ :=\ point\ [p2.1\ +\ c2,\ p2.2\ +\ s2,\ z,\ p2.4]}
\newline
{\tt 15.\ \ \ \ [[p1,\ p2,\ p3],\ [p2,\ p4]]}\newline
\endImportant

Read the file and then create
an arrow from the point \spad{origin} to the point \spad{unit}.
\xtc{
Read the input file defining \userfun{makeArrow}.
}{
\spadpaste{)read arrows\bound{v1}}
}
\xtc{
Construct the arrow (a list of two curves).
}{
\spadpaste{arrow := makeArrow(origin,unit)\bound{v2}\free{v1 d3 d4}}
}
\xtc{
Create an empty object \spad{sp} of type \spad{ThreeSpace}.
}{
\spadpaste{sp := createThreeSpace()\bound{c1}}
}
\xtc{
Add each curve of the arrow to the space \spad{sp}.
}{
\spadpaste{for a in arrow repeat sp := curve(sp,a)
\bound{v3}\free{v2}\free{c1}}
}
\psXtc{
Create a \threedim{} viewport containing that space.
}{
\graphpaste{vp := makeViewport3D(sp,"Arrow")\bound{v4}\free{v3}}
}{
\epsffile[0 0 295 295]{../ps/arrow.ps}
}
```

```
\psXtc{
Here is a better viewing angle.
}{
\spadpaste{rotate(vp,200,-60)\bound{v5}\free{v4}}
}{
\epsffile[0 0 295 295]{../ps/arrowr.ps}
}


\endscroll
\autobuttons
\end{page}

\begin{patch}{ugIntProgPLCPagePatch1}
\begin{paste}{ugIntProgPLCPageFull1}{ugIntProgPLCPageEmpty1}
\pastebutton{ugIntProgPLCPageFull1}{\hidepaste}
\tab{5}\spadcommand{zero := 0.0@DFLOAT\bound{d1 }}
\indentrel{3}\begin{verbatim}
   (1)  0.0
                                     Type: DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntProgPLCPageEmpty1}
\begin{paste}{ugIntProgPLCPageEmpty1}{ugIntProgPLCPagePatch1}
\pastebutton{ugIntProgPLCPageEmpty1}{\showpaste}
\tab{5}\spadcommand{zero := 0.0@DFLOAT\bound{d1 }}
\end{paste}\end{patch}

\begin{patch}{ugIntProgPLCPagePatch2}
\begin{paste}{ugIntProgPLCPageFull2}{ugIntProgPLCPageEmpty2}
\pastebutton{ugIntProgPLCPageFull2}{\hidepaste}
\tab{5}\spadcommand{one := 1.0@DFLOAT\bound{d2 }}
\indentrel{3}\begin{verbatim}
   (2)  1.0
                                     Type: DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntProgPLCPageEmpty2}
\begin{paste}{ugIntProgPLCPageEmpty2}{ugIntProgPLCPagePatch2}
\pastebutton{ugIntProgPLCPageEmpty2}{\showpaste}
\tab{5}\spadcommand{one := 1.0@DFLOAT\bound{d2 }}
\end{paste}\end{patch}

\begin{patch}{ugIntProgPLCPagePatch3}
\begin{paste}{ugIntProgPLCPageFull3}{ugIntProgPLCPageEmpty3}
\pastebutton{ugIntProgPLCPageFull3}{\hidepaste}
\tab{5}\spadcommand{origin := point [zero,zero,zero,zero]\free{d1 }\bound{d3 }}
\indentrel{3}\begin{verbatim}
```

```
   (3)  [0.0,0.0,0.0,0.0]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntProgPLCPageEmpty3}
\begin{paste}{ugIntProgPLCPageEmpty3}{ugIntProgPLCPagePatch3}
\pastebutton{ugIntProgPLCPageEmpty3}{\showpaste}
\tab{5}\spadcommand{origin := point [zero,zero,zero,zero]\free{d1 }\bound{d3 }}
\end{paste}\end{patch}

\begin{patch}{ugIntProgPLCPagePatch4}
\begin{paste}{ugIntProgPLCPageFull4}{ugIntProgPLCPageEmpty4}
\pastebutton{ugIntProgPLCPageFull4}{\hidepaste}
\tab{5}\spadcommand{unit := point [one,one,one,zero]\free{d1 d2 }\bound{d4 }}
\indentrel{3}\begin{verbatim}
   (4)  [1.0,1.0,1.0,0.0]
                                Type: Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntProgPLCPageEmpty4}
\begin{paste}{ugIntProgPLCPageEmpty4}{ugIntProgPLCPagePatch4}
\pastebutton{ugIntProgPLCPageEmpty4}{\showpaste}
\tab{5}\spadcommand{unit := point [one,one,one,zero]\free{d1 d2 }\bound{d4 }}
\end{paste}\end{patch}

\begin{patch}{ugIntProgPLCPagePatch5}
\begin{paste}{ugIntProgPLCPageFull5}{ugIntProgPLCPageEmpty5}
\pastebutton{ugIntProgPLCPageFull5}{\hidepaste}
\tab{5}\spadcommand{line := [origin, unit]\free{d3 d4 }\bound{d5 }}
\indentrel{3}\begin{verbatim}
   (5)  [[0.0,0.0,0.0,0.0],[1.0,1.0,1.0,0.0]]
                           Type: List Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntProgPLCPageEmpty5}
\begin{paste}{ugIntProgPLCPageEmpty5}{ugIntProgPLCPagePatch5}
\pastebutton{ugIntProgPLCPageEmpty5}{\showpaste}
\tab{5}\spadcommand{line := [origin, unit]\free{d3 d4 }\bound{d5 }}
\end{paste}\end{patch}

\begin{patch}{ugIntProgPLCPagePatch6}
\begin{paste}{ugIntProgPLCPageFull6}{ugIntProgPLCPageEmpty6}
\pastebutton{ugIntProgPLCPageFull6}{\hidepaste}
\tab{5}\spadcommand{)read arrows\bound{v1 }}
\indentrel{3}\begin{verbatim}
   (6)  2.8274333882308138
                                      Type: DoubleFloat
```

```
   (7)  0.20000000000000001
                                  Type: DoubleFloat
                                         Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntProgPLCPageEmpty6}
\begin{paste}{ugIntProgPLCPageEmpty6}{ugIntProgPLCPagePatch6}
\pastebutton{ugIntProgPLCPageEmpty6}{\showpaste}
\tab{5}\spadcommand{)read arrows\bound{v1 }}
\end{paste}\end{patch}

\begin{patch}{ugIntProgPLCPagePatch7}
\begin{paste}{ugIntProgPLCPageFull7}{ugIntProgPLCPageEmpty7}
\pastebutton{ugIntProgPLCPageFull7}{\hidepaste}
\tab{5}\spadcommand{arrow := makeArrow(origin,unit)\bound{v2 }\free{v1 d3 d4 }}
\indentrel{3}\begin{verbatim}
   (9)
   [
     [[0.0,0.0,0.0,0.0], [1.0,1.0,1.0,0.0],

       [0.69134628604607973, 0.842733077659504,
        0.80000000000000004, 0.0]
       ]
     ,

     [[1.0,1.0,1.0,0.0],

       [0.842733077659504, 0.69134628604607973,
        0.80000000000000004, 0.0]
       ]
     ]
                      Type: List List Point DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntProgPLCPageEmpty7}
\begin{paste}{ugIntProgPLCPageEmpty7}{ugIntProgPLCPagePatch7}
\pastebutton{ugIntProgPLCPageEmpty7}{\showpaste}
\tab{5}\spadcommand{arrow := makeArrow(origin,unit)\bound{v2 }\free{v1 d3 d4 }}
\end{paste}\end{patch}

\begin{patch}{ugIntProgPLCPagePatch8}
\begin{paste}{ugIntProgPLCPageFull8}{ugIntProgPLCPageEmpty8}
\pastebutton{ugIntProgPLCPageFull8}{\hidepaste}
\tab{5}\spadcommand{sp := createThreeSpace()\bound{c1 }}
\indentrel{3}\begin{verbatim}
   (10)  3-Space with 0 components
                           Type: ThreeSpace DoubleFloat
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntProgPLCPageEmpty8}
\begin{paste}{ugIntProgPLCPageEmpty8}{ugIntProgPLCPagePatch8}
\pastebutton{ugIntProgPLCPageEmpty8}{\showpaste}
\tab{5}\spadcommand{sp := createThreeSpace()\bound{c1 }}
\end{paste}\end{patch}

\begin{patch}{ugIntProgPLCPagePatch9}
\begin{paste}{ugIntProgPLCPageFull9}{ugIntProgPLCPageEmpty9}
\pastebutton{ugIntProgPLCPageFull9}{\hidepaste}
\tab{5}\spadcommand{for a in arrow repeat sp := curve(sp,a)\bound{v3 }\free{v2 }\free{c1 }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntProgPLCPageEmpty9}
\begin{paste}{ugIntProgPLCPageEmpty9}{ugIntProgPLCPagePatch9}
\pastebutton{ugIntProgPLCPageEmpty9}{\showpaste}
\tab{5}\spadcommand{for a in arrow repeat sp := curve(sp,a)\bound{v3 }\free{v2 }\free{c1 }}
\end{paste}\end{patch}

\begin{patch}{ugIntProgPLCPagePatch10}
\begin{paste}{ugIntProgPLCPageFull10}{ugIntProgPLCPageEmpty10}
\pastebutton{ugIntProgPLCPageFull10}{\hidepaste}
\tab{5}\spadgraph{vp := makeViewport3D(sp,"Arrow")\bound{v4 }\free{v3 }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/ugintprogplcpage10.view/image}}{viewa
\end{paste}\end{patch}

\begin{patch}{ugIntProgPLCPageEmpty10}
\begin{paste}{ugIntProgPLCPageEmpty10}{ugIntProgPLCPagePatch10}
\pastebutton{ugIntProgPLCPageEmpty10}{\showpaste}
\tab{5}\spadgraph{vp := makeViewport3D(sp,"Arrow")\bound{v4 }\free{v3 }}
\end{paste}\end{patch}

\begin{patch}{ugIntProgPLCPagePatch11}
\begin{paste}{ugIntProgPLCPageFull11}{ugIntProgPLCPageEmpty11}
\pastebutton{ugIntProgPLCPageFull11}{\hidepaste}
\tab{5}\spadcommand{rotate(vp,200,-60)\bound{v5 }\free{v4 }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntProgPLCPageEmpty11}
\begin{paste}{ugIntProgPLCPageEmpty11}{ugIntProgPLCPagePatch11}
\pastebutton{ugIntProgPLCPageEmpty11}{\showpaste}
\tab{5}\spadcommand{rotate(vp,200,-60)\bound{v5 }\free{v4 }}
\end{paste}\end{patch}
```

A Bouquet of Arrows

— ug10.ht —

```
\begin{page}{ugIntProgColorArrPage}{10.5. A Bouquet of Arrows}
\beginscroll

Let's draw a ``bouquet'' of arrows.
Each arrow is identical. The arrowheads are
uniformly placed on a circle parallel to the \smath{xy}-plane.
Thus the position of each arrow differs only
by the angle \texht{$\theta$}{theta},
\texht{$0 \leq \theta < 2\pi$}{\spad{0 <= theta < 2*\%pi}},
between the arrow and
the \smath{x}-axis on the \smath{xy}-plane.

Our bouquet is rather special: each arrow has a different
color (which won't be evident here, unfortunately).
This is arranged by letting the color of each successive arrow be
denoted by \texht{$\theta$}{theta}.
In this way, the color of arrows ranges from red to green to violet.
Here is a program to draw a bouquet of \smath{n} arrows.

\beginImportant

\noindent
{\tt 1.\ \ \ drawBouquet(n,title)\ ==}\newline
{\tt 2.\ \ \ \ \ angle\ :=\ 0.0@DFLOAT}\newline
{\tt 3.\ \ \ \ \ sp\ :=\ createThreeSpace()}\newline
{\tt 4.\ \ \ \ \ for\ i\ in\ 0..n-1\ repeat}\newline
{\tt 5.\ \ \ \ \ \ \ start\ :=\ point\
[0.0@DFLOAT,0.0@DFLOAT,0.0@DFLOAT,angle]\ }\newline
{\tt 6.\ \ \ \ \ \ \ end\ \ \ :=\ point\
[cos\ angle,\ sin\ angle,\ 1.0@DFLOAT,\ angle]}\newline
{\tt 7.\ \ \ \ \ \ \ arrow\ :=\ makeArrow(start,end)}\newline
{\tt 8.\ \ \ \ \ \ \ for\ a\ in\ makeArrow(start,end)\ repeat\ }\newline
{\tt 9.\ \ \ \ \ \ \ \ \ curve(sp,a)}\newline
{\tt 10.\ \ \ \ \ \ angle\ :=\ angle\ +\ 2*\%pi/n}\newline
{\tt 11.\ \ \ \ makeViewport3D(sp,title)}\newline
\endImportant

\xtc{
Read the input file.
}{
```

```
\spadpaste{)read bouquet\bound{b1}}
}
\psXtc{
A bouquet of a dozen arrows.
}{
\graphpaste{drawBouquet(12,"A Dozen Arrows")\free{b1}}
}{
\epsffile[0 0 295 295]{../ps/bouquet.ps}
}
\

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugIntProgColorArrPagePatch1}
\begin{paste}{ugIntProgColorArrPageFull1}{ugIntProgColorArrPageEmpty1}
\pastebutton{ugIntProgColorArrPageFull1}{\hidepaste}
\tab{5}\spadcommand{)read bouquet\bound{b1 }}
\indentrel{3}\begin{verbatim}
   (1)  0.20000000000000001
                                      Type: DoubleFloat
   (2)  2.8274333882308138
                                      Type: DoubleFloat
                                             Type: Void
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntProgColorArrPageEmpty1}
\begin{paste}{ugIntProgColorArrPageEmpty1}{ugIntProgColorArrPagePatch1}
\pastebutton{ugIntProgColorArrPageEmpty1}{\showpaste}
\tab{5}\spadcommand{)read bouquet\bound{b1 }}
\end{paste}\end{patch}

\begin{patch}{ugIntProgColorArrPagePatch2}
\begin{paste}{ugIntProgColorArrPageFull2}{ugIntProgColorArrPageEmpty2}
\pastebutton{ugIntProgColorArrPageFull2}{\hidepaste}
\tab{5}\spadgraph{drawBouquet(12,"A Dozen Arrows")\free{b1 }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/ugintprogcolorarrpage2.view/image}}{v
\end{paste}\end{patch}

\begin{patch}{ugIntProgColorArrPageEmpty2}
\begin{paste}{ugIntProgColorArrPageEmpty2}{ugIntProgColorArrPagePatch2}
\pastebutton{ugIntProgColorArrPageEmpty2}{\showpaste}
\tab{5}\spadgraph{drawBouquet(12,"A Dozen Arrows")\free{b1 }}
\end{paste}\end{patch}
```

Drawing Complex Vector Fields

— ug10.ht —

```
\begin{page}{ugIntProgVecFieldsPage}{10.6. Drawing Complex Vector Fields}
\beginscroll

We now put our arrows to good use drawing complex vector fields.
These vector fields give a representation of complex-valued
functions of complex variables.
Consider a Cartesian coordinate grid of points \smath{(x, y)} in
the plane, and some complex-valued function \smath{f} defined on
this grid.
At every point on this grid, compute the value of \texht{$f(x +
iy)$}{f(x + y*\%i)} and call it \smath{z}.
Since \smath{z} has both a real and imaginary value for a given
\smath{(x,y)} grid point, there are four dimensions to plot.
What do we do?
We represent the values of \smath{z} by arrows planted at each
grid point.
Each arrow represents the value of \smath{z} in polar coordinates
\texht{$(r,\theta)$}{(r, theta)}.
The length of the arrow is proportional to \smath{r}.
Its direction is given by \texht{$\theta$}{theta}.

The code for drawing vector fields is in the file {\bf vectors.input}.
We discuss its contents from top to bottom.

Before showing you the code, we have two small
matters to take care of.
First, what if the function has large spikes, say, ones that go off
to infinity?
We define a variable \spad{clipValue} for this purpose. When
\spad{r} exceeds the value of \spad{clipValue}, then the value of
\spad{clipValue} is used instead of that for \spad{r}.
For convenience, we define a function \spad{clipFun(x)} which uses
\spad{clipValue} to ``clip'' the value of \spad{x}.

%
\beginImportant

\noindent
{\tt 1.\ \ \ clipValue\ :\ DFLOAT\ :=\ 6}\newline
{\tt 2.\ \ \ clipFun(x)\ ==\ min(max(x,-clipValue),clipValue)}\newline
\endImportant

Notice that we identify \spad{clipValue} as a small float but do
not declare the type of the function \userfun{clipFun}.
As it turns out, \userfun{clipFun} is called with a
```

```
small float value.
This declaration ensures that \userfun{clipFun} never does a
conversion when it is called.

The second matter concerns the possible ‘‘poles’’ of a
function, the actual points where the spikes have infinite
values.
Axiom uses normal \spadtype{DoubleFloat} arithmetic  which
does not directly handle infinite values.
If your function has poles, you must adjust your step size to
avoid landing directly on them (Axiom calls \spadfun{error}
when asked to divide a value by \axiom{0}, for example).

We set the variables \spad{realSteps} and \spad{imagSteps} to
hold the number of steps taken in the real and imaginary
directions, respectively.
Most examples will have ranges centered around the origin.
To avoid a pole at the origin, the number of points is taken
to be odd.

\beginImportant

\noindent
{\tt 1.\ \ \ realSteps:\ INT\ :=\ 25}\newline
{\tt 2.\ \ \ imagSteps:\ INT\ :=\ 25}\newline
{\tt 3.\ \ \ )read\ arrows}\newline
\endImportant

Now define the function \userfun{drawComplexVectorField} to draw the
arrows.  It is good practice to declare the type of the main function
in the file.  This one declaration is usually sufficient to ensure
that other lower-level functions are compiled with the correct types.

\beginImportant

\noindent
{\tt 4.\ \ \ C\ :=\ Complex\ DoubleFloat}\newline
{\tt 5.\ \ \ S\ :=\ Segment\ DoubleFloat}\newline
{\tt 6.\ \ \ drawComplexVectorField:\ (C\ ->\ C,\ S,\ S)\ ->\ VIEW3D}\newline
\endImportant

The first argument is a function mapping complex small floats into
complex small floats.
The second and third arguments give the range of real and
imaginary values as segments like \spad{a..b}.
The result is a \threedim{} viewport.
Here is the full function definition:

\beginImportant
```

```
\noindent
{\tt 7.\ \ \ drawComplexVectorField(f,\ realRange,imagRange)\ ==}\newline
{\tt 8.\ \ \ \ \ delReal\ :=\ (hi(realRange)-lo(realRange))/realSteps}
\newline
{\tt 9.\ \ \ \ \ delImag\ :=\ (hi(imagRange)-lo(imagRange))/imagSteps}
\newline
{\tt 10.\ \ \ \ sp\ :=\ createThreeSpace()}\newline
{\tt 11.\ \ \ \ real\ :=\ lo(realRange)}\newline
{\tt 12.\ \ \ \ for\ i\ in\ 1..realSteps+1\ repeat}\newline
{\tt 13.\ \ \ \ \ \ imag\ :=\ lo(imagRange)}\newline
{\tt 14.\ \ \ \ \ \ for\ j\ in\ 1..imagSteps+1\ repeat}\newline
{\tt 15.\ \ \ \ \ \ \ \ z\ :=\ f\ complex(real,imag)}\newline
{\tt 16.\ \ \ \ \ \ \ \ arg\ :=\ argument\ z}\newline
{\tt 17.\ \ \ \ \ \ \ \ len\ :=\ clipFun\ sqrt\ norm\ z}\newline
{\tt 18.\ \ \ \ \ \ \ \ p1\ :=\ \ point\
[real,\ imag,\ 0.0@DFLOAT,\ arg]}\newline
{\tt 19.\ \ \ \ \ \ \ \ scaleLen\ :=\ delReal\ *\ len}\newline
{\tt 20.\ \ \ \ \ \ \ \ p2\ :=\ point\
[p1.1\ +\ scaleLen*cos(arg),}\newline
{\tt 21.\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ p1.2\ +\
scaleLen*sin(arg),0.0@DFLOAT,\ arg]}\newline
{\tt 22.\ \ \ \ \ \ \ \ arrow\ :=\ makeArrow(p1,\ p2)}\newline
{\tt 23.\ \ \ \ \ \ \ \ for\ a\ in\ arrow\ repeat\ curve(sp,\ a)}\newline
{\tt 24.\ \ \ \ \ \ \ \ imag\ :=\ imag\ +\ delImag}\newline
{\tt 25.\ \ \ \ \ \ real\ :=\ real\ +\ delReal}\newline
{\tt 26.\ \ \ \ makeViewport3D(sp,\ "Complex\ Vector\ Field")}\newline
\endImportant

As a first example, let us draw \spad{f(z) == sin(z)}.  There is no
need to create a user function: just pass the
\spadfunFrom{sin}{Complex DoubleFloat} from
\spadtype{Complex DoubleFloat}.
\xtc{
Read the file.
}{
\spadpaste{)read vectors \bound{readVI}}
}
\psXtc{
Draw the complex vector field of \spad{sin(x)}.
}{
\graphpaste{drawComplexVectorField(sin,-2..2,-2..2) \free{readVI}}
}{
\epsffile[0 0 295 295]{../ps/vectorsin.ps}
}
\

\endscroll
\autobuttons
\end{page}
```

```
\begin{patch}{ugIntProgVecFieldsPagePatch1}
\begin{paste}{ugIntProgVecFieldsPageFull1}{ugIntProgVecFieldsPageEmpty1}
\pastebutton{ugIntProgVecFieldsPageFull1}{\hidepaste}
\tab{5}\spadcommand{)read vectors\bound{readVI }}
\indentrel{3}\begin{verbatim}
   (1)  2.8274333882308138
                                      Type: DoubleFloat
   (2)  0.20000000000000001
                                      Type: DoubleFloat
                                             Type: Void
   (4)  6.0
                                      Type: DoubleFloat
                                             Type: Void
   (6)  25
                                          Type: Integer
   (7)  25
                                          Type: Integer
   (8)  Complex DoubleFloat
                                           Type: Domain
   (9)  Segment DoubleFloat
                                           Type: Domain
                                             Type: Void
                                             Type: Void
                                             Type: Void
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntProgVecFieldsPageEmpty1}
\begin{paste}{ugIntProgVecFieldsPageEmpty1}{ugIntProgVecFieldsPagePatch1}
\pastebutton{ugIntProgVecFieldsPageEmpty1}{\showpaste}
\tab{5}\spadcommand{)read vectors\bound{readVI }}
\end{paste}\end{patch}

\begin{patch}{ugIntProgVecFieldsPagePatch2}
\begin{paste}{ugIntProgVecFieldsPageFull2}{ugIntProgVecFieldsPageEmpty2}
\pastebutton{ugIntProgVecFieldsPageFull2}{\hidepaste}
\tab{5}\spadgraph{drawComplexVectorField(sin,-2..2,-2..2)\free{readVI }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/ugintprogvecfieldspage2.view/image}}{
\end{paste}\end{patch}

\begin{patch}{ugIntProgVecFieldsPageEmpty2}
\begin{paste}{ugIntProgVecFieldsPageEmpty2}{ugIntProgVecFieldsPagePatch2}
\pastebutton{ugIntProgVecFieldsPageEmpty2}{\showpaste}
\tab{5}\spadgraph{drawComplexVectorField(sin,-2..2,-2..2)\free{readVI }}
\end{paste}\end{patch}
```

Drawing Complex Functions

⇒ "notitle" (ugGraphPage) 11 on page 1967

— **ug10.ht** —

```
\begin{page}{ugIntProgCompFunsPage}{10.7. Drawing Complex Functions}
\beginscroll

Here is another way to graph a complex function of complex arguments.
For each complex value \smath{z}, compute \smath{f(z)}, again
expressing the value in polar coordinates \smath{(r,\theta{})}.  We
draw the complex valued function, again considering the
\smath{(x,y)}-plane as the complex plane, using \smath{r} as the
height (or \smath{z}-coordinate) and \smath{\theta} as the color.
This is a standard plot---we learned how to do this in
\downlink{``Graphics''}{ugGraphPage} in Chapter
7\ignore{ugGraph}---but here we write a new program to
illustrate the creation of polygon meshes, or grids.

Call this function \userfun{drawComplex}.
It displays the points using the ``mesh'' of points.
The function definition is in three parts.

\beginImportant

\noindent
{\tt 1.\ \ \ drawComplex:\ (C\ ->\ C,\ S,\ S)\ ->\ VIEW3D}\newline
{\tt 2.\ \ \ drawComplex(f,\ realRange,\ imagRange)\ ==}\newline
{\tt 3.\ \ \ \ \ delReal\ :=\ (hi(realRange)-lo(realRange))/realSteps}
\newline
{\tt 4.\ \ \ \ \ delImag\ :=\ (hi(imagRange)-lo(imagRange))/imagSteps}
\newline
{\tt 5.\ \ \ \ \ llp:List\ List\ Point\ DFLOAT\ :=\ []}\newline
\endImportant

Variables \spad{delReal} and \spad{delImag} give the step sizes along
the real and imaginary directions as computed by the values of the
global variables \spad{realSteps} and \spad{imagSteps}.  The mesh is
represented by a list of lists of points \spad{llp}, initially empty.
Now \spad{[ ]} alone is ambiguous, so to set this initial value you
have to tell Axiom what type of empty list it is.  Next comes the loop
which builds \spad{llp}.

\beginImportant

\noindent
{\tt 1.\ \ \ \ \ real\ :=\ lo(realRange)}\newline
{\tt 2.\ \ \ \ \ for\ i\ in\ 1..realSteps+1\ repeat}\newline
{\tt 3.\ \ \ \ \ \ \ imag\ :=\ lo(imagRange)}\newline
```

```
{\tt 4.\ \ \ \ \ \ \ lp\ :=\ []\$(List\ Point\ DFLOAT)}\newline
{\tt 5.\ \ \ \ \ \ \ for\ j\ in\ 1..imagSteps+1\ repeat}\newline
{\tt 6.\ \ \ \ \ \ \ \ \ z\ :=\ f\ complex(real,imag)}\newline
{\tt 7.\ \ \ \ \ \ \ \ \ pt\ :=\ point\
[real,imag,\ clipFun\ sqrt\ norm\ z,\ }\newline
{\tt 8.\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ argument\ z]}
\newline
{\tt 9.\ \ \ \ \ \ \ \ \ lp\ :=\ cons(pt,lp)}\newline
{\tt 10.\ \ \ \ \ \ \ \ imag\ :=\ imag\ +\ delImag}\newline
{\tt 11.\ \ \ \ \ \ real\ :=\ real\ +\ delReal}\newline
{\tt 12.\ \ \ \ \ \ llp\ :=\ cons(lp,\ llp)}\newline
\endImportant

The code consists of both an inner and outer loop.  Each pass through
the inner loop adds one list \spad{lp} of points to the list of lists
of points \spad{llp}.  The elements of \spad{lp} are collected in
reverse order.

\beginImportant

\noindent
{\tt 13.\ \ \ \ makeViewport3D(mesh(llp),\ "Complex\ Function")}\newline
\endImportant

The operation \spadfun{mesh} then creates an object of type
\spadtype{ThreeSpace(DoubleFloat)} from the list of lists of points.
This is then passed to \spadfun{makeViewport3D} to display the
image.

Now add this function directly to your {\bf vectors.input} file and
re-read the file using \spad{)read vectors}.  We try
\userfun{drawComplex} using a user-defined function \spad{f}.

\xtc{
Read the file.
}{
\spadpaste{)read vectors \bound{readVI}}
}
\xtc{
This one has a pole at \smath{z=0}.
}{
\spadpaste{f(z) == exp(1/z)\bound{e1}}
}
\psXtc{
Draw it with an odd number of steps to avoid the pole.
}{
\graphpaste{drawComplex(f,-2..2,-2..2)\free{e1 readVI}}
}{
\epsffile[0 0 295 295]{../ps/complexexp.ps}
}
```

```
\

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugIntProgCompFunsPagePatch1}
\begin{paste}{ugIntProgCompFunsPageFull1}{ugIntProgCompFunsPageEmpty1}
\pastebutton{ugIntProgCompFunsPageFull1}{\hidepaste}
\tab{5}\spadcommand{)read vectors\bound{readVI }}
\indentrel{3}\begin{verbatim}
   (1)  2.8274333882308138
                                      Type: DoubleFloat
   (2)  0.20000000000000001
                                      Type: DoubleFloat
                                             Type: Void
   (4)  6.0
                                      Type: DoubleFloat
                                             Type: Void
   (6)  25
                                          Type: Integer
   (7)  25
                                          Type: Integer
   (8)  Complex DoubleFloat
                                           Type: Domain
   (9)  Segment DoubleFloat
                                           Type: Domain
                                             Type: Void
                                             Type: Void
                                             Type: Void
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntProgCompFunsPageEmpty1}
\begin{paste}{ugIntProgCompFunsPageEmpty1}{ugIntProgCompFunsPagePatch1}
\pastebutton{ugIntProgCompFunsPageEmpty1}{\showpaste}
\tab{5}\spadcommand{)read vectors\bound{readVI }}
\end{paste}\end{patch}

\begin{patch}{ugIntProgCompFunsPagePatch2}
\begin{paste}{ugIntProgCompFunsPageFull2}{ugIntProgCompFunsPageEmpty2}
\pastebutton{ugIntProgCompFunsPageFull2}{\hidepaste}
\tab{5}\spadcommand{f(z) == exp(1/z)\bound{e1 }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntProgCompFunsPageEmpty2}
```

```
\begin{paste}{ugIntProgCompFunsPageEmpty2}{ugIntProgCompFunsPagePatch2}
\pastebutton{ugIntProgCompFunsPageEmpty2}{\showpaste}
\tab{5}\spadcommand{f(z) == exp(1/z)\bound{e1 }}
\end{paste}\end{patch}

\begin{patch}{ugIntProgCompFunsPagePatch3}
\begin{paste}{ugIntProgCompFunsPageFull3}{ugIntProgCompFunsPageEmpty3}
\pastebutton{ugIntProgCompFunsPageFull3}{\hidepaste}
\tab{5}\spadgraph{drawComplex(f,-2..2,-2..2)\free{e1 readVI }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/ugintprogcompfunspage3.view/image}}{v
\end{paste}\end{patch}

\begin{patch}{ugIntProgCompFunsPageEmpty3}
\begin{paste}{ugIntProgCompFunsPageEmpty3}{ugIntProgCompFunsPagePatch3}
\pastebutton{ugIntProgCompFunsPageEmpty3}{\showpaste}
\tab{5}\spadgraph{drawComplex(f,-2..2,-2..2)\free{e1 readVI }}
\end{paste}\end{patch}
```

———-

Functions Producing Functions

⇒ "notitle" (ugUserMakePage) 10 on page 1889

— **ug10.ht** —

```
\begin{page}{ugIntProgFunctionsPage}{10.8. Functions Producing Functions}
\beginscroll

In \downlink{``Making Functions from Objects''}{ugUserMakePage} in Section
6.14\ignore{ugUserMake}, you learned how to use the
operation \spadfun{function} to create a function from symbolic
formulas.  Here we introduce a similar operation which not only
creates functions, but functions from functions.

The facility we need is provided by the package
\spadtype{MakeUnaryCompiledFunction(E,S,T)}.  This package produces a
unary (one-argument) compiled function from some symbolic data
generated by a previous computation.\footnote{%
\spadtype{MakeBinaryCompiledFunction} is available for binary
functions.}  The \spad{E} tells where the symbolic data comes from;
the \spad{S} and \spad{T} give Axiom the source and target type of the
function, respectively.  The compiled function produced has type
\spadsig{\spad{S}}{\spad{T}}.  To produce a compiled function with
definition \spad{p(x) == expr}, call \spad{compiledFunction(expr, x)}
from this package.  The function you get has no name.  You must to
assign the function to the variable \spad{p} to give it that name.  %
\xtc{ Do some computation.  }{ \spadpaste{(x+1/3)**5\bound{p1}} }
```

```
\xtc{ Convert this to an anonymous function of \spad{x}.  Assign it to
the variable \spad{p} to give the function a name.  }{ \spadpaste{p :=
compiledFunction(\%,x)\$MakeUnaryCompiledFunction(POLY FRAC
INT,DFLOAT,DFLOAT)\bound{p2}\free{p1}} } \xtc{ Apply the function.  }{
\spadpaste{p(sin(1.3))\bound{p3}\free{p2}} }

For a more sophisticated application, read on.

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugIntProgFunctionsPagePatch1}
\begin{paste}{ugIntProgFunctionsPageFull1}{ugIntProgFunctionsPageEmpty1}
\pastebutton{ugIntProgFunctionsPageFull1}{\hidepaste}
\tab{5}\spadcommand{(x+1/3)**5\bound{p1 }}
\indentrel{3}\begin{verbatim}
         5   5  4   10  3   10  2    5      1
   (1)  x  +  x  +  x  +  x  +  x +
             3       9      27      81     243
                      Type: Polynomial Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntProgFunctionsPageEmpty1}
\begin{paste}{ugIntProgFunctionsPageEmpty1}{ugIntProgFunctionsPagePatch1}
\pastebutton{ugIntProgFunctionsPageEmpty1}{\showpaste}
\tab{5}\spadcommand{(x+1/3)**5\bound{p1 }}
\end{paste}\end{patch}

\begin{patch}{ugIntProgFunctionsPagePatch2}
\begin{paste}{ugIntProgFunctionsPageFull2}{ugIntProgFunctionsPageEmpty2}
\pastebutton{ugIntProgFunctionsPageFull2}{\hidepaste}
\tab{5}\spadcommand{p := compiledFunction(\%,x)$MakeUnaryCompiledFunction(POLY FRAC INT,DFLOAT,DFLOAT)\bound
\indentrel{3}\begin{verbatim}
   (2)  theMap(MKUCFUNC;unaryFunction;SM;2!0,350)
                     Type: (DoubleFloat -> DoubleFloat)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntProgFunctionsPageEmpty2}
\begin{paste}{ugIntProgFunctionsPageEmpty2}{ugIntProgFunctionsPagePatch2}
\pastebutton{ugIntProgFunctionsPageEmpty2}{\showpaste}
\tab{5}\spadcommand{p := compiledFunction(\%,x)$MakeUnaryCompiledFunction(POLY FRAC INT,DFLOAT,DFLOAT)\bound
\end{paste}\end{patch}

\begin{patch}{ugIntProgFunctionsPagePatch3}
\begin{paste}{ugIntProgFunctionsPageFull3}{ugIntProgFunctionsPageEmpty3}
\pastebutton{ugIntProgFunctionsPageFull3}{\hidepaste}
\tab{5}\spadcommand{p(sin(1.3))\bound{p3 }\free{p2 }}
```

```
\indentrel{3}\begin{verbatim}
   (3)  3.668751115057229
                                        Type: DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntProgFunctionsPageEmpty3}
\begin{paste}{ugIntProgFunctionsPageEmpty3}{ugIntProgFunctionsPagePatch3}
\pastebutton{ugIntProgFunctionsPageEmpty3}{\showpaste}
\tab{5}\spadcommand{p(sin(1.3))\bound{p3 }\free{p2 }}
\end{paste}\end{patch}
```

———-

Automatic Newton Iteration Formulas

⇒ “notitle” (MappingPackageOneXmpPage) 3.73 on page 965
— **ug10.ht** —

```
\begin{page}{ugIntProgNewtonPage}
{10.9. Automatic Newton Iteration Formulas}
\beginscroll

We resume
our continuing saga of arrows and complex functions.
Suppose we want to investigate the behavior of Newton's iteration function
in the complex plane.
Given a function \smath{f}, we want to find the complex values
\smath{z} such that \smath{f(z) = 0}.

The first step is to produce a Newton iteration formula for
a given \smath{f}:
\texht{$x_{n+1} = x_n - {{f(x_n)}\over{f'(x_n)}}.$}{%
\spad{x(n+1) = x(n) - f(x(n))/f'(x(n))}.}
We represent this formula by a function \smath{g}
that performs the computation on the right-hand side, that is,
\texht{$x_{n+1} = {g}(x_n)$}{\spad{x(n+1) = g(x(n))}}.

The type \spadtype{Expression Integer} (abbreviated \spadtype{EXPR
INT}) is used to represent general symbolic expressions in
Axiom.
To make our facility as general as possible, we assume
\smath{f} has this type.
Given \smath{f}, we want
to produce a Newton iteration function \spad{g} which,
given a complex point \texht{$x_n$}{x(n)}, delivers the next
Newton iteration point \texht{$x_{n+1}$}{x(n+1)}.
```

This time we write an input file called {\bf newton.input}.
We need to import \spadtype{MakeUnaryCompiledFunction} (discussed
in the last section), call it with appropriate types, and then define
the function \spad{newtonStep} which references it.
Here is the function \spad{newtonStep}:

```
\beginImportant

\noindent
{\tt 1.\ \ \ C\ :=\ Complex\ DoubleFloat}\newline
{\tt 2.\ \ \ complexFunPack:=MakeUnaryCompiledFunction(EXPR\ INT,C,C)}
\newline
{\tt 3.\ \ \ }\newline
{\tt 4.\ \ \ newtonStep(f)\ ==}\newline
{\tt 5.\ \ \ \ \ fun\ \ :=\ complexNumericFunction\ f}\newline
{\tt 6.\ \ \ \ \ deriv\ :=\ complexDerivativeFunction(f,1)}\newline
{\tt 7.\ \ \ \ \ (x:C):C\ +->}\newline
{\tt 8.\ \ \ \ \ \ \ x\ -\ fun(x)/deriv(x)}\newline
{\tt 9.\ \ \ }\newline
{\tt 10.\ \ complexNumericFunction\ f\ ==}\newline
{\tt 11.\ \ \ \ v\ :=\ theVariableIn\ f}\newline
{\tt 12.\ \ \ \ compiledFunction(f,\ v)\$complexFunPack}\newline
{\tt 13.\ \ }\newline
{\tt 14.\ \ complexDerivativeFunction(f,n)\ ==}\newline
{\tt 15.\ \ \ \ v\ :=\ theVariableIn\ f}\newline
{\tt 16.\ \ \ \ df\ :=\ D(f,v,n)}\newline
{\tt 17.\ \ \ \ compiledFunction(df,\ v)\$complexFunPack}\newline
{\tt 18.\ \ }\newline
{\tt 19.\ \ theVariableIn\ f\ ==\ \ }\newline
{\tt 20.\ \ \ \ vl\ :=\ variables\ f}\newline
{\tt 21.\ \ \ \ nv\ :=\ \#\ vl}\newline
{\tt 22.\ \ \ \ nv\ >\ 1\ =>\ error\ "Expression\ is\ not\ univariate."}
\newline
{\tt 23.\ \ \ \ nv\ =\ 0\ =>\ 'x}\newline
{\tt 24.\ \ \ \ first\ vl}\newline
\endImportant
```

Do you see what is going on here?
A formula \spad{f} is passed into the function \userfun{newtonStep}.
First, the function turns \spad{f} into a compiled program mapping
complex numbers into complex numbers. Next, it does the same thing
for the derivative of \spad{f}. Finally, it returns a function which
computes a single step of Newton's iteration.

The function \userfun{complexNumericFunction} extracts the variable
from the expression \spad{f} and then turns \spad{f} into a function
which maps complex numbers into complex numbers. The function
\userfun{complexDerivativeFunction} does the same thing for the
derivative of \spad{f}. The function \userfun{theVariableIn}

```
extracts the variable from the expression \spad{f}, calling the function
\spadfun{error} if \spad{f} has more than one variable.
It returns the dummy variable \spad{x} if \spad{f} has no variables.

Let's now apply \userfun{newtonStep} to the formula for computing
cube roots of two.
%
\xtc{
Read the input file with the definitions.
}{
\spadpaste{)read newton\bound{n1}}
}
\xtc{}{
\spadpaste{)read vectors \bound{n1a}}
}

\xtc{
The cube root of two.
}{
\spadpaste{f := x**3 - 2\bound{n2}\free{n1 n1a}}
}
\xtc{
Get Newton's iteration formula.
}{
\spadpaste{g := newtonStep f\bound{n3}\free{n2}}
}
\xtc{
Let \spad{a} denote the result of
applying Newton's iteration once to the complex number \spad{1 + \%i}.
}{
\spadpaste{a := g(1.0 + \%i)\bound{n4}\free{n3}}
}
\xtc{
Now apply it repeatedly. How fast does it converge?
}{
\spadpaste{[(a := g(a)) for i in 1..]\bound{n5}\free{n4}}
}
\xtc{
Check the accuracy of the last iterate.
}{
\spadpaste{a**3\bound{n6}\free{n5}}
}

In
\downlink{`MappingPackage1'}{MappingPackageOneXmpPage}
\ignore{MappingPackage1},
we show how functions can be manipulated as objects in Axiom.  A
useful operation to consider here is \spadop{*}, which means
composition.  For example \spad{g*g} causes the Newton iteration
formula to be applied twice.  Correspondingly, \spad{g**n} means to
```

```
apply the iteration formula \spad{n} times.

%
\xtc{
Apply \spad{g} twice to the point \spad{1 + \%i}.
}{
\spadpaste{(g*g) (1.0 + \%i)\bound{n10}\free{n3}}
}
\xtc{
Apply \spad{g} 11 times.
}{
\spadpaste{(g**11) (1.0 + \%i)\bound{n11}\free{n10}}
}

Look now at the vector field and surface generated
after two steps of Newton's formula for the cube root of two.
The poles in these pictures represent bad starting values, and the
flat areas are the regions of convergence to the three roots.
%
\psXtc{
The vector field.
}{
\graphpaste{drawComplexVectorField(g**3,-3..3,-3..3)\free{n3}}
}{
\epsffile[0 0 295 295]{../ps/vectorroot.ps}
}
\psXtc{
The surface.
}{
\graphpaste{drawComplex(g**3,-3..3,-3..3)\free{n3}}
}{
\epsffile[0 0 295 295]{../ps/complexroot.ps}
}
\

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugIntProgNewtonPagePatch1}
\begin{paste}{ugIntProgNewtonPageFull1}{ugIntProgNewtonPageEmpty1}
\pastebutton{ugIntProgNewtonPageFull1}{\hidepaste}
\tab{5}\spadcommand{)read newton\bound{n1 }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
   (2)
  MakeUnaryCompiledFunction(Expression Integer,Complex Do
  ubleFloat,Complex DoubleFloat)
                                           Type: Domain
                                             Type: Void
```

```
                                              Type: Void
                                              Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntProgNewtonPageEmpty1}
\begin{paste}{ugIntProgNewtonPageEmpty1}{ugIntProgNewtonPagePatch1}
\pastebutton{ugIntProgNewtonPageEmpty1}{\showpaste}
\tab{5}\spadcommand{)read newton\bound{n1 }}
\end{paste}\end{patch}

\begin{patch}{ugIntProgNewtonPagePatch2}
\begin{paste}{ugIntProgNewtonPageFull2}{ugIntProgNewtonPageEmpty2}
\pastebutton{ugIntProgNewtonPageFull2}{\hidepaste}
\tab{5}\spadcommand{)read vectors\bound{n1a }}
\indentrel{3}\begin{verbatim}
   (6)  2.8274333882308138
                                       Type: DoubleFloat
   (7)  0.20000000000000001
                                       Type: DoubleFloat
                                              Type: Void
   (9)  6.0
                                       Type: DoubleFloat
                                              Type: Void
   (11)  25
                                           Type: Integer
   (12)  25
                                           Type: Integer
   (13)  Complex DoubleFloat
                                            Type: Domain
   (14)  Segment DoubleFloat
                                            Type: Domain
                                              Type: Void
                                              Type: Void
                                              Type: Void
                                              Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntProgNewtonPageEmpty2}
\begin{paste}{ugIntProgNewtonPageEmpty2}{ugIntProgNewtonPagePatch2}
\pastebutton{ugIntProgNewtonPageEmpty2}{\showpaste}
\tab{5}\spadcommand{)read vectors\bound{n1a }}
\end{paste}\end{patch}

\begin{patch}{ugIntProgNewtonPagePatch3}
\begin{paste}{ugIntProgNewtonPageFull3}{ugIntProgNewtonPageEmpty3}
\pastebutton{ugIntProgNewtonPageFull3}{\hidepaste}
\tab{5}\spadcommand{f := x**3 - 2\bound{n2 }\free{n1 n1a }}
\indentrel{3}\begin{verbatim}
```

```
         3
   (19)  x  - 2
                                Type: Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntProgNewtonPageEmpty3}
\begin{paste}{ugIntProgNewtonPageEmpty3}{ugIntProgNewtonPagePatch3}
\pastebutton{ugIntProgNewtonPageEmpty3}{\showpaste}
\tab{5}\spadcommand{f := x**3 - 2\bound{n2 }\free{n1 n1a }}
\end{paste}\end{patch}

\begin{patch}{ugIntProgNewtonPagePatch4}
\begin{paste}{ugIntProgNewtonPageFull4}{ugIntProgNewtonPageEmpty4}
\pastebutton{ugIntProgNewtonPageFull4}{\hidepaste}
\tab{5}\spadcommand{g := newtonStep f\bound{n3 }\free{n2 }}
\indentrel{3}\begin{verbatim}
   (20)  theMap(LAMBDA_f3lkwh_704,484)
     Type: (Complex DoubleFloat -> Complex DoubleFloat)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntProgNewtonPageEmpty4}
\begin{paste}{ugIntProgNewtonPageEmpty4}{ugIntProgNewtonPagePatch4}
\pastebutton{ugIntProgNewtonPageEmpty4}{\showpaste}
\tab{5}\spadcommand{g := newtonStep f\bound{n3 }\free{n2 }}
\end{paste}\end{patch}

\begin{patch}{ugIntProgNewtonPagePatch5}
\begin{paste}{ugIntProgNewtonPageFull5}{ugIntProgNewtonPageEmpty5}
\pastebutton{ugIntProgNewtonPageFull5}{\hidepaste}
\tab{5}\spadcommand{a := g(1.0 + \%i)\bound{n4 }\free{n3 }}
\indentrel{3}\begin{verbatim}
   (21)  0.66666666666666674 + 0.33333333333333337%i
                               Type: Complex DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntProgNewtonPageEmpty5}
\begin{paste}{ugIntProgNewtonPageEmpty5}{ugIntProgNewtonPagePatch5}
\pastebutton{ugIntProgNewtonPageEmpty5}{\showpaste}
\tab{5}\spadcommand{a := g(1.0 + \%i)\bound{n4 }\free{n3 }}
\end{paste}\end{patch}

\begin{patch}{ugIntProgNewtonPagePatch6}
\begin{paste}{ugIntProgNewtonPageFull6}{ugIntProgNewtonPageEmpty6}
\pastebutton{ugIntProgNewtonPageFull6}{\hidepaste}
\tab{5}\spadcommand{[(a := g(a)) for i in 1..]\bound{n5 }\free{n4 }}
\indentrel{3}\begin{verbatim}
   (22)
```

```
   [1.1644444444444444 - 0.73777777777777775%i,
    0.92614004697164776 - 0.17463006425584393%i,
    1.3164444838140228 + 0.15690694583015852%i,
    1.2462991025761463 + 0.015454763610132094%i,
    1.2598725296532081 - 0.00033827162059311272%i,
    1.259920960928212 + 2.6023534653422681e-08%i,
    1.259921049894879 - 3.6751942591616685e-15%i,
    1.2599210498948732 - 3.3132158019282496e-29%i,
    1.2599210498948732 - 5.6051938572992683e-45%i,
    1.2599210498948732, ...]
                       Type: Stream Complex DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntProgNewtonPageEmpty6}
\begin{paste}{ugIntProgNewtonPageEmpty6}{ugIntProgNewtonPagePatch6}
\pastebutton{ugIntProgNewtonPageEmpty6}{\showpaste}
\tab{5}\spadcommand{[(a := g(a)) for i in 1..]\bound{n5 }\free{n4 }}
\end{paste}\end{patch}

\begin{patch}{ugIntProgNewtonPagePatch7}
\begin{paste}{ugIntProgNewtonPageFull7}{ugIntProgNewtonPageEmpty7}
\pastebutton{ugIntProgNewtonPageFull7}{\hidepaste}
\tab{5}\spadcommand{a**3\bound{n6 }\free{n5 }}
\indentrel{3}\begin{verbatim}
   (23)  2.0
                              Type: Complex DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntProgNewtonPageEmpty7}
\begin{paste}{ugIntProgNewtonPageEmpty7}{ugIntProgNewtonPagePatch7}
\pastebutton{ugIntProgNewtonPageEmpty7}{\showpaste}
\tab{5}\spadcommand{a**3\bound{n6 }\free{n5 }}
\end{paste}\end{patch}

\begin{patch}{ugIntProgNewtonPagePatch8}
\begin{paste}{ugIntProgNewtonPageFull8}{ugIntProgNewtonPageEmpty8}
\pastebutton{ugIntProgNewtonPageFull8}{\hidepaste}
\tab{5}\spadcommand{(g*g) (1.0 + \%i)\bound{n10 }\free{n3 }}
\indentrel{3}\begin{verbatim}
   (24)  1.1644444444444444 - 0.73777777777777775%i
                              Type: Complex DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntProgNewtonPageEmpty8}
\begin{paste}{ugIntProgNewtonPageEmpty8}{ugIntProgNewtonPagePatch8}
\pastebutton{ugIntProgNewtonPageEmpty8}{\showpaste}
\tab{5}\spadcommand{(g*g) (1.0 + \%i)\bound{n10 }\free{n3 }}
```

```
\end{paste}\end{patch}

\begin{patch}{ugIntProgNewtonPagePatch9}
\begin{paste}{ugIntProgNewtonPageFull9}{ugIntProgNewtonPageEmpty9}
\pastebutton{ugIntProgNewtonPageFull9}{\hidepaste}
\tab{5}\spadcommand{(g**11) (1.0 + \%i)\bound{n11 }\free{n10 }}
\indentrel{3}\begin{verbatim}
   (25)  1.2599210498948732
                              Type: Complex DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntProgNewtonPageEmpty9}
\begin{paste}{ugIntProgNewtonPageEmpty9}{ugIntProgNewtonPagePatch9}
\pastebutton{ugIntProgNewtonPageEmpty9}{\showpaste}
\tab{5}\spadcommand{(g**11) (1.0 + \%i)\bound{n11 }\free{n10 }}
\end{paste}\end{patch}

\begin{patch}{ugIntProgNewtonPagePatch10}
\begin{paste}{ugIntProgNewtonPageFull10}{ugIntProgNewtonPageEmpty10}
\pastebutton{ugIntProgNewtonPageFull10}{\hidepaste}
\tab{5}\spadgraph{drawComplexVectorField(g**3,-3..3,-3..3)\free{n3 }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/ugintprognewtonpage10.view/image}}{viewalone\spa
\end{paste}\end{patch}

\begin{patch}{ugIntProgNewtonPageEmpty10}
\begin{paste}{ugIntProgNewtonPageEmpty10}{ugIntProgNewtonPagePatch10}
\pastebutton{ugIntProgNewtonPageEmpty10}{\showpaste}
\tab{5}\spadgraph{drawComplexVectorField(g**3,-3..3,-3..3)\free{n3 }}
\end{paste}\end{patch}

\begin{patch}{ugIntProgNewtonPagePatch11}
\begin{paste}{ugIntProgNewtonPageFull11}{ugIntProgNewtonPageEmpty11}
\pastebutton{ugIntProgNewtonPageFull11}{\hidepaste}
\tab{5}\spadgraph{drawComplex(g**3,-3..3,-3..3)\free{n3 }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/ugintprognewtonpage11.view/image}}{viewalone\spa
\end{paste}\end{patch}

\begin{patch}{ugIntProgNewtonPageEmpty11}
\begin{paste}{ugIntProgNewtonPageEmpty11}{ugIntProgNewtonPagePatch11}
\pastebutton{ugIntProgNewtonPageEmpty11}{\showpaste}
\tab{5}\spadgraph{drawComplex(g**3,-3..3,-3..3)\free{n3 }}
\end{paste}\end{patch}

\begin{patch}{ugIntProgNewtonPagePatch12}
\begin{paste}{ugIntProgNewtonPageFull12}{ugIntProgNewtonPageEmpty12}
\pastebutton{ugIntProgNewtonPageFull12}{\hidepaste}
\tab{5}\spadcommand{all}
\indentrel{3}\begin{verbatim}
   (28)  all
```

```
                                   Type: Variable all
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugIntProgNewtonPageEmpty12}
\begin{paste}{ugIntProgNewtonPageEmpty12}{ugIntProgNewtonPagePatch12}
\pastebutton{ugIntProgNewtonPageEmpty12}{\showpaste}
\tab{5}\spadcommand{all}
\end{paste}\end{patch}
```

Chapter 14

Users Guide Chapter 11 (ug11.ht)

Packages

⇒ "notitle" (ugIntProgPage) 13 on page 2337
⇒ "notitle" (ugIntProgPage) 13 on page 2337
⇒ "notitle" (ugPackagesNamesPage) 14 on page 2377
⇒ "notitle" (ugPackagesSyntaxPage) 14 on page 2379
⇒ "notitle" (ugPackagesAbstractPage) 14 on page 2380
⇒ "notitle" (ugPackagesCapsulesPage) 14 on page 2381
⇒ "notitle" (ugPackagesInputFilesPage) 14 on page 2382
⇒ "notitle" (ugPackagesPackagesPage) 14 on page 2383
⇒ "notitle" (ugPackagesParametersPage) 14 on page 2387
⇒ "notitle" (ugPackagesCondsPage) 14 on page 2390
⇒ "notitle" (ugPackagesCompilingPage) 14 on page 2392
⇒ "notitle" (ugPackagesHowPage) 14 on page 2399

— **ug11.ht** —

```
\begin{page}{ugPackagesPage}{11. Packages}
\beginscroll

Packages provide the bulk of
Axiom's algorithmic library, from numeric packages for computing
special functions to symbolic facilities for
differential equations, symbolic integration, and limits.

In \downlink{``Interactive Programming''}{ugIntProgPage} in Chapter
10\ignore{ugIntProg}, we developed several useful
functions for drawing vector fields and complex functions.  We now
```

```
show you how you can add these functions to the Axiom library to make
them available for general use.

The way we created the functions in
\downlink{``Interactive Programming''}{ugIntProgPage} in Chapter
10\ignore{ugIntProg} is typical of how you, as an
advanced Axiom user, may interact with Axiom.  You have an
application.  You go to your editor and create an input file defining
some functions for the application.  Then you run the file and try the
functions.  Once you get them all to work, you will often want to
extend them, add new features, perhaps write additional functions.

Eventually, when you have a useful set of functions for your
application, you may want to add them to your local Axiom library.  To
do this, you embed these function definitions in a package and add
that package to the library.

To introduce new packages, categories, and domains into the system,
you need to use the Axiom compiler to convert the constructors into
executable machine code.  An existing compiler in Axiom is available
on an ``as-is'' basis.  A new, faster compiler will be available in
version 2.0 of Axiom.

\beginImportant

\noindent
\label{pak-cdraw}
{\tt 1.\ \ \ C\ \ \ \ \ \ ==>\ Complex\ DoubleFloat}\newline
{\tt 2.\ \ \ S\ \ \ \ \ \ ==>\ Segment\ DoubleFloat}\newline
{\tt 3.\ \ \ INT\ \ \ \ ==>\ Integer}\newline
{\tt 4.\ \ \ DFLOAT\ ==>\ DoubleFloat}\newline
{\tt 5.\ \ \ VIEW3D\ ==>\ ThreeDimensionalViewport}\newline
{\tt 6.\ \ \ CURVE\ \ ==>\ List\ List\ Point\ DFLOAT}\newline
{\tt 7.\ \ \ }\newline
{\tt 8.\ \ \ )abbrev\ package\ DRAWCX\ DrawComplex}\newline
{\tt 9.\ \ \ DrawComplex():\ Exports\ ==\ Implementation\ where}\newline
{\tt 10.\ \ }\newline
{\tt 11.\ \ \ \ Exports\ ==\ with}\newline
{\tt 12.\ \ \ \ \ \ drawComplex:\ (C\ ->\ C,S,S,Boolean)\ ->\ VIEW3D}\newline
{\tt 13.\ \ \ \ \ \
drawComplexVectorField:\ (C\ ->\ C,S,S)\ ->\ VIEW3D}\newline
{\tt 14.\ \ \ \ \ \ setRealSteps:\ INT\ ->\ INT}\newline
{\tt 15.\ \ \ \ \ \ setImagSteps:\ INT\ ->\ INT}\newline
{\tt 16.\ \ \ \ \ \ setClipValue:\ DFLOAT->\ DFLOAT}\newline
{\tt 17.\ \ }\newline
{\tt 18.\ \ \ \ Implementation\ ==\ add}\newline
{\tt 19.\ \ \ \ \ \ arrowScale\ :\ DFLOAT\ :=\ (0.2)::DFLOAT\ --relative\ size}\newline
{\tt 20.\ \ \ \ \ \
arrowAngle\ :\ DFLOAT\ :=\ pi()-pi()/(20::DFLOAT)}\newline
{\tt 21.\ \ \ \ \ \
```

```
realSteps\ \ :\ INT\ :=\ 11\ --\#\ real\ steps}\newline
{\tt 22.\ \ \ \ \ \
imagSteps\ \ :\ INT\ :=\ 11\ --\#\ imaginary\ steps}\newline
{\tt 23.\ \ \ \ \ \
clipValue\ \ :\ DFLOAT\ \ :=\ 10::DFLOAT\ --maximum\ vector\ length}\newline
{\tt 24.\ \ }\newline
{\tt 25.\ \ \ \ \ \ setRealSteps(n)\ ==\ realSteps\ :=\ n}\newline
{\tt 26.\ \ \ \ \ \ setImagSteps(n)\ ==\ imagSteps\ :=\ n}\newline
{\tt 27.\ \ \ \ \ \ setClipValue(c)\ ==\ clipValue\ :=\ c}\newline
{\tt 28.\ \ }\newline
{\tt 29.\ \ \ \ \ \
clipFun:\ DFLOAT\ ->\ DFLOAT\ --Clip\ large\ magnitudes.}\newline
{\tt 30.\ \ \ \ \ \
clipFun(x)\ ==\ min(max(x,\ -clipValue),\ clipValue)}\newline
{\tt 31.\ \ }\newline
{\tt 32.\ \ \ \ \ \
makeArrow:\ (Point\ DFLOAT,Point\ DFLOAT,DFLOAT,DFLOAT)\ ->\ CURVE}\newline
{\tt 33.\ \ \ \ \ \ makeArrow(p1,\ p2,\ len,\ arg)\ ==\ ...}\newline
{\tt 34.\ \ }\newline
{\tt 35.\ \ \ \ \ \
drawComplex(f,\ realRange,\ imagRange,\ arrows?)\ ==\ ...}\newline
\caption{The DrawComplex package.}\label{fig-pak-cdraw}
\endImportant

\beginmenu
\menudownlink{{11.1. Names, Abbreviations, and File Structure}}
{ugPackagesNamesPage}
\menudownlink{{11.2. Syntax}}{ugPackagesSyntaxPage}
\menudownlink{{11.3. Abstract Datatypes}}{ugPackagesAbstractPage}
\menudownlink{{11.4. Capsules}}{ugPackagesCapsulesPage}
\menudownlink{{11.5. Input Files vs. Packages}}{ugPackagesInputFilesPage}
\menudownlink{{11.6. Compiling Packages}}{ugPackagesPackagesPage}
\menudownlink{{11.7. Parameters}}{ugPackagesParametersPage}
\menudownlink{{11.8. Conditionals}}{ugPackagesCondsPage}
\menudownlink{{11.9. Testing}}{ugPackagesCompilingPage}
\menudownlink{{11.10. How Packages Work}}{ugPackagesHowPage}
\endmenu
\endscroll
\autobuttons
\end{page}
```

———-

Names, Abbreviations, and File Structure

⇒ "notitle" (ugIntProgPage) 13 on page 2337

⇒ “notitle” (ugTypesWritingAbbrPage) 7 on page 1638

— ug11.ht —

```
\begin{page}{ugPackagesNamesPage}
{11.1. Names, Abbreviations, and File Structure}
\beginscroll
%

Each package has a name and an abbreviation.  For a package of the
complex draw functions from
\downlink{‘‘Interactive Programming’’}{ugIntProgPage} in Chapter
10\ignore{ugIntProg}, we choose the name
\nonLibAxiomType{DrawComplex} and abbreviation
\nonLibAxiomType{DRAWCX}.\footnote{An abbreviation can be any string
of between two and seven capital letters and digits, beginning with a
letter.  See
\downlink{‘‘Abbreviations’’}{ugTypesWritingAbbrPage} in
Section 2.5.5\ignore{ugTypesWritingAbbr} for more
information.}  To be sure that you have not chosen a name or
abbreviation already used by the system, issue the system command
\spadcmd{)show} for both the name and the abbreviation.

Once you have named the package and its abbreviation, you can choose
any new filename you like with extension ‘‘{\bf \spadFileExt{}}’’ to
hold the definition of your package.  We choose the name {\bf
drawpak\spadFileExt{}}.  If your application involves more than one
package, you can put them all in the same file.  Axiom assumes no
relationship between the name of a library file, and the name or
abbreviation of a package.

Near the top of the ‘‘{\bf \spadFileExt{}}’’ file, list all the
abbreviations for the packages using \spadcmd{)abbrev}, each command
beginning in column one.  Macros giving names to Axiom expressions can
also be placed near the top of the file.  The macros are only usable
from their point of definition until the end of the file.

Consider the definition of
\nonLibAxiomType{DrawComplex} in Figure \ref{fig-pak-cdraw}.
After the macro
definition
\begin{verbatim}
S      ==> Segment DoubleFloat
\end{verbatim}
the name
{\tt S} can be used in the file as a
shorthand for \axiomType{Segment DoubleFloat}.
\footnote{The interpreter also allows
{\tt macro} for macro definitions.}
The abbreviation command for the package
\begin{verbatim}
```

```
)abbrev package DRAWCX DrawComplex
\end{verbatim}
is given after the macros (although it could precede them).

\endscroll
\autobuttons
\end{page}
```

———

Syntax

— **ug11.ht** —

```
\begin{page}{ugPackagesSyntaxPage}{11.2. Syntax}
\beginscroll

The definition of a package has the syntax:
\centerline{{\frenchspacing{\it PackageForm {\tt :}
Exports\quad{\tt ==}\quad Implementation}}}
The syntax for defining a package
constructor is the same as that for defining any function in Axiom.
In practice, the definition extends over many lines so that this
syntax is not practical.  Also, the type of a package is expressed by
the operator \axiom{with} \spadkey{with} followed by an explicit list
of operations.  A preferable way to write the definition of a package
is with a \axiom{where} \spadkey{where} expression:

\beginImportant
The definition of a package usually has the form: \newline
{\tt%
{\it PackageForm} : Exports  ==  Implementation where \newline
\texht{\hspace*{.75pc}}{\tab{3}} {\it optional type declarations}\newline
\texht{\hspace*{.75pc}}{\tab{3}} Exports  ==   with \newline
\texht{\hspace*{2.0pc}}{\tab{6}}   {\it list of exported operations}\newline
\texht{\hspace*{.75pc}}{\tab{3}} Implementation == add \newline
\texht{\hspace*{2.0pc}}{\tab{6}}
{\it list of function definitions for exported operations}
}
\endImportant

The \axiomType{DrawComplex} package takes no parameters and exports five
operations, each a separate item of a \spadgloss{pile}.
Each operation is described as a \spadgloss{declaration}: a name, followed
by a colon (\axiomSyntax{:}), followed by the type of the operation.
All operations have types expressed as \spadglossSee{mappings}{mapping} with
the syntax
```

```
\centerline{{{\it}}
\centerline{{source\quad{\tt ->}\quad target}}
\centerline{{}}}

\endscroll
\autobuttons
\end{page}
```

———-

Abstract Datatypes

⇒ "notitle" (ugIntProgCompFunsPage) 13 on page 2361

— ug11.ht —

```
\begin{page}{ugPackagesAbstractPage}{11.3. Abstract Datatypes}
\beginscroll

A constructor as defined in Axiom is called an \spadgloss{abstract
datatype} in the computer science literature.  Abstract datatypes
separate ``specification'' (what operations are provided) from
``implementation'' (how the operations are implemented).  The {\tt
Exports} (specification) part of a constructor is said to be
``public'' (it provides the user interface to the package) whereas the
{\tt Implementation} part is ``private'' (information here is
effectively hidden---programs cannot take advantage of it).

The {\tt Exports} part specifies what operations the package provides
to users.  As an author of a package, you must ensure that the {\tt
Implementation} part provides a function for each operation in the
{\tt Exports} part.\footnote{The \spadtype{DrawComplex} package
enhances the facility described in
\downlink{``Drawing Complex Functions''}{ugIntProgCompFunsPage} in
Chapter 10.7\ignore{ugIntProgCompFuns} by allowing
a complex function to have arrows emanating from the surface to
indicate the direction of the complex argument.}

An important difference between interactive programming and the use of
packages is in the handling of global variables such as
\axiom{realSteps} and \axiom{imagSteps}.  In interactive programming,
you simply change the values of variables by \spadgloss{assignment}.
With packages, such variables are local to the package---their values
can only be set using functions exported by the package.  In our
example package, we provide two functions \fakeAxiomFun{setRealSteps}
and \fakeAxiomFun{setImagSteps} for this purpose.

Another local variable is \axiom{clipValue} which can be changed using
```

```
the exported operation \fakeAxiomFun{setClipValue}.  This value is
referenced by the internal function \fakeAxiomFun{clipFun} that
decides whether to use the computed value of the function at a point
or, if the magnitude of that value is too large, the value assigned to
\axiom{clipValue} (with the appropriate sign).

\endscroll
\autobuttons
\end{page}
```

———-

Capsules

— ug11.ht —

```
\begin{page}{ugPackagesCapsulesPage}{11.4. Capsules}
\beginscroll
%
The part to the right of {\tt add} in the {\tt Implementation}
\spadkey{add}
part of the definition is called a \spadgloss{capsule}.
The purpose of a capsule is:
\indent{4}
\beginitems
\item[-] to define a function for each exported operation, and
\item[-] to define a \spadgloss{local environment}
for these functions to run.
\enditems
\indent{0}

What is a local environment?
First, what is an environment?
Think of the capsule as an input file that Axiom reads from top to
bottom.
Think of the input file as having a \axiom{)clear all} at the top
so that initially no variables or functions are defined.
When this file is read, variables such as \axiom{realSteps} and
\axiom{arrowSize} in \nonLibAxiomType{DrawComplex} are set to initial
values. Also, all the functions defined in the capsule are compiled.
These include those that are exported (like \axiom{drawComplex}), and
those that are not (like \axiom{makeArrow}).
At the end, you get a set of name-value pairs:
variable names (like \axiom{realSteps} and \axiom{arrowSize})
are paired with assigned values, while
operation names (like \axiom{drawComplex} and \axiom{makeArrow})
are paired with function values.
```

```
This set of name-value pairs is called an \spadgloss{environment}.
Actually, we call this environment the ``initial environment'' of a
package: it is the environment that exists immediately after the
package is first built.  Afterwards, functions of this capsule can
access or reset a variable in the environment.  The environment is
called {\it local} since any changes to the value of a variable in
this environment can be seen {\it only} by these functions.

Only the functions from the package can change the variables in the
local environment.  When two functions are called successively from a
package, any changes caused by the first function called are seen by
the second.

Since the environment is local to the package, its names don't get
mixed up with others in the system or your workspace.  If you happen
to have a variable called \axiom{realSteps} in your workspace, it does
not affect what the \nonLibAxiomType{DrawComplex} functions do in any
way.

The functions in a package are compiled into machine code.  Unlike
function definitions in input files that may be compiled repeatedly as
you use them with varying argument types, functions in packages have a
unique type (generally parameterized by the argument parameters of a
package) and a unique compilation residing on disk.

The capsule itself is turned into a compiled function.  This so-called
{\it capsule function} is what builds the initial environment spoken
of above.  If the package has arguments (see below), then each call to
the package constructor with a distinct pair of arguments builds a
distinct package, each with its own local environment.

\endscroll
\autobuttons
\end{page}
```

———-

Input Files vs. Packages

— **ug11.ht** —

```
\begin{page}{ugPackagesInputFilesPage}{11.5. Input Files vs. Packages}
\beginscroll
%
A good question at this point would be
``Is writing a package more difficult than writing an input file?''
```

```
The programs in input files are designed for flexibility and
ease-of-use.  Axiom can usually work out all of your types as it reads
your program and does the computations you request.  Let's say that
you define a one-argument function without giving its type.  When you
first apply the function to a value, this value is understood by Axiom
as identifying the type for the argument parameter.  Most of the time
Axiom goes through the body of your function and figures out the
target type that you have in mind.  Axiom sometimes fails to get it
right.  Then---and only then---do you need a declaration to tell Axiom
what type you want.

Input files are usually written to be read by Axiom---and by you.
Without suitable documentation and declarations, your input files
are likely incomprehensible to a colleague---and to you some
months later!

Packages are designed for legibility, as well as run-time efficiency.
There are few new concepts you need to learn to write
packages. Rather, you just have to be explicit about types and type
conversions.  The types of all functions are pre-declared so that
Axiom---and the reader--- knows precisely what types of arguments can
be passed to and from the functions (certainly you don't want a
colleague to guess or to have to work this out from context!).  The
types of local variables are also declared.  Type conversions are
explicit, never automatic.\footnote{There is one exception to this
rule: conversions from a subdomain to a domain are automatic.  After
all, the objects both have the domain as a common type.}

In summary, packages are more tedious to write than input files.  When
writing input files, you can casually go ahead, giving some facts now,
leaving others for later.  Writing packages requires forethought, care
and discipline.

\endscroll
\autobuttons
\end{page}
```

———-

Compiling Packages

— ug11.ht —

```
\begin{page}{ugPackagesPackagesPage}{11.6. Compiling Packages}
\beginscroll
%
```

```
Once you have defined the package \nonLibAxiomType{DrawComplex}, you
need to compile and test it.  To compile the package, issue the system
command \spadcmd{)compile drawpak}.  Axiom reads the file {\bf
drawpak\spadFileExt{}} and compiles its contents into machine binary.
If all goes well, the file {\bf DRAWCX.nrlib} is created in your local
directory for the package.  To test the package, you must load the
package before trying an operation.

\nullXtc{
Compile the package.
}{
\spadpaste{)compile drawpak}
}
\xtc{
Expose the package.
}{
\spadpaste{)expose DRAWCX \bound{dp}}
}
\xtc{
Use an odd step size to avoid
a pole at the origin.
}{
\spadpaste{setRealSteps 51 \free{dp}\bound{srs}}
}
\xtc{
}{
\spadpaste{setImagSteps 51 \free{dp}\bound{scs}}
}
\xtc{
Define \userfun{f} to be the Gamma function.
}{
\spadpaste{f(z) == Gamma(z) \bound{f}}
}
\xtc{
Clip values of function with magnitude larger than 7.
}{
\spadpaste{setClipValue 7}
}
\psXtc{
Draw the \spadfun{Gamma} function.
}{
\graphpaste{drawComplex(f,-\%pi..\%pi,-\%pi..\%pi, false) \free{srs scs f}}
}{
\epsffile[0 0 300 300]{../ps/3dgamma11.ps}
}

\endscroll
\autobuttons
\end{page}
```

```
\begin{patch}{ugPackagesPackagesPagePatch1}
\begin{paste}{ugPackagesPackagesPageFull1}{ugPackagesPackagesPageEmpty1}
\pastebutton{ugPackagesPackagesPageFull1}{\hidepaste}
\tab{5}\spadcommand{)compile drawpak}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugPackagesPackagesPageEmpty1}
\begin{paste}{ugPackagesPackagesPageEmpty1}{ugPackagesPackagesPagePatch1}
\pastebutton{ugPackagesPackagesPageEmpty1}{\showpaste}
\tab{5}\spadcommand{)compile drawpak}
\end{paste}\end{patch}

\begin{patch}{ugPackagesPackagesPagePatch2}
\begin{paste}{ugPackagesPackagesPageFull2}{ugPackagesPackagesPageEmpty2}
\pastebutton{ugPackagesPackagesPageFull2}{\hidepaste}
\tab{5}\spadcommand{)expose DRAWCX\bound{dp }}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugPackagesPackagesPageEmpty2}
\begin{paste}{ugPackagesPackagesPageEmpty2}{ugPackagesPackagesPagePatch2}
\pastebutton{ugPackagesPackagesPageEmpty2}{\showpaste}
\tab{5}\spadcommand{)expose DRAWCX\bound{dp }}
\end{paste}\end{patch}

\begin{patch}{ugPackagesPackagesPagePatch3}
\begin{paste}{ugPackagesPackagesPageFull3}{ugPackagesPackagesPageEmpty3}
\pastebutton{ugPackagesPackagesPageFull3}{\hidepaste}
\tab{5}\spadcommand{setRealSteps 51\free{dp }\bound{srs }}
\indentrel{3}\begin{verbatim}
   (1)  51
                                   Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugPackagesPackagesPageEmpty3}
\begin{paste}{ugPackagesPackagesPageEmpty3}{ugPackagesPackagesPagePatch3}
\pastebutton{ugPackagesPackagesPageEmpty3}{\showpaste}
\tab{5}\spadcommand{setRealSteps 51\free{dp }\bound{srs }}
\end{paste}\end{patch}

\begin{patch}{ugPackagesPackagesPagePatch4}
\begin{paste}{ugPackagesPackagesPageFull4}{ugPackagesPackagesPageEmpty4}
\pastebutton{ugPackagesPackagesPageFull4}{\hidepaste}
\tab{5}\spadcommand{setImagSteps 51\free{dp }\bound{scs }}
\indentrel{3}\begin{verbatim}
```

```
   (2)  51
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugPackagesPackagesPageEmpty4}
\begin{paste}{ugPackagesPackagesPageEmpty4}{ugPackagesPackagesPagePatch4}
\pastebutton{ugPackagesPackagesPageEmpty4}{\showpaste}
\tab{5}\spadcommand{setImagSteps 51\free{dp }\bound{scs }}
\end{paste}\end{patch}

\begin{patch}{ugPackagesPackagesPagePatch5}
\begin{paste}{ugPackagesPackagesPageFull5}{ugPackagesPackagesPageEmpty5}
\pastebutton{ugPackagesPackagesPageFull5}{\hidepaste}
\tab{5}\spadcommand{f(z) == Gamma(z)\bound{f }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugPackagesPackagesPageEmpty5}
\begin{paste}{ugPackagesPackagesPageEmpty5}{ugPackagesPackagesPagePatch5}
\pastebutton{ugPackagesPackagesPageEmpty5}{\showpaste}
\tab{5}\spadcommand{f(z) == Gamma(z)\bound{f }}
\end{paste}\end{patch}

\begin{patch}{ugPackagesPackagesPagePatch6}
\begin{paste}{ugPackagesPackagesPageFull6}{ugPackagesPackagesPageEmpty6}
\pastebutton{ugPackagesPackagesPageFull6}{\hidepaste}
\tab{5}\spadcommand{setClipValue 7}
\indentrel{3}\begin{verbatim}
   (4)  7.0
                                    Type: DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugPackagesPackagesPageEmpty6}
\begin{paste}{ugPackagesPackagesPageEmpty6}{ugPackagesPackagesPagePatch6}
\pastebutton{ugPackagesPackagesPageEmpty6}{\showpaste}
\tab{5}\spadcommand{setClipValue 7}
\end{paste}\end{patch}

\begin{patch}{ugPackagesPackagesPagePatch7}
\begin{paste}{ugPackagesPackagesPageFull7}{ugPackagesPackagesPageEmpty7}
\pastebutton{ugPackagesPackagesPageFull7}{\hidepaste}
\tab{5}\spadgraph{drawComplex(f,-\%pi..\%pi,-\%pi..\%pi, false)\free{srs scs f }}
\center{\unixcommand{\inputimage{\env{AXIOM}/doc/viewports/ugpackagespackagespage7.view/image}}{
\end{paste}\end{patch}

\begin{patch}{ugPackagesPackagesPageEmpty7}
```

```
\begin{paste}{ugPackagesPackagesPageEmpty7}{ugPackagesPackagesPagePatch7}
\pastebutton{ugPackagesPackagesPageEmpty7}{\showpaste}
\tab{5}\spadgraph{drawComplex(f,-\%pi..\%pi,-\%pi..\%pi, false)\free{srs scs f }}
\end{paste}\end{patch}
```

————-

Parameters

⇒ "notitle" (ugTypesPage) 7 on page 1613
⇒ "notitle" (ugUserBlocksPage) 10 on page 1898
⇒ "notitle" (ugCategoriesAttributesPage) 15 on page 2416

— **ug11.ht** —

```
\begin{page}{ugPackagesParametersPage}{11.7. Parameters}
\beginscroll

The power of packages becomes evident when packages have parameters.
Usually these parameters are domains and the exported operations have
types involving these parameters.

In \downlink{``Using Types and Modes''}{ugTypesPage} in Chapter
2\ignore{ugTypes}, you learned that categories denote
classes of domains.  Although we cover this notion in detail in the
next chapter, we now give you a sneak preview of its usefulness.

In \downlink{``Functions Defined with Blocks''}{ugUserBlocksPage} in
Section 6.15\ignore{ugUserBlocks}, we defined functions
\axiom{bubbleSort(m)} and \axiom{insertionSort(m)} to sort a list of
integers.  If you look at the code for these functions, you see that
they may be used to sort {\it any} structure \axiom{m} with the right
properties.  Also, the functions can be used to sort lists of {\it
any} elements---not just integers.  Let us now recall the code for
\axiom{bubbleSort}.

\begin{verbatim}
bubbleSort(m) ==
  n := #m
  for i in 1..(n-1) repeat
    for j in n..(i+1) by -1 repeat
      if m.j < m.(j-1) then swap!(m,j,j-1)
  m
\end{verbatim}

What properties of ``lists of integers'' are assumed by the sorting
algorithm?  In the first line, the operation \spadfun{\#} computes the
```

```
maximum index of the list.  The first obvious property is that
\axiom{m} must have a finite number of elements.  In Axiom, this is
done by your telling Axiom that \axiom{m} has the ``attribute''
\spadatt{finiteAggregate}.  An \spadgloss{attribute} is a property
that a domain either has or does not have.  As we show later in
\downlink{``Attributes''}{ugCategoriesAttributesPage}
in Section 12.9\ignore{ugCategoriesAttributes}, programs
can query domains as to the presence or absence of an attribute.

The operation \spadfunX{swap} swaps elements of \axiom{m}.
Using \Browse{}, you find that \spadfunX{swap} requires its
elements to come from a domain of category
\axiomType{IndexedAggregate} with attribute
\spadatt{shallowlyMutable}.
This attribute means that you can change the internal components
of \axiom{m} without changing its external structure.
Shallowly-mutable data structures include lists, streams, one- and
two-dimensional arrays, vectors, and matrices.

The category \axiomType{IndexedAggregate} designates the class of
aggregates whose elements can be accessed by the notation
\axiom{m.s} for suitable selectors \axiom{s}.
The category \axiomType{IndexedAggregate} takes two arguments:
\axiom{Index}, a domain of selectors for the aggregate, and
\axiom{Entry}, a domain of entries for the aggregate.
Since the sort functions access elements by integers, we must
choose \axiom{Index = }\axiomType{Integer}.
The most general class of domains for which \axiom{bubbleSort} and
\axiom{insertionSort} are defined are those of
category \spadtype{IndexedAggregate(Integer,Entry)} with the two
attributes \spadatt{shallowlyMutable} and
\spadatt{finiteAggregate}.

Using \Browse{}, you can also discover that Axiom has many kinds of
domains with attribute \spadatt{shallowlyMutable}.  Those of class
\axiomType{IndexedAggregate(Integer,Entry)} include \axiomType{Bits},
\axiomType{FlexibleArray}, \axiomType{OneDimensionalArray},
\axiomType{List}, \axiomType{String}, and \axiomType{Vector}, and also
\axiomType{HashTable} and \axiomType{EqTable} with integer keys.
Although you may never want to sort all such structures, we
nonetheless demonstrate Axiom's ability to do so.

Another requirement is that \nonLibAxiomType{Entry} has an operation
\axiomOp{<}.  One way to get this operation is to assume that
\nonLibAxiomType{Entry} has category \axiomType{OrderedSet}.  By
definition, will then export a \axiomOp{<} operation.  A more general
approach is to allow any comparison function \axiom{f} to be used for
sorting.  This function will be passed as an argument to the sorting
functions.
```

```
Our sorting package then takes two arguments: a domain \axiom{S} of
objects of {\it any} type, and a domain \axiom{A}, an aggregate of
type \axiomType{IndexedAggregate(Integer, S)} with the above two
attributes.  Here is its definition using what are close to the
original definitions of \axiom{bubbleSort} and \axiom{insertionSort}
for sorting lists of integers.  The symbol \axiomSyntax{!} is added to
the ends of the operation names.  This uniform naming convention is
used for Axiom operation names that destructively change one or more
of their arguments.

\beginImportant

\noindent
{\tt 1.\ \ \ SortPackage(S,A)\ :\ Exports\ ==\ Implementation\ where}
\newline
{\tt 2.\ \ \ \ \ S:\ Object}\newline
{\tt 3.\ \ \ \ \ A:\ IndexedAggregate(Integer,S)}\newline
{\tt 4.\ \ \ \ \ \ \ with\ (finiteAggregate;\ shallowlyMutable)}
\newline
{\tt 5.\ \ \ }\newline
{\tt 6.\ \ \ \ \ Exports\ ==\ with}\newline
{\tt 7.\ \ \ \ \ \ \ bubbleSort!:\ (A,(S,S)\ ->\ Boolean)\ ->\ A}
\newline
{\tt 8.\ \ \ \ \ \ \
insertionSort!:\ (A,\ (S,S)\ ->\ Boolean)\ ->\ A}\newline
{\tt 9.\ \ \ }\newline
{\tt 10.\ \ \ \ Implementation\ ==\ add}\newline
{\tt 11.\ \ \ \ \ \ bubbleSort!(m,f)\ ==}\newline
{\tt 12.\ \ \ \ \ \ \ \ n\ :=\ \#m}\newline
{\tt 13.\ \ \ \ \ \ \ \ for\ i\ in\ 1..(n-1)\ repeat}\newline
{\tt 14.\ \ \ \ \ \ \ \ \ \
for\ j\ in\ n..(i+1)\ by\ -1\ repeat}\newline
{\tt 15.\ \ \ \ \ \ \ \ \ \ \ \
if\ f(m.j,m.(j-1))\ then\ swap!(m,j,j-1)}\newline
{\tt 16.\ \ \ \ \ \ \ \ m}\newline
{\tt 17.\ \ \ \ \ \ insertionSort!(m,f)\ ==}\newline
{\tt 18.\ \ \ \ \ \ \ \ for\ i\ in\ 2..\#m\ repeat}\newline
{\tt 19.\ \ \ \ \ \ \ \ \ \ j\ :=\ i}\newline
{\tt 20.\ \ \ \ \ \ \ \ \ \
while\ j\ >\ 1\ and\ f(m.j,m.(j-1))\ repeat}\newline
{\tt 21.\ \ \ \ \ \ \ \ \ \ \ \ swap!(m,j,j-1)}\newline
{\tt 22.\ \ \ \ \ \ \ \ \ \ \ \
j\ :=\ (j\ -\ 1)\ pretend\ PositiveInteger}\newline
{\tt 23.\ \ \ \ \ \ \ \ m}\newline
\endImportant

\endscroll
\autobuttons
\end{page}
```

———-

Conditionals

⇒ "notitle" (ugUserBlocksPage) 10 on page 1898
— **ug11.ht** —

```
\begin{page}{ugPackagesCondsPage}{11.8. Conditionals}
\beginscroll

When packages have parameters, you can say that an operation is or is
not exported depending on the values of those parameters.  When the
domain of objects \axiom{S} has an \axiomOp{<} operation, we can
supply one-argument versions of \axiom{bubbleSort} and
\axiom{insertionSort} which use this operation for sorting.  The
presence of the operation \axiomOp{<} is guaranteed when \axiom{S} is
an ordered set.

\beginImportant

\noindent
{\tt 1.\ \ \ Exports\ ==\ with}\newline
{\tt 2.\ \ \ \ \ \ \ bubbleSort!:\ (A,(S,S)\ ->\ Boolean)\ ->\ A}
\newline
{\tt 3.\ \ \ \ \ \ \
insertionSort!:\ (A,\ (S,S)\ ->\ Boolean)\ ->\ A}\newline
{\tt 4.\ \ \ }\newline
{\tt 5.\ \ \ \ \ \ \ if\ S\ has\ OrderedSet\ then}\newline
{\tt 6.\ \ \ \ \ \ \ \ \ bubbleSort!:\ A\ ->\ A}\newline
{\tt 7.\ \ \ \ \ \ \ \ \ insertionSort!:\ A\ ->\ A}\newline
\endImportant

In addition to exporting the one-argument sort operations
conditionally, we must provide conditional definitions for the
operations in the {\tt Implementation} part.
This is easy: just have the one-argument functions call the
corresponding two-argument functions with the operation
\axiomOp{<} from \axiom{S}.

\beginImportant

\noindent
{\tt 1.\ \ \ \ \ Implementation\ ==\ add}\newline
{\tt 2.\ \ \ \ \ \ \ \ \ \ ...}\newline
{\tt 3.\ \ \ \ \ \ \ if\ S\ has\ OrderedSet\ then}\newline
{\tt 4.\ \ \ \ \ \ \ \ \
bubbleSort!(m)\ ==\ bubbleSort!(m,<\$S)}\newline
{\tt 5.\ \ \ \ \ \ \ \ \
```

```
insertionSort!(m)\ ==\ insertionSort!(m,<\$S)}\newline
\endImportant

In \downlink{``Functions Defined with Blocks''}{ugUserBlocksPage} in
Section 6.15\ignore{ugUserBlocks}, we give an alternative
definition of \fakeAxiomFun{bubbleSort} using
\spadfunFrom{first}{List} and \spadfunFrom{rest}{List} that is more
efficient for a list (for which access to any element requires
traversing the list from its first node).  To implement a more
efficient algorithm for lists, we need the operation \spadfun{setelt}
which allows us to destructively change the \spadfun{first} and
\spadfun{rest} of a list.  Using \Browse{}, you find that these
operations come from category \axiomType{UnaryRecursiveAggregate}.
Several aggregate types are unary recursive aggregates including those
of \axiomType{List} and \axiomType{AssociationList}.  We provide two
different implementations for \fakeAxiomFun{bubbleSort!} and
\fakeAxiomFun{insertionSort!}: one for list-like structures, another
for array-like structures.

\beginImportant

\noindent
{\tt 1.\ \ \ Implementation\ ==\ add}\newline
{\tt 2.\ \ \ \ \ \ \ \ \ \ \ ...}\newline
{\tt 3.\ \ \ \ \ \ \ if\ A\ has\ UnaryRecursiveAggregate(S)\ then}
\newline
{\tt 4.\ \ \ \ \ \ \ \ \ bubbleSort!(m,fn)\ ==}\newline
{\tt 5.\ \ \ \ \ \ \ \ \ \ \ empty?\ m\ =>\ m}\newline
{\tt 6.\ \ \ \ \ \ \ \ \ \ \ l\ :=\ m}\newline
{\tt 7.\ \ \ \ \ \ \ \ \ \ \
while\ not\ empty?\ (r\ :=\ l.rest)\ repeat}\newline
{\tt 8.\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ r\ :=\ bubbleSort!\ r}\newline
{\tt 9.\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ x\ :=\ l.first}\newline
{\tt 10.\ \ \ \ \ \ \ \ \ \ \ \ \ \ if\ fn(r.first,x)\ then}\newline
{\tt 11.\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ l.first\ :=\ r.first}\newline
{\tt 12.\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ r.first\ :=\ x}\newline
{\tt 13.\ \ \ \ \ \ \ \ \ \ \ \ \ \ l.rest\ :=\ r}\newline
{\tt 14.\ \ \ \ \ \ \ \ \ \ \ \ \ \ l\ :=\ l.rest}\newline
{\tt 15.\ \ \ \ \ \ \ \ \ \ \ m}\newline
{\tt 16.\ \ \ \ \ \ \ \ \ insertionSort!(m,fn)\ ==}\newline
{\tt 17.\ \ \ \ \ \ \ \ \ \ \ ...}\newline
\endImportant

The ordering of definitions is important.
The standard definitions come first and
then the predicate
\begin{verbatim}
A has UnaryRecursiveAggregate(S)
\end{verbatim}
is evaluated.
```

```
If {\tt true}, the special definitions cover up the standard ones.

Another equivalent way to write the capsule is to use an
\axiom{if-then-else} expression:
\spadkey{if}

\beginImportant

\noindent
{\tt 1.\ \ \ \ \ \ \ \ if\ A\ has\ UnaryRecursiveAggregate(S)\ then}
\newline
{\tt 2.\ \ \ \ \ \ \ \ \ \ \ ...}\newline
{\tt 3.\ \ \ \ \ \ \ \ else}\newline
{\tt 4.\ \ \ \ \ \ \ \ \ \ \ ...}\newline
\endImportant

\endscroll
\autobuttons
\end{page}
```

———-

Testing

⇒ "notitle" (EqTableXmpPage) 3.26 on page 386

— **ug11.ht** —

```
\begin{page}{ugPackagesCompilingPage}{11.9. Testing}
\beginscroll

Once you have written the package, embed it in a file, for example,
{\bf sortpak\spadFileExt{}}.  Be sure to include an \axiom{)abbrev}
command at the top of the file:
\begin{verbatim}
)abbrev package SORTPAK SortPackage
\end{verbatim}
Now compile the file (using \spadcmd{)compile sortpak\spadFileExt{}}).
\xtc{
Expose the constructor.
You are then ready to begin testing.
}{
\spadpaste{)expose SORTPAK}
}
\xtc{
Define a list.
}{
\spadpaste{l := [1,7,4,2,11,-7,3,2]}
```

```
}
\xtc{
Since the integers are an ordered set,
a one-argument operation will do.
}{
\spadpaste{bubbleSort!(l)}
}
\xtc{
Re-sort it using ``greater than.''
}{
\spadpaste{bubbleSort!(l,(x,y) +-> x > y)}
}
\xtc{
Now sort it again using \axiomOp{<} on integers.
}{
\spadpaste{bubbleSort!(l, <\$Integer)}
}
\xtc{
A string is an aggregate of characters so we can sort them as well.
}{
\spadpaste{bubbleSort! "Mathematical Sciences"}
}
\xtc{
Is \axiomOp{<} defined on booleans?
}{
\spadpaste{false < true}
}
\xtc{
Good! Create a bit string representing ten consecutive
boolean values \axiom{true}.
}{
\spadpaste{u : Bits := new(10,true)}
}
\xtc{
Set bits 3 through 5 to \axiom{false}, then display the result.
}{
\spadpaste{u(3..5) := false; u}
}
\xtc{
Now sort these booleans.
}{
\spadpaste{bubbleSort! u}
}
\xtc{
Create an ``eq-table'' (see
\downlink{`EqTable'}{EqTableXmpPage}\ignore{EqTable}), a
table having integers as keys
and strings as values.
}{
\spadpaste{t : EqTable(Integer,String) := table()}
```

```
}
\xtc{
Give the table a first entry.
}{
\spadpaste{t.1 := "robert"}
}
\xtc{
And a second.
}{
\spadpaste{t.2 := "richard"}
}
\xtc{
What does the table look like?
}{
\spadpaste{t}
}
\xtc{
Now sort it.
}{
\spadpaste{bubbleSort! t}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugPackagesCompilingPagePatch1}
\begin{paste}{ugPackagesCompilingPageFull1}{ugPackagesCompilingPageEmpty1}
\pastebutton{ugPackagesCompilingPageFull1}{\hidepaste}
\tab{5}\spadcommand{)expose SORTPAK}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugPackagesCompilingPageEmpty1}
\begin{paste}{ugPackagesCompilingPageEmpty1}{ugPackagesCompilingPagePatch1}
\pastebutton{ugPackagesCompilingPageEmpty1}{\showpaste}
\tab{5}\spadcommand{)expose SORTPAK}
\end{paste}\end{patch}

\begin{patch}{ugPackagesCompilingPagePatch2}
\begin{paste}{ugPackagesCompilingPageFull2}{ugPackagesCompilingPageEmpty2}
\pastebutton{ugPackagesCompilingPageFull2}{\hidepaste}
\tab{5}\spadcommand{l := [1,7,4,2,11,-7,3,2]}
\indentrel{3}\begin{verbatim}
   (1)  [1,7,4,2,11,- 7,3,2]
                                     Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugPackagesCompilingPageEmpty2}
\begin{paste}{ugPackagesCompilingPageEmpty2}{ugPackagesCompilingPagePatch2}
\pastebutton{ugPackagesCompilingPageEmpty2}{\showpaste}
\tab{5}\spadcommand{l := [1,7,4,2,11,-7,3,2]}
\end{paste}\end{patch}

\begin{patch}{ugPackagesCompilingPagePatch3}
\begin{paste}{ugPackagesCompilingPageFull3}{ugPackagesCompilingPageEmpty3}
\pastebutton{ugPackagesCompilingPageFull3}{\hidepaste}
\tab{5}\spadcommand{bubbleSort!(l)}
\indentrel{3}\begin{verbatim}
   (2)  [- 7,1,2,2,3,4,7,11]
                                     Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugPackagesCompilingPageEmpty3}
\begin{paste}{ugPackagesCompilingPageEmpty3}{ugPackagesCompilingPagePatch3}
\pastebutton{ugPackagesCompilingPageEmpty3}{\showpaste}
\tab{5}\spadcommand{bubbleSort!(l)}
\end{paste}\end{patch}

\begin{patch}{ugPackagesCompilingPagePatch4}
\begin{paste}{ugPackagesCompilingPageFull4}{ugPackagesCompilingPageEmpty4}
\pastebutton{ugPackagesCompilingPageFull4}{\hidepaste}
\tab{5}\spadcommand{bubbleSort!(l,(x,y) +-> x > y)}
\indentrel{3}\begin{verbatim}
   (3)  [11,7,4,3,2,2,1,- 7]
                                     Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugPackagesCompilingPageEmpty4}
\begin{paste}{ugPackagesCompilingPageEmpty4}{ugPackagesCompilingPagePatch4}
\pastebutton{ugPackagesCompilingPageEmpty4}{\showpaste}
\tab{5}\spadcommand{bubbleSort!(l,(x,y) +-> x > y)}
\end{paste}\end{patch}

\begin{patch}{ugPackagesCompilingPagePatch5}
\begin{paste}{ugPackagesCompilingPageFull5}{ugPackagesCompilingPageEmpty5}
\pastebutton{ugPackagesCompilingPageFull5}{\hidepaste}
\tab{5}\spadcommand{bubbleSort!(l, <$Integer)}
\indentrel{3}\begin{verbatim}
   (4)  [- 7,1,2,2,3,4,7,11]
                                     Type: List Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugPackagesCompilingPageEmpty5}
\begin{paste}{ugPackagesCompilingPageEmpty5}{ugPackagesCompilingPagePatch5}
```

```
\pastebutton{ugPackagesCompilingPageEmpty5}{\showpaste}
\tab{5}\spadcommand{bubbleSort!(l, <$Integer)}
\end{paste}\end{patch}

\begin{patch}{ugPackagesCompilingPagePatch6}
\begin{paste}{ugPackagesCompilingPageFull6}{ugPackagesCompilingPageEmpty6}
\pastebutton{ugPackagesCompilingPageFull6}{\hidepaste}
\tab{5}\spadcommand{bubbleSort! "Mathematical Sciences"}
\indentrel{3}\begin{verbatim}
   (5)  " MSaaaccceeehiilmnstt"
                                          Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugPackagesCompilingPageEmpty6}
\begin{paste}{ugPackagesCompilingPageEmpty6}{ugPackagesCompilingPagePatch6}
\pastebutton{ugPackagesCompilingPageEmpty6}{\showpaste}
\tab{5}\spadcommand{bubbleSort! "Mathematical Sciences"}
\end{paste}\end{patch}

\begin{patch}{ugPackagesCompilingPagePatch7}
\begin{paste}{ugPackagesCompilingPageFull7}{ugPackagesCompilingPageEmpty7}
\pastebutton{ugPackagesCompilingPageFull7}{\hidepaste}
\tab{5}\spadcommand{false < true}
\indentrel{3}\begin{verbatim}
   (6)  true
                                         Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugPackagesCompilingPageEmpty7}
\begin{paste}{ugPackagesCompilingPageEmpty7}{ugPackagesCompilingPagePatch7}
\pastebutton{ugPackagesCompilingPageEmpty7}{\showpaste}
\tab{5}\spadcommand{false < true}
\end{paste}\end{patch}

\begin{patch}{ugPackagesCompilingPagePatch8}
\begin{paste}{ugPackagesCompilingPageFull8}{ugPackagesCompilingPageEmpty8}
\pastebutton{ugPackagesCompilingPageFull8}{\hidepaste}
\tab{5}\spadcommand{u : Bits := new(10,true)}
\indentrel{3}\begin{verbatim}
   (7)  "1111111111"
                                            Type: Bits
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugPackagesCompilingPageEmpty8}
\begin{paste}{ugPackagesCompilingPageEmpty8}{ugPackagesCompilingPagePatch8}
\pastebutton{ugPackagesCompilingPageEmpty8}{\showpaste}
\tab{5}\spadcommand{u : Bits := new(10,true)}
```

```
\end{paste}\end{patch}

\begin{patch}{ugPackagesCompilingPagePatch9}
\begin{paste}{ugPackagesCompilingPageFull9}{ugPackagesCompilingPageEmpty9}
\pastebutton{ugPackagesCompilingPageFull9}{\hidepaste}
\tab{5}\spadcommand{u(3..5) := false; u}
\indentrel{3}\begin{verbatim}
   (8)  "1100011111"
                                             Type: Bits
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugPackagesCompilingPageEmpty9}
\begin{paste}{ugPackagesCompilingPageEmpty9}{ugPackagesCompilingPagePatch9}
\pastebutton{ugPackagesCompilingPageEmpty9}{\showpaste}
\tab{5}\spadcommand{u(3..5) := false; u}
\end{paste}\end{patch}

\begin{patch}{ugPackagesCompilingPagePatch10}
\begin{paste}{ugPackagesCompilingPageFull10}{ugPackagesCompilingPageEmpty10}
\pastebutton{ugPackagesCompilingPageFull10}{\hidepaste}
\tab{5}\spadcommand{bubbleSort! u}
\indentrel{3}\begin{verbatim}
   (9)  "0001111111"
                                             Type: Bits
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugPackagesCompilingPageEmpty10}
\begin{paste}{ugPackagesCompilingPageEmpty10}{ugPackagesCompilingPagePatch10}
\pastebutton{ugPackagesCompilingPageEmpty10}{\showpaste}
\tab{5}\spadcommand{bubbleSort! u}
\end{paste}\end{patch}

\begin{patch}{ugPackagesCompilingPagePatch11}
\begin{paste}{ugPackagesCompilingPageFull11}{ugPackagesCompilingPageEmpty11}
\pastebutton{ugPackagesCompilingPageFull11}{\hidepaste}
\tab{5}\spadcommand{t : EqTable(Integer,String) := table()}
\indentrel{3}\begin{verbatim}
   (10)  table()
                            Type: EqTable(Integer,String)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugPackagesCompilingPageEmpty11}
\begin{paste}{ugPackagesCompilingPageEmpty11}{ugPackagesCompilingPagePatch11}
\pastebutton{ugPackagesCompilingPageEmpty11}{\showpaste}
\tab{5}\spadcommand{t : EqTable(Integer,String) := table()}
\end{paste}\end{patch}
```

```
\begin{patch}{ugPackagesCompilingPagePatch12}
\begin{paste}{ugPackagesCompilingPageFull12}{ugPackagesCompilingPageEmpty12}
\pastebutton{ugPackagesCompilingPageFull12}{\hidepaste}
\tab{5}\spadcommand{t.1 := "robert"}
\indentrel{3}\begin{verbatim}
   (11)  "robert"
                                            Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugPackagesCompilingPageEmpty12}
\begin{paste}{ugPackagesCompilingPageEmpty12}{ugPackagesCompilingPagePatch12}
\pastebutton{ugPackagesCompilingPageEmpty12}{\showpaste}
\tab{5}\spadcommand{t.1 := "robert"}
\end{paste}\end{patch}

\begin{patch}{ugPackagesCompilingPagePatch13}
\begin{paste}{ugPackagesCompilingPageFull13}{ugPackagesCompilingPageEmpty13}
\pastebutton{ugPackagesCompilingPageFull13}{\hidepaste}
\tab{5}\spadcommand{t.2 := "richard"}
\indentrel{3}\begin{verbatim}
   (12)  "richard"
                                            Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugPackagesCompilingPageEmpty13}
\begin{paste}{ugPackagesCompilingPageEmpty13}{ugPackagesCompilingPagePatch13}
\pastebutton{ugPackagesCompilingPageEmpty13}{\showpaste}
\tab{5}\spadcommand{t.2 := "richard"}
\end{paste}\end{patch}

\begin{patch}{ugPackagesCompilingPagePatch14}
\begin{paste}{ugPackagesCompilingPageFull14}{ugPackagesCompilingPageEmpty14}
\pastebutton{ugPackagesCompilingPageFull14}{\hidepaste}
\tab{5}\spadcommand{t}
\indentrel{3}\begin{verbatim}
   (13)  table(2= "richard",1= "robert")
                          Type: EqTable(Integer,String)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugPackagesCompilingPageEmpty14}
\begin{paste}{ugPackagesCompilingPageEmpty14}{ugPackagesCompilingPagePatch14}
\pastebutton{ugPackagesCompilingPageEmpty14}{\showpaste}
\tab{5}\spadcommand{t}
\end{paste}\end{patch}

\begin{patch}{ugPackagesCompilingPagePatch15}
\begin{paste}{ugPackagesCompilingPageFull15}{ugPackagesCompilingPageEmpty15}
```

```
\pastebutton{ugPackagesCompilingPageFull15}{\hidepaste}
\tab{5}\spadcommand{bubbleSort! t}
\indentrel{3}\begin{verbatim}
   (14)  table(2= "robert",1= "richard")
                           Type: EqTable(Integer,String)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugPackagesCompilingPageEmpty15}
\begin{paste}{ugPackagesCompilingPageEmpty15}{ugPackagesCompilingPagePatch15}
\pastebutton{ugPackagesCompilingPageEmpty15}{\showpaste}
\tab{5}\spadcommand{bubbleSort! t}
\end{paste}\end{patch}
```

———-

How Packages Work

⇒ "notitle" (ugCategoriesHierPage) 15 on page 2410

— **ug11.ht** —

```
\begin{page}{ugPackagesHowPage}{11.10. How Packages Work}
\beginscroll

Recall that packages as abstract datatypes are compiled independently
and put into the library.  The curious reader may ask: ``How is the
interpreter able to find an operation such as
\fakeAxiomFun{bubbleSort!}?  Also, how is a single compiled function
such as \fakeAxiomFun{bubbleSort!} able to sort data of different
types?''

After the interpreter loads the package \nonLibAxiomType{SortPackage},
the four operations from the package become known to the interpreter.
Each of these operations is expressed as a {\it modemap} in which the
type of the operation is written in terms of symbolic domains.

\nullXtc{
See the modemaps for \fakeAxiomFun{bubbleSort!}.
}{
\spadpaste{)display op bubbleSort!}
}
\begin{verbatim}
There are 2 exposed functions called bubbleSort! :

   [1] D1 -> D1 from SortPackage(D2,D1)
         if D2 has ORDSET and D2 has OBJECT and D1 has
         IndexedAggregate(Integer, D2) with
```

```
              finiteAggregate
              shallowlyMutable

   [2] (D1,((D3,D3) -> Boolean)) -> D1 from SortPackage(D3,D1)
         if D3 has OBJECT and D1 has
         IndexedAggregate(Integer,D3) with
              finiteAggregate
              shallowlyMutable
\end{verbatim}

What happens if you ask for \axiom{bubbleSort!([1,-5,3])}?
There is a unique modemap for an operation named
\fakeAxiomFun{bubbleSort!} with one argument.
Since \axiom{[1,-5,3]} is a list of integers, the symbolic domain
\axiom{D1} is defined as \axiomType{List(Integer)}.
For some operation to apply, it must satisfy the predicate for
some \axiom{D2}.
What \axiom{D2}?
The third expression of the \axiom{and} requires {\tt D1 has
IndexedAggregate(Integer, D2) with} two attributes.
So the interpreter searches for an \axiomType{IndexedAggregate}
among the ancestors of \axiomType{List (Integer)} (see

\downlink{``Hierarchies''}{ugCategoriesHierPage} in Section
12.4\ignore{ugCategoriesHier}).  It finds one:
\axiomType{IndexedAggregate(Integer, Integer)}.  The interpreter tries
defining \axiom{D2} as \axiomType{Integer}.  After substituting for
\axiom{D1} and \axiom{D2}, the predicate evaluates to \axiom{true}.
An applicable operation has been found!

Now Axiom builds the package \axiomType{SortPackage(List(Integer),
Integer)}.  According to its definition, this package exports the
required operation: \fakeAxiomFun{bubbleSort!}: \spadsig{List
Integer}{List Integer}.  The interpreter then asks the package for a
function implementing this operation.  The package gets all the
functions it needs (for example, \axiomFun{rest} and \axiomFunX{swap})
from the appropriate domains and then it returns a
\fakeAxiomFun{bubbleSort!} to the interpreter together with the local
environment for \fakeAxiomFun{bubbleSort!}.  The interpreter applies
the function to the argument \axiom{[1,-5,3]}.  The
\fakeAxiomFun{bubbleSort!} function is executed in its local
environment and produces the result.  \endscroll \autobuttons
\end{page}

\begin{patch}{ugPackagesHowPagePatch1}
\begin{paste}{ugPackagesHowPageFull1}{ugPackagesHowPageEmpty1}
\pastebutton{ugPackagesHowPageFull1}{\hidepaste}
\tab{5}\spadcommand{)display op bubbleSort!}
\indentrel{3}\begin{verbatim}
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugPackagesHowPageEmpty1}
\begin{paste}{ugPackagesHowPageEmpty1}{ugPackagesHowPagePatch1}
\pastebutton{ugPackagesHowPageEmpty1}{\showpaste}
\tab{5}\spadcommand{)display op bubbleSort!}
\end{paste}\end{patch}
```

Chapter 15

Users Guide Chapter 12 (ug12.ht)

Categories

⇐ "Domain Constructors" (ugTypesBasicDomainConsPage) 7 on page 1619
⇒ "Domain Constructors" (ugTypesBasicDomainConsPage) 7 on page 1619
⇒ "Definitions" (ugCategoriesDefsPage) 15 on page 2405
⇒ "Exports" (ugCategoriesExportsPage) 15 on page 2407
⇒ "Documentation" (ugCategoriesDocPage) 15 on page 2408
⇒ "Hierarchies" (ugCategoriesHierPage) 15 on page 2410
⇒ "Membership" (ugCategoriesMembershipPage) 15 on page 2411
⇒ "Defaults" (ugCategoriesDefaultsPage) 15 on page 2412
⇒ "Axioms" (ugCategoriesAxiomsPage) 15 on page 2414
⇒ "Correctness" (ugCategoriesCorrectnessPage) 15 on page 2415
⇒ "Attributes" (ugCategoriesAttributesPage) 15 on page 2416
⇒ "Parameters" (ugCategoriesParametersPage) 15 on page 2419
⇒ "Conditionals" (ugCategoriesConditionalsPage) 15 on page 2420
⇒ "Anonymous Categories" (ugCategoriesAndPackagesPage) 15 on page 2422

— **ug12.ht** —

```
\begin{page}{ugCategoriesPage}{12. Categories}
\beginscroll

This chapter unravels the mysteries of categories---what
they are, how they are related to domains and packages,
how they are defined in Axiom, and how you can extend the
system to include new categories of your own.

We assume that you have read the introductory material on domains and
```

```
categories in
\downlink{``Domain Constructors''}{ugTypesBasicDomainConsPage}
in Section 2.1.1\ignore{ugTypesBasicDomainCons}.  There
you learned that the notion of packages covered in the previous
chapter are special cases of domains.  While this is in fact the case,
it is useful here to regard domains as distinct from packages.

Think of a domain as a datatype, a collection of objects (the objects
of the domain).  From your ``sneak preview'' in the previous chapter,
you might conclude that categories are simply named clusters of
operations exported by domains.  As it turns out, categories have a
much deeper meaning.  Categories are fundamental to the design of
Axiom.  They control the interactions between domains and algorithmic
packages, and, in fact, between all the components of Axiom.

Categories form hierarchies as shown on the inside cover pages of this
book.  The inside front-cover pages illustrate the basic algebraic
hierarchy of the Axiom programming language.  The inside back-cover
pages show the hierarchy for data structures.

Think of the category structures of Axiom as a foundation for a city
on which superstructures (domains) are built.  The algebraic
hierarchy, for example, serves as a foundation for constructive
mathematical algorithms embedded in the domains of Axiom.  Once in
place, domains can be constructed, either independently or from one
another.

Superstructures are built for quality---domains are compiled into
machine code for run-time efficiency.  You can extend the foundation
in directions beyond the space directly beneath the superstructures,
then extend selected superstructures to cover the space.  Because of
the compilation strategy, changing components of the foundation
generally means that the existing superstructures (domains) built on
the changed parts of the foundation (categories) have to be
rebuilt---that is, recompiled.

Before delving into some of the interesting facts about categories,
let's see how you define them in Axiom.

\beginmenu
\menudownlink{{12.1. Definitions}}{ugCategoriesDefsPage}
\menudownlink{{12.2. Exports}}{ugCategoriesExportsPage}
\menudownlink{{12.3. Documentation}}{ugCategoriesDocPage}
\menudownlink{{12.4. Hierarchies}}{ugCategoriesHierPage}
\menudownlink{{12.5. Membership}}{ugCategoriesMembershipPage}
\menudownlink{{12.6. Defaults}}{ugCategoriesDefaultsPage}
\menudownlink{{12.7. Axioms}}{ugCategoriesAxiomsPage}
\menudownlink{{12.8. Correctness}}{ugCategoriesCorrectnessPage}
\menudownlink{{12.9. Attributes}}{ugCategoriesAttributesPage}
\menudownlink{{12.10. Parameters}}{ugCategoriesParametersPage}
```

```
\menudownlink{{12.11. Conditionals}}{ugCategoriesConditionalsPage}
\menudownlink{{12.12. Anonymous Categories}}{ugCategoriesAndPackagesPage}
\endmenu
\endscroll
\autobuttons
\end{page}
```

———-

Definitions

⇐ "Categories" (ugCategoriesPage) 15 on page 2403
⇒ "notitle" (ugTypesPage) 7 on page 1613

— **ug12.ht** —

```
\begin{page}{ugCategoriesDefsPage}{12.1. Definitions}
\beginscroll

A category is defined by a function with exactly the same format as
any other function in Axiom.

\beginImportant
The definition of a category has the syntax:
\centerline{{{\it CategoryForm} : {\tt Category\quad{}==\quad{}}
{\it Extensions} {\tt [ with} {\it Exports} {\tt ]}}}

The brackets {\tt [ ]} here indicate optionality.
\endImportant


The first example of a category definition is
\spadtype{SetCategory},
the most basic of the algebraic categories in Axiom.

\beginImportant

\noindent
{\tt 1.\ \ \ SetCategory():\ Category\ ==}\newline
{\tt 2.\ \ \ \ \ \ Join(Type,CoercibleTo\ OutputForm)\ with}\newline
{\tt 3.\ \ \ \ \ \ \ \ \ "="\ :\ (\$,\ \$)\ ->\ Boolean}\newline
\endImportant

The definition starts off with the name of the
category (\spadtype{SetCategory}); this is
always in column one in the source file.
%% maybe talk about naming conventions for source files? .spad or .ax?
```

```
All parts of a category definition are then indented with respect to this
first line.

In \downlink{``Using Types and Modes''}{ugTypesPage} in Chapter
2\ignore{ugTypes}, we talked about \spadtype{Ring} as
denoting the class of all domains that are rings, in short, the class
of all rings.  While this is the usual naming convention in Axiom, it
is also common to use the word ``Category'' at the end of a category
name for clarity.  The interpretation of the name
\spadtype{SetCategory} is, then, ``the category of all domains that
are (mathematical) sets.''

The name \spadtype{SetCategory} is followed in the definition by its
formal parameters enclosed in parentheses \spadSyntax{()}.
Here there are no parameters.
As required, the type of the result of this category function is the
distinguished name {\sf Category}.

Then comes the \spadSyntax{==}.
As usual, what appears to the right of the \spadSyntax{==} is a
definition, here, a category definition.
A category definition always has two parts separated by the reserved word
\spadkey{with}
\spad{with}.
%\footnote{Debugging hint: it is very easy to forget
%the \spad{with}!}

The first part tells what categories the category extends.
Here, the category extends two categories: \spadtype{Type}, the
category of all domains, and
\spadtype{CoercibleTo(OutputForm)}.
%\footnote{\spadtype{CoercibleTo(OutputForm)}
%can also be written (and is written in the definition above) without
%parentheses.}
The operation \spad{Join} is a system-defined operation that
\spadkey{Join}
forms a single category from two or more other categories.

Every category other than \spadtype{Type} is an extension of some other
category.
If, for example, \spadtype{SetCategory} extended only the category
\spadtype{Type}, the definition here would read ``{\tt Type with
...}''.
In fact, the {\tt Type} is optional in this line; ``{\tt with
...}'' suffices.

\endscroll
\autobuttons
\end{page}
```

———-

Exports

⇐ "Categories" (ugCategoriesPage) 15 on page 2403

— ug12.ht —

```
\begin{page}{ugCategoriesExportsPage}{12.2. Exports}
\beginscroll

To the right of the \spad{with} is a list of
\spadkey{with}
all the \spadglossSee{exports}{export} of the category.
Each exported operation has a name and a type expressed by a
\spadgloss{declaration} of the form
``{\frenchspacing\tt {\it name}: {\it type}}''.

Categories can export symbols, as well as
{\tt 0} and {\tt 1} which denote
domain constants.\footnote{The
numbers {\tt 0} and {\tt 1} are operation names in Axiom.}
In the current implementation, all other exports are operations with
types expressed as \spadglossSee{mappings}{mapping} with the syntax
\centerline{{{\it}}
\centerline{{source\quad{\tt ->}\quad target}}
\centerline{{}}}

The category \spadtype{SetCategory} has a single export: the operation
\spadop{=} whose type is given by the mapping {\tt (\$, \$) -> Boolean}.
The \spadSyntax{\$} in a mapping type always means ``the domain.'' Thus
the operation \spadop{=} takes two arguments from the domain and
returns a value of type \spadtype{Boolean}.

The source part of the mapping here is given by a {\it tuple}
consisting of two or more types separated by commas and enclosed in
parentheses.
If an operation takes only one argument, you can drop the parentheses
around the source type.
If the mapping has no arguments, the source part of the mapping is either
left blank or written as \spadSyntax{()}.
Here are examples of formats of various operations with some
contrived names.

\begin{verbatim}
someIntegerConstant  :    $
aZeroArgumentOperation:   () -> Integer
aOneArgumentOperation:    Integer -> $
aTwoArgumentOperation:    (Integer,$) -> Void
```

```
aThreeArgumentOperation:  ($,Integer,$) -> Fraction($)
\end{verbatim}

\endscroll
\autobuttons
\end{page}
```

———-

Documentation

⇐ “Categories” (ugCategoriesPage) 15 on page 2403

— **ug12.ht** —

```
\begin{page}{ugCategoriesDocPage}{12.3. Documentation}
\beginscroll

The definition of \spadtype{SetCategory} above is  missing
an important component: its library documentation.
Here is its definition, complete with documentation.

\beginImportant

\noindent
{\tt 1.\ \ \ ++\ Description:}\newline
{\tt 2.\ \ \ ++\
\bs{}axiomType\{SetCategory\}\ is\ the\ basic\ category}\newline
{\tt 3.\ \ \ ++\
for\ describing\ a\ collection\ of\ elements\ with}\newline
{\tt 4.\ \ \ ++\
\bs{}axiomOp\{=\}\ (equality)\ and\ a\ \bs{}axiomFun\{coerce\}}\newline
{\tt 5.\ \ \ ++\ to\ \bs{}axiomType\{OutputForm\}.}\newline
{\tt 6.\ \ \ }\newline
{\tt 7.\ \ \ SetCategory():\ Category\ ==}\newline
{\tt 8.\ \ \ \ \ Join(Type,\ CoercibleTo\ OutputForm)\ with}\newline
{\tt 9.\ \ \ \ \ \ \ "=":\ (\$,\ \$)\ ->\ Boolean}\newline
{\tt 10.\ \ \ \ \ \ \ \ ++\
\bs{}axiom\{x\ =\ y\}\ tests\ if\ \bs{}axiom\{x\}\ and}\newline
{\tt 11.\ \ \ \ \ \ \ \ ++\ \bs{}axiom\{y\}\ are\ equal.}\newline
\endImportant

Documentary comments are an important part of constructor definitions.
Documentation is given both for the category itself and for
each export.
A description for the category precedes the code.
Each line of the description begins in column one with \axiomSyntax{++}.
The description starts with the word {\tt Description:}.\footnote{Other
```

```
information such as the author's name, date of creation, and so on,
can go in this
area as well but are currently ignored by Axiom.}
All lines of the description following the initial line are
indented by the same amount.

{\texht{\sloppy}{} Surround the name of any constructor (with or
without parameters) with an
\texht{\verb+\axiomType{}+}{\\axiomType\{\}}.  Similarly, surround an
operator name with \texht{\verb+\axiomOp{}+}{\\axiomOp\{\}}, an Axiom
operation with \texht{\verb+\axiomFun{}+}{\\axiomFun\{\}}, and a
variable or Axiom expression with
\texht{\verb+\axiom{}+}{\\axiom\{\}}.  Library documentation is given
in a \TeX{}-like language so that it can be used both for hard-copy
and for \Browse{}.  These different wrappings cause operations and
types to have mouse-active buttons in \Browse{}.  For hard-copy
output, wrapped expressions appear in a different font.  The above
documentation appears in hard-copy as:

}
%
\texht{\begin{quotation}}{\indent{3}}
%
\axiomType{SetCategory} is the basic category
for describing a collection of elements with \axiomOp{=}
(equality) and a \spadfun{coerce} to \axiomType{OutputForm}.
%
\texht{\end{quotation}}{\indent{0}}
%
and
%
\texht{\begin{quotation}}{\indent{3}}
%
\axiom{x = y} tests if \axiom{x} and \axiom{y} are equal.
%
\texht{\end{quotation}}{\indent{0}}
%

For our purposes in this chapter, we omit the documentation from further
category descriptions.

\endscroll
\autobuttons
\end{page}
```

Hierarchies

⇐ “Categories” (ugCategoriesPage) 15 on page 2403

— **ug12.ht** —

```
\begin{page}{ugCategoriesHierPage}{12.4. Hierarchies}
\beginscroll

A second example of a category is
\spadtype{SemiGroup}, defined by:

\beginImportant

\noindent
{\tt 1.\ \ \ SemiGroup():\ Category\ ==\ SetCategory\ with}\newline
{\tt 2.\ \ \ \ \ \ \ \ \ "*":\ \ (\$,\$)\ ->\ \$}\newline
{\tt 3.\ \ \ \ \ \ \ \ \ "**":\ (\$,\ PositiveInteger)\ ->\ \$}\newline
\endImportant

This definition is as simple as that for \spadtype{SetCategory},
except that there are two exported operations.
Multiple exported operations are written as a \spadgloss{pile},
that is, they all begin in the same column.
Here you see that the category mentions another type,
\spadtype{PositiveInteger}, in a signature.
Any domain can be used in a signature.

Since categories extend one another, they form hierarchies.
Each category other than \spadtype{Type} has one or more parents given
by the one or more categories mentioned before the \spad{with} part of
the definition.
\spadtype{SemiGroup} extends \spadtype{SetCategory} and
\spadtype{SetCategory} extends both \spadtype{Type} and
\spadtype{CoercibleTo (OutputForm)}.
Since \spadtype{CoercibleTo (OutputForm)} also extends \spadtype{Type},
the mention of \spadtype{Type} in the definition is unnecessary but
included for emphasis.

\endscroll
\autobuttons
\end{page}
```

———

Membership

⇐ “Categories” (ugCategoriesPage) 15 on page 2403

— **ug12.ht** —

```
\begin{page}{ugCategoriesMembershipPage}{12.5. Membership}
\beginscroll

We say a category designates a class of domains.
What class of domains?
That is, how does Axiom know what domains belong to what categories?
The simple answer to this basic question is key to the design of
Axiom:

\beginImportant
\centerline{{\bf Domains belong to categories by assertion.}}
\endImportant

When a domain is defined, it is asserted to belong to one or more
categories.
Suppose, for example, that an author of domain \spadtype{String} wishes to
use the binary operator \spadop{*} to denote concatenation.
Thus \spad{"hello " * "there"} would produce the string
\spad{"hello there"}\footnote{Actually, concatenation of strings in
Axiom is done by juxtaposition or by using the operation
\spadfunFrom{concat}{String}.
The expression \spad{"hello " "there"} produces the string
\spad{"hello there"}.}.
The author of \spadtype{String} could then assert that \spadtype{String}
is a member of \spadtype{SemiGroup}.
According to our definition of \spadtype{SemiGroup}, strings
would then also have the operation \spadop{**} defined automatically.
Then \spad{"--" ** 4} would produce a string of eight dashes
\spad{"--------"}.
Since \spadtype{String} is a member of \spadtype{SemiGroup}, it also is
a member of \spadtype{SetCategory} and thus has an operation
\spadop{=} for testing that two strings are equal.

\texht{Now turn to the algebraic category hierarchy inside the
front cover of this book.}{}
Any domain that is a member of a
category extending \spadtype{SemiGroup} is a member of
\spadtype{SemiGroup} (that is, it {\it is} a semigroup).
In particular, any domain asserted to be a \spadtype{Ring} is a
semigroup since \spadtype{Ring} extends \spadtype{Monoid}, that,
in turn, extends \spadtype{SemiGroup}.
The definition of \spadtype{Integer} in Axiom asserts that
\spadtype{Integer} is a member of category
\spadtype{IntegerNumberSystem}, that, in turn, asserts that it is
```

```
a member of \spadtype{EuclideanDomain}.
Now \spadtype{EuclideanDomain} extends
\spadtype{PrincipalIdealDomain} and so on.
If you trace up the hierarchy, you see that
\spadtype{EuclideanDomain} extends \spadtype{Ring}, and,
therefore, \spadtype{SemiGroup}.
Thus \spadtype{Integer} is a semigroup and also exports the
operations \spadop{*} and \spadop{**}.

\endscroll
\autobuttons
\end{page}
```

———-

Defaults

⇐ “Categories” (ugCategoriesPage) 15 on page 2403
⇒ “Hierarchies” (ugCategoriesHierPage) 15 on page 2410
⇒ “Packages” (ugPackagesPage) 14 on page 2375

— **ug12.ht** —

```
\begin{page}{ugCategoriesDefaultsPage}{12.6. Defaults}
\beginscroll

We actually omitted the last
part of the definition of
\spadtype{SemiGroup} in
\downlink{‘‘Hierarchies’’}{ugCategoriesHierPage}
in Section 12.4\ignore{ugCategoriesHier}.
Here now is its complete Axiom definition.

\beginImportant

\noindent
{\tt 1.\ \ \ SemiGroup():\ Category\ ==\ SetCategory\ with}\newline
{\tt 2.\ \ \ \ \ \ \ \ \ "*":\ (\$,\ \$)\ ->\ \$}\newline
{\tt 3.\ \ \ \ \ \ \ \ \ "**":\ (\$,\ PositiveInteger)\ ->\ \$}\newline
{\tt 4.\ \ \ \ \ \ \ add}\newline
{\tt 5.\ \ \ \ \ \ \ \ \ import\ RepeatedSquaring(\$)}\newline
{\tt 6.\ \ \ \ \ \ \ \ \
x:\ \$\ **\ n:\ PositiveInteger\ ==\ expt(x,n)}\newline
\endImportant

The \spad{add} part at the end is used to give ‘‘default definitions’’
for \spadkey{add} exported operations.  Once you have a multiplication
```

```
operation \spadop{*}, you can define exponentiation for positive
integer exponents using repeated multiplication:
\centerline{{\spad{x ** n = x * x * ... * x}
( \spad{n} copies of \spad{x} )}} This
definition for \spadop{**} is called a {\it default} definition.  In
general, a category can give default definitions for any operation it
exports.  Since \spadtype{SemiGroup} and all its category descendants
in the hierarchy export \spadop{**}, any descendant category may
redefine \spadop{**} as well.

A domain of category \spadtype{SemiGroup} (such as \spadtype{Integer})
may or may not choose to define its own \spadop{**} operation.  If it
does not, a default definition that is closest (in a ``tree-distance''
sense of the hierarchy) to the domain is chosen.

The part of the category definition following an \spadop{add} operation
is a \spadgloss{capsule}, as discussed in
\texht{the previous chapter.}
{\downlink{``Packages''}{ugPackagesPage}
in Chapter 11\ignore{ugPackages}.}
The line
\begin{verbatim}
import RepeatedSquaring($)
\end{verbatim}
references the package
\spadtype{RepeatedSquaring(\$)}, that is, the package
\spadtype{RepeatedSquaring} that takes ``this domain'' as its
parameter.
For example, if the semigroup \spadtype{Polynomial (Integer)}
does not define its own exponentiation operation, the
definition used may come from the package
\spadtype{RepeatedSquaring (Polynomial (Integer))}.
The next line gives the definition in terms of \spadfun{expt} from that
package.

The default definitions are collected to form a ``default package''
for the category.  The name of the package is the same as the category
but with an ampersand (\spadSyntax{\&}) added at the end.  A default
package always takes an additional argument relative to the category.
Here is the definition of the default package \pspadtype{SemiGroup\&}
as automatically generated by Axiom from the above definition of
\spadtype{SemiGroup}.

\beginImportant

\noindent
{\tt 1.\ \ \ SemiGroup\_\&(\$):\ Exports\ ==\ Implementation\ where}
\newline
{\tt 2.\ \ \ \ \ \$:\ SemiGroup}\newline
{\tt 3.\ \ \ \ \ Exports\ ==\ with}\newline
```

```
{\tt 4.\ \ \ \ \ \ \ "**":\ (\$,\ PositiveInteger)\ ->\ \$}\newline
{\tt 5.\ \ \ \ \ Implementation\ ==\ add}\newline
{\tt 6.\ \ \ \ \ \ \ import\ RepeatedSquaring(\$)}\newline
{\tt 7.\ \ \ \ \ \ \ x:\$\ **\ n:PositiveInteger\ ==\ expt(x,n)}\newline
\endImportant

\endscroll
\autobuttons
\end{page}
```

———-

Axioms

⇐ “Categories” (ugCategoriesPage) 15 on page 2403
⇒ “Defaults” (ugCategoriesDefaultsPage) 15 on page 2412

— **ug12.ht** —

```
\begin{page}{ugCategoriesAxiomsPage}{12.7. Axioms}
\beginscroll

In \texht{the previous section}
{\downlink{‘‘Defaults’’}{ugCategoriesDefaultsPage}
in Section 12.6\ignore{ugCategoriesDefaults}}
you saw the complete Axiom program defining \index{axiom}
\spadtype{SemiGroup}.  According to this definition, semigroups (that
is, are sets with the operations \spadopFrom{*}{SemiGroup} and
\spadopFrom{**}{SemiGroup}.

You might ask: ‘‘Aside from the notion of default packages, isn’t a
category just a \spadgloss{macro}, that is, a shorthand equivalent to
the two operations \spadop{*} and \spadop{**} with their types?’’ If a
category were a macro, every time you saw the word
\spadtype{SemiGroup}, you would rewrite it by its list of exported
operations.  Furthermore, every time you saw the exported operations
of \spadtype{SemiGroup} among the exports of a constructor, you could
conclude that the constructor exported \spadtype{SemiGroup}.

A category is {\it not} a macro and here is why.
The definition for \spadtype{SemiGroup} has documentation that states:

\texht{\begin{quotation}}{\indent{3}}
    Category \spadtype{SemiGroup} denotes the class of all multiplicative
    semigroups, that is, a set with an associative operation \spadop{*}.

    \texht{\vskip .5\baselineskip}{}
```

```
    {Axioms:}

    {\small\tt associative("*" : (\$,\$)->\$) -- (x*y)*z = x*(y*z)}
\texht{\end{quotation}}{\indent{3}}

According to the author's remarks, the mere exporting of an operation
named \spadop{*} and \spadop{**} is not enough to qualify the domain
as a \spadtype{SemiGroup}.  In fact, a domain can be a semigroup only
if it explicitly exports a \spadop{**} and a \spadop{*} satisfying the
associativity axiom.

In general, a category name implies a set of axioms, even mathematical
theorems.  There are numerous axioms from \spadtype{Ring}, for
example, that are well-understood from the literature.  No attempt is
made to list them all.  Nonetheless, all such mathematical facts are
implicit by the use of the name \spadtype{Ring}.

\endscroll
\autobuttons
\end{page}
```

Correctness

⇐ "Categories" (ugCategoriesPage) 15 on page 2403

— ug12.ht —

```
\begin{page}{ugCategoriesCorrectnessPage}{12.8. Correctness}
\beginscroll

While such statements are only comments,
Axiom can enforce their intention simply by shifting the burden of
responsibility onto the author of a domain.
A domain belongs to category \spad{Ring} only if the
author asserts that the domain  belongs to \spadtype{Ring} or
to a category that extends \spadtype{Ring}.

This principle of assertion is important for large user-extendable
systems.
Axiom has a large library of operations offering facilities in
many areas.
Names such as \spadfun{norm} and \spadfun{product}, for example, have
diverse meanings in diverse contexts.
An inescapable hindrance to users would be to force those who wish to
extend Axiom to always invent new names for operations.
%>> I don't think disambiguate is really a word, though I like it
```

```
Axiom allows you to reuse names, and then use context to disambiguate one
from another.

Here is another example of why this is important.  Some languages,
such as {\bf APL}, denote the \spadtype{Boolean} constants \spad{true}
and \spad{false} by the integers \spad{1} and \spad{0}.  You may want
to let infix operators \spadop{+} and \spadop{*} serve as the logical
operators \spadfun{or} and \spadfun{and}, respectively.  But note
this: \spadtype{Boolean} is not a ring.  The {\it inverse axiom} for
\spadtype{Ring} states:

\centerline{
{Every element \spad{x} has an additive inverse \spad{y} such that}}
\centerline{{\spad{x + y = 0}.}}
%
\spadtype{Boolean} is not a ring since \spad{true} has
no inverse---there is no inverse element \spad{a} such that
\spad{1 + a = 0} (in terms of booleans, \spad{(true or a) = false}).
Nonetheless, Axiom {\it could} easily and correctly implement
\spadtype{Boolean} this way.
\spadtype{Boolean} simply would not assert that it is of category
\spadtype{Ring}.
Thus the \spadop{+} for \spadtype{Boolean} values
is not confused with the one for \spadtype{Ring}.
Since the \spadtype{Polynomial} constructor requires its argument
to be a ring, Axiom would then refuse to build the
domain \spadtype{Polynomial(Boolean)}. Also, Axiom would refuse to
wrongfully apply algorithms to \spadtype{Boolean} elements that
presume that the ring axioms for \spadop{+} hold.

\endscroll
\autobuttons
\end{page}
```

Attributes

⇐ "Categories" (ugCategoriesPage) 15 on page 2403
⇒ "Category Assertions" (ugDomainsAssertionsPage) 16 on page 2429

— **ug12.ht** —

```
\begin{page}{ugCategoriesAttributesPage}{12.9. Attributes}
\beginscroll

Most axioms are not computationally useful.
```

```
Those that are can be explicitly expressed by what Axiom calls an
\spadgloss{attribute}.
The attribute \spadatt{commutative("*")}, for example, is used to assert
that a domain has commutative multiplication.
Its definition is given by its documentation:

\texht{\begingroup \parindent=1pc \narrower\noindent}{\indent{3}}%
    A domain \spad{R} has \spadatt{commutative("*")}
    if it has an operation "*": \spadsig{(R,R)}{R}
such that \spad{x * y = y * x}.
\texht{\par\endgroup}{\indent{0}}

Just as you can test whether a domain has the category \spadtype{Ring}, you
can test that a domain has a given attribute.

\xtc{
Do polynomials over the integers
have commutative multiplication?
}{
\spadpaste{Polynomial Integer has commutative("*")}
}
\xtc{
Do matrices over the integers
have commutative multiplication?
}{
\spadpaste{Matrix Integer has commutative("*")}
}

Attributes are used to conditionally export and define
operations for a domain (see
\downlink{``Category Assertions''}{ugDomainsAssertionsPage} in
Section 13.3\ignore{ugDomainsAssertions}).
Attributes can also be asserted in a category definition.

After mentioning category \spadtype{Ring} many times in this book,
it is high time that we show you its definition:

\beginImportant

\noindent
{\tt 1.\ \ \ Ring():\ Category\ ==}\newline
{\tt 2.\ \ \ \ \ Join(Rng,Monoid,LeftModule(\$:\ Rng))\ with}\newline
{\tt 3.\ \ \ \ \ \ \ \ \ characteristic:\ ->\ NonNegativeInteger}\newline
{\tt 4.\ \ \ \ \ \ \ \ \ coerce:\ Integer\ ->\ \$}\newline
{\tt 5.\ \ \ \ \ \ \ \ \ unitsKnown}\newline
{\tt 6.\ \ \ \ \ \ \ add}\newline
{\tt 7.\ \ \ \ \ \ \ \ \ n:Integer}\newline
{\tt 8.\ \ \ \ \ \ \ \ \ coerce(n)\ ==\ n\ *\ 1\$\$}\newline
\endImportant
```

```
There are only two new things here.  First, look at the
\spadSyntax{\$\$} on the last line.  This is not a typographic error!
The first \spadSyntax{\$} says that the \spad{1} is to come from some
domain.  The second \spadSyntax{\$} says that the domain is ``this
domain.''  If \spadSyntax{\$} is \spadtype{Fraction(Integer)}, this
line reads {\tt coerce(n) == n * 1\$Fraction(Integer)}.

The second new thing is the presence of attribute
``\spad{unitsKnown}''.  Axiom can always distinguish an attribute from
an operation.  An operation has a name and a type. An attribute has no
type.  The attribute \spadatt{unitsKnown} asserts a rather subtle
mathematical fact that is normally taken for granted when working with
rings.\footnote{With this axiom, the units of a domain are the set of
elements \spad{x} that each have a multiplicative inverse \spad{y} in
the domain.  Thus \spad{1} and \spad{-1} are units in domain
\spadtype{Integer}.  Also, for \spadtype{Fraction Integer}, the domain
of rational numbers, all non-zero elements are units.}  Because
programs can test for this attribute, Axiom can correctly handle
rather more complicated mathematical structures (ones that are similar
to rings but do not have this attribute).

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugCategoriesAttributesPagePatch1}
\begin{paste}{ugCategoriesAttributesPageFull1}{ugCategoriesAttributesPageEmpty1}
\pastebutton{ugCategoriesAttributesPageFull1}{\hidepaste}
\tab{5}\spadcommand{Polynomial Integer has commutative("*")}
\indentrel{3}\begin{verbatim}
   (1)  true
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugCategoriesAttributesPageEmpty1}
\begin{paste}{ugCategoriesAttributesPageEmpty1}{ugCategoriesAttributesPagePatch1}
\pastebutton{ugCategoriesAttributesPageEmpty1}{\showpaste}
\tab{5}\spadcommand{Polynomial Integer has commutative("*")}
\end{paste}\end{patch}

\begin{patch}{ugCategoriesAttributesPagePatch2}
\begin{paste}{ugCategoriesAttributesPageFull2}{ugCategoriesAttributesPageEmpty2}
\pastebutton{ugCategoriesAttributesPageFull2}{\hidepaste}
\tab{5}\spadcommand{Matrix Integer has commutative("*")}
\indentrel{3}\begin{verbatim}
   (2)  false
                                          Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugCategoriesAttributesPageEmpty2}
\begin{paste}{ugCategoriesAttributesPageEmpty2}{ugCategoriesAttributesPagePatch2}
\pastebutton{ugCategoriesAttributesPageEmpty2}{\showpaste}
\tab{5}\spadcommand{Matrix Integer has commutative("*")}
\end{paste}\end{patch}
```

———-

Parameters

⇐ "Categories" (ugCategoriesPage) 15 on page 2403

— **ug12.ht** —

```
\begin{page}{ugCategoriesParametersPage}{12.10. Parameters}
\beginscroll

Like domain constructors, category constructors can also have
parameters.  For example, category \spadtype{MatrixCategory} is a
parameterized category for defining matrices over a ring \spad{R} so
that the matrix domains can have different representations and
indexing schemes.  Its definition has the form:

\beginImportant

\noindent
{\tt 1.\ \ \ MatrixCategory(R,Row,Col):\ Category\ ==}\newline
{\tt 2.\ \ \ \ \ \ \ TwoDimensionalArrayCategory(R,Row,Col)\ with\ ...}\newline
\endImportant

The category extends \spadtype{TwoDimensionalArrayCategory} with the
same arguments.  You cannot find
\spadtype{TwoDimensionalArrayCategory} in the algebraic hierarchy
listing.  Rather, it is a member of the data structure hierarchy,
given inside the back cover of this book.  In particular,
\spadtype{TwoDimensionalArrayCategory} is an extension of
\spadtype{HomogeneousAggregate} since its elements are all one type.

The domain \spadtype{Matrix(R)}, the class of matrices with coefficients
from domain \spad{R}, asserts that it is a member of category
\spadtype{MatrixCategory(R, Vector(R), Vector(R))}.
The parameters of a category must also have types.
The first parameter to \spadtype{MatrixCategory}
\spad{R} is required to be a ring.
The second and third are required to be domains of category
\spadtype{FiniteLinearAggregate(R)}.\footnote{%
This is another extension of
```

```
\spadtype{HomogeneousAggregate} that you can see in
the data structure hierarchy.}
In practice, examples of categories having parameters other than
domains are rare.

Adding the declarations for parameters to the definition for
\spadtype{MatrixCategory}, we have:

\beginImportant

\noindent
{\tt 1.\ \ \ R:\ Ring}\newline
{\tt 2.\ \ \ (Row,\ Col):\ FiniteLinearAggregate(R)}\newline
{\tt 3.\ \ \ }\newline
{\tt 4.\ \ \ MatrixCategory(R,\ Row,\ Col):\ Category\ ==}\newline
{\tt 5.\ \ \ \ \ \ \
TwoDimensionalArrayCategory(R,\ Row,\ Col)\ with\ ...}\newline
\endImportant

\endscroll
\autobuttons
\end{page}
```

———-

Conditionals

⇐ "Categories" (ugCategoriesPage) 15 on page 2403
⇒ "Conditionals" (ugPackagesCondsPage) 14 on page 2390

— **ug12.ht** —

```
\begin{page}{ugCategoriesConditionalsPage}{12.11. Conditionals}
\beginscroll

As categories have parameters, the actual operations exported by a
category can depend on these parameters.
As an example, the operation \spadfunFrom{determinant}{MatrixCategory}
from category \spadtype{MatrixCategory} is only exported when the
underlying domain \spad{R} has commutative multiplication:

\begin{verbatim}
if R has commutative("*") then
   determinant: $ -> R
\end{verbatim}

Conditionals can also define conditional extensions of a category.
```

Here is a portion of the definition of \spadtype{QuotientFieldCategory}:

\beginImportant

\noindent
{\tt 1.\ \ \ QuotientFieldCategory(R)\ :\ Category\ ==\ ...\ with\ ...}\newline
{\tt 2.\ \ \ \ \ \ \ \ if\ R\ has\ OrderedSet\ then\ OrderedSet}\newline
{\tt 3.\ \ \ \ \ \ \ \ if\ R\ has\ IntegerNumberSystem\ then}\newline
{\tt 4.\ \ \ \ \ \ \ \ \ \ ceiling:\ \$\ ->\ R}\newline
{\tt 5.\ \ \ \ \ \ \ \ \ \ \ \ ...}\newline
\endImportant

Think of category \spadtype{QuotientFieldCategory(R)} as
denoting the domain \spadtype{Fraction(R)}, the
class of all fractions of the form \smath{a/b} for elements of \spad{R}.
The first conditional means in English:
''If the elements of \spad{R} are totally ordered (\spad{R}
is an \spadtype{OrderedSet}), then so are the fractions \smath{a/b}''.

The second conditional is used to conditionally export an
operation \spadfun{ceiling} which returns the smallest integer
greater than or equal to its argument.
Clearly, ''ceiling'' makes sense for integers but not for
polynomials and other algebraic structures.
Because of this conditional,
the domain \spadtype{Fraction(Integer)} exports
an operation
\spadfun{ceiling}: \spadsig{Fraction Integer}{Integer}, but
\spadtype{Fraction Polynomial Integer} does not.

Conditionals can also appear in the default definitions for the
operations of a category.
For example, a default definition for \spadfunFrom{ceiling}{Field}
within the part following the \spadop{add} reads:

```
\begin{verbatim}
if R has IntegerNumberSystem then
    ceiling x == ...
\end{verbatim}
```

Here the predicate used is identical to the predicate in the {\tt
Exports} part. This need not be the case. See
\downlink{''Conditionals''}{ugPackagesCondsPage} in Section
11.8\ignore{ugPackagesConds} for a more complicated
example.

\endscroll
\autobuttons
\end{page}

———-

Anonymous Categories

⇐ "Categories" (ugCategoriesPage) 15 on page 2403
⇒ "Abstract Datatypes" (ugPackagesAbstractPage) 14 on page 2380

— **ug12.ht** —

```
\begin{page}{ugCategoriesAndPackagesPage}{12.12. Anonymous Categories}
\beginscroll

The part of a category to the right of a {\tt with} is also regarded
as a category---an ``anonymous category.''  Thus you have already seen
a category definition The {\tt Exports} part of the package
\pspadtype{DrawComplex}
(\downlink{``Abstract Datatypes''}{ugPackagesAbstractPage} in
Section 11.3\ignore{ugPackagesAbstract}) is an
anonymous category.  This is not necessary.  We could, instead, give
this category a name:

%
\beginImportant

\noindent
{\tt 1.\ \ \ DrawComplexCategory():\ Category\ ==\ with}\newline
{\tt 2.\ \ \ \ \ \
drawComplex:\ (C\ ->\ C,S,S,Boolean)\ ->\ VIEW3D}\newline
{\tt 3.\ \ \ \ \ \
drawComplexVectorField:\ (C\ ->\ C,S,S)\ ->\ VIEW3D}\newline
{\tt 4.\ \ \ \ \ \ setRealSteps:\ INT\ ->\ INT}\newline
{\tt 5.\ \ \ \ \ \ setImagSteps:\ INT\ ->\ INT}\newline
{\tt 6.\ \ \ \ \ \ setClipValue:\ DFLOAT->\ DFLOAT}\newline
\endImportant
%
and then define \spadtype{DrawComplex} by:
%
\beginImportant

\noindent
{\tt 1.\ \ \ DrawComplex():\ DrawComplexCategory\ ==\ Implementation}
\newline
{\tt 2.\ \ \ \ \ \ where}\newline
{\tt 3.\ \ \ \ \ \ \ \ \ ...}\newline
\endImportant
%

There is no reason, however, to give this list of exports a name
since no other domain or package exports it.
```

```
In fact, it is rare for a package to export a named category.
As you will see in the next chapter, however, it is very common
for the definition of domains to mention one or more category
before the {\tt with}.
\spadkey{with}
\endscroll
\autobuttons
\end{page}
```

Chapter 16

Users Guide Chapter 13 (ug13.ht)

Domains

— ug13.ht —

```
\begin{page}{ugDomainsPage}{13. Domains}
\beginscroll

We finally come to the \spadgloss{domain constructor}.  A few subtle
differences between packages and domains turn up some interesting
issues.  We first discuss these differences then describe the
resulting issues by illustrating a program for the
\axiomType{QuadraticForm} constructor.  After a short example of an
algebraic constructor, \axiomType{CliffordAlgebra}, we show how you
use domain constructors to build a database query facility.
```

```
\beginmenu
\menudownlink{{13.1. Domains vs. Packages}}{ugPackagesDomsPage}
\menudownlink{{13.2. Definitions}}{ugDomainsDefsPage}
\menudownlink{{13.3. Category Assertions}}{ugDomainsAssertionsPage}
\menudownlink{{13.4. A Demo}}{ugDomainsDemoPage}
\menudownlink{{13.5. Browse}}{ugDomainsBrowsePage}
\menudownlink{{13.6. Representation}}{ugDomainsRepPage}
\menudownlink{{13.7. Multiple Representations}}{ugDomainsMultipleRepsPage}
\menudownlink{{13.8. Add Domain}}{ugDomainsAddDomainPage}
\menudownlink{{13.9. Defaults}}{ugDomainsDefaultsPage}
\menudownlink{{13.10. Origins}}{ugDomainsOriginsPage}
\menudownlink{{13.11. Short Forms}}{ugDomainsShortFormsPage}
\menudownlink{{13.12. Example 1: Clifford Algebra}}{ugDomainsCliffordPage}
\menudownlink{{13.13. Example 2: Building A Query Facility}}
{ugDomsinsDatabasePage}
\endmenu
\endscroll
\autobuttons
\end{page}
```

———-

Domains vs. Packages

⇒ "notitle" (ugPackagesPage) 14 on page 2375
— **ug13.ht** —

```
\begin{page}{ugPackagesDomsPage}{13.1. Domains vs. Packages}
\beginscroll

Packages are special cases of domains.  What is the difference between
a package and a domain that is not a package?  By definition, there is
only one difference: a domain that is not a package has the symbol
\axiomSyntax{\$} appearing somewhere among the types of its exported
operations.  The \axiomSyntax{\$} denotes ``this domain.'' If the
\axiomSyntax{\$} appears before the \axiomSyntax{->} in the type of a
signature, it means the operation takes an element from the domain as
an argument.  If it appears after the \axiomSyntax{->}, then the
operation returns an element of the domain.

If no exported operations mention \axiomSyntax{\$}, then evidently
there is nothing of interest to do with the objects of the domain.
You might then say that a package is a ``boring'' domain!  But, as you
saw in \downlink{``Packages''}{ugPackagesPage} in Chapter
11\ignore{ugPackages}, packages are a very useful
notion indeed.  The exported operations of a package depend solely on
```

```
the parameters to the package constructor and other explicit domains.

To summarize, domain constructors are versatile structures that serve
two distinct practical purposes: Those like \axiomType{Polynomial} and
\axiomType{List} describe classes of computational objects; others,
like \pspadtype{SortPackage}, describe packages of useful operations.
As in the last chapter, we focus here on the first kind.

\endscroll
\autobuttons
\end{page}
```

———-

Definitions

— ug13.ht —

```
\begin{page}{ugDomainsDefsPage}{13.2. Definitions}
\beginscroll
%

The syntax for defining a domain constructor is the same as for any
function in Axiom:
\centerline{{\frenchspacing{\tt {\it DomainForm} : {\it Exports} ==
{\it Implementation}}}}
As this definition usually extends over many lines, a
\axiom{where} expression is generally used instead.
\spadkey{where}

\beginImportant
A recommended format for the definition of a domain is:\newline
{\tt%
{\it DomainForm} : Exports  ==  Implementation where \newline
\texht{\hspace*{.75pc}}{\tab{8}}
{\it optional type declarations} \newline
\texht{\hspace*{.75pc}}{\tab{3}}
Exports  ==  [{\it Category Assertions}] with \newline
\texht{\hspace*{2.0pc}}{\tab{8}}
{\it list of exported operations} \newline
\texht{\hspace*{.75pc}}{\tab{3}}
Implementation  ==  [{\it Add Domain}] add \newline
\texht{\hspace*{2.0pc}}{\tab{6}}
[Rep := {\it Representation}] \newline
\texht{\hspace*{2.0pc}}{\tab{8}}
{\it list of function definitions for exported operations}
}
```

```
\texht{\vskip 4pt}{}
Note: The brackets {\tt [ ]} here denote optionality.
\endImportant

A complete domain constructor definition for
\axiomType{QuadraticForm} is shown in Figure \ref{fig-quadform}.
Interestingly, this little domain illustrates all the new
concepts you need to learn.

\beginImportant

\noindent
{\tt 1.\ \ \ )abbrev\ domain\ QFORM\ QuadraticForm}\newline
{\tt 2.\ \ \ }\newline
{\tt 3.\ \ \ ++\ Description:}\newline
{\tt 4.\ \ \ ++\ \ \
This\ domain\ provides\ modest\ support\ for}\newline
{\tt 5.\ \ \ ++\ \ \ quadratic\ forms.}\newline
{\tt 6.\ \ \
QuadraticForm(n,\ K):\ Exports\ ==\ Implementation\ where}\newline
{\tt 7.\ \ \ \ \ \ \ n:\ PositiveInteger}\newline
{\tt 8.\ \ \ \ \ \ \ K:\ Field}\newline
{\tt 9.\ \ \ }\newline
{\tt 10.\ \ \ \ \ \ Exports\ ==\ AbelianGroup\ with}\newline
{\tt 11.\ \ \ \ \ \ \ \ quadraticForm:\ SquareMatrix(n,K)\ ->\ \$}\newline
{\tt 12.\ \ \ \ \ \ \ \ \ \ ++\
\bs{}axiom\{quadraticForm(m)\}\ creates\ a\ quadratic}\newline
{\tt 13.\ \ \ \ \ \ \ \ \ \ ++\
quadratic\ form\ from\ a\ symmetric,}\newline
{\tt 14.\ \ \ \ \ \ \ \ \ \ ++\ square\ matrix\ \bs{}axiom\{m\}.}\newline
{\tt 15.\ \ \ \ \ \ \ \ matrix:\ \$\ ->\ SquareMatrix(n,K)}\newline
{\tt 16.\ \ \ \ \ \ \ \ \ \ ++\
\bs{}axiom\{matrix(qf)\}\ creates\ a\ square\ matrix}\newline
{\tt 17.\ \ \ \ \ \ \ \ \ \ ++\
from\ the\ quadratic\ form\ \bs{}axiom\{qf\}.}\newline
{\tt 18.\ \ \ \ \ \ \ \ elt:\ (\$,\ DirectProduct(n,K))\ ->\ K}\newline
{\tt 19.\ \ \ \ \ \ \ \ \ \ ++\
\bs{}axiom\{qf(v)\}\ evaluates\ the\ quadratic\ form}\newline
{\tt 20.\ \ \ \ \ \ \ \ \ \ ++\
\bs{}axiom\{qf\}\ on\ the\ vector\ \bs{}axiom\{v\},}\newline
{\tt 21.\ \ \ \ \ \ \ \ \ \ ++\ producing\ a\ scalar.}\newline
{\tt 22.\ \ }\newline
{\tt 23.\ \ \ \ \ \ Implementation\ ==\ SquareMatrix(n,K)\ add}\newline
{\tt 24.\ \ \ \ \ \ \ \ Rep\ :=\ SquareMatrix(n,K)}\newline
{\tt 25.\ \ \ \ \ \ \ \ quadraticForm\ m\ ==}\newline
{\tt 26.\ \ \ \ \ \ \ \ \ \ not\ symmetric?\ m\ =>\ error}\newline
{\tt 27.\ \ \ \ \ \ \ \ \ \ \ \
"quadraticForm\ requires\ a\ symmetric\ matrix"}\newline
{\tt 28.\ \ \ \ \ \ \ \ \ \ m\ ::\ \$}\newline
```

```
{\tt 29.\ \ \ \ \ \ \ \ matrix\ q\ ==\ q\ ::\ Rep}\newline
{\tt 30.\ \ \ \ \ \ \ \ elt(q,v)\ ==\ dot(v,\ (matrix\ q\ *\ v))}\newline
%
\caption{The \protect\axiomType{QuadraticForm} domain.}\label{fig-quadform}
\endImportant

A domain constructor can take any number and type of parameters.
\axiomType{QuadraticForm} takes a positive integer \axiom{n} and a field
\axiom{K} as arguments.
Like a package, a domain has a set of explicit exports and an
implementation described by a capsule.
Domain constructors are documented in the same way as package constructors.

Domain \axiomType{QuadraticForm(n, K)}, for a given positive integer
\axiom{n} and domain \axiom{K}, explicitly exports three operations:
%
\indent{4}
\beginitems
\item\axiom{quadraticForm(A)} creates a quadratic form from a matrix
\axiom{A}.
\item\axiom{matrix(q)} returns the matrix \axiom{A} used to create
the quadratic form \axiom{q}.
\item\axiom{q.v} computes the scalar \texht{$v^TAv$}{transpose(v)*A*v}
for a given vector \axiom{v}.
\enditems
\indent{0}

Compared with the corresponding syntax given for the definition of a
package, you see that a domain constructor has three optional parts to
its definition: {\it Category Assertions}, {\it Add Domain}, and
{\it Representation}.

\endscroll
\autobuttons
\end{page}
```

———

Category Assertions

⇒ “notitle” (ugCategoriesCorrectnessPage) 15 on page 2415
⇒ “notitle” (ugCategoriesConditionalsPage) 15 on page 2420

— ug13.ht —

```
\begin{page}{ugDomainsAssertionsPage}{13.3. Category Assertions}
\beginscroll
```

```
%

The {\it Category Assertions} part of your domain constructor
definition lists those categories of which all domains created by the
constructor are unconditionally members.  The word ``unconditionally''
means that membership in a category does not depend on the values of
the parameters to the domain constructor.  This part thus defines the
link between the domains and the category hierarchies given on the
inside covers of this book.  As described in
\downlink{``Correctness''}{ugCategoriesCorrectnessPage}
in Section 12.8\ignore{ugCategoriesCorrectness}, it is
this link that makes it possible for you to pass objects of the
domains as arguments to other operations in Axiom.

Every \axiomType{QuadraticForm} domain is declared to be
unconditionally a member of category \axiomType{AbelianGroup}.  An
abelian group is a collection of elements closed under addition.
Every object {\it x} of an abelian group has an additive inverse {\it
y} such that \texht{$x + y = 0$}{\axiom{{\it x} + {\it y} = 0}}.  The
exports of an abelian group include \axiom{0}, \axiomOp{+},
\axiomOp{-}, and scalar multiplication by an integer.  After asserting
that \axiomType{QuadraticForm} domains are abelian groups, it is
possible to pass quadratic forms to algorithms that only assume
arguments to have these abelian group properties.

In
\downlink{``Conditionals''}{ugCategoriesConditionalsPage}
in Section 12.11\ignore{ugCategoriesConditionals}, you
saw that \axiomType{Fraction(R)}, a member of
\axiomType{QuotientFieldCategory(R)}, is a member of
\axiomType{OrderedSet} if \axiom{R} is a member of
\axiomType{OrderedSet}.  Likewise, from the {\tt Exports} part of the
definition of \axiomType{ModMonic(R, S)}, \begin{verbatim}
UnivariatePolynomialCategory(R) with
  if R has Finite then Finite
     ...
\end{verbatim}
you see that \axiomType{ModMonic(R, S)} is a member of
\axiomType{Finite} is \axiom{R} is.

The {\tt Exports} part of a domain definition is the same kind of
expression that can appear to the right of an \axiomSyntax{==} in a
category definition.  If a domain constructor is unconditionally a
member of two or more categories, a \axiom{Join} form is used.
\spadkey{Join} The {\tt Exports} part of the definition of
\axiomType{FlexibleArray(S)} reads, for example:
\begin{verbatim}
Join(ExtensibleLinearAggregate(S),
     OneDimensionalArrayAggregate(S)) with...
\end{verbatim}
```

```
\endscroll
\autobuttons
\end{page}
```

———-

A Demo

— ug13.ht —

```
\begin{page}{ugDomainsDemoPage}{13.4. A Demo}
\beginscroll
%
Before looking at the {\it Implementation} part of \axiomType{QuadraticForm},
let's try some examples.

\texht{\vskip 2pc}{}
\xtc{
Build a domain \axiom{QF}.
}{
\spadpaste{QF := QuadraticForm(2,Fraction Integer)\bound{x2}\free{x1}}
}
\xtc{
Define a matrix to be used to construct
a quadratic form.
}{
\spadpaste{A := matrix [[-1,1/2],[1/2,1]]\bound{x3}\free{x2}}
}
\xtc{
Construct the quadratic form.
A package call {\tt \$QF} is necessary since there
are other \axiomType{QuadraticForm} domains.
}{
\spadpaste{q : QF := quadraticForm(A)\bound{x4}\free{x3}}
}
\xtc{
Looks like a matrix. Try computing
the number of rows.
Axiom won't let you.
}{
\spadpaste{nrows q\free{x3}}
}
\xtc{
Create a direct product element \axiom{v}.
A package call is again necessary, but Axiom
understands your list as denoting a vector.
```

```
}{
\spadpaste{v := directProduct([2,-1])\$DirectProduct(2,Fraction Integer)
\bound{x5}\free{x4}}
}
\xtc{
Compute the product \texht{$v^TAv$}{transpose(v)*A*v}.
}{
\spadpaste{q.v\free{x5}}
}
\xtc{
What is 3 times \axiom{q} minus \axiom{q} plus \axiom{q}?
}{
\spadpaste{3*q-q+q\free{x4}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugDomainsDemoPagePatch1}
\begin{paste}{ugDomainsDemoPageFull1}{ugDomainsDemoPageEmpty1}
\pastebutton{ugDomainsDemoPageFull1}{\hidepaste}
\tab{5}\spadcommand{QF := QuadraticForm(2,Fraction Integer)\bound{x2 }\free{x1 }}
\indentrel{3}\begin{verbatim}
   (1)  QuadraticForm(2,Fraction Integer)
                                        Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugDomainsDemoPageEmpty1}
\begin{paste}{ugDomainsDemoPageEmpty1}{ugDomainsDemoPagePatch1}
\pastebutton{ugDomainsDemoPageEmpty1}{\showpaste}
\tab{5}\spadcommand{QF := QuadraticForm(2,Fraction Integer)\bound{x2 }\free{x1 }}
\end{paste}\end{patch}

\begin{patch}{ugDomainsDemoPagePatch2}
\begin{paste}{ugDomainsDemoPageFull2}{ugDomainsDemoPageEmpty2}
\pastebutton{ugDomainsDemoPageFull2}{\hidepaste}
\tab{5}\spadcommand{A := matrix [[-1,1/2],[1/2,1]]\bound{x3 }\free{x2 }}
\indentrel{3}\begin{verbatim}
             1
        - 1
             2
   (2)
         1
            1
         2
                         Type: Matrix Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{ugDomainsDemoPageEmpty2}
\begin{paste}{ugDomainsDemoPageEmpty2}{ugDomainsDemoPagePatch2}
\pastebutton{ugDomainsDemoPageEmpty2}{\showpaste}
\tab{5}\spadcommand{A := matrix [[-1,1/2],[1/2,1]]\bound{x3 }\free{x2 }}
\end{paste}\end{patch}

\begin{patch}{ugDomainsDemoPagePatch3}
\begin{paste}{ugDomainsDemoPageFull3}{ugDomainsDemoPageEmpty3}
\pastebutton{ugDomainsDemoPageFull3}{\hidepaste}
\tab{5}\spadcommand{q : QF := quadraticForm(A)\bound{x4 }\free{x3 }}
\indentrel{3}\begin{verbatim}
            1
        - 1
            2
   (3)
         1
            1
         2
                Type: QuadraticForm(2,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugDomainsDemoPageEmpty3}
\begin{paste}{ugDomainsDemoPageEmpty3}{ugDomainsDemoPagePatch3}
\pastebutton{ugDomainsDemoPageEmpty3}{\showpaste}
\tab{5}\spadcommand{q : QF := quadraticForm(A)\bound{x4 }\free{x3 }}
\end{paste}\end{patch}

\begin{patch}{ugDomainsDemoPagePatch4}
\begin{paste}{ugDomainsDemoPageFull4}{ugDomainsDemoPageEmpty4}
\pastebutton{ugDomainsDemoPageFull4}{\hidepaste}
\tab{5}\spadcommand{nrows q\free{x3 }}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugDomainsDemoPageEmpty4}
\begin{paste}{ugDomainsDemoPageEmpty4}{ugDomainsDemoPagePatch4}
\pastebutton{ugDomainsDemoPageEmpty4}{\showpaste}
\tab{5}\spadcommand{nrows q\free{x3 }}
\end{paste}\end{patch}

\begin{patch}{ugDomainsDemoPagePatch5}
\begin{paste}{ugDomainsDemoPageFull5}{ugDomainsDemoPageEmpty5}
\pastebutton{ugDomainsDemoPageFull5}{\hidepaste}
\tab{5}\spadcommand{v := directProduct([2,-1])$DirectProduct(2,Fraction Integer)\bound{x5 }\free{x4 }}
\indentrel{3}\begin{verbatim}
   (4)  [2,- 1]
                Type: DirectProduct(2,Fraction Integer)
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugDomainsDemoPageEmpty5}
\begin{paste}{ugDomainsDemoPageEmpty5}{ugDomainsDemoPagePatch5}
\pastebutton{ugDomainsDemoPageEmpty5}{\showpaste}
\tab{5}\spadcommand{v := directProduct([2,-1])$DirectProduct(2,Fraction Integer)\bound{x5 }\free
\end{paste}\end{patch}

\begin{patch}{ugDomainsDemoPagePatch6}
\begin{paste}{ugDomainsDemoPageFull6}{ugDomainsDemoPageEmpty6}
\pastebutton{ugDomainsDemoPageFull6}{\hidepaste}
\tab{5}\spadcommand{q.v\free{x5 }}
\indentrel{3}\begin{verbatim}
   (5)  - 5
                                 Type: Fraction Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugDomainsDemoPageEmpty6}
\begin{paste}{ugDomainsDemoPageEmpty6}{ugDomainsDemoPagePatch6}
\pastebutton{ugDomainsDemoPageEmpty6}{\showpaste}
\tab{5}\spadcommand{q.v\free{x5 }}
\end{paste}\end{patch}

\begin{patch}{ugDomainsDemoPagePatch7}
\begin{paste}{ugDomainsDemoPageFull7}{ugDomainsDemoPageEmpty7}
\pastebutton{ugDomainsDemoPageFull7}{\hidepaste}
\tab{5}\spadcommand{3*q-q+q\free{x4 }}
\indentrel{3}\begin{verbatim}
             3
        - 3
             2
   (6)
         3
            3
         2
                Type: QuadraticForm(2,Fraction Integer)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugDomainsDemoPageEmpty7}
\begin{paste}{ugDomainsDemoPageEmpty7}{ugDomainsDemoPagePatch7}
\pastebutton{ugDomainsDemoPageEmpty7}{\showpaste}
\tab{5}\spadcommand{3*q-q+q\free{x4 }}
\end{paste}\end{patch}
```

———-

Browse

— ug13.ht —

```
\begin{page}{ugDomainsBrowsePage}{13.5. Browse}
\beginscroll

The \Browse{} facility of Hyperdoc is useful for
investigating
the properties of domains, packages, and categories.
From the main Hyperdoc menu, move your mouse to {\bf Browse} and
click on the left mouse button.
This brings up the \Browse{} first page.
Now, with your mouse pointer somewhere in this window, enter the
string ‘‘quadraticform’’ into the input area (all lower case
letters will do).
Move your mouse to {\bf Constructors} and click.
Up comes a page describing \axiomType{QuadraticForm}.

From here, click on {\bf Description}.
This gives you a page that includes a part labeled by ‘‘{\it
Description:}’’.
You also see the types for arguments \axiom{n} and \axiom{K}
displayed as well as the fact that \axiomType{QuadraticForm}
returns an \axiomType{AbelianGroup}.
You can go and experiment a bit by selecting {\bf Field} with
your mouse.
Eventually, use
\UpButton{}
several times to return to the first page on
\axiomType{QuadraticForm}.

Select {\bf Operations} to get a list of operations for
\axiomType{QuadraticForm}.
You can select an operation by clicking on it
to get an individual page with information about that operation.
Or you can select the buttons along the bottom to see alternative
views or get additional information on the operations.
Then return to the page on \axiomType{QuadraticForm}.

Select {\bf Cross Reference} to get another menu.
This menu has buttons for {\bf Parents}, {\bf Ancestors}, and
others.
Clicking on {\bf Parents}, you see that \axiomType{QuadraticForm}
has one parent \axiomType{AbelianMonoid}.

\endscroll
\autobuttons
\end{page}
```

———-

Representation

⇒ “notitle” (ugDomainsDemoPage) 16 on page 2431

— **ug13.ht** —

```
\begin{page}{ugDomainsRepPage}{13.6. Representation}
\beginscroll
%
The {\tt Implementation} part of an Axiom capsule for a
domain constructor uses the special variable \axiom{Rep} to
identify the lower level data type used to represent the objects
of the domain.
The \axiom{Rep} for quadratic forms is \axiomType{SquareMatrix(n, K)}.
This means that all objects of the domain are required to be
\axiom{n} by \axiom{n} matrices with elements from \axiomType{K}.

The code for \axiomFun{quadraticForm} in Figure \ref{fig-quadform} on
page \pageref{fig-quadform} checks that the matrix is symmetric and
then converts it to \axiomSyntax{\$}, which means, as usual, ‘‘this
domain.’’ Such explicit conversions \index{conversion} are generally
required by the compiler.  Aside from checking that the matrix is
symmetric, the code for this function essentially does nothing.  The
{\frenchspacing\tt m :: \$} on line 28 coerces \axiom{m} to a
quadratic form.  In fact, the quadratic form you created in step (3)
of \downlink{‘‘A Demo’’}{ugDomainsDemoPage} in Section
13.4\ignore{ugDomainsDemo} is just the matrix you
passed it in disguise!  Without seeing this definition, you would not
know that.  Nor can you take advantage of this fact now that you do
know!  When we try in the next step of
\downlink{‘‘A Demo’’}{ugDomainsDemoPage} in Section
13.4\ignore{ugDomainsDemo} to regard \axiom{q} as a
matrix by asking for \axiomFun{nrows}, the number of its rows, Axiom
gives you an error message saying, in effect, ‘‘Good try, but this
won’t work!’’

The definition for the \spadfunFrom{matrix}{QuadraticForm}
function could hardly be simpler:
it just returns its argument after explicitly
\spadglossSee{coercing}{coerce} its argument to a matrix.
Since the argument is already a matrix, this coercion does no
computation.

Within the context of a capsule, an object of \axiomSyntax{\$} is
regarded both as a quadratic form {\it and} as a
```

```
matrix.\footnote{In case each of \axiomSyntax{\$} and \axiom{Rep}
have the same named operation available,
the one from \axiom{\$} takes precedence.
Thus, if you want the one from \axiomSyntax{Rep}, you must
package call it using a \axiomSyntax{\$Rep} suffix.}
This makes the definition of \axiom{q.v} easy---it
just calls the \spadfunFrom{dot}{DirectProduct} product from
\axiomType{DirectProduct} to perform the indicated operation.

\endscroll
\autobuttons
\end{page}
```

———-

Multiple Representations

⇒ "notitle" (ugTypesUnionsPage) 7 on page 1656

— **ug13.ht** —

```
\begin{page}{ugDomainsMultipleRepsPage}{13.7. Multiple Representations}
\beginscroll
%

To write functions that implement the operations of a domain, you
want to choose the most computationally efficient
data structure to represent the elements of your domain.

A classic problem in computer algebra is the optimal choice for an
internal representation of polynomials.
If you create a polynomial, say \texht{$3x^2+ 5$}{\axiom{3*x**2 + 5}},
how does Axiom hold this value internally?
There are many ways.
Axiom has nearly a dozen different representations of
polynomials, one to suit almost any purpose.
Algorithms for solving polynomial equations work most
efficiently with polynomials represented one way, whereas those for
factoring polynomials are most efficient using another.
One often-used representation is  a list of terms, each term
consisting of exponent-coefficient records written in the order
of decreasing exponents.
For example, the polynomial \texht{$3x^2+5$}{3*x**2+5} is
%>> I changed the k's in next line to e's as I thought that was
%>> clearer.
represented by the list \axiom{[[e:2, c:3], [e:0, c:5]]}.

What is the optimal data structure for a matrix?
```

```
It depends on the application.
For large sparse matrices, a linked-list structure of records
holding only the non-zero elements may be optimal.
If the elements can be defined by a simple formula
\texht{$f(i,j)$}{\axiom{f(i,j)}}, then a compiled function for
\axiom{f} may be optimal.
Some programmers prefer to represent ordinary matrices as vectors
of vectors.
Others prefer to represent matrices by one big linear array where
elements are accessed with linearly computable indexes.

While all these simultaneous structures tend to be confusing,
Axiom provides a helpful organizational tool for such a purpose:
categories.
\axiomType{PolynomialCategory}, for example, provides a uniform user
interface across all polynomial types.
Each kind of polynomial implements functions for
all these operations, each in its own way.
If you use only the top-level operations in
\axiomType{PolynomialCategory} you usually do not care what kind
of polynomial implementation is used.

%>> I've often thought, though, that it would be nice to be
%>> be able to use conditionals for representations.

Within a given domain, however, you define (at most) one
representation.\footnote{You can make that representation a
\pspadtype{Union} type, however.  See
\downlink{``Unions''}{ugTypesUnionsPage} in Section
2.5\ignore{ugTypesUnions} for examples of unions.}
If you want to have multiple representations (that is, several
domains, each with its own representation), use a category to describe
the {\tt Exports}, then define separate domains for each
representation.

\endscroll
\autobuttons
\end{page}
```

———-

Add Domain

⇒ "notitle" (ugDomainsDemoPage) 16 on page 2431

— **ug13.ht** —

```
\begin{page}{ugDomainsAddDomainPage}{13.8. Add Domain}
```

```
\beginscroll
%

The capsule part of {\tt Implementation} defines functions that
implement the operations exported by the domain---usually only some of
the operations.  In our demo in
\downlink{``A Demo''}{ugDomainsDemoPage} in Section
13.4\ignore{ugDomainsDemo}, we asked for the value of
\axiom{3*q-q+q}.  Where do the operations \axiomOp{*}, \axiomOp{+},
and \axiomOp{-} come from?  There is no definition for them in the
capsule!

The {\tt Implementation} part of a definition can optionally specify
an ``add-domain'' to the left of an {\tt add} \spadkey{add} (for
\axiomType{QuadraticForm}, defines \axiomType{SquareMatrix(n,K)} is
the add-domain).  The meaning of an add-domain is simply this: if the
capsule part of the {\tt Implementation} does not supply a function
for an operation, Axiom goes to the add-domain to find the function.
So do \axiomOpFrom{*}{QuadraticForm}, \axiomOpFrom{+}{QuadraticForm}
and \axiomOpFrom{-}{QuadraticForm} come from
\axiomType{SquareMatrix(n,K)}?

\endscroll
\autobuttons
\end{page}
```

———

Defaults

⇒ “notitle” (ugPackagesPage) 14 on page 2375
⇒ “notitle” (ugCategoriesDefaultsPage) 15 on page 2412

— **ug13.ht** —

```
\begin{page}{ugDomainsDefaultsPage}{13.9. Defaults}
\beginscroll
%
In \downlink{``Packages''}{ugPackagesPage} in
Chapter 11\ignore{ugPackages}, we saw that
categories can provide
default implementations for their operations.
How and when are they used?
When Axiom finds that \axiomType{QuadraticForm(2, Fraction Integer)}
does not implement the operations \axiomOp{*},
\axiomOp{+}, and \axiomOp{-}, it goes to
\axiomType{SquareMatrix(2,Fraction Integer)} to find it.
```

```
As it turns out, \axiomType{SquareMatrix(2, Fraction Integer)} does
not implement {\it any} of these operations!

What does Axiom do then?  Here is its overall strategy.  First, Axiom
looks for a function in the capsule for the domain.  If it is not
there, Axiom looks in the add-domain for the operation.  If that
fails, Axiom searches the add-domain of the add-domain, and so on.  If
all those fail, it then searches the default packages for the
categories of which the domain is a member.  In the case of
\axiomType{QuadraticForm}, it searches \axiomType{AbelianGroup}, then
its parents, grandparents, and so on.  If this fails, it then searches
the default packages of the add-domain.  Whenever a function is found,
the search stops immediately and the function is returned.  When all
fails, the system calls \axiomFun{error} to report this unfortunate
news to you.  To find out the actual order of constructors searched
for \axiomType{QuadraticForm}, consult \Browse{}: from the
\axiomType{QuadraticForm}, click on {\bf Cross Reference}, then on
{\bf Lineage}.

Let's apply this search strategy for our example \axiom{3*q-q+q}.  The
scalar multiplication comes first.  Axiom finds a default
implementation in \axiomType{AbelianGroup\&}.  Remember from
\downlink{``Defaults''}{ugCategoriesDefaultsPage} in
Section 12.6\ignore{ugCategoriesDefaults} that
\axiomType{SemiGroup} provides a default definition for
\texht{$x^n$}{x**n} by repeated squaring?  \axiomType{AbelianGroup}
similarly provides a definition for \texht{$n x$}{n*x} by repeated
doubling.

But the search of the defaults for \axiomType{QuadraticForm} fails to
find any \axiomOp{+} or \axiomOp{*} in the default packages for the
ancestors of \axiomType{QuadraticForm}.  So it now searches among
those for \axiomType{SquareMatrix}.  Category
\axiomType{MatrixCategory}, which provides a uniform interface for all
matrix domains, is a grandparent of \axiomType{SquareMatrix} and has a
capsule defining many functions for matrices, including matrix
addition, subtraction, and scalar multiplication.  The default package
\axiomType{MatrixCategory\&} is where the functions for
\axiomOpFrom{+}{QuadraticForm} and \spadfunFrom{-}{QuadraticForm} come
from.

You can use \Browse{} to discover where the operations for
\axiomType{QuadraticForm} are implemented.  First, get the page
describing \axiomType{QuadraticForm}.  With your mouse somewhere in
this window, type a ``2'', press the \texht{\fbox{\bf Tab}}{{\bf Tab}}
key, and then enter ``Fraction Integer'' to indicate that you want the
domain \axiomType{QuadraticForm(2, Fraction Integer)}.  Now click on
{\bf Operations} to get a table of operations and on \axiomOp{*} to
get a page describing the \axiomOp{*} operation.  Finally, click on
{\bf implementation} at the bottom.
```

```
\endscroll
\autobuttons
\end{page}
```

———-

Origins

— ug13.ht —

```
\begin{page}{ugDomainsOriginsPage}{13.10. Origins}
\beginscroll
%

Aside from the notion of where an operation is implemented, a useful
notion is the {\it origin} or ‘‘home’’ of an operation.  When an
operation (such as \spadfunFrom{quadraticForm}{QuadraticForm}) is
explicitly exported by a domain (such as \axiomType{QuadraticForm}),
you can say that the origin of that operation is that domain.  If an
operation is not explicitly exported from a domain, it is inherited
from, and has as origin, the (closest) category that explicitly
exports it.  The operations \axiomOpFrom{+}{AbelianMonoid} and
\axiomOpFrom{-}{AbelianMonoid} of \axiomType{QuadraticForm}, for
example, are inherited from \axiomType{AbelianMonoid}.  As it turns
out, \axiomType{AbelianMonoid} is the origin of virtually every
\axiomOp{+} operation in Axiom!

Again, you can use \Browse{} to discover the origins of operations.
From the \Browse{} page on \axiomType{QuadraticForm}, click on {\bf
Operations}, then on {\bf origins} at the bottom of the page.

The origin of the operation is the {\it only} place where on-line
documentation is given.  However, you can re-export an operation to
give it special documentation.  Suppose you have just invented the
world’s fastest algorithm for inverting matrices using a particular
internal representation for matrices.  If your matrix domain just
declares that it exports \axiomType{MatrixCategory}, it exports the
\axiomFun{inverse} operation, but the documentation the user gets from
\Browse{} is the standard one from \axiomType{MatrixCategory}.  To
give your version of \axiomFun{inverse} the attention it deserves,
simply export the operation explicitly with new documentation.  This
redundancy gives \axiomFun{inverse} a new origin and tells \Browse{}
to present your new documentation.

\endscroll
\autobuttons
```

```
\end{page}
```

———-

Short Forms

— ug13.ht —

```
\begin{page}{ugDomainsShortFormsPage}{13.11. Short Forms}
\beginscroll

In Axiom, a domain could be defined using only an add-domain and no
capsule.  Although we talk about rational numbers as quotients of
integers, there is no type \pspadtype{RationalNumber} in Axiom.  To
create such a type, you could compile the following ‘‘short-form’’
definition:

\beginImportant

\noindent
{\tt 1.\ \ \ RationalNumber()\ ==\ Fraction(Integer)}\newline
\endImportant

The {\tt Exports} part of this definition is missing and is taken to
be equivalent to that of \axiomType{Fraction(Integer)}.  Because of
the add-domain philosophy, you get precisely what you want.  The
effect is to create a little stub of a domain.  When a user asks to
add two rational numbers, Axiom would ask \pspadtype{RationalNumber}
for a function implementing this \axiomOp{+}.  Since the domain has no
capsule, the domain then immediately sends its request to
\axiomType{Fraction (Integer)}.

The short form definition for domains is used to
define such domains as \axiomType{MultivariatePolynomial}:

\beginImportant

\noindent
{\tt 1.\ \ \ MultivariatePolynomial(vl:\ List\ Symbol,\ R:\ Ring)\ ==}
\newline
{\tt 2.\ \ \ \ \ \ SparseMultivariatePolynomial(R,}\newline
{\tt 3.\ \ \ \ \ \ \ \ \ OrderedVariableList\ vl)}\newline
\endImportant

\endscroll
\autobuttons
\end{page}
```

———-

Example 1: Clifford Algebra

⇒ "notitle" (CliffordAlgebraXmpPage) 3.15 on page 234

— **ug13.ht** —

```
\begin{page}{ugDomainsCliffordPage}{13.12. Example 1: Clifford Algebra}
\beginscroll
%

Now that we have \axiomType{QuadraticForm} available,
let's put it to use.
Given some quadratic form \smath{Q} described by an
\smath{n} by \smath{n} matrix over a field \smath{K}, the domain
\axiomType{CliffordAlgebra(n, K, Q)} defines a vector space of
dimension \texht{$2^n$}{2**n} over \smath{K}.
This is an interesting domain since complex numbers, quaternions,
exterior algebras and spin algebras are all examples of Clifford
algebras.

The basic idea is this:
the quadratic form \axiom{Q} defines a basis
\texht{$e_1,e_2\ldots,e_n$}{e1,e2,...,en} for the
vector space \texht{$K^n$}{K**n}---the direct product of \axiom{K}
with itself \axiom{n} times.
From this, the Clifford algebra generates a basis of
\texht{$2^n$}{2**n} elements given by all the possible products
of the \texht{$e_i$}{\axiom{ei}} in order without duplicates, that is,
\texht{
1,
$e_1$,
$e_2$,
$e_1e_2$,
$e_3$,
$e_1e_3$,
$e_2e_3$,
$e_1e_2,e_3$}{1, e1, e2, e1*e2, e3, e1*e3, e2*e3, e1*e2*e3},
and so on.

The algebra is defined by the relations
\begin{verbatim}
ei*ei = Q(ei)
ei*ej = -ej*ei,  i ^= j
\end{verbatim}
```

```
Now look at the snapshot of its definition given in Figure
\ref{fig-clifalg}.
Lines 9-10 show part of the definitions of the {\tt Exports}.
A Clifford algebra over a field \axiom{K} is asserted to be a ring,
an algebra over \axiom{K}, and a vector space over \axiom{K}.
Its explicit exports include
\axiom{e(n),} which returns the \eth{\axiom{n}} unit element.

\beginImportant

\noindent
{\tt 1.\ \ \ NNI\ ==>\ NonNegativeInteger}\newline
{\tt 2.\ \ \ PI\ \ ==>\ PositiveInteger}\newline
{\tt 3.\ \ \ }\newline
{\tt 4.\ \ \
CliffordAlgebra(n,K,q):\ Exports\ ==\ Implementation\ where}\newline
{\tt 5.\ \ \ \ \ \ \ n:\ PI}\newline
{\tt 6.\ \ \ \ \ \ \ K:\ Field}\newline
{\tt 7.\ \ \ \ \ \ \ q:\ QuadraticForm(n,\ K)}\newline
{\tt 8.\ \ \ }\newline
{\tt 9.\ \ \ \ \ \ \
Exports\ ==\ Join(Ring,Algebra(K),VectorSpace(K))\ with}\newline
{\tt 10.\ \ \ \ \ \ \ \ e:\ PI\ ->\ \$}\newline
{\tt 11.\ \ \ \ \ \ \ \ \ \ \ \ ...\ \ \ \ \ \ \ \ }\newline
{\tt 12.\ \ }\newline
{\tt 13.\ \ \ \ \ \ Implementation\ ==\ add}\newline
{\tt 14.\ \ \ \ \ \ \ \ Qeelist\ :=\ \ }\newline
{\tt 15.\ \ \ \ \ \ \ \ \ \
[q.unitVector(i::PI)\ for\ i\ in\ 1..n]}\newline
{\tt 16.\ \ \ \ \ \ \ \ dim\ \ \ \ \ :=\ \ 2**n}\newline
{\tt 17.\ \ \ \ \ \ \ \ Rep\ \ \ \ \ :=\ PrimitiveArray\ K}\newline
{\tt 18.\ \ \ \ \ \ \ \ New\ ==>\ new(dim,\ 0\$K)\$Rep}\newline
{\tt 19.\ \ \ \ \ \ \ \ x\ +\ y\ ==}\newline
{\tt 20.\ \ \ \ \ \ \ \ \ \ z\ :=\ New}\newline
{\tt 21.\ \ \ \ \ \ \ \ \ \
for\ i\ in\ 0..dim-1\ repeat\ z.i\ :=\ x.i\ +\ y.i}\newline
{\tt 22.\ \ \ \ \ \ \ \ \ \ z}\newline
{\tt 23.\ \ \ \ \ \ \ \
addMonomProd:\ (K,\ NNI,\ K,\ NNI,\ \$)\ ->\ \$}\newline
{\tt 24.\ \ \ \ \ \ \ \
addMonomProd(c1,\ b1,\ c2,\ b2,\ z)\ ==\ \ ...}\newline
{\tt 25.\ \ \ \ \ \ \ \ x\ *\ y\ ==}\newline
{\tt 26.\ \ \ \ \ \ \ \ \ \ z\ :=\ New}\newline
{\tt 27.\ \ \ \ \ \ \ \ \ \ for\ ix\ in\ 0..dim-1\ repeat}\newline
{\tt 28.\ \ \ \ \ \ \ \ \ \ \ \
if\ x.ix\ \notequal{}\ 0\ then\ for\ iy\ in\ 0..dim-1\ repeat}\newline
{\tt 29.\ \ \ \ \ \ \ \ \ \ \ \ \ \ if\ y.iy\ \notequal{}\ 0}\newline
{\tt 30.\ \ \ \ \ \ \ \ \ \ \ \ \ \
then\ addMonomProd(x.ix,ix,y.iy,iy,z)}\newline
{\tt 31.\ \ \ \ \ \ \ \ \ \ \ \ z}\newline
```

```
{\tt 32.\ \ \ \ \ \ \ \ \ \ \ \ \ ...}\newline
\caption{Part of the \protect\axiomType{CliffordAlgebra} domain.}
\label{fig-clifalg}
\endImportant

The {\tt Implementation} part begins by defining a local variable
\axiom{Qeelist} to hold the list of all \axiom{q.v} where \axiom{v}
runs over the unit vectors from 1 to the dimension \axiom{n}.
Another local variable \axiom{dim} is set to \texht{$2^n$}{2**n},
computed once and for all.
The representation for the domain is
\axiomType{PrimitiveArray(K)},
which is a basic array of elements from domain \axiom{K}.
Line 18 defines \axiom{New} as shorthand for the more lengthy
expression \axiom{new(dim, 0\$K)\$Rep}, which computes a primitive
array of length \texht{$2^n$}{2**n} filled with \axiom{0}'s from
domain \axiom{K}.

Lines 19-22 define the sum of two elements \axiom{x} and \axiom{y}
straightforwardly.
First, a new array of all \axiom{0}'s is created, then filled with
the sum of the corresponding elements.
Indexing for primitive arrays starts at 0.
The definition of the product of \axiom{x} and \axiom{y} first requires
the definition of a local function \userfun{addMonomProd}.
Axiom knows it is local since it is not an exported function.
The types of all local functions must be declared.

For a demonstration of \axiomType{CliffordAlgebra}, see
\downlink{`CliffordAlgebra'}{CliffordAlgebraXmpPage}
\ignore{CliffordAlgebra}.

\endscroll
\autobuttons
\end{page}
```

———

Example 2: Building A Query Facility

⇒ "notitle" (ugDomainsQueryLanguagePage) 16 on page 2447
⇒ "notitle" (ugDomainsDatabaseConstructorPage) 16 on page 2450
⇒ "notitle" (ugDomainsQueryEquationsPage) 16 on page 2452
⇒ "notitle" (ugDomainsDataListsPage) 16 on page 2454
⇒ "notitle" (ugDomainsDatabasePage) 16 on page 2455
⇒ "notitle" (ugDomainsCreatingPage) 16 on page 2455

⇒ "notitle" (ugDomainsPuttingPage) 16 on page 2456
⇒ "notitle" (ugDomainsExamplesPage) 16 on page 2457
— **ug13.ht** —

```
\begin{page}{ugDomsinsDatabasePage}
{13.13. Example 2: Building A Query Facility}
\beginscroll
%
We now turn to an entirely different kind of application,
building a query language for a database.

Here is the practical problem to solve.
The \Browse{} facility of Axiom has a
database for all operations and constructors which is
stored on disk and accessed by Hyperdoc.
For our purposes here, we regard each line of this file as having
eight fields:
{\tt class, name, type, nargs, exposed, kind, origin,} and {\tt condition.}
Here is an example entry:

\begin{verbatim}
o`determinant`$->R`1`x`d`Matrix(R)`has(R,commutative("*"))
\end{verbatim}

In English, the entry means:
\texht{\begin{quotation}}{\newline}
\texht{\raggedright}{}
The operation \axiomFun{determinant}: \spadsig{\$}{R} with {\it 1}
argument, is
{\it exposed} and is exported by {\it domain} \axiomType{Matrix(R)}
if {\tt R has commutative("*")}.
\texht{\end{quotation}}{\newline}

Our task is to create a little query language that allows us
to get useful information from this database.

\beginmenu
\menudownlink{{13.13.1. A Little Query Language}}
{ugDomainsQueryLanguagePage}
\menudownlink{{13.13.2. The Database Constructor}}
{ugDomainsDatabaseConstructorPage}
\menudownlink{{13.13.3. Query Equations}}{ugDomainsQueryEquationsPage}
\menudownlink{{13.13.4. DataLists}}{ugDomainsDataListsPage}
\menudownlink{{13.13.5. Index Cards}}{ugDomainsDatabasePage}
\menudownlink{{13.13.6. Creating a Database}}{ugDomainsCreatingPage}
\menudownlink{{13.13.7. Putting It All Together}}{ugDomainsPuttingPage}
\menudownlink{{13.13.8. Example Queries}}{ugDomainsExamplesPage}
\endmenu
\endscroll
\autobuttons
```

```
\end{page}
```

———

A Little Query Language

— **ug13.ht** —

```
\begin{page}{ugDomainsQueryLanguagePage}{13.13.1. A Little Query Language}
\beginscroll

First we design a simple language for accessing information from
the database.
We have the following simple model in mind for its design.
Think of the database as a box of index cards.
There is only one search operation---it
takes the name of a field and a predicate
(a boolean-valued function) defined on the fields of the
index cards.
When applied, the search operation goes through the entire box
selecting only those index cards for which the predicate is true.
The result of a search is a new box of index cards.
This process can be repeated again and again.

The predicates all have a particularly simple form: {\it symbol}
{\tt =} {\it pattern}, where {\it symbol} designates one of the
fields, and {\it pattern} is a ``search string''---a string
that may contain a ``{\tt *}'' as a
wildcard.
Wildcards match any substring, including the empty string.
Thus the pattern {\tt "*ma*t"} matches
{\tt "mat", "doormat"} and {\tt "smart".}

To illustrate how queries are given, we give you a sneak preview
of the facility we are about to create.

\xtc{
Extract the database of all Axiom operations.
}{
\spadpaste{ops := getDatabase("o")\bound{o1}}
}
\xtc{
How many exposed three-argument \axiomFun{map} operations involving
streams?
}{
\spadpaste{ops.(name="map").(nargs="3").(type="*Stream*")
\bound{o2}\free{o1}}
```

```
}

As usual, the arguments of \axiomFun{elt} (\axiomSyntax{.})
associate to the left.
The first \axiomFun{elt} produces the set of all operations with
name {\tt map}.
The second \axiomFun{elt} produces the set of all map operations
with three arguments.
The third \axiomFun{elt} produces the set of all three-argument map
operations having a type mentioning \axiomType{Stream}.

Another thing we'd like to do is to extract one field from each of
the index cards in the box and look at the result.
Here is an example of that kind of request.

\xtc{
What constructors explicitly export a \axiomFun{determinant} operation?
}{
\spadpaste{
elt(elt(elt(elt(ops,name="determinant"),origin),sort),unique)\free{o1}}
}

The first \axiomFun{elt} produces the set of all index cards with
name {\tt determinant}.
The second \axiomFun{elt} extracts the {\tt origin} component from
each index card. Each origin component
is the name of a constructor which directly
exports the operation represented by the index card.
Extracting a component from each index card produces what we call
a {\it datalist}.
The third \axiomFun{elt}, {\tt sort}, causes the datalist of
origins to be sorted in alphabetic
order.
The fourth, {\tt unique}, causes duplicates to be removed.

Before giving you a more extensive demo of this facility,
we now build the necessary domains and packages to implement it.
%We will introduce a few of our minor conveniences.

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugDomainsQueryLanguagePagePatch1}
\begin{paste}{ugDomainsQueryLanguagePageFull1}{ugDomainsQueryLanguagePageEmpty1}
\pastebutton{ugDomainsQueryLanguagePageFull1}{\hidepaste}
\tab{5}\spadcommand{ops := getDatabase("o")\bound{o1 }}
\indentrel{3}\begin{verbatim}
   (1)  6315
                               Type: Database IndexCard
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugDomainsQueryLanguagePageEmpty1}
\begin{paste}{ugDomainsQueryLanguagePageEmpty1}{ugDomainsQueryLanguagePagePatch1}
\pastebutton{ugDomainsQueryLanguagePageEmpty1}{\showpaste}
\tab{5}\spadcommand{ops := getDatabase("o")\bound{o1 }}
\end{paste}\end{patch}

\begin{patch}{ugDomainsQueryLanguagePagePatch2}
\begin{paste}{ugDomainsQueryLanguagePageFull2}{ugDomainsQueryLanguagePageEmpty2}
\pastebutton{ugDomainsQueryLanguagePageFull2}{\hidepaste}
\tab{5}\spadcommand{ops.(name="map").(nargs="3").(type="*Stream*")\bound{o2 }\free{o1 }}
\indentrel{3}\begin{verbatim}
   (2)  3
                              Type: Database IndexCard
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugDomainsQueryLanguagePageEmpty2}
\begin{paste}{ugDomainsQueryLanguagePageEmpty2}{ugDomainsQueryLanguagePagePatch2}
\pastebutton{ugDomainsQueryLanguagePageEmpty2}{\showpaste}
\tab{5}\spadcommand{ops.(name="map").(nargs="3").(type="*Stream*")\bound{o2 }\free{o1 }}
\end{paste}\end{patch}

\begin{patch}{ugDomainsQueryLanguagePagePatch3}
\begin{paste}{ugDomainsQueryLanguagePageFull3}{ugDomainsQueryLanguagePageEmpty3}
\pastebutton{ugDomainsQueryLanguagePageFull3}{\hidepaste}
\tab{5}\spadcommand{elt(elt(elt(elt(ops,name="determinant"),origin),sort),unique)\free{o1 }}
\indentrel{3}\begin{verbatim}
   (3)
   ["InnerMatrixLinearAlgebraFunctions",
    "MatrixCategory", "MatrixLinearAlgebraFunctions",
    "SquareMatrixCategory"]
                                 Type: DataList String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugDomainsQueryLanguagePageEmpty3}
\begin{paste}{ugDomainsQueryLanguagePageEmpty3}{ugDomainsQueryLanguagePagePatch3}
\pastebutton{ugDomainsQueryLanguagePageEmpty3}{\showpaste}
\tab{5}\spadcommand{elt(elt(elt(elt(ops,name="determinant"),origin),sort),unique)\free{o1 }}
\end{paste}\end{patch}
```

The Database Constructor

— ug13.ht —

```
\begin{page}{ugDomainsDatabaseConstructorPage}
{13.13.2. The Database Constructor}
\beginscroll

We work from the top down. First, we define a database,
our box of index cards, as an abstract datatype.
For sake of illustration and generality,
we assume that an index card is some type \axiom{S}, and
that a database is a box of objects of type \axiom{S}.
Here is the Axiom program defining the \pspadtype{Database}
domain.

\beginImportant

\noindent
{\tt 1.\ \ \ PI\ ==>\ PositiveInteger}\newline
{\tt 2.\ \ \ Database(S):\ Exports\ ==\ Implementation\ where}\newline
{\tt 3.\ \ \ \ \ S:\ Object\ with\ }\newline
{\tt 4.\ \ \ \ \ \ \ elt:\ (\$,\ Symbol)\ ->\ String}\newline
{\tt 5.\ \ \ \ \ \ \ display:\ \$\ ->\ Void}\newline
{\tt 6.\ \ \ \ \ \ \ fullDisplay:\ \$\ ->\ Void}\newline
{\tt 7.\ \ \ }\newline
{\tt 8.\ \ \ \ \ Exports\ ==\ with}\newline
{\tt 9.\ \ \ \ \ \ \ elt:\ (\$,QueryEquation)\ ->\ \$}\newline
{\tt 10.\ \ \ \ \ \ elt:\ (\$,\ Symbol)\ ->\ DataList\ String}\newline
{\tt 11.\ \ \ \ \ \ "+":\ (\$,\$)\ ->\ \$}\newline
{\tt 12.\ \ \ \ \ \ "-":\ (\$,\$)\ ->\ \$}\newline
{\tt 13.\ \ \ \ \ \ display:\ \$\ ->\ Void}\newline
{\tt 14.\ \ \ \ \ \ fullDisplay:\ \$\ ->\ Void}\newline
{\tt 15.\ \ \ \ \ \ fullDisplay:\ (\$,PI,PI)\ ->\ Void}\newline
{\tt 16.\ \ \ \ \ \ coerce:\ \$\ ->\ OutputForm}\newline
{\tt 17.\ \ \ \ Implementation\ ==\ add}\newline
{\tt 18.\ \ \ \ \ \ \ \ \ ...}\newline
\endImportant

The domain constructor takes a parameter \axiom{S}, which
stands for the class of index cards.
We describe an index card later.
Here think of an index card as a string which has
the eight fields mentioned above.

First, we tell Axiom what operations we are going to require
from index cards.
We need an \axiomFun{elt} to extract the contents of a field
(such as {\tt name} and {\tt type}) as a string.
```

```
For example,
\axiom{c.name} returns a string that is the content of the
\axiom{name} field on the index card \axiom{c}.
We need to display an index card in two ways:
\pspadfun{display} shows only the name and type of an
operation;
\pspadfun{fullDisplay} displays all fields.
The display operations return no useful information and thus have
return type \axiomType{Void}.

Next, we tell Axiom what operations the user can apply
to the database.
This part defines our little query language.
The most important operation is
{\frenchspacing\tt db . field = pattern} which
returns a new database, consisting of all index
cards of {\tt db} such that the \axiom{field} part of the index
card is matched by the string pattern called \axiom{pattern}.
The expression {\tt field = pattern} is an object of type
\axiomType{QueryEquation} (defined in the next section).

Another \axiomFun{elt} is needed to produce a \pspadtype{DataList}
object.
Operation \axiomOp{+} is to merge two databases together;
\axiomOp{-} is used to subtract away common entries in a second
database from an initial database.
There are three display functions.
The \pspadfun{fullDisplay} function has two versions: one
that prints all the records, the other that prints only a fixed
number of records.
A \axiomFun{coerce} to \axiomType{OutputForm} creates a display
object.

The {\tt Implementation} part of \axiomType{Database} is straightforward.
\beginImportant

\noindent
{\tt 1.\ \ \ \ \ Implementation\ ==\ add}\newline
{\tt 2.\ \ \ \ \ \ \ s:\ Symbol}\newline
{\tt 3.\ \ \ \ \ \ \ Rep\ :=\ List\ S}\newline
{\tt 4.\ \ \ \ \ \ \ elt(db,equation)\ ==\ ...}\newline
{\tt 5.\ \ \ \ \ \ \
elt(db,key)\ ==\ [x.key\ for\ x\ in\ db]::DataList(String)}\newline
{\tt 6.\ \ \ \ \ \ \
display(db)\ ==\ \ for\ x\ in\ db\ repeat\ display\ x}\newline
{\tt 7.\ \ \ \ \ \ \
fullDisplay(db)\ ==\ for\ x\ in\ db\ repeat\ fullDisplay\ x}\newline
{\tt 8.\ \ \ \ \ \ \
fullDisplay(db,\ n,\ m)\ ==\ for\ x\ in\ db\ for\ i\ in\ 1..m}\newline
{\tt 9.\ \ \ \ \ \ \ \ \ repeat}\newline
```

```
{\tt 10.\ \ \ \ \ \ \ \ \ \ if\ i\ >=\ n\ then\ fullDisplay\ x}\newline
{\tt 11.\ \ \ \ \ \ x+y\ ==\ removeDuplicates!\ merge(x,y)}\newline
{\tt 12.\ \ \ \ \ \ x-y\ ==\ mergeDifference(copy(x::Rep),}\newline
{\tt 13.\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
y::Rep)\$MergeThing(S)}\newline
{\tt 14.\ \ \ \ \ \ coerce(db):\ OutputForm\ ==\ (\#db)::\ OutputForm}
\newline
\endImportant

The database is represented by a list of elements of \axiom{S}
(index cards).
We leave the definition of the first \axiomFun{elt} operation
(on line 4) until the next section.
The second \axiomFun{elt} collects all the strings with field name
{\it key} into a list.
The \axiomFun{display} function and first \axiomFun{fullDisplay} function
simply call the corresponding functions from \axiom{S}.
The second \axiomFun{fullDisplay} function provides an efficient way of
printing out a portion of a large list.
The \axiomOp{+} is defined by using the existing
\spadfunFrom{merge}{List} operation defined on lists, then
removing duplicates from the result.
The \axiomOp{-} operation requires writing a corresponding
subtraction operation.
A package \axiomType{MergeThing} (not shown) provides this.

The \axiomFun{coerce} function converts the database to an
\axiomType{OutputForm} by computing the number of index cards.
This is a good example of the independence of
the representation of an Axiom object from how it presents
itself to the user. We usually do not want to look at a database---but
do care how many ``hits'' we get for a given query.
So we define the output representation of a database to be simply
the number of index cards our query finds.
\endscroll
\autobuttons
\end{page}
```

———-

Query Equations

— ug13.ht —

```
\begin{page}{ugDomainsQueryEquationsPage}{13.13.3. Query Equations}
\beginscroll
```

```
The predicate for our search is given by an object of type
\pspadtype{QueryEquation}.
Axiom does not have such an object yet so we
have to invent it.

\beginImportant

\noindent
{\tt 1.\ \ \ QueryEquation():\ Exports\ ==\ Implementation\ where}\newline
{\tt 2.\ \ \ \ \ Exports\ ==\ with}\newline
{\tt 3.\ \ \ \ \ \ \ \ equation:\ (Symbol,\ String)\ ->\ \$}\newline
{\tt 4.\ \ \ \ \ \ \ \ variable:\ \$\ ->\ Symbol}\newline
{\tt 5.\ \ \ \ \ \ \ \ value:\\ \\ \\ \\ \$\ ->\ String}\newline
{\tt 6.\ \ \ }\newline
{\tt 7.\ \ \ \ \ Implementation\ ==\ add}\newline
{\tt 8.\ \ \ \ \ \ \ \ Rep\ :=\ Record(var:Symbol,\ val:String)}\newline
{\tt 9.\ \ \ \ \ \ \ \ equation(x,\ s)\ ==\ [x,\ s]}\newline
{\tt 10.\ \ \ \ \ \ \ variable\ q\ ==\ q.var}\newline
{\tt 11.\ \ \ \ \ \ \ value\\ \\ \\ \\ q\ ==\ q.val}\newline
\endImportant

Axiom converts an input expression of the form
\axiom{{\it a} = {\it b}} to \axiom{equation({\it a, b})}.
Our equations always have a symbol on the left and a string
on the right.
The {\tt Exports} part thus specifies an operation
\axiomFun{equation} to create a query equation, and
\pspadfun{variable} and \pspadfun{value} to select the left- and
right-hand sides.
The {\tt Implementation} part uses \pspadtype{Record} for a
space-efficient representation of an equation.

Here is the missing definition for the \axiomFun{elt} function of
\pspadtype{Database} in the last section:

\beginImportant

\noindent
{\tt 1.\ \ \ \ \ \ \ \ elt(db,eq)\ ==}\newline
{\tt 2.\ \ \ \ \ \ \ \ \ \ field\\ \ :=\ variable\ eq}\newline
{\tt 3.\ \ \ \ \ \ \ \ \ \ value\ :=\ value\ eq}\newline
{\tt 4.\ \ \ \ \ \ \ \ \ \
[x\ for\ x\ in\ db\ |\ matches?(value,x.field)]}\newline
\endImportant

Recall that a database is represented by a list.
Line 4 simply runs over that list collecting all elements
such that the pattern (that is, \axiom{value})
matches the selected field of the element.
```

```
\endscroll
\autobuttons
\end{page}
```

———-

DataLists

— ug13.ht —

```
\begin{page}{ugDomainsDataListsPage}{13.13.4. DataLists}
\beginscroll

Type \pspadtype{DataList} is a new type invented to hold the result
of selecting one field from each of the index cards in the box.
It is useful to make datalists extensions of lists---lists that
have special \axiomFun{elt} operations defined on them for
sorting and removing duplicates.

\beginImportant

\noindent
{\tt 1.\ \ \ DataList(S:OrderedSet)\ :\ Exports\ ==\
Implementation\ where}\newline
{\tt 2.\ \ \ \ \ Exports\ ==\ ListAggregate(S)\ with}\newline
{\tt 3.\ \ \ \ \ \ \ elt:\ (\$,"unique")\ ->\ \$}\newline
{\tt 4.\ \ \ \ \ \ \ elt:\ (\$,"sort")\ ->\ \$}\newline
{\tt 5.\ \ \ \ \ \ \ elt:\ (\$,"count")\ ->\ NonNegativeInteger}\newline
{\tt 6.\ \ \ \ \ \ \ coerce:\ List\ S\ ->\ \$}\newline
{\tt 7.\ \ \ }\newline
{\tt 8.\ \ \ \ \ Implementation\ ==\ \ List(S)\ add}\newline
{\tt 9.\ \ \ \ \ \ \ Rep\ :=\ List\ S}\newline
{\tt 10.\ \ \ \ \ \ elt(x,"unique")\ ==\ removeDuplicates(x)}\newline
{\tt 11.\ \ \ \ \ \ elt(x,"sort")\ ==\ sort(x)}\newline
{\tt 12.\ \ \ \ \ \ elt(x,"count")\ ==\ \#x}\newline
{\tt 13.\ \ \ \ \ \ coerce(x:List\ S)\ ==\ x\ ::\ \$}\newline
\endImportant

The {\tt Exports} part asserts that datalists belong to the
category \axiomType{ListAggregate}.
Therefore, you can use all the usual list operations on datalists,
such as \spadfunFrom{first}{List}, \spadfunFrom{rest}{List}, and
\spadfunFrom{concat}{List}.
In addition, datalists have four explicit operations.
Besides the three \axiomFun{elt} operations, there is a
\axiomFun{coerce} operation that creates datalists from lists.
```

```
The {\tt Implementation} part needs only to define four functions.
All the rest are obtained from \axiomType{List(S)}.

\endscroll
\autobuttons
\end{page}
```

———-

Index Cards

— ug13.ht —

```
\begin{page}{ugDomainsDatabasePage}{13.13.5. Index Cards}
\beginscroll

An index card comes from a file as one long string.  We define
functions that extract substrings from the long string.  Each field
has a name that is passed as a second argument to \axiomFun{elt}.

\beginImportant

\noindent
{\tt 1.\ \ \ IndexCard()\ ==\ Implementation\ where}\newline
{\tt 2.\ \ \ \ \ Exports\ ==\ with}\newline
{\tt 3.\ \ \ \ \ \ \ elt:\ (\$,\ Symbol)\ ->\ String}\newline
{\tt 4.\ \ \ \ \ \ \ display:\ \$\ ->\ Void}\newline
{\tt 5.\ \ \ \ \ \ \ fullDisplay:\ \$\ ->\ Void}\newline
{\tt 6.\ \ \ \ \ \ \ coerce:\ String\ ->\ \$}\newline
{\tt 7.\ \ \ \ \ Implementation\ ==\ String\ add\ ...}\newline
\endImportant

We leave the {\tt Implementation} part to the reader.
All operations involve straightforward string manipulations.

\endscroll
\autobuttons
\end{page}
```

———-

Creating a Database

— ug13.ht —

```
\begin{page}{ugDomainsCreatingPage}{13.13.6. Creating a Database}
\beginscroll

We must not forget one important operation: one that builds the
database in the first place!  We'll name it \pspadfun{getDatabase} and
put it in a package.  This function is implemented by calling the
\Lisp{} function \axiom{getBrowseDatabase(s)} to get appropriate
information from \Browse{}.  This operation takes a string indicating
which lines you want from the database: \axiom{"o"} gives you all
operation lines, and \axiom{"k"}, all constructor lines.  Similarly,
\axiom{"c"}, \axiom{"d"}, and \axiom{"p"} give you all category,
domain and package lines respectively.

\beginImportant

\noindent
{\tt 1.\ \ \ OperationsQuery():\ Exports\ ==\ Implementation\ where}
\newline
{\tt 2.\ \ \ \ \ Exports\ ==\ with}\newline
{\tt 3.\ \ \ \ \ \ \ getDatabase:\ String\ ->\ Database(IndexCard)}
\newline
{\tt 4.\ \ \ }\newline
{\tt 5.\ \ \ \ \ Implementation\ ==\ add}\newline
{\tt 6.\ \ \ \ \ \ \ getDatabase(s)\ ==\ getBrowseDatabase(s)\$Lisp}
\newline
\endImportant

We do not bother creating a special name for databases of index
cards.
\pspadtype{Database (IndexCard)} will do.
Notice that we used the package \pspadtype{OperationsQuery} to
create, in effect,
a new kind of domain: \pspadtype{Database(IndexCard)}.

\endscroll
\autobuttons
\end{page}
```

———-

Putting It All Together

— ug13.ht —

```
\begin{page}{ugDomainsPuttingPage}{13.13.7. Putting It All Together}
\beginscroll
```

```
To create the database facility, you put all these constructors
into one file.\footnote{You could use separate files, but we
are putting them all together because, organizationally, that is
the logical thing to do.}
At the top of the file put \spadcmd{)abbrev} commands, giving the
constructor abbreviations you created.

\beginImportant

\noindent
{\tt 1.\ \ \ )abbrev\ domain\ \ ICARD\ \ \ IndexCard}\newline
{\tt 2.\ \ \ )abbrev\ domain\ \ QEQUAT\ \ QueryEquation}\newline
{\tt 3.\ \ \ )abbrev\ domain\ \ MTHING\ \ MergeThing}\newline
{\tt 4.\ \ \ )abbrev\ domain\ \ DLIST\ \ \ DataList}\newline
{\tt 5.\ \ \ )abbrev\ domain\ \ DBASE\ \ \ Database}\newline
{\tt 6.\ \ \ )abbrev\ package\ OPQUERY\ OperationsQuery}\newline
\endImportant

With all this in {\bf alql.spad}, for example, compile it using
\begin{verbatim}
)compile alql
\end{verbatim}
and then load each of the constructors:
\begin{verbatim}
)load ICARD QEQUAT MTHING DLIST DBASE OPQUERY
\end{verbatim}
You are ready to try some sample queries.

\endscroll
\autobuttons
\end{page}
```

———-

Example Queries

— ug13.ht —

```
\begin{page}{ugDomainsExamplesPage}{13.13.8. Example Queries}
\beginscroll

Our first set of queries give some statistics on constructors in
the current Axiom system.

\xtc{
How many constructors does Axiom have?
}{
```

```
\spadpaste{ks := getDatabase "k"\bound{q1}}
}
\xtc{
Break this down into the number of categories, domains, and packages.
}{
\spadpaste{[ks.(kind=k) for k in ["c","d","p"]]\bound{q3}\free{q1}}
}
\xtc{
What are all the domain constructors that take no parameters?
}{
\spadpaste{elt(ks.(kind="d").(nargs="0"),name)\bound{q4}\free{q1}}
}
\xtc{
How many constructors have ‘‘Matrix’’ in their name?
}{
\spadpaste{mk := ks.(name="*Matrix*")\bound{q5}\free{q1}}
}
\xtc{
What are the names of those that are domains?
}{
\spadpaste{elt(mk.(kind="d"),name)\bound{q6}\free{q5}}
}
\xtc{
How many operations are there in the library?
}{
\spadpaste{o := getDatabase "o"\bound{o1}}
}
\xtc{
Break this down into categories, domains, and packages.
}{
\spadpaste{[o.(kind=k) for k in ["c","d","p"]]\free{o1}}
}

The query language is helpful in getting information about a
particular operation you might like to apply.
While this information can be obtained with
\Browse{}, the use of the query database gives you data that you
can manipulate in the workspace.

\xtc{
How many operations have ‘‘eigen’’ in the name?
}{
\spadpaste{eigens := o.(name="*eigen*")\bound{eigens}\free{o1}}
}
\xtc{
What are their names?
}{
\spadpaste{elt(eigens,name)\free{eigens}}
}
\xtc{
```

```
Where do they come from?
}{
\spadpaste{elt(elt(elt(eigens,origin),sort),unique) \free{eigens}}
}

The operations \axiomOp{+} and \axiomOp{-} are useful for
constructing small databases and combining them.
However, remember that the only matching you can do is string
matching.
Thus a pattern such as {\tt "*Matrix*"} on the type field
matches
any type containing \axiomType{Matrix}, \axiomType{MatrixCategory},
\axiomType{SquareMatrix}, and so on.

\xtc{
How many operations mention ``Matrix'' in their type?
}{
\spadpaste{tm := o.(type="*Matrix*")\bound{x10}\free{o1}}
}
\xtc{
How many operations come from constructors with ``Matrix'' in
their name?
}{
\spadpaste{fm := o.(origin="*Matrix*")\bound{x11}\free{o1}}
}
\xtc{
How many operations are in \axiom{fm} but not in \axiom{tm}?
}{
\spadpaste{fm-tm \bound{x12}\free{x10 x11}}
}
\xtc{
Display the operations that both mention ``Matrix'' in their type
and come from a constructor having ``Matrix'' in their name.
}{
\spadpaste{fullDisplay(fm-\%) \bound{x13}\free{x12}}
}
\xtc{
How many operations involve matrices?
}{
\spadpaste{m := tm+fm \bound{x14}\free{x10 x11}}
}
\xtc{
Display 4 of them.
}{
\spadpaste{fullDisplay(m, 202, 205) \free{x14}}
}
\xtc{
How many distinct names of operations involving matrices are there?
}{
\spadpaste{elt(elt(elt(m,name),unique),count) \free{x14}}
```

```
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{ugDomainsExamplesPagePatch1}
\begin{paste}{ugDomainsExamplesPageFull1}{ugDomainsExamplesPageEmpty1}
\pastebutton{ugDomainsExamplesPageFull1}{\hidepaste}
\tab{5}\spadcommand{ks := getDatabase "k"\bound{q1 }}
\indentrel{3}\begin{verbatim}
   (1)  1067
                               Type: Database IndexCard
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugDomainsExamplesPageEmpty1}
\begin{paste}{ugDomainsExamplesPageEmpty1}{ugDomainsExamplesPagePatch1}
\pastebutton{ugDomainsExamplesPageEmpty1}{\showpaste}
\tab{5}\spadcommand{ks := getDatabase "k"\bound{q1 }}
\end{paste}\end{patch}

\begin{patch}{ugDomainsExamplesPagePatch2}
\begin{paste}{ugDomainsExamplesPageFull2}{ugDomainsExamplesPageEmpty2}
\pastebutton{ugDomainsExamplesPageFull2}{\hidepaste}
\tab{5}\spadcommand{[ks.(kind=k) for k in ["c","d","p"]]\bound{q3 }\free{q1 }}
\indentrel{3}\begin{verbatim}
   (2)  [205,393,469]
                          Type: List Database IndexCard
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugDomainsExamplesPageEmpty2}
\begin{paste}{ugDomainsExamplesPageEmpty2}{ugDomainsExamplesPagePatch2}
\pastebutton{ugDomainsExamplesPageEmpty2}{\showpaste}
\tab{5}\spadcommand{[ks.(kind=k) for k in ["c","d","p"]]\bound{q3 }\free{q1 }}
\end{paste}\end{patch}

\begin{patch}{ugDomainsExamplesPagePatch3}
\begin{paste}{ugDomainsExamplesPageFull3}{ugDomainsExamplesPageEmpty3}
\pastebutton{ugDomainsExamplesPageFull3}{\hidepaste}
\tab{5}\spadcommand{elt(ks.(kind="d").(nargs="0"),name)\bound{q4 }\free{q1 }}
\indentrel{3}\begin{verbatim}
   (3)
   ["AlgebraicNumber", "AnonymousFunction", "Any",
    "AttributeButtons", "BasicFunctions",
    "BasicOperator", "BinaryExpansion", "BinaryFile",
    "Bits", "Boolean", "CardinalNumber",
    "CharacterClass", "Character", "Color", "Commutator",
    "DecimalExpansion", "DoubleFloat", "DrawOption",
```

```
    "Exit", "ExtAlgBasis", "FileName", "Float",
    "FortranCode", "FortranScalarType",
    "FortranTemplate", "FortranType", "GraphImage",
    "HexadecimalExpansion", "IVBaseColor", "IVBasicNode",
    "IVCoordinate3", "IVCoordinate4", "IVFaceSet",
    "IVField", "IVGroup", "IVIndexedLineSet",
    "IVNodeConnection", "IVNodeObject", "IVPointSet",
    "IVQuadMesh", "IVSeparator", "IVSimpleInnerNode",
    "IVUtilities", "IVValue", "IndexCard",
    "InnerAlgebraicNumber", "InputForm", "Integer",
    "IntegrationFunctionsTable", "InventorDataSink",
    "InventorRenderPackage", "InventorViewPort",
    "Library", "MachineComplex", "MachineFloat",
    "MachineInteger",
    "NagDiscreteFourierTransformInterfacePackage",
    "NagEigenInterfacePackage",
    "NagOptimisationInterfacePackage",
    "NagQuadratureInterfacePackage", "NagResultChecks",
    "NagSpecialFunctionsInterfacePackage",
    "NonNegativeInteger", "None",
    "NumericalIntegrationProblem", "NumericalODEProblem",
    "NumericalOptimizationProblem",
    "NumericalPDEProblem", "ODEIntensityFunctionsTable",
    "OrdSetInts", "OutputForm", "Palette", "Partition",
    "Pi", "PlaneAlgebraicCurvePlot", "Plot3D", "Plot",
    "PositiveInteger", "QueryEquation", "RenderTools",
    "Result", "RomanNumeral", "RoutinesTable",
    "SExpression", "ScriptFormulaFormat",
    "SingleInteger", "SingletonAsOrderedSet", "String",
    "SubSpaceComponentProperty", "Switch", "SymbolTable",
    "Symbol", "TexFormat", "TextFile", "TheSymbolTable",
    "ThreeDimensionalViewport", "Timer",
    "TwoDimensionalViewport", "Void",
    "d01TransformFunctionType", "d01ajfAnnaType",
    "d01akfAnnaType", "d01alfAnnaType", "d01amfAnnaType",
    "d01anfAnnaType", "d01apfAnnaType", "d01aqfAnnaType",
    "d01asfAnnaType", "d01fcfAnnaType", "d01gbfAnnaType",
    "d02bbfAnnaType", "d02bhfAnnaType", "d02cjfAnnaType",
    "d02ejfAnnaType", "d03eefAnnaType", "d03fafAnnaType",
    "e04dgfAnnaType", "e04fdfAnnaType", "e04gcfAnnaType",
    "e04jafAnnaType", "e04mbfAnnaType", "e04nafAnnaType",
    "e04ucfAnnaType"]
                                  Type: DataList String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugDomainsExamplesPageEmpty3}
\begin{paste}{ugDomainsExamplesPageEmpty3}{ugDomainsExamplesPagePatch3}
\pastebutton{ugDomainsExamplesPageEmpty3}{\showpaste}
\tab{5}\spadcommand{elt(ks.(kind="d").(nargs="0"),name)\bound{q4 }\free{q1 }}
```

```
\end{paste}\end{patch}

\begin{patch}{ugDomainsExamplesPagePatch4}
\begin{paste}{ugDomainsExamplesPageFull4}{ugDomainsExamplesPageEmpty4}
\pastebutton{ugDomainsExamplesPageFull4}{\hidepaste}
\tab{5}\spadcommand{mk := ks.(name="*Matrix*")\bound{q5 }\free{q1 }}
\indentrel{3}\begin{verbatim}
   (4)  26
                              Type: Database IndexCard
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugDomainsExamplesPageEmpty4}
\begin{paste}{ugDomainsExamplesPageEmpty4}{ugDomainsExamplesPagePatch4}
\pastebutton{ugDomainsExamplesPageEmpty4}{\showpaste}
\tab{5}\spadcommand{mk := ks.(name="*Matrix*")\bound{q5 }\free{q1 }}
\end{paste}\end{patch}

\begin{patch}{ugDomainsExamplesPagePatch5}
\begin{paste}{ugDomainsExamplesPageFull5}{ugDomainsExamplesPageEmpty5}
\pastebutton{ugDomainsExamplesPageFull5}{\hidepaste}
\tab{5}\spadcommand{elt(mk.(kind="d"),name)\bound{q6 }\free{q5 }}
\indentrel{3}\begin{verbatim}
   (5)
   ["DenavitHartenbergMatrix",
    "DirectProductMatrixModule", "IndexedMatrix",
    "LieSquareMatrix", "Matrix", "RectangularMatrix",
    "SquareMatrix", "ThreeDimensionalMatrix"]
                                 Type: DataList String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugDomainsExamplesPageEmpty5}
\begin{paste}{ugDomainsExamplesPageEmpty5}{ugDomainsExamplesPagePatch5}
\pastebutton{ugDomainsExamplesPageEmpty5}{\showpaste}
\tab{5}\spadcommand{elt(mk.(kind="d"),name)\bound{q6 }\free{q5 }}
\end{paste}\end{patch}

\begin{patch}{ugDomainsExamplesPagePatch6}
\begin{paste}{ugDomainsExamplesPageFull6}{ugDomainsExamplesPageEmpty6}
\pastebutton{ugDomainsExamplesPageFull6}{\hidepaste}
\tab{5}\spadcommand{o := getDatabase "o"\bound{o1 }}
\indentrel{3}\begin{verbatim}
   (6)  6315
                              Type: Database IndexCard
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugDomainsExamplesPageEmpty6}
\begin{paste}{ugDomainsExamplesPageEmpty6}{ugDomainsExamplesPagePatch6}
```

```
\pastebutton{ugDomainsExamplesPageEmpty6}{\showpaste}
\tab{5}\spadcommand{o := getDatabase "o"\bound{o1 }}
\end{paste}\end{patch}

\begin{patch}{ugDomainsExamplesPagePatch7}
\begin{paste}{ugDomainsExamplesPageFull7}{ugDomainsExamplesPageEmpty7}
\pastebutton{ugDomainsExamplesPageFull7}{\hidepaste}
\tab{5}\spadcommand{[o.(kind=k) for k in ["c","d","p"]]\free{o1 }}
\indentrel{3}\begin{verbatim}
   (7)  [1646,2040,2629]
                          Type: List Database IndexCard
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugDomainsExamplesPageEmpty7}
\begin{paste}{ugDomainsExamplesPageEmpty7}{ugDomainsExamplesPagePatch7}
\pastebutton{ugDomainsExamplesPageEmpty7}{\showpaste}
\tab{5}\spadcommand{[o.(kind=k) for k in ["c","d","p"]]\free{o1 }}
\end{paste}\end{patch}

\begin{patch}{ugDomainsExamplesPagePatch8}
\begin{paste}{ugDomainsExamplesPageFull8}{ugDomainsExamplesPageEmpty8}
\pastebutton{ugDomainsExamplesPageFull8}{\hidepaste}
\tab{5}\spadcommand{eigens := o.(name="*eigen*")\bound{eigens }\free{o1 }}
\indentrel{3}\begin{verbatim}
   (8)  4
                               Type: Database IndexCard
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugDomainsExamplesPageEmpty8}
\begin{paste}{ugDomainsExamplesPageEmpty8}{ugDomainsExamplesPagePatch8}
\pastebutton{ugDomainsExamplesPageEmpty8}{\showpaste}
\tab{5}\spadcommand{eigens := o.(name="*eigen*")\bound{eigens }\free{o1 }}
\end{paste}\end{patch}

\begin{patch}{ugDomainsExamplesPagePatch9}
\begin{paste}{ugDomainsExamplesPageFull9}{ugDomainsExamplesPageEmpty9}
\pastebutton{ugDomainsExamplesPageFull9}{\hidepaste}
\tab{5}\spadcommand{elt(eigens,name)\free{eigens }}
\indentrel{3}\begin{verbatim}
   (9)
   ["eigenMatrix", "eigenvalues", "eigenvector",
    "eigenvectors"]
                                  Type: DataList String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugDomainsExamplesPageEmpty9}
\begin{paste}{ugDomainsExamplesPageEmpty9}{ugDomainsExamplesPagePatch9}
```

```
\pastebutton{ugDomainsExamplesPageEmpty9}{\showpaste}
\tab{5}\spadcommand{elt(eigens,name)\free{eigens }}
\end{paste}\end{patch}

\begin{patch}{ugDomainsExamplesPagePatch10}
\begin{paste}{ugDomainsExamplesPageFull10}{ugDomainsExamplesPageEmpty10}
\pastebutton{ugDomainsExamplesPageFull10}{\hidepaste}
\tab{5}\spadcommand{elt(elt(elt(eigens,origin),sort),unique)\free{eigens }}
\indentrel{3}\begin{verbatim}
   (10)  ["EigenPackage","RadicalEigenPackage"]
                                   Type: DataList String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugDomainsExamplesPageEmpty10}
\begin{paste}{ugDomainsExamplesPageEmpty10}{ugDomainsExamplesPagePatch10}
\pastebutton{ugDomainsExamplesPageEmpty10}{\showpaste}
\tab{5}\spadcommand{elt(elt(elt(eigens,origin),sort),unique)\free{eigens }}
\end{paste}\end{patch}

\begin{patch}{ugDomainsExamplesPagePatch11}
\begin{paste}{ugDomainsExamplesPageFull11}{ugDomainsExamplesPageEmpty11}
\pastebutton{ugDomainsExamplesPageFull11}{\hidepaste}
\tab{5}\spadcommand{tm := o.(type="*Matrix*")\bound{x10 }\free{o1 }}
\indentrel{3}\begin{verbatim}
   (11)  353
                                Type: Database IndexCard
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugDomainsExamplesPageEmpty11}
\begin{paste}{ugDomainsExamplesPageEmpty11}{ugDomainsExamplesPagePatch11}
\pastebutton{ugDomainsExamplesPageEmpty11}{\showpaste}
\tab{5}\spadcommand{tm := o.(type="*Matrix*")\bound{x10 }\free{o1 }}
\end{paste}\end{patch}

\begin{patch}{ugDomainsExamplesPagePatch12}
\begin{paste}{ugDomainsExamplesPageFull12}{ugDomainsExamplesPageEmpty12}
\pastebutton{ugDomainsExamplesPageFull12}{\hidepaste}
\tab{5}\spadcommand{fm := o.(origin="*Matrix*")\bound{x11 }\free{o1 }}
\indentrel{3}\begin{verbatim}
   (12)  192
                                Type: Database IndexCard
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugDomainsExamplesPageEmpty12}
\begin{paste}{ugDomainsExamplesPageEmpty12}{ugDomainsExamplesPagePatch12}
\pastebutton{ugDomainsExamplesPageEmpty12}{\showpaste}
\tab{5}\spadcommand{fm := o.(origin="*Matrix*")\bound{x11 }\free{o1 }}
```

```
\end{paste}\end{patch}

\begin{patch}{ugDomainsExamplesPagePatch13}
\begin{paste}{ugDomainsExamplesPageFull13}{ugDomainsExamplesPageEmpty13}
\pastebutton{ugDomainsExamplesPageFull13}{\hidepaste}
\tab{5}\spadcommand{fm-tm\bound{x12 }\free{x10 x11 }}
\indentrel{3}\begin{verbatim}
   (13)  146
                               Type: Database IndexCard
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugDomainsExamplesPageEmpty13}
\begin{paste}{ugDomainsExamplesPageEmpty13}{ugDomainsExamplesPagePatch13}
\pastebutton{ugDomainsExamplesPageEmpty13}{\showpaste}
\tab{5}\spadcommand{fm-tm\bound{x12 }\free{x10 x11 }}
\end{paste}\end{patch}

\begin{patch}{ugDomainsExamplesPagePatch14}
\begin{paste}{ugDomainsExamplesPageFull14}{ugDomainsExamplesPageEmpty14}
\pastebutton{ugDomainsExamplesPageFull14}{\hidepaste}
\tab{5}\spadcommand{fullDisplay(fm-\%)\bound{x13 }\free{x12 }}
\indentrel{3}\begin{verbatim}
   ** : (Matrix(R),NonNegativeInteger)->Matrix(R)
      from StorageEfficientMatrixOperations(R) (unexposed)
   clearDenominator : (Matrix(Q))->Matrix(R)
      from MatrixCommonDenominator(R,Q)
   coerceP
      : (Vector(Matrix(R)))->Vector(Matrix(Polynomial(R)))
      from CoerceVectorMatrixPackage(R) (unexposed)
   coerce
      :
      (Vector(Matrix(R)))->Vector(Matrix(Fraction(Polynom
      ial(R))))
      from CoerceVectorMatrixPackage(R) (unexposed)
   coerce : (_$)->Matrix(R)
      from RectangularMatrix(m,n,R) (unexposed)
   coerce : (_$)->Matrix(R)
      from SquareMatrix(ndim,R) (unexposed)
   coerce : (Matrix(MachineFloat))->_$
      from FortranMatrixCategory
   commonDenominator : (Matrix(Q))->R
      from MatrixCommonDenominator(R,Q)
   copy! : (Matrix(R),Matrix(R))->Matrix(R)
      from StorageEfficientMatrixOperations(R) (unexposed)
   f01brf
      :
      (Integer,Integer,Integer,Integer,DoubleFloat,Boolea
      n,Boolean,List(Boolean),Matrix(DoubleFloat),Matrix(
      Integer),Matrix(Integer),Integer)->Result
```

```
   from NagMatrixOperationsPackage
f01bsf
   :
   (Integer,Integer,Integer,Matrix(Integer),Matrix(Int
   eger),Matrix(Integer),Matrix(Integer),Boolean,Doubl
   eFloat,Boolean,Matrix(Integer),Matrix(DoubleFloat),
   Integer)->Result
   from NagMatrixOperationsPackage
f01maf
   :
   (Integer,Integer,Integer,Integer,List(Boolean),Matr
   ix(DoubleFloat),Matrix(Integer),Matrix(Integer),Dou
   bleFloat,DoubleFloat,Integer)->Result
   from NagMatrixOperationsPackage
f01mcf
   :
   (Integer,Matrix(DoubleFloat),Integer,Matrix(Integer
   ),Integer)->Result
   from NagMatrixOperationsPackage
f01qcf
   :
   (Integer,Integer,Integer,Matrix(DoubleFloat),Intege
   r)->Result
   from NagMatrixOperationsPackage
f01qdf
   :
   (String,String,Integer,Integer,Matrix(DoubleFloat),
   Integer,Matrix(DoubleFloat),Integer,Integer,Matrix(
   DoubleFloat),Integer)->Result
   from NagMatrixOperationsPackage
f01qef
   :
   (String,Integer,Integer,Integer,Integer,Matrix(Doub
   leFloat),Matrix(DoubleFloat),Integer)->Result
   from NagMatrixOperationsPackage
f01rcf
   :
   (Integer,Integer,Integer,Matrix(Complex(DoubleFloat
   )),Integer)->Result
   from NagMatrixOperationsPackage
f01rdf
   :
   (String,String,Integer,Integer,Matrix(Complex(Doubl
   eFloat)),Integer,Matrix(Complex(DoubleFloat)),Integ
   er,Integer,Matrix(Complex(DoubleFloat)),Integer)->R
   esult
   from NagMatrixOperationsPackage
f01ref
   :
   (String,Integer,Integer,Integer,Integer,Matrix(Comp
```

```
   lex(DoubleFloat)),Matrix(Complex(DoubleFloat)),Inte
   ger)->Result
   from NagMatrixOperationsPackage
hasSolution? : (Matrix(F),Vector(F))->Boolean
   from LinearSystemMatrixPackage1(F)
leftScalarTimes! : (Matrix(R),R,Matrix(R))->Matrix(R)
   from StorageEfficientMatrixOperations(R) (unexposed)
minus! : (Matrix(R),Matrix(R))->Matrix(R)
   from StorageEfficientMatrixOperations(R) (unexposed)
minus! : (Matrix(R),Matrix(R),Matrix(R))->Matrix(R)
   from StorageEfficientMatrixOperations(R) (unexposed)
particularSolution
   : (Matrix(F),Vector(F))->Union(Vector(F),"failed")
   from LinearSystemMatrixPackage1(F)
plus! : (Matrix(R),Matrix(R),Matrix(R))->Matrix(R)
   from StorageEfficientMatrixOperations(R) (unexposed)
power!
   :
   (Matrix(R),Matrix(R),Matrix(R),Matrix(R),NonNegativ
   eInteger)->Matrix(R)
   from StorageEfficientMatrixOperations(R) (unexposed)
rank : (Matrix(F),Vector(F))->NonNegativeInteger
   from LinearSystemMatrixPackage1(F)
rectangularMatrix : (Matrix(R))->_$
   from RectangularMatrix(m,n,R) (unexposed)
retractIfCan
   : (Matrix(Expression(Float)))->Union(_$,"failed")
   from FortranMatrixFunctionCategory
retractIfCan
   : (Matrix(Expression(Integer)))->Union(_$,"failed")
   from FortranMatrixFunctionCategory
retractIfCan
   :
   (Matrix(Fraction(Polynomial(Float))))->Union(_$,"fa
   iled")
   from FortranMatrixFunctionCategory
retractIfCan
   :
   (Matrix(Fraction(Polynomial(Integer))))->Union(_$,"
   failed")
   from FortranMatrixFunctionCategory
retractIfCan
   : (Matrix(Polynomial(Float)))->Union(_$,"failed")
   from FortranMatrixFunctionCategory
retractIfCan
   : (Matrix(Polynomial(Integer)))->Union(_$,"failed")
   from FortranMatrixFunctionCategory
retract : (Matrix(Expression(Float)))->_$
   from FortranMatrixFunctionCategory
retract : (Matrix(Expression(Integer)))->_$
```

```
      from FortranMatrixFunctionCategory
   retract : (Matrix(Fraction(Polynomial(Float))))->_$
      from FortranMatrixFunctionCategory
   retract : (Matrix(Fraction(Polynomial(Integer))))->_$
      from FortranMatrixFunctionCategory
   retract : (Matrix(Polynomial(Float)))->_$
      from FortranMatrixFunctionCategory
   retract : (Matrix(Polynomial(Integer)))->_$
      from FortranMatrixFunctionCategory
   rightScalarTimes! : (Matrix(R),Matrix(R),R)->Matrix(R)
      from StorageEfficientMatrixOperations(R) (unexposed)
   solve
      :
      (Matrix(F),List(Vector(F)))->List(Record(particular
      :Union(Vector(F),"failed"),basis:List(Vector(F))))
      from LinearSystemMatrixPackage1(F)
   solve
      :
      (Matrix(F),Vector(F))->Record(particular:Union(Vect
      or(F),"failed"),basis:List(Vector(F)))
      from LinearSystemMatrixPackage1(F)
   splitDenominator
      : (Matrix(Q))->Record(num:Matrix(R),den:R)
      from MatrixCommonDenominator(R,Q)
   squareMatrix : (Matrix(R))->_$
      from SquareMatrix(ndim,R) (unexposed)
   times! : (Matrix(R),Matrix(R),Matrix(R))->Matrix(R)
      from StorageEfficientMatrixOperations(R) (unexposed)
                                       Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugDomainsExamplesPageEmpty14}
\begin{paste}{ugDomainsExamplesPageEmpty14}{ugDomainsExamplesPagePatch14}
\pastebutton{ugDomainsExamplesPageEmpty14}{\showpaste}
\tab{5}\spadcommand{fullDisplay(fm-\%)\bound{x13 }\free{x12 }}
\end{paste}\end{patch}

\begin{patch}{ugDomainsExamplesPagePatch15}
\begin{paste}{ugDomainsExamplesPageFull15}{ugDomainsExamplesPageEmpty15}
\pastebutton{ugDomainsExamplesPageFull15}{\hidepaste}
\tab{5}\spadcommand{m := tm+fm\bound{x14 }\free{x10 x11 }}
\indentrel{3}\begin{verbatim}
   (15)  499
                              Type: Database IndexCard
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugDomainsExamplesPageEmpty15}
\begin{paste}{ugDomainsExamplesPageEmpty15}{ugDomainsExamplesPagePatch15}
```

```
\pastebutton{ugDomainsExamplesPageEmpty15}{\showpaste}
\tab{5}\spadcommand{m := tm+fm\bound{x14 }\free{x10 x11 }}
\end{paste}\end{patch}

\begin{patch}{ugDomainsExamplesPagePatch16}
\begin{paste}{ugDomainsExamplesPageFull16}{ugDomainsExamplesPageEmpty16}
\pastebutton{ugDomainsExamplesPageFull16}{\hidepaste}
\tab{5}\spadcommand{fullDisplay(m, 202, 205)\free{x14 }}
\indentrel{3}\begin{verbatim}
   elt : (_$,List(Integer),List(Integer))->_$
      from MatrixCategory(R,Row,Col)
   elt : (_$,Integer,Integer,R)->R
      from RectangularMatrixCategory(m,n,R,Row,Col)
   elt
      :
      (_$,NonNegativeInteger,NonNegativeInteger,NonNegati
      veInteger)->R
      from ThreeDimensionalMatrix(R)
   eval
      :
      (Matrix(Expression(DoubleFloat)),List(Symbol),Vecto
      r(Expression(DoubleFloat)))->Matrix(Expression(Doub
      leFloat))
      from d02AgentsPackage
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugDomainsExamplesPageEmpty16}
\begin{paste}{ugDomainsExamplesPageEmpty16}{ugDomainsExamplesPagePatch16}
\pastebutton{ugDomainsExamplesPageEmpty16}{\showpaste}
\tab{5}\spadcommand{fullDisplay(m, 202, 205)\free{x14 }}
\end{paste}\end{patch}

\begin{patch}{ugDomainsExamplesPagePatch17}
\begin{paste}{ugDomainsExamplesPageFull17}{ugDomainsExamplesPageEmpty17}
\pastebutton{ugDomainsExamplesPageFull17}{\hidepaste}
\tab{5}\spadcommand{elt(elt(elt(m,name),unique),count)\free{x14 }}
\indentrel{3}\begin{verbatim}
   (17)  317
                                  Type: PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugDomainsExamplesPageEmpty17}
\begin{paste}{ugDomainsExamplesPageEmpty17}{ugDomainsExamplesPagePatch17}
\pastebutton{ugDomainsExamplesPageEmpty17}{\showpaste}
\tab{5}\spadcommand{elt(elt(elt(m,name),unique),count)\free{x14 }}
\end{paste}\end{patch}
```

Chapter 17

Users Guide Chapter 14 (ug14.ht)

Browse

⇒ "notitle" (ugBrowseStartPage) 17 on page 2472
⇒ "notitle" (ugBrowseDomainPage) 17 on page 2474
⇒ "notitle" (ugBrowseMiscellaneousFeaturesPage) 17 on page 2485

— **ug14.ht** —

```
\begin{page}{ugBrowsePage}{14. Browse}
\beginscroll

This chapter discusses the \Browse{}
component of Hyperdoc.
We suggest you invoke Axiom and work through this
chapter, section by section, following our examples to gain some
familiarity with \Browse{}.

\beginmenu
    \menudownlink{{14.1. The Front Page: Searching the Library}}
{ugBrowseStartPage}
    \menudownlink{{14.2. The Constructor Page}}{ugBrowseDomainPage}
    \menudownlink{{14.3. Miscellaneous Features of Browse}}
{ugBrowseMiscellaneousFeaturesPage}
\endmenu
\endscroll
\autobuttons
\end{page}
```

———-

The Front Page: Searching the Library

⇒ "notitle" (ugBrowseCapitalizationConventionPage) 17 on page 2490
⇒ "notitle" (ugBrowseDomainPage) 17 on page 2474
⇒ "notitle" (ugBrowseViewsOfConstructorsPage) 17 on page 2482
⇒ "notitle" (ugBrowseViewsOfOperationsPage) 17 on page 2487

— **ug14.ht** —

```
\begin{page}{ugBrowseStartPage}
{14.1. The Front Page: Searching the Library}
\beginscroll
To enter \Browse{}, click on {\bf Browse} on the top level page
of Hyperdoc to get the {\it front page} of \Browse{}.
%
%324pt is 4.5",180pt is 2.5",432pt is 6"=textwidth,54=(432-324)/2
%ps files are 4.5"x2.5" except source 4.5"x2.5"
%

To use this page, you first enter a \spadgloss{search string} into
the input area at the top, then click on one of the buttons below.
We show the use of each of the buttons by example.

\subsubsection{Constructors}

First enter the search string {\tt Matrix} into the input area and
click on {\bf Constructors}.  What you get is the {\it constructor
page} for \axiomType{Matrix}.  We show and describe this page in
detail in \downlink{``The Constructor Page''}{ugBrowseDomainPage} in
Section 14.2\ignore{ugBrowseDomain}.  By convention,
Axiom does a case-insensitive search for a match.  Thus {\tt matrix}
is just as good as {\tt Matrix}, has the same effect as {\tt MaTrix},
and so on.  We recommend that you generally use small letters for
names however.  A search string with only capital letters has a
special meaning (see
\downlink{``Capitalization Convention''}
{ugBrowseCapitalizationConventionPage}
in Section 14.3.3\ignore{ugBrowseCapitalizationConvention}).


Click on \UpBitmap{} to return to the \Browse{} front page.

Use the symbol ``{\tt *}'' in search strings as a \spadgloss{wild
card}.
A wild card matches any substring, including the empty string.
For example, enter the search string {\tt *matrix*} into the input
```

```
area and click on {\bf Constructors}.\footnote{To get only
categories, domains, or packages, rather than all constructors,
you can click on the corresponding button to the right of {\bf
Constructors}.}
What you get is a table of all constructors whose names contain
the string ``{\tt matrix}.''

All constructors containing the string are listed, whether
\spadglossSee{exposed}{expose} or \spadglossSee{unexposed}{expose}.
You can hide the names of the unexposed constructors by clicking on
the {\it *=}{\bf unexposed} button in the {\it Views} panel at the
bottom of the window.  (The button will change to {\bf exposed} {\it
only}.)

One of the names in this table is \axiomType{Matrix}.  Click on
\axiomType{Matrix}.  What you get is again the constructor page for
\axiomType{Matrix}.  As you see, \Browse{} gives you a large network
of information in which there are many ways to reach the same pages.

Again click on the \UpBitmap{} to return to the table of constructors
whose names contain {\tt matrix}.  Below the table is a {\it Views}
panel.  This panel contains buttons that let you view constructors in
different ways.  To learn about views of constructors, skip to
\downlink{``Views Of Constructors''}{ugBrowseViewsOfConstructorsPage}
in Section 14.2.3\ignore{ugBrowseViewsOfConstructors}.

Click on \UpBitmap{} to return to the \Browse{} front page.

\subsubsection{Operations}

Enter {\tt *matrix} into the input area and click on {\bf Operations}.
This time you get a table of {\it operations} whose names end with
{\tt matrix} or {\tt Matrix}.


If you select an operation name, you go to a page describing all the
operations in Axiom of that name.  At the bottom of an operation page
is another kind of {\it Views} panel, one for operation pages.  To
learn more about these views, skip to
\downlink{``Views of Operations''}{ugBrowseViewsOfOperationsPage}
in Section 14.3.2\ignore{ugBrowseViewsOfOperations}.

Click on \UpBitmap{} to return to the \Browse{} front page.

\subsubsection{Attributes}

This button gives you a table of attribute names that match the
search string. Enter the search string {\tt *} and click on
{\bf Attributes} to get a list
of all system attributes.
```

```
Click on \UpBitmap{} to return to the \Browse{} front page.

Again there is a {\it Views} panel at the bottom with buttons that let
you view the attributes in different ways.

\subsubsection{General}

This button does a general search for all constructor, operation, and
attribute names matching the search string.  Enter the search string
\allowbreak {\tt *matrix*} into the input area.  Click on {\bf
General} to find all constructs that have {\tt matrix} as a part of
their name.


The summary gives you all the names under a heading when the number of
entries is less than 10.

Click on \UpBitmap{} to return to the \Browse{} front page.

\subsubsection{Documentation}

Again enter the search key {\tt *matrix*} and this time click on
{\bf Documentation}.
This search matches any constructor, operation, or attribute
name whose documentation contains a substring matching {\tt
matrix}.


Click on \UpBitmap{} to return to the \Browse{} front page.

\subsubsection{Complete}

This search combines both {\bf General} and {\bf Documentation}.

\endscroll
\autobuttons
\end{page}
```

———-

The Constructor Page

— ug14.ht —

```
\begin{page}{ugBrowseDomainPage}{14.2. The Constructor Page}
\beginscroll
```

```
In this section we look in detail at a constructor page for domain
\axiomType{Matrix}.
Enter {\tt matrix} into the input area on the main \Browse{} page
and click on {\bf Constructors}.



The header part tells you that \axiomType{Matrix} has abbreviation
\axiomType{MATRIX} and one argument called {\tt R} that must be a
domain of category \axiomType{Ring}.
Just what domains can be arguments of \axiomType{Matrix}?
To find this out, click on the {\tt R} on the second line of the
heading.
What you get is a table of all acceptable domain parameter values
of {\tt R}, or a table of \spadgloss{rings} in Axiom.


Click on \UpBitmap{} to return to the constructor page for
\axiomType{Matrix}.
\texht{\newpage}{}

If you have access to the source code of Axiom, the third
line of the heading gives you the name of the source file
containing the definition of \axiomType{Matrix}.
Click on it to pop up an editor window containing the source code
of \axiomType{Matrix}.


We recommend that you leave the editor window up while working
through this chapter as you occasionally may want to refer to it.
\texht{\newpage}{}

\beginmenu
\menudownlink{{14.2.1. Constructor Page Buttons}}
{ugBrowseDomainButtonsPage}
\menudownlink{{14.2.2. Cross Reference}}
{ugBrowseCrossReferencePage}
\menudownlink{{14.2.3. Views Of Constructors}}
{ugBrowseViewsOfConstructorsPage}
\menudownlink{{14.2.4. Giving Parameters to Constructors}}
{ugBrowseGivingParametersPage}
\endmenu
\endscroll
\autobuttons
\end{page}
```

———

Constructor Page Buttons

⇒ "notitle" (ugBrowseViewsOfOperationsPage) 17 on page 2487

— **ug14.ht** —

```
\begin{page}{ugBrowseDomainButtonsPage}{14.2.1. Constructor Page Buttons}
\beginscroll

We examine each button on this page in order.

\labelSpace{2pc}

\subsubsection{Description}

Click here to bring up a page with a brief description of
constructor \axiomType{Matrix}.
If you have access to system source code, note that these comments
can be found directly over the constructor definition.


\subsubsection{Operations}

Click here to get a table of operations exported by
\axiomType{Matrix}.
You may wish to widen the window to have multiple columns as
below.


If you click on an operation name, you bring up a description page for
the operations.  For a detailed description of these pages, skip to
\downlink{``Views of Operations''}{ugBrowseViewsOfOperationsPage}
in Section 14.3.2\ignore{ugBrowseViewsOfOperations}.

\subsubsection{Attributes}

Click here to get a table of the two attributes exported by
\axiomType{Matrix}:
\spadatt{\texht{fi\-nite\-Ag\-gre\-gate}{finiteAggregate}} and
\spadatt{shallowlyMutable}.
These are two computational properties that result from
\axiomType{Matrix} being regarded as a data structure.

\subsubsection{Examples}

Click here to get an {\it examples page} with examples of operations to
create and manipulate matrices.

Read through this section.
Try selecting the various buttons.
```

```
Notice that if you click on an operation name, such as
\spadfunFrom{new}{Matrix}, you bring up a description page for that
operation from \axiomType{Matrix}.

Example pages have several examples of Axiom commands.
Each example has an active button to its left.
Click on it!
A pre-computed answer is pasted into the page immediately following the
command.
If you click on the button a second time, the answer disappears.
This button thus acts as a toggle:
``now you see it; now you don't.''

Note also that the Axiom commands themselves are active.
If you want to see Axiom execute the command, then click on it!
A new Axiom window appears on your screen and the command is
executed.

\httex{At the end of the page is generally a menu of buttons that lead
you to further sections.
Select one of these topics to explore its contents.}{}

\subsubsection{Exports}

Click here to see a page describing the exports of \axiomType{Matrix}
exactly as described by the source code.


As you see, \axiomType{Matrix} declares that it exports all the operations
and attributes exported by category
\axiomType{MatrixCategory(R, Row, Col)}.
In addition, two operations, \axiomFun{diagonalMatrix} and
\axiomFun{inverse}, are explicitly exported.

To learn a little about the structure of Axiom, we suggest you do
the following exercise.
Otherwise, click on \UpButton{} and go on to the next section.
\axiomType{Matrix} explicitly exports only two operations.
The other operations are thus exports of \axiomType{MatrixCategory}.
In general, operations are usually not explicitly exported by a domain.
Typically they are \spadglossSee{inherited}{inherit} from several
different categories.
Let's find out from where the operations of \axiomType{Matrix} come.

\indent{4}
\beginitems
\item[1. ] Click on {\bf MatrixCategory}, then on {\bf Exports}.
Here you see that {\bf MatrixCategory} explicitly exports many matrix
operations.
Also, it inherits its operations from
```

```
\axiomType{TwoDimensionalArrayCategory}.

\item[2. ] Click on {\bf TwoDimensionalArrayCategory}, then on
{\bf Exports}.  Here you see explicit operations dealing with rows
and columns.  In addition, it inherits operations from
\axiomType{HomogeneousAggregate}.

%\item Click on {\bf HomogeneousAggregate}, then on {\bf Exports}.
%And so on.
%If you continue doing this, eventually you will

\item[3. ] Click on \UpBitmap{} and then
click on {\bf Object}, then on {\bf Exports}, where you see
there are no exports.

\item[4. ] Click on \UpBitmap{} repeatedly to return to the constructor
page for \axiomType{Matrix}.

\enditems
\indent{0}

\subsubsection{Related Operations}

Click here bringing up a table of operations that are exported by
\spadglossSee{packages}{package} but not by \axiomType{Matrix} itself.


To see a table of such packages, use the {\bf Relatives} button on the
{\bf Cross Reference} page described next.


\endscroll
\autobuttons
\end{page}
```

Cross Reference

— ug14.ht —

```
\begin{page}{ugBrowseCrossReferencePage}{14.2.2. Cross Reference}
\beginscroll
Click on the {\bf Cross Reference} button on the main constructor page
for \axiomType{Matrix}.
This gives you a page having various cross reference information stored
under the respective buttons.
```

```
\subsubsection{Parents}

The parents of a domain are the same as the categories mentioned under
the {\bf Exports} button on the first page.
Domain \axiomType{Matrix} has only one parent but in general a domain can
have any number.

\subsubsection{Ancestors}

The \spadglossSee{ancestors}{ancestor} of a constructor consist of its
parents, the parents of its parents, and so on.  Did you perform the
exercise in the last section under {\bf Exports}?  If so, you see here
all the categories you found while ascending the {\bf Exports} chain
for \axiomType{Matrix}.

\subsubsection{Relatives}

The \spadglossSee{relatives}{relative} of a domain constructor are package
constructors that provide operations in addition to those
\spadglossSee{exported}{export} by the domain.

Try this exercise.
\indent{4}
\beginitems
\item[1. ] Click on {\bf Relatives}, bringing up a list of
\spadglossSee{packages}{package}.

\item[2. ] Click on {\bf LinearSystemMatrixPackage} bringing up its
constructor page.\footnote{You may want to widen your Hyperdoc
window to make what follows more legible.}

\item[3. ] Click on {\bf Operations}.
Here you see \axiomFun{rank}, an operation also exported by
\axiomType{Matrix} itself.

\item[4. ] Click on {\bf rank}.
This \spadfunFrom{rank}{LinearSystemMatrixPackage} has two arguments and
thus is different from the \spadfunFrom{rank}{Matrix} from
\axiomType{Matrix}.

\item[5. ] Click on \UpBitmap{} to return to the list of operations for
the package \axiomType{LinearSystemMatrixPackage}.

\item[6. ] Click on {\bf solve} to bring up a
\spadfunFrom{solve}{LinearSystemMatrixPackage} for linear systems of
equations.

\item[7. ] Click on \UpBitmap{} several times to return to the cross
```

```
reference page for \axiomType{Matrix}.
\enditems
\indent{0}

\subsubsection{Dependents}

The \spadglossSee{dependents}{dependent} of a constructor are those
\spadglossSee{domains}{domain} or \spadglossSee{packages}{package}
that mention that constructor either as an argument or in its
\spadglossSee{exports}{export}.

If you click on {\bf Dependents} two entries may surprise you:
\axiomType{RectangularMatrix} and \axiomType{SquareMatrix}.
This happens because \axiomType{Matrix}, as it turns out, appears in
signatures of operations exported by these domains.

\subsubsection{Lineage}

The term \spadgloss{lineage} refers to the {\it search order} for
functions.
If you are an expert user or curious about how the Axiom system
works, try the following exercise.
Otherwise, you best skip this button and go on to {\bf Clients}.

Clicking on {\bf Lineage} gives you a
list of domain constructors:
\axiomType{InnerIndexedTwoDimensionalArray},
\aliascon{MatrixCategory\&}{MATCAT-},
\aliascon{TwoDimensionalArrayCategory\&}{ARR2CAT-},
\aliascon{HomogeneousAggregate\&}{HOAGG-},
\aliascon{Aggregate\&}{AGG-}.
What are these constructors and how are they used?

We explain by an example.  Suppose you create a matrix using the
interpreter, then ask for its \axiomFun{rank}.  Axiom must then find a
function implementing the \axiomFun{rank} operation for matrices.  The
first place Axiom looks for \axiomFun{rank} is in the
\axiomType{Matrix} domain.

If not there, the lineage of \axiomType{Matrix} tells Axiom where else
to look.  Associated with the matrix domain are five other lineage
domains.  Their order is important.  Axiom first searches the first
one, \axiomType{InnerIndexedTwoDimensionalArray}.  If not there, it
searches the second \aliascon{MatrixCategory\&}{MATCAT-}.  And so on.

Where do these {\it lineage constructors} come from?
The source code for \axiomType{Matrix} contains this syntax for the
\spadgloss{function body} of
\axiomType{Matrix}:\footnote{\axiomType{InnerIndexedTwoDimensionalArray}
is a special domain implemented for matrix-like domains to provide
```

```
efficient implementations of \twodim{} arrays.
For example, domains of category \axiomType{TwoDimensionalArrayCategory}
can have any integer as their \spad{minIndex}.
Matrices and other members of this special ``inner'' array have their
\spad{minIndex} defined as \spad{1}.}
\begin{verbatim}
InnerIndexedTwoDimensionalArray(R,mnRow,mnCol,Row,Col)
   add ...
\end{verbatim}
where the ``{\tt ...}'' denotes all the code that follows.
In English, this means:
``The functions for matrices are defined as those from
\axiomType{InnerIndexedTwoDimensionalArray} domain augmented by those
defined in `{\tt ...}',''  where the latter take precedence.

This explains \axiomType{InnerIndexedTwoDimensionalArray}.
The other names, those with names ending with an ampersand \axiomSyntax{\&} are
\spadglossSee{default packages}{default package}
for categories to which \axiomType{Matrix} belongs.
Default packages are ordered by the notion of ``closest ancestor.''

\subsubsection{Clients}

A client of \axiomType{Matrix} is any constructor that uses
\axiomType{Matrix} in its implementation.
For example, \axiomType{Complex} is a client of \axiomType{Matrix}; it
exports several operations that take matrices as arguments or return
matrices as values.\footnote{A constructor is a client of
\axiomType{Matrix} if it handles any matrix.
For example, a constructor having internal (unexported) operations
dealing with matrices is also a client.}

\subsubsection{Benefactors}

A \spadgloss{benefactor} of \axiomType{Matrix} is any constructor that
\axiomType{Matrix} uses in its implementation.
This information, like that for clients, is gathered from run-time
structures.\footnote{The benefactors exclude constructors such as
\axiomType{PrimitiveArray} whose operations macro-expand and so vanish
from sight!}

Cross reference pages for categories have some different buttons on
them.
Starting with the constructor page of \axiomType{Matrix}, click on
\axiomType{Ring} producing its constructor page.
Click on {\bf Cross Reference},
producing the cross-reference page for \axiomType{Ring}.
Here are buttons {\bf Parents} and {\bf Ancestors} similar to the notion
for domains, except for categories the relationship between parent and
child is defined through \spadgloss{category extension}.
```

```
\subsubsection{Children}

Category hierarchies go both ways.
There are children as well as parents.
A child can have any number of parents, but always at least one.
Every category is therefore a descendant of exactly one category:
\axiomType{Object}.

\subsubsection{Descendants}

These are children, children of children, and so on.

Category hierarchies are complicated by the fact that categories take
parameters.
Where a parameterized category fits into a hierarchy {\it may} depend on
values of its parameters.
In general, the set of categories in Axiom forms a {\it directed
acyclic graph}, that is, a graph with directed arcs and no cycles.

\subsubsection{Domains}

This produces a table of all domain constructors that can possibly be
rings (members of category \axiomType{Ring}).
Some domains are unconditional rings.
Others are rings for some parameters and not for others.
To find out which, select the {\bf conditions} button in the views
panel.
For example, \axiomType{DirectProduct(n, R)} is a ring if {\tt R} is a
ring.



\endscroll
\autobuttons
\end{page}
```

———-

Views Of Constructors

— ug14.ht —

```
\begin{page}{ugBrowseViewsOfConstructorsPage}{14.2.3. Views Of Constructors}
\beginscroll

Below every constructor table page is a {\it Views} panel.
```

```
As an example, click on {\bf Cross Reference} from
the constructor page of \axiomType{Matrix},
then on {\bf Benefactors} to produce a
short table of constructor names.

The {\it Views} panel is at the bottom of the page.
Two items, {\it names} and {\it conditions,} are in italics.
Others are active buttons.
The active buttons are those that give you useful alternative views
on this table of constructors.
Once you select a view, you notice that the button turns
off (becomes italicized) so that you cannot reselect it.

\subsubsection{names}

This view gives you a table of names.
Selecting any of these names brings up the constructor page for that
constructor.

\subsubsection{abbrs}

This view gives you a table of abbreviations, in the same order as the
original constructor names.
nAbbreviations are in capitals and are limited to 7 characters.
They can be used interchangeably with constructor names in input areas.

\subsubsection{kinds}

This view organizes constructor names into
the three kinds: categories, domains and packages.

\subsubsection{files}

This view gives a table of file names for the source
code of the constructors in alphabetic order after removing
duplicates.

\subsubsection{parameters}

This view presents constructors with the arguments.
This view of the benefactors of \axiomType{Matrix} shows that
\axiomType{Matrix} uses as many as five different \axiomType{List} domains
in its implementation.

\subsubsection{filter}

This button is used to refine the list of names or abbreviations.
Starting with the {\it names} view, enter {\tt m*} into the input area
and click on {\bf filter}.
You then get a shorter table with only the names beginning with {\tt m}.
```

```
\subsubsection{documentation}

This gives you documentation for each of the constructors.

\subsubsection{conditions}

This page organizes the constructors according to predicates.  The
view is not available for your example page since all constructors are
unconditional.  For a table with conditions, return to the {\bf Cross
Reference} page for \axiomType{Matrix}, click on {\bf Ancestors}, then
on {\bf conditions} in the view panel.  This page shows you that
\axiomType{CoercibleTo(OutputForm)} and \axiomType{SetCategory} are
ancestors of \axiomType{Matrix(R)} only if {\tt R} belongs to category
\axiomType{SetCategory}.

\endscroll
\autobuttons
\end{page}
```

———-

Giving Parameters to Constructors

— ug14.ht —

```
\begin{page}{ugBrowseGivingParametersPage}
{14.2.4. Giving Parameters to Constructors}
\beginscroll
%***********************************************************************

Notice the input area at the bottom of the constructor page.  If you
leave this blank, then the information you get is for the domain
constructor \axiomType{Matrix(R)}, that is, \axiomType{Matrix} for an
arbitrary underlying domain {\tt R}.

In general, however, the exports and other information {\it do} usually
depend on the actual value of {\tt R}.
For example, \axiomType{Matrix} exports the \axiomFun{inverse} operation
only if the domain {\tt R} is a \axiomType{Field}.
To see this, try this from the main constructor page:

\indent{4}
\beginitems
\item[1. ] Enter {\tt Integer} into the input area at the bottom of
the page.
```

```
\item[2. ] Click on {\bf Operations}, producing a table of operations.
Note the number of operation names that appear at the top of the
page.

\item[3. ] Click on \UpBitmap{} to return to the constructor page.

\item[4. ] Use the
\texht{\fbox{\bf Delete}}{{\bf Delete}}
or
\texht{\fbox{\bf Backspace}}{{\bf Backspace}}
keys to erase {\tt Integer} from the input area.

\item[5. ] Click on {\bf Operations} to produce a new table of operations.
Look at the number of operations you get.
This number is greater than what you had before.
Find, for example, the operation \axiomFun{inverse}.

\item[6. ] Click on {\bf inverse} to produce a page describing the
operation \axiomFun{inverse}.  At the bottom of the description, you
notice that the {\bf Conditions} line says ``{\tt R} has
\axiomType{Field}.''  This operation is {\it not} exported by
\axiomType{Matrix(Integer)} since \axiomType{Integer} is not a
\spadgloss{field}.

Try putting the name of a domain such as \axiomType{Fraction Integer}
(which is a field) into the input area, then clicking on {\bf Operations}.
As you see, the operation \axiomFun{inverse} is exported.
\enditems
\indent{0}

\endscroll
\autobuttons
\end{page}
```

———-

Miscellaneous Features of Browse

— **ug14.ht** —

```
\begin{page}{ugBrowseMiscellaneousFeaturesPage}
{14.3. Miscellaneous Features of Browse}
\beginscroll
\labelSpace{4pc}
\beginmenu
\menudownlink{{14.3.1. The Description Page for Operations}}
{ugBrowseDescriptionPagePage}
```

```
\menudownlink{{14.3.2. Views of Operations}}
{ugBrowseViewsOfOperationsPage}
\menudownlink{{14.3.3. Capitalization Convention}}
{ugBrowseCapitalizationConventionPage}
\endmenu
\endscroll
\autobuttons
\end{page}
```

———-

The Description Page for Operations

— ug14.ht —

```
\begin{page}{ugBrowseDescriptionPagePage}
{14.3.1. The Description Page for Operations}
\beginscroll
From the constructor page of \axiomType{Matrix},
click on {\bf Operations} to bring up the table of operations
for \axiomType{Matrix}.

Find the operation {\bf inverse} in the table and click on it.
This takes you to a page showing the documentation for this operation.


Here is the significance of the headings you see.

\subsubsection{Arguments}

This lists each of the arguments of the operation in turn,
paraphrasing the \spadgloss{signature} of the operation.  As for
signatures, a \axiomSyntax{\$} is used to designate {\em this domain},
that is, \axiomType{Matrix(R)}.

\subsubsection{Returns}

This describes the return value for the operation, analogous to the {\bf
Arguments} part.

\subsubsection{Origin}

This tells you which domain or category explicitly exports the
operation.
In this example, the domain itself is the {\it Origin}.
```

```
\subsubsection{Conditions}

This tells you that the operation is exported by
\axiomType{Matrix(R)} only if
``{\tt R} has \axiomType{Field},'' that is, ``{\tt R} is a member of
category \axiomType{Field}.''
When no {\bf Conditions} part is given, the operation is exported for
all values of {\tt R}.

\subsubsection{Description}

Here are the \axiomSyntax{++} comments
that appear in the source code of its {\it Origin}, here
\axiomType{Matrix}.
You find these comments in the source code for \axiomType{Matrix}.


Click on \UpBitmap{} to return to the table of operations.
Click on {\bf map}.
Here you find three different operations named \axiomFun{map}.
This should not surprise you.
Operations are identified by name and \spadgloss{signature}.
There are three operations named \axiomFun{map}, each with
different signatures.
What you see is the {\it descriptions} view of the operations.
If you like, select the button in the heading of one of these
descriptions to get {\it only} that operation.

\subsubsection{Where}

This part qualifies domain parameters mentioned in the arguments to the
operation.

\endscroll
\autobuttons
\end{page}
```

Views of Operations

— ug14.ht —

```
\begin{page}{ugBrowseViewsOfOperationsPage}{14.3.2. Views of Operations}
\beginscroll

We suggest that you go to the constructor page for \axiomType{Matrix}
```

```
and click on {\bf Operations} to bring up a table of operations
with a {\it Views} panel at the bottom.

\subsubsection{names}

This view lists the names of the operations.
Unlike constructors, however, there may be several operations with the
same name.
The heading for the page tells you the number of unique names and the
number of distinct operations when these numbers are different.

\subsubsection{filter}

As for constructors, you can use this button to cut down the list of
operations you are looking at.
Enter, for example, {\tt m*} into the input area to the right of {\bf
filter} then click on {\bf filter}.
As usual, any logical expression is permitted.
For example, use
\begin{verbatim}
*! or *?
\end{verbatim}
to get a list of destructive operations and predicates.

\subsubsection{documentation}

This gives you the most information:
a detailed description of all the operations in the form you have seen
before.
Every other button summarizes these operations in some form.

\subsubsection{signatures}

This views the operations by showing their signatures.

\subsubsection{parameters}

This views the operations by their distinct syntactic forms with
parameters.

\subsubsection{origins}

This organizes the operations according to the constructor that
explicitly exports them.

\subsubsection{conditions}

This view organizes the operations into conditional and unconditional
operations.
```

```
\subsubsection{usage}

This button is only available if your user-level is set to {\it
development}.
The {\bf usage} button produces a table of constructors that reference this
operation.\footnote{Axiom requires an especially long time to
produce this table, so anticipate this when requesting this
information.}

\subsubsection{implementation}

This button is only available if your user-level is set to {\it
development}.
If you enter values for all domain parameters on the constructor page,
then the {\bf implementation} button appears in place of the {\bf
conditions} button.
This button tells you what domains or packages actually implement the
various operations.\footnote{This button often takes a long time; expect
a delay while you wait for an answer.}

With your user-level set to {\it development}, we suggest you try this
exercise.
Return to the main constructor page for \axiomType{Matrix}, then enter
{\tt Integer} into the input area at the bottom as the value of {\tt R}.
Then click on {\bf Operations} to produce a table of operations.
Note that the {\bf conditions} part of the {\it Views} table is
replaced by {\bf implementation}.
Click on {\bf implementation}.
After some delay, you get a page describing what implements each of
the matrix operations, organized by the various domains and packages.


\subsubsection{generalize}

This button only appears for an operation page of a constructor
involving a unique operation name.

From an operations page for \axiomType{Matrix}, select any
operation name, say {\bf rank}.
In the views panel, the {\bf filter} button is  replaced by
{\bf generalize}.
Click on it!
%% Replaced {\bf threshold} with 10 below.  MGR 1995oct31
What you get is a description of all Axiom operations
named \axiomFun{rank}.\footnote{If there were more than 10
operations of the name, you get instead a page
with a {\it Views} panel at the bottom and the message to {\bf
Select a view below}.
To get the descriptions of all these operations as mentioned
above, select the {\bf description} button.}
```

```
\subsubsection{all domains}

This button only appears on an operation page resulting from a
search from the front page of \Browse{} or from selecting
{\bf generalize} from an operation page for a constructor.

Note that the {\bf filter} button in the {\it Views} panel is
replaced by {\bf all domains}.
Click on it to produce a table of {\it all} domains or packages that
export a \axiomFun{rank} operation.


We note that this table specifically refers to all the \axiomFun{rank}
operations shown in the preceding page.
Return to the descriptions of all the \axiomFun{rank} operations and
select one of them by clicking on the button in its heading.
Select {\bf all domains}.
As you see, you have a smaller table of constructors.
When there is only one constructor, you get the
constructor page for that constructor.
\texht{\newpage}{}

\endscroll
\autobuttons
\end{page}
```

———-

Capitalization Convention

— ug14.ht —

```
\begin{page}{ugBrowseCapitalizationConventionPage}
{14.3.3. Capitalization Convention}
\beginscroll

When entering search keys for constructors, you can use capital
letters to search for abbreviations.
For example, enter {\tt UTS} into the input area and click on {\bf
Constructors}.
Up comes a page describing \axiomType{UnivariateTaylorSeries}
whose abbreviation is \axiomType{UTS}.

Constructor abbreviations always have three or more capital
letters.
For short constructor names (six letters or less), abbreviations
```

```
are not generally helpful as their abbreviation is typically the
constructor name in capitals.
For example, the abbreviation for \axiomType{Matrix} is
\axiomType{MATRIX}.

Abbreviations can also contain numbers.
For example, \axiomType{POLY2} is the abbreviation for constructor
\axiomType{PolynomialFunctions2}.
For default packages, the abbreviation is the same as the
abbreviation for the corresponding category with the ``\&''
replaced by ``-''.
For example, for the category default package
\aliascon{MatrixCategory\&}{MATCAT-} the abbreviation is
\axiomType{MATCAT-} since the corresponding category
\axiomType{MatrixCategory} has abbreviation \axiomType{MATCAT}.

\endscroll
\autobuttons
\end{page}
```

———-

Chapter 18

Users Guide Chapter 15 (ug15.ht)

What's New in Axiom Version 2.0

— ug15.ht —

```
\begin{page}{ugWhatsNewPage}{15. What's New in Axiom Version 2.0}
\beginscroll

Many things have changed in this new version of Axiom and
we describe many of the more important topics here.

\beginmenu
\menudownlink{{15.1. Important Things to Read First}}
{ugWhatsNewImportantPage}
\menudownlink{{15.3. The NAG Library Link}}{nagLinkIntroPage}
\menudownlink{{15.4. Interactive Front-end and Language}}
{ugWhatsNewLanguagePage}
\menudownlink{{15.5. Library}}{ugWhatsNewLibraryPage}
\menudownlink{{15.6. \HyperName}}{ugWhatsNewHyperDocPage}
\menudownlink{{15.7. Documentation}}{ugWhatsNewDocumentationPage}
\endmenu
\endscroll
\autobuttons
\end{page}
```

Important Things to Read First

— **ug15.ht** —

```
\begin{page}{ugWhatsNewImportantPage}
{15.1. Important Things to Read First}
\beginscroll

If you have any private {\tt .spad} files (that is, library files
which were not shipped with Axiom) you will need to
recompile them.  For example, if you wrote the file {\tt
regress.spad} then you should issue {\tt )compile regress.spad}
before trying to use it.

The internal representation of \axiomType{Union}  has changed.
This means that \texht{\linebreak}{} Axiom data saved
with Release 1.x may not
be readable by this Release. If you cannot recreate the saved data
by recomputing in Release 2.0, please contact NAG for assistance.

\endscroll
\autobuttons
\end{page}
```

———

The NAG Library Link

⇒ “notitle” (htxl1) 3.63 on page 845
⇒ “notitle” (nagDocumentationPage) 18 on page 2495
⇒ “notitle” (nagLinkUsagePage) 18 on page 2498
⇒ “notitle” (aspSectionPage) 18 on page 2501
⇒ “notitle” (generalFortranPage) 18 on page 2505
⇒ “notitle” (nagTechnicalPage) 18 on page 2530

— **ug15.ht** —

```
\begin{page}{nagLinkIntroPage}{15.3. The NAG Library Link}
\beginscroll

The \naglib{} link allows you to call NAG Fortran
routines from within Axiom, passing Axiom objects as parameters
and getting them back as results.

The \naglib{} and, consequently, the link are divided into {\em
chapters}, which cover different areas of numerical analysis.  The
```

```
statistical and sorting {\em chapters} of the Library, however, are
not included in the link and various support and utility routines
(mainly the F06 and X {\em chapters}) have been omitted.

Each {\em chapter} has a short (at most three-letter) name; for
example, the {\em chapter} devoted to the solution of ordinary
differential equations is called D02.  When using the link via the
\downlink{Hyperdoc interface}{htxl1}, you will be presented with a
complete menu of these {\em chapters}. The names of individual
routines within each {\em chapter} are formed by adding three letters
to the {\em chapter} name, so for example the routine for solving ODEs
by Adams method is called
\axiomFunFrom{d02cjf}{NagOrdinaryDifferentialEquationsPackage}.

\beginmenu
    \menudownlink{{15.3.1. Interpreting NAG Documentation}}
{nagDocumentationPage}
    \menudownlink{{15.3.2. Using the Link}}{nagLinkUsagePage}
    \menudownlink{{15.3.3. Providing values for Argument Subprograms}}
{aspSectionPage}
    \menudownlink{{15.3.4. General Fortran-generation utilities in Axiom}}
{generalFortranPage}
    \menudownlink{{15.3.5. Some technical information}}{nagTechnicalPage}
\endmenu
\endscroll
\autobuttons
\end{page}
```

———-

Interpreting NAG Documentation

⇒ "notitle" (manpageXXintro) 22.1 on page 2727
⇒ "notitle" (manpageXXonline) 22.1 on page 2703
⇒ "notitle" (FoundationLibraryDocPage) 3.1 on page 125
⇒ "notitle" (aspSectionPage) 18 on page 2501

— **ug15.ht** —

```
\begin{page}{nagDocumentationPage}{15.3.1. Interpreting NAG Documentation}
\beginscroll

Information about using the \naglib{} in general, and about using
individual routines in particular, can be accessed via Hyperdoc.
This documentation refers to the Fortran routines directly; the
purpose of this subsection is to explain how this corresponds to the
Axiom routines.
```

```
For general information about the \naglib{} users should consult
\downlink{Essential Introduction to the NAG Foundation Library}
{manpageXXintro}.
The documentation is in ASCII format, and a description of the conventions
used to represent mathematical symbols is given in
\downlink{Introduction to NAG On-Line Documentation}{manpageXXonline}.
Advice about choosing a routine from a particular {\em chapter} can
be found in the \downlink{Chapter Documents}{FoundationLibraryDocPage}.

\subsubsection{Correspondence Between Fortran and Axiom types}

The NAG documentation refers to the Fortran types of objects; in
general, the correspondence to Axiom types is as follows.
\indent{4}
\beginitems
\item[-] Fortran INTEGER corresponds to Axiom \axiomType{Integer}.
\item[-] Fortran DOUBLE PRECISION corresponds to Axiom
\axiomType{DoubleFloat}.
\item[-] Fortran COMPLEX corresponds to Axiom
\axiomType{Complex DoubleFloat}.
\item[-] Fortran LOGICAL corresponds to Axiom \axiomType{Boolean}.
\item[-] Fortran CHARACTER*(*) corresponds to Axiom \axiomType{String}.
\enditems
\indent{0}
(Exceptionally, for NAG EXTERNAL parameters -- ASPs in link parlance
-- REAL and COMPLEX correspond to \axiomType{MachineFloat} and
\axiomType{MachineComplex},
respectively; see
\downlink{``Providing values for Argument Subprograms''}
{aspSectionPage} in Section 15.3.3\ignore{aspSection}.)

The correspondence for aggregates is as follows.
\indent{4}
\beginitems
\item[-] A one-dimensional Fortran array corresponds to an Axiom
\texht{\linebreak}{}
      \axiomType{Matrix} with one column.
\item[-] A two-dimensional Fortran ARRAY corresponds to an Axiom
\texht{\linebreak}{}
      \axiomType{Matrix}.
\item[-] A three-dimensional Fortran ARRAY corresponds to an Axiom
\texht{\linebreak}{}
      \axiomType{ThreeDimensionalMatrix}.
\enditems
\indent{0}
Higher-dimensional arrays are not currently needed for the \naglib{}.

Arguments which are Fortran FUNCTIONs or SUBROUTINEs correspond to
special ASP domains in Axiom. See
```

```
\downlink{``Providing values for Argument Subprograms''}
{aspSectionPage} in Section 15.3.3\ignore{aspSection}.

\subsubsection{Classification of NAG parameters}

NAG parameters are classified as belonging to one (or more) of the
following categories: {\tt Input}, {\tt Output}, {\tt Workspace} or
{\tt External} procedure.  Within {\tt External} procedures a similar
classification is used, and parameters may also be {\tt Dummies}, or
{\tt User Workspace} (data structures not used by the NAG routine but
provided for the convenience of the user).

When calling a NAG routine via the link the user only provides values
for {\tt Input} and {\tt External} parameters.

The order of the parameters is, in general, different from  the order
specified in the \naglib{} documentation. The Browser description
for each routine helps in determining the correspondence. As a rule of
thumb, {\tt Input} parameters come first followed by {\tt Input/Output}
parameters. The {\tt External} parameters are always found at the end.


\subsubsection{IFAIL}

NAG routines often return diagnostic information through a parameter
called \axiom{ifail}.  With a few exceptions, the principle is that on
input \axiom{ifail} takes one of the values $-1,0,1$.  This determines
how the routine behaves when it encounters an error:
\indent{4}
\beginitems
\item[-] a value of 1 causes the NAG routine to return without printing
an error message;
\item[-] a value of 0 causes the NAG routine to print an error message
and abort;
\item[-] a value of -1 causes the NAG routine to return and print an
error message.
\enditems
\indent{0}

The user is STRONGLY ADVISED to set \axiom{ifail} to \texht{$-1$}{-1}
when using the link.  If \axiom{ifail} has been set to \texht{$1$}{1}
or \texht{$-1$}{-1} on input, then its value on output will determine
the possible cause of any error.  A value of \texht{$0$}{0} indicates
successful completion, otherwise it provides an index into a table of
diagnostics provided as part of the routine documentation (accessible
via \Browse{}).

\endscroll
\autobuttons
\end{page}
```

———-

Using the Link

⇒ "notitle" (htxl1) 3.63 on page 845

— **ug15.ht** —

```
\begin{page}{nagLinkUsagePage}{15.3.2. Using the Link}
\beginscroll

The easiest way to use the link is via the
\downlink{Hyperdoc interface}{htxl1}.
You will be presented with a set of fill-in forms where
you can specify the parameters for each call.  Initially, the forms
contain example values, demonstrating the use of each routine (these,
in fact, correspond to the standard NAG example program for the
routine in question).  For some parameters, these values can provide
reasonable defaults; others, of course, represent data.  When you
change a parameter which controls the size of an array, the data in
that array are reset to a ``neutral'' value -- usually zero.

When you are satisfied with the values entered, clicking on the
``Continue'' button will display the Axiom command needed to
run the chosen NAG routine with these values.  Clicking on the
``Do It'' button will then cause Axiom to execute this command
and return the result in the parent Axiom session, as described
below.  Note that, for some routines, multiple HyperDoc ``pages'' are
required, due to the structure of the data.  For these, returning to
an earlier page causes HyperDoc to reset the later pages (this is a
general feature of HyperDoc); in such a case, the simplest way to
repeat a call, varying a parameter on an earlier page, is probably to
modify the call displayed in the parent session.

An alternative approach is to call NAG routines directly in your
normal Axiom session (that is, using the Axiom
interpreter).  Such calls return an
object of type \axiomType{Result}.  As not
all parameters in the underlying NAG routine are required in the
Axiom call (and the parameter ordering may be different), before
calling a NAG routine you should consult the description of the
Axiom operation in the Browser.  (The quickest route to this
is to type the routine name, in lower case, into the Browser's
input area, then click on {\tt Operations}.)  The parameter names
used coincide with NAG's, although they will appear here in lower
case.  Of course, it is also possible to become familiar with the
Axiom form of a routine by first using it through the
```

```
\downlink{Hyperdoc interface}{htxl1}.

\xtc{
As an example of this mode of working, we can find a zero
of a function, lying between 3 and 4, as follows:
}{
\spadpaste{answer:=c05adf(3.0,4.0,1.0e-5,0.0,-1,sin(X)::ASP1(F))
\bound{answer} }
}
\xtc{
By default, \axiomType{Result} only displays the type of returned values,
since the amount of information returned can be quite large.  Individual
components can be examined as follows:
}{
\spadpaste{answer . x\free{answer}}
}
\xtc{
}{
\spadpaste{answer . ifail\free{answer}}
}
\xtc{
In order to avoid conflict with names defined in the workspace, you
can also get the values by using the \axiomType{String} type (the
interpreter automatically coerces them to \axiomType{Symbol})
}{
\spadpaste{answer "x"\free{answer}}
}


It is possible to have Axiom display the values of scalar or array
results automatically.  For more details, see the commands
\axiomFunFrom{showScalarValues}{Result}
and \axiomFunFrom{showArrayValues}{Result}.

\xtc{
There is also a {\bf .input} file for each NAG routine, containing
Axiom interpreter commands to set up and run the standard NAG
example for that routine.
}{
\spadpaste{)read c05adf.input}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{nagLinkUsagePagePatch1}
\begin{paste}{nagLinkUsagePageFull1}{nagLinkUsagePageEmpty1}
\pastebutton{nagLinkUsagePageFull1}{\hidepaste}
\tab{5}\spadcommand{answer:=c05adf(3.0,4.0,1.0e-5,0.0,-1,sin(X)::ASP1(F))\bound{answer }}
```

```
\indentrel{3}\begin{verbatim}
   (1)  [ifail: Integer,x: DoubleFloat]
                                              Type: Result
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{nagLinkUsagePageEmpty1}
\begin{paste}{nagLinkUsagePageEmpty1}{nagLinkUsagePagePatch1}
\pastebutton{nagLinkUsagePageEmpty1}{\showpaste}
\tab{5}\spadcommand{answer:=c05adf(3.0,4.0,1.0e-5,0.0,-1,sin(X)::ASP1(F))\bound{answer }}
\end{paste}\end{patch}

\begin{patch}{nagLinkUsagePagePatch2}
\begin{paste}{nagLinkUsagePageFull2}{nagLinkUsagePageEmpty2}
\pastebutton{nagLinkUsagePageFull2}{\hidepaste}
\tab{5}\spadcommand{answer . x\free{answer }}
\indentrel{3}\begin{verbatim}
   (2)  3.14159265545896
                                         Type: DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{nagLinkUsagePageEmpty2}
\begin{paste}{nagLinkUsagePageEmpty2}{nagLinkUsagePagePatch2}
\pastebutton{nagLinkUsagePageEmpty2}{\showpaste}
\tab{5}\spadcommand{answer . x\free{answer }}
\end{paste}\end{patch}

\begin{patch}{nagLinkUsagePagePatch3}
\begin{paste}{nagLinkUsagePageFull3}{nagLinkUsagePageEmpty3}
\pastebutton{nagLinkUsagePageFull3}{\hidepaste}
\tab{5}\spadcommand{answer . ifail\free{answer }}
\indentrel{3}\begin{verbatim}
   (3)  0
                                             Type: Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{nagLinkUsagePageEmpty3}
\begin{paste}{nagLinkUsagePageEmpty3}{nagLinkUsagePagePatch3}
\pastebutton{nagLinkUsagePageEmpty3}{\showpaste}
\tab{5}\spadcommand{answer . ifail\free{answer }}
\end{paste}\end{patch}

\begin{patch}{nagLinkUsagePagePatch4}
\begin{paste}{nagLinkUsagePageFull4}{nagLinkUsagePageEmpty4}
\pastebutton{nagLinkUsagePageFull4}{\hidepaste}
\tab{5}\spadcommand{answer "x"\free{answer }}
\indentrel{3}\begin{verbatim}
   (4)  3.14159265545896
```

```
                                      Type: DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{nagLinkUsagePageEmpty4}
\begin{paste}{nagLinkUsagePageEmpty4}{nagLinkUsagePagePatch4}
\pastebutton{nagLinkUsagePageEmpty4}{\showpaste}
\tab{5}\spadcommand{answer "x"\free{answer }}
\end{paste}\end{patch}

\begin{patch}{nagLinkUsagePagePatch5}
\begin{paste}{nagLinkUsagePageFull5}{nagLinkUsagePageEmpty5}
\pastebutton{nagLinkUsagePageFull5}{\hidepaste}
\tab{5}\spadcommand{)read c05adf.input}
\indentrel{3}\begin{verbatim}
   (1)  true
                                          Type: Boolean
   (2)  true
                                          Type: Boolean
   (3)  F
                                           Type: Asp1 F
   (4)  0.0
                                      Type: DoubleFloat
   (5)  1.0
                                      Type: DoubleFloat
   (6)  1.0e-05
                                      Type: DoubleFloat
   (7)  0.0
                                      Type: DoubleFloat
   (8)  [ifail: 0,x: 0.567143306604963]
                                           Type: Result
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{nagLinkUsagePageEmpty5}
\begin{paste}{nagLinkUsagePageEmpty5}{nagLinkUsagePagePatch5}
\pastebutton{nagLinkUsagePageEmpty5}{\showpaste}
\tab{5}\spadcommand{)read c05adf.input}
\end{paste}\end{patch}
```

————-

Providing values for Argument Subprograms

⇒ "notitle" (generalFortranPage) 18 on page 2505

— **ug15.ht** —

```
\begin{page}{aspSectionPage}
{15.3.3. Providing values for Argument Subprograms}
\beginscroll

There are a number of ways in which users can provide values for argument
subprograms (ASPs).  At the top level the user will see that NAG routines
require
an object from the \axiomType{Union} of a \axiomType{Filename} and an ASP.
\xtc{
For example \axiomFun{c05adf} requires an object of type
\texht{\linebreak}{} \axiomType{Union}(fn:
\axiomType{FileName},fp: \axiomType{Asp1 F})
}{
\spadpaste{)display operation c05adf}
}

The user thus has a choice of providing the name of a file containing
Fortran source code, or of somehow generating the ASP within Axiom.
If a filename is specified, it is searched for in the {\it local}
machine, i.e., the machine that Axiom is running on.

\subsubsection{Providing ASPs via \axiomType{FortranExpression}}

The \axiomType{FortranExpression} domain is used to represent
expressions which can be translated into Fortran under certain
circumstances.  It is very similar to \axiomType{Expression} except
that only operators which exist in Fortran can be used, and only
certain variables can occur.  For example the instantiation
\axiomType{FortranExpression([X],[M],MachineFloat)} is the domain of
expressions containing the scalar \axiom{X} and the array \axiom{M}.
\xtc{
This allows us to create expressions like:
}{
\spadpaste{f : FortranExpression([X],[M],MachineFloat) := sin(X)+M[3,1]}
}
\xtc{
but not
}{
\spadpaste{f : FortranExpression([X],[M],MachineFloat) := sin(M)+Y}
}

Those ASPs which represent expressions usually export a
\axiomFun{coerce} from an appropriate instantiation of
\axiomType{FortranExpression} (or perhaps \axiomType{Vector
FortranExpression} etc.).  For convenience there are also retractions
from appropriate instantiations of \axiomType{Expression},
\axiomType{Polynomial} and \axiomType{Fraction Polynomial}.

\subsubsection{Providing ASPs via \axiomType{FortranCode}}
```

```
\texht{\exptypeindex{FortranCode}}{} \axiomType{FortranCode} allows us
to build arbitrarily complex ASPs via a kind of pseudo-code.  It is
described fully in
\downlink{``General Fortran-generation utilities in Axiom''}
{generalFortranPage} in Section 15.3.4\ignore{generalFortran}.

Every ASP exports two \axiomFun{coerce} functions: one from
\axiomType{FortranCode} and one from \axiomType{List FortranCode}.
There is also a \axiomFun{coerce} from \texht{\linebreak}{}
\axiomType{Record( localSymbols: SymbolTable, code: List FortranCode)}
which is used for passing extra symbol information about the ASP.

\xtc{
So for example, to integrate the function abs(x) we could use the
built-in \axiomFun{abs} function.  But suppose we want to get back to
basics and define it directly, then we could do the following:
}{
\spadpaste{d01ajf(-1.0, 1.0, 0.0, 1.0e-5, 800, 200, -1, cond(LT(X,0),
assign(F,-X), assign(F,X))) result }
}

The \axiomFunFrom{cond}{FortranCode} operation creates a conditional
clause and the \axiomFunFrom{assign}{FortranCode} an assignment
statement.

\subsubsection{Providing ASPs via \axiomType{FileName}}

Suppose we have created the file ``asp.f'' as follows:
\begin{verbatim}
      DOUBLE PRECISION FUNCTION F(X)
      DOUBLE PRECISION X
      F=4.0D0/(X*X+1.0D0)
      RETURN
      END
\end{verbatim}
and wish to pass it to the NAG
routine \axiomFun{d01ajf} which performs one-dimensional quadrature.
We can do this as follows:
\begin{verbatim}
d01ajf(0.0 ,1.0, 0.0, 1.0e-5, 800, 200, -1, "asp.f")
\end{verbatim}

\endscroll
\autobuttons
\end{page}

\begin{patch}{aspSectionPagePatch1}
\begin{paste}{aspSectionPageFull1}{aspSectionPageEmpty1}
\pastebutton{aspSectionPageFull1}{\hidepaste}
\tab{5}\spadcommand{)display operation c05adf}
```

```
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{aspSectionPageEmpty1}
\begin{paste}{aspSectionPageEmpty1}{aspSectionPagePatch1}
\pastebutton{aspSectionPageEmpty1}{\showpaste}
\tab{5}\spadcommand{)display operation c05adf}
\end{paste}\end{patch}

\begin{patch}{aspSectionPagePatch2}
\begin{paste}{aspSectionPageFull2}{aspSectionPageEmpty2}
\pastebutton{aspSectionPageFull2}{\hidepaste}
\tab{5}\spadcommand{f : FortranExpression([X],[M],MachineFloat) := sin(X)+M[3,1]}
\indentrel{3}\begin{verbatim}
   (1)  sin(X) + M
                  3,1
          Type: FortranExpression([X],[M],MachineFloat)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{aspSectionPageEmpty2}
\begin{paste}{aspSectionPageEmpty2}{aspSectionPagePatch2}
\pastebutton{aspSectionPageEmpty2}{\showpaste}
\tab{5}\spadcommand{f : FortranExpression([X],[M],MachineFloat) := sin(X)+M[3,1]}
\end{paste}\end{patch}

\begin{patch}{aspSectionPagePatch3}
\begin{paste}{aspSectionPageFull3}{aspSectionPageEmpty3}
\pastebutton{aspSectionPageFull3}{\hidepaste}
\tab{5}\spadcommand{f : FortranExpression([X],[M],MachineFloat) := sin(M)+Y}
\indentrel{3}\begin{verbatim}
   sin(M) + Y
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{aspSectionPageEmpty3}
\begin{paste}{aspSectionPageEmpty3}{aspSectionPagePatch3}
\pastebutton{aspSectionPageEmpty3}{\showpaste}
\tab{5}\spadcommand{f : FortranExpression([X],[M],MachineFloat) := sin(M)+Y}
\end{paste}\end{patch}

\begin{patch}{aspSectionPagePatch4}
\begin{paste}{aspSectionPageFull4}{aspSectionPageEmpty4}
\pastebutton{aspSectionPageFull4}{\hidepaste}
\tab{5}\spadcommand{d01ajf(-1.0, 1.0, 0.0, 1.0e-5, 800, 200, -1, cond(LT(X,0), assign(F,-X), ass
\indentrel{3}\begin{verbatim}
   (2)  1.0
                                      Type: DoubleFloat
\end{verbatim}
```

```
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{aspSectionPageEmpty4}
\begin{paste}{aspSectionPageEmpty4}{aspSectionPagePatch4}
\pastebutton{aspSectionPageEmpty4}{\showpaste}
\tab{5}\spadcommand{d01ajf(-1.0, 1.0, 0.0, 1.0e-5, 800, 200, -1, cond(LT(X,0), assign(F,-X), assign(F,X)))
\end{paste}\end{patch}
```

General Fortran-generation utilities in Axiom

— ug15.ht —

```
\begin{page}{generalFortranPage}
{15.3.4. General Fortran-generation utilities in Axiom}
\beginscroll

This section describes more advanced facilities which are available to
users who wish to generate Fortran code from within Axiom.  There are
facilities to manipulate templates, store type information, and
generate code fragments or complete programs.

\subsubsection{Template Manipulation}

A template is a skeletal program which is ``fleshed out'' with data when
it is processed.  It is a sequence of {\em active} and {\em passive} parts:
active parts are sequences of Axiom commands which are processed as if they
had been typed into the interpreter; passive parts are simply echoed
verbatim on the Fortran output stream.

Suppose, for example, that we have the following template, stored in
the file ``test.tem'':
\begin{verbatim}
-- A simple template
beginVerbatim
      DOUBLE PRECISION FUNCTION F(X)
      DOUBLE PRECISION X
endVerbatim
outputAsFortran("F",f)
beginVerbatim
      RETURN
      END
endVerbatim
\end{verbatim}
The passive parts lie between the two tokens {\tt beginVerbatim} and
\texht{\linebreak}{} {\tt endVerbatim}.  There are two active
```

```
statements: one which is simply an Axiom ( \texht{\verb+--+}{\-\-})
comment, and one which produces an assignment to the current value of
{\tt f}.  We could use it as follows:
\begin{verbatim}
(4) ->f := 4.0/(1+X**2)

          4
   (4)   ------
          2
         X  + 1

(5) ->processTemplate "test.tem"
      DOUBLE PRECISION FUNCTION F(X)
      DOUBLE PRECISION X
      F=4.0D0/(X*X+1.0D0)
      RETURN
      END

   (5)  "CONSOLE"
\end{verbatim}

(A more reliable method of specifying the filename will be introduced
below.)  Note that the Fortran assignment {\tt F=4.0D0/(X*X+1.0D0)}
automatically converted 4.0 and 1 into DOUBLE PRECISION numbers; in
general, the Axiom Fortran generation facility will convert
anything which should be a floating point object into either
a Fortran REAL or DOUBLE PRECISION object.
\xtc{
Which alternative is used is determined by the command
}{
\spadpaste{)set fortran precision}
}

It is sometimes useful to end a template before the file itself ends
(e.g. to allow the template to be tested incrementally or so that a
piece of text describing how the template works can be included).  It
is of course possible to ``comment-out'' the remainder of the file.
Alternatively, the single token {\tt endInput} as part of an active
portion of the template will cause processing to be ended prematurely
at that point.

The \axiomFun{processTemplate} command comes in two flavours.  In the
first case, illustrated above, it takes one argument of domain
\axiomType{FileName}, the name of the template to be processed, and
writes its output on the current Fortran output stream.  In general, a
filename can be generated from {\em directory}, {\em name} and {\em
extension} components, using the operation \axiomFun{filename}, as in
\begin{verbatim}
processTemplate filename("","test","tem")
\end{verbatim}
```

```
There is an alternative version of \axiomFun{processTemplate}, which
takes two arguments (both of domain \axiomType{FileName}).  In this
case the first argument is the name of the template to be processed,
and the second is the file in which to write the results.  Both
versions return the location of the generated Fortran code as their
result ({\tt "CONSOLE"} in the above example).

It is sometimes useful to be able to mix active and passive parts of a
line or statement.  For example you might want to generate a Fortran
Comment describing your data set.  For this kind of application we
provide three functions as follows:
\texht
{
\begin{tabular}{p{1.8in}p{2.6in}}
\axiomFun{fortranLiteral} & writes a string on the Fortran output
stream \\
 & \\
\axiomFun{fortranCarriageReturn} & writes a carriage return on the
Fortran output stream \\
& \\
\axiomFun{fortranLiteralLine} & writes a string followed by a return
on the Fortran output stream \\
\end{tabular}
}
{
\newline
\axiomFun{fortranLiteral}\tab{25}writes a string on the Fortran
output stream\newline
\axiomFun{fortranCarriageReturn}\tab{25}writes a carriage return
on the Fortran output stream\newline
\axiomFun{fortranLiteralLine}\tab{25}writes a string followed by
a return on the Fortran output stream\newline
}
\xtc{
So we could create our comment as follows:
}{
\spadpaste{m := matrix [[1,2,3],[4,5,6]]\bound{m}}
}
\xtc{
}{
\spadpaste{fortranLiteralLine concat
["C\ \ \ \ \ \ The\ Matrix\ has\ ", nrows(m)::String, "\ rows\
and\ ", ncols(m)::String, "\ columns"]\free{m}}
}
\xtc{
or, alternatively:
}{
\spadpaste{fortranLiteral "C\ \ \ \ \ \ The\ Matrix\ has\ "}
}
\xtc{
```

```
}{
\spadpaste{fortranLiteral(nrows(m)::String)}
}
\xtc{
}{
\spadpaste{fortranLiteral "\ rows\ and\ "}
}
\xtc{
}{
\spadpaste{fortranLiteral(ncols(m)::String)\free{m}}
}
\xtc{
}{
\spadpaste{fortranLiteral "\ columns"}
}
\xtc{
}{
\spadpaste{fortranCarriageReturn()}
}

We should stress that these functions, together with the
\axiomFun{outputAsFortran} function are the {\em only} sure ways of
getting output to appear on the Fortran output stream.  Attempts to
use Axiom commands such as \axiomFun{output} or \axiomFunX{writeline}
may appear to give the required result when displayed on the console,
but will give the wrong result when Fortran and algebraic output are
sent to differing locations.  On the other hand, these functions can
be used to send helpful messages to the user, without interfering with
the generated Fortran.

\subsubsection{Manipulating the Fortran Output Stream}
\texht{\exptypeindex{FortranOutputStackPackage}}{}

Sometimes it is useful to manipulate the Fortran output stream in a
program, possibly without being aware of its current value.  The main
use of this is for gathering type declarations (see ``Fortran Types''
below) but it can be useful in other contexts as well.  Thus we
provide a set of commands to manipulate a stack of (open) output
streams.  Only one stream can be written to at any given time.  The
stack is never empty---its initial value is the console or the current
value of the Fortran output stream, and can be determined using
\xtc{
}{
\spadpaste{topFortranOutputStack()}
}
(see below).
The commands available to manipulate the stack are:
\texht{
\begin{tabular}{ll}
\axiomFun{clearFortranOutputStack} & resets the stack to the console \\
```

```
 & \\
\axiomFun{pushFortranOutputStack} & pushes a
\axiomType{FileName} onto the stack \\
 & \\
\axiomFun{popFortranOutputStack} & pops the stack \\
 & \\
\axiomFun{showFortranOutputStack} & returns the current stack \\
 & \\
\axiomFun{topFortranOutputStack} & returns the top element of the stack \\
\end{tabular}
}
{
\newline
\axiomFun{clearFortranOutputStack}\tab{25}resets the stack\newline
\axiomFun{pushFortranOutputStack}\tab{25}pushes a
\axiomType{FileName} onto the stack\newline
\axiomFun{popFortranOutputStack}\tab{25}pops the stack\newline
\axiomFun{showFortranOutputStack}\tab{25}returns the current stack\newline
\axiomFun{topFortranOutputStack}\tab{25}returns the
top element of the stack\newline
}
These commands are all part of \axiomType{FortranOutputStackPackage}.

\subsubsection{Fortran Types}

When generating code it is important to keep track of the Fortran
types of the objects which we are generating.  This is useful for a
number of reasons, not least to ensure that we are actually generating
legal Fortran code.  The current type system is built up in several
layers, and we shall describe each in turn.

\subsubsection{FortranScalarType}
\texht{\exptypeindex{FortranScalarType}}{}

This domain represents the simple Fortran datatypes: REAL,
DOUBLE PRECISION, COMPLEX, LOGICAL, INTEGER, and CHARACTER.
It is possible to \axiomFun{coerce} a \axiomType{String} or
\axiomType{Symbol} into the domain, test whether two objects are
equal, and also apply the predicate functions
\axiomFunFrom{real?}{FortranScalarType} etc.

\subsubsection{FortranType}
\texht{\exptypeindex{FortranType}}{}

This domain represents ``full'' types: i.e., datatype plus array
dimensions (where appropriate) plus whether or not the parameter is an
external subprogram.  It is possible to \axiomFun{coerce} an object of
\axiomType{FortranScalarType} into the domain or \axiomFun{construct}
one from an element of \axiomType{FortranScalarType}, a list of
\axiomType{Polynomial Integer}s (which can of course be simple
```

```
integers or symbols) representing its dimensions, and a
\axiomType{Boolean} declaring whether it is external or not.  The list
of dimensions must be empty if the \axiomType{Boolean} is {\tt true}.
The functions \axiomFun{scalarTypeOf}, \axiomFun{dimensionsOf} and
\axiomFun{external?} return the appropriate parts, and it is possible
to get the various basic Fortran Types via functions like
\axiomFun{fortranReal}.
\xtc{
For example:
}{
\spadpaste{type:=construct(real,[i,10],false)$FortranType}
}
\xtc{
or
}{
\spadpaste{type:=[real,[i,10],false]$FortranType\bound{type}}
}
\xtc{
}{
\spadpaste{scalarTypeOf type\free{type}}
}
\xtc{
}{
\spadpaste{dimensionsOf type\free{type}}
}
\xtc{
}{
\spadpaste{external?  type\free{type}}
}
\xtc{
}{
\spadpaste{fortranLogical()}
}
\xtc{
}{
\spadpaste{construct(integer,[],true)$FortranType}
}

\subsubsection{SymbolTable}
\texht{\exptypeindex{SymbolTable}}{}

This domain creates and manipulates a symbol table for generated
Fortran code.  This is used by \axiomType{FortranProgram} to represent
the types of objects in a subprogram.  The commands available are:
\texht{
\begin{tabular}{ll}
\axiomFun{empty} & creates a new \axiomType{SymbolTable} \\
 & \\
\axiomFunX{declare} & creates a new entry in a table \\
 & \\
```

```
\axiomFun{fortranTypeOf} & returns the type of an object in a table \\
 & \\
\axiomFun{parametersOf} & returns a list of all the symbols in the table \\
 & \\
\axiomFun{typeList} & returns a list of all objects of a given type \\
 & \\
\axiomFun{typeLists} &
returns a list of lists of all objects sorted by type \\
 & \\
\axiomFun{externalList} & returns a list of all {\tt EXTERNAL} objects \\
 & \\
\axiomFun{printTypes} & produces Fortran type declarations from a table\\
\end{tabular}
}
{
\newline
\axiomFun{empty}\tab{25}creates a new \axiomType{SymbolTable}\newline
\axiomFunX{declare}\tab{25}creates a new entry in a table \newline
\axiomFun{fortranTypeOf}
\tab{25}returns the type of an object in a table \newline
\axiomFun{parametersOf}
\tab{25}returns a list of all the symbols in the table \newline
\axiomFun{typeList}
\tab{25}returns a list of all objects of a given type \newline
\axiomFun{typeLists}
\tab{25}returns a list of lists of all objects sorted by type \newline
\axiomFun{externalList}
\tab{25}returns a list of all {\tt EXTERNAL} objects \newline
\axiomFun{printTypes}
\tab{25}produces Fortran type declarations from a table\newline
}
\xtc{
}{
\spadpaste{symbols := empty()$SymbolTable\bound{symbols}}
}
\xtc{
}{
\spadpaste{declare!(X,fortranReal(),symbols)\free{symbols}}
}
\xtc{
}{
\spadpaste{declare!(M,construct(real,[i,j],false)$FortranType,symbols)
\free{symbols}}
}
\xtc{
}{
\spadpaste{declare!([i,j],fortranInteger(),symbols)\free{symbols}}
}
\xtc{
}{
```

```
\spadpaste{symbols\free{symbols}}
}
\xtc{
}{
\spadpaste{fortranTypeOf(i,symbols)\free{symbols}}
}
\xtc{
}{
\spadpaste{typeList(real,symbols)\free{symbols}}
}
\xtc{
}{
\spadpaste{printTypes symbols\free{symbols}}
}

\subsubsection{TheSymbolTable}
\texht{\exptypeindex{TheSymbolTable}}{}

This domain creates and manipulates one global symbol table to be
used, for example, during template processing. It is also used when
linking to external Fortran routines. The information stored for each
subprogram (and the main program segment, where relevant) is:
\indent{4}
\beginitems
\item[-] its name;
\item[-] its return type;
\item[-] its argument list;
\item[-] and its argument types.
\enditems
\indent{0}
Initially, any information provided is deemed to be for the main program
segment.
\xtc{
Issuing the following command indicates that from now on all information
refers to the subprogram \axiom{F}.
}{
\spadpaste{newSubProgram F}
}
\xtc{
It is possible to return to processing the main program segment by issuing
the command:
}{
\spadpaste{endSubProgram()}
}
The following commands exist:
\texht{
\begin{tabular}{p{1.6in}p{2.8in}}
\axiomFunX{returnType}
& declares the return type of the current subprogram \\
 & \\
```

```
\axiomFun{returnTypeOf}
& returns the return type of a subprogram \\
 & \\
\axiomFunX{argumentList}
&  declares the argument list of the current subprogram \\
 & \\
\axiomFun{argumentListOf}
&  returns the argument list of a subprogram \\
 & \\
\axiomFunX{declare}
& provides type declarations for parameters of the current subprogram \\
 & \\
\axiomFun{symbolTableOf}
& returns the symbol table  of a subprogram \\
 & \\
\axiomFun{printHeader}
& produces the Fortran header for the current subprogram \\
\end{tabular}
}
{
\newline
\axiomFunX{returnType}
\tab{25}declares the return type of the current subprogram \newline
\axiomFun{returnTypeOf}
\tab{25}returns the return type of a subprogram \newline
\axiomFunX{argumentList}
\tab{25}declares the argument list of the current subprogram \newline
\axiomFun{argumentListOf}
\tab{25}returns the argument list of a subprogram \newline
\axiomFunX{declare}
\tab{25}provides type declarations for parameters of the
current subprogram \newline
\axiomFun{symbolTableOf}\tab{25}returns the symbol table
of a subprogram \newline
\axiomFun{printHeader}\tab{25}produce the Fortran header
for the current subprogram \newline
}
In addition there are versions of these commands which are
parameterised by the name of a subprogram, and others parameterised by
both the name of a subprogram and by an instance of
\axiomType{TheSymbolTable}.
\xtc{
}{
\spadpaste{newSubProgram F \bound{forPleasure}}
}
\xtc{
}{
\spadpaste{argumentList!(F,[X])\free{forPleasure}}
}
\xtc{
```

```
}{
\spadpaste{returnType!(F,real)\free{forPleasure}}
}
\xtc{
}{
\spadpaste{declare!(X,fortranReal(),F)\free{forPleasure}}
}
\xtc{
}{
\spadpaste{printHeader F\free{forPleasure}}
}

\subsubsection{Advanced Fortran Code Generation}

This section describes facilities for representing Fortran statements, and
building up complete subprograms from them.

\subsubsection{Switch}
\texht{\exptypeindex{Switch}}{}

This domain is used to represent statements like {\tt x < y}.  Although
these can be represented directly in Axiom, it is a little cumbersome,
since Axiom evaluates the last statement, for example, to \axiom{true}
(since \axiom{x} is  lexicographically less than \axiom{y}).

Instead we have a set of operations,
such as \axiomFun{LT} to represent \axiom{<},
to let us build such statements.  The available constructors are:
\texht{
\centerline{{\begin{tabular}{ll}}}
\centerline{{\axiomFun{LT} & $<$ }}
\centerline{{\axiomFun{GT} & $>$ }}
\centerline{{\axiomFun{LE} & $\leq$ }}
\centerline{{\axiomFun{GE} & $\geq$ }}
\centerline{{\axiomFun{EQ} & $=$ }}
\centerline{{\axiomFun{AND} & $and$}}
\centerline{{\axiomFun{OR} & $or$ }}
\centerline{{\axiomFun{NOT} & $not$ }}
\centerline{{\end{tabular}}}
}
{
\newline
\axiomFun{LT}\tab{25}\texht{$<$}{<} \newline
\axiomFun{GT}\tab{25}\texht{$>$}{>} \newline
\axiomFun{LE}\tab{25}\texht{$\leq$}{<=} \newline
\axiomFun{GE}\tab{25}\texht{$\geq$}{>=} \newline
\axiomFun{EQ}\tab{25}\texht{$=$}{=} \newline
\axiomFun{AND}\tab{25}\texht{$and$}{{\tt and}}\newline
\axiomFun{OR}\tab{25}\texht{$or$}{{\tt or}} \newline
\axiomFun{NOT}\tab{25}\texht{$not$}{{\tt not}} \newline
```

```
}
\xtc{
So for example:
}{
\spadpaste{LT(x,y)}
}

\subsubsection{FortranCode}

This domain represents code segments or operations: currently assignments,
conditionals, blocks, comments, gotos, continues, various kinds of loops,
and return statements.
\xtc{
For example we can create quite a complicated conditional statement using
assignments, and then turn it into Fortran code:
}{
\spadpaste{
c := cond(LT(X,Y),assign(F,X),cond(GT(Y,Z),assign(F,Y),assign(F,Z)))
\bound{c}}
}
\xtc{
}{
\spadpaste{printCode c\free{c}}
}

The Fortran code is printed
on the current Fortran output stream.

\subsubsection{FortranProgram}
\texht{\exptypeindex{FortranProgram}}{}

This domain is used to construct complete Fortran subprograms out of
elements of \axiomType{FortranCode}.  It is parameterised by the name
of the target subprogram (a \axiomType{Symbol}), its return type (from
\axiomType{Union}(\axiomType{FortranScalarType},``void'')), its
arguments (from \axiomType{List Symbol}), and its symbol table (from
\axiomType{SymbolTable}).  One can \axiomFun{coerce} elements of
either \axiomType{FortranCode} or \axiomType{Expression} into it.

\xtc{
First of all we create a symbol table:
}{
\spadpaste{symbols := empty()$SymbolTable\bound{symbols}}
}
\xtc{
Now put some type declarations into it:
}{
\spadpaste{declare!([X,Y],fortranReal(),symbols)\free{symbols}}
}
\xtc{
```

```
Then (for convenience)
we set up the particular instantiation of \axiomType{FortranProgram}
}{
\spadpaste{FP := FortranProgram(F,real,[X,Y],symbols)\free{symbols}
\bound{FP}}
}
\xtc{
Create an object of type \axiomType{Expression(Integer)}:
}{
\spadpaste{asp := X*sin(Y)\bound{asp}}
}
\xtc{
Now \axiomFun{coerce} it into \axiomType{FP}, and print its Fortran form:
}{
\spadpaste{outputAsFortran(asp::FP)\free{FP asp}}
}

We can generate a \axiomType{FortranProgram} using \axiom{FortranCode}.
For example:
\xtc{
Augment our symbol table:
}{
\spadpaste{declare!(Z,fortranReal(),symbols)\free{symbols}\bound{Z}}
}
\xtc{
and transform the conditional expression we prepared earlier:
}{
\spadpaste{outputAsFortran([c,returns()]::FP) \free{FP c Z}}
}

\endscroll
\autobuttons
\end{page}

\begin{patch}{generalFortranPagePatch1}
\begin{paste}{generalFortranPageFull1}{generalFortranPageEmpty1}
\pastebutton{generalFortranPageFull1}{\hidepaste}
\tab{5}\spadcommand{)set fortran precision}
\indentrel{3}\begin{verbatim}
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{generalFortranPageEmpty1}
\begin{paste}{generalFortranPageEmpty1}{generalFortranPagePatch1}
\pastebutton{generalFortranPageEmpty1}{\showpaste}
\tab{5}\spadcommand{)set fortran precision}
\end{paste}\end{patch}

\begin{patch}{generalFortranPagePatch2}
\begin{paste}{generalFortranPageFull2}{generalFortranPageEmpty2}
```

```
\pastebutton{generalFortranPageFull2}{\hidepaste}
\tab{5}\spadcommand{m := matrix [[1,2,3],[4,5,6]]\bound{m }}
\indentrel{3}\begin{verbatim}
        1  2  3
   (1)
        4  5  6
                                   Type: Matrix Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{generalFortranPageEmpty2}
\begin{paste}{generalFortranPageEmpty2}{generalFortranPagePatch2}
\pastebutton{generalFortranPageEmpty2}{\showpaste}
\tab{5}\spadcommand{m := matrix [[1,2,3],[4,5,6]]\bound{m }}
\end{paste}\end{patch}

\begin{patch}{generalFortranPagePatch3}
\begin{paste}{generalFortranPageFull3}{generalFortranPageEmpty3}
\pastebutton{generalFortranPageFull3}{\hidepaste}
\tab{5}\spadcommand{fortranLiteralLine concat ["C      The Matrix has ", nrows(m)::String, " rows and ", ncc
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{generalFortranPageEmpty3}
\begin{paste}{generalFortranPageEmpty3}{generalFortranPagePatch3}
\pastebutton{generalFortranPageEmpty3}{\showpaste}
\tab{5}\spadcommand{fortranLiteralLine concat ["C      The Matrix has ", nrows(m)::String, " rows and ", ncc
\end{paste}\end{patch}

\begin{patch}{generalFortranPagePatch4}
\begin{paste}{generalFortranPageFull4}{generalFortranPageEmpty4}
\pastebutton{generalFortranPageFull4}{\hidepaste}
\tab{5}\spadcommand{fortranLiteral "C      The Matrix has "}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{generalFortranPageEmpty4}
\begin{paste}{generalFortranPageEmpty4}{generalFortranPagePatch4}
\pastebutton{generalFortranPageEmpty4}{\showpaste}
\tab{5}\spadcommand{fortranLiteral "C      The Matrix has "}
\end{paste}\end{patch}

\begin{patch}{generalFortranPagePatch5}
\begin{paste}{generalFortranPageFull5}{generalFortranPageEmpty5}
\pastebutton{generalFortranPageFull5}{\hidepaste}
\tab{5}\spadcommand{fortranLiteral(nrows(m)::String)}
```

```
\indentrel{3}\begin{verbatim}
                                          Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{generalFortranPageEmpty5}
\begin{paste}{generalFortranPageEmpty5}{generalFortranPagePatch5}
\pastebutton{generalFortranPageEmpty5}{\showpaste}
\tab{5}\spadcommand{fortranLiteral(nrows(m)::String)}
\end{paste}\end{patch}

\begin{patch}{generalFortranPagePatch6}
\begin{paste}{generalFortranPageFull6}{generalFortranPageEmpty6}
\pastebutton{generalFortranPageFull6}{\hidepaste}
\tab{5}\spadcommand{fortranLiteral " rows and "}
\indentrel{3}\begin{verbatim}
                                          Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{generalFortranPageEmpty6}
\begin{paste}{generalFortranPageEmpty6}{generalFortranPagePatch6}
\pastebutton{generalFortranPageEmpty6}{\showpaste}
\tab{5}\spadcommand{fortranLiteral " rows and "}
\end{paste}\end{patch}

\begin{patch}{generalFortranPagePatch7}
\begin{paste}{generalFortranPageFull7}{generalFortranPageEmpty7}
\pastebutton{generalFortranPageFull7}{\hidepaste}
\tab{5}\spadcommand{fortranLiteral(ncols(m)::String)\free{m }}
\indentrel{3}\begin{verbatim}
                                          Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{generalFortranPageEmpty7}
\begin{paste}{generalFortranPageEmpty7}{generalFortranPagePatch7}
\pastebutton{generalFortranPageEmpty7}{\showpaste}
\tab{5}\spadcommand{fortranLiteral(ncols(m)::String)\free{m }}
\end{paste}\end{patch}

\begin{patch}{generalFortranPagePatch8}
\begin{paste}{generalFortranPageFull8}{generalFortranPageEmpty8}
\pastebutton{generalFortranPageFull8}{\hidepaste}
\tab{5}\spadcommand{fortranLiteral " columns"}
\indentrel{3}\begin{verbatim}
                                          Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{generalFortranPageEmpty8}
\begin{paste}{generalFortranPageEmpty8}{generalFortranPagePatch8}
\pastebutton{generalFortranPageEmpty8}{\showpaste}
\tab{5}\spadcommand{fortranLiteral " columns"}
\end{paste}\end{patch}

\begin{patch}{generalFortranPagePatch9}
\begin{paste}{generalFortranPageFull9}{generalFortranPageEmpty9}
\pastebutton{generalFortranPageFull9}{\hidepaste}
\tab{5}\spadcommand{fortranCarriageReturn()}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{generalFortranPageEmpty9}
\begin{paste}{generalFortranPageEmpty9}{generalFortranPagePatch9}
\pastebutton{generalFortranPageEmpty9}{\showpaste}
\tab{5}\spadcommand{fortranCarriageReturn()}
\end{paste}\end{patch}

\begin{patch}{generalFortranPagePatch10}
\begin{paste}{generalFortranPageFull10}{generalFortranPageEmpty10}
\pastebutton{generalFortranPageFull10}{\hidepaste}
\tab{5}\spadcommand{topFortranOutputStack()}
\indentrel{3}\begin{verbatim}
   (9)  "CONSOLE"
                                           Type: String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{generalFortranPageEmpty10}
\begin{paste}{generalFortranPageEmpty10}{generalFortranPagePatch10}
\pastebutton{generalFortranPageEmpty10}{\showpaste}
\tab{5}\spadcommand{topFortranOutputStack()}
\end{paste}\end{patch}

\begin{patch}{generalFortranPagePatch11}
\begin{paste}{generalFortranPageFull11}{generalFortranPageEmpty11}
\pastebutton{generalFortranPageFull11}{\hidepaste}
\tab{5}\spadcommand{type:=construct(real,[i,10],false)$FortranType}
\indentrel{3}\begin{verbatim}
   (10)  REAL
             (i,10)
                                      Type: FortranType
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{generalFortranPageEmpty11}
\begin{paste}{generalFortranPageEmpty11}{generalFortranPagePatch11}
```

```
\pastebutton{generalFortranPageEmpty11}{\showpaste}
\tab{5}\spadcommand{type:=construct(real,[i,10],false)$FortranType}
\end{paste}\end{patch}

\begin{patch}{generalFortranPagePatch12}
\begin{paste}{generalFortranPageFull12}{generalFortranPageEmpty12}
\pastebutton{generalFortranPageFull12}{\hidepaste}
\tab{5}\spadcommand{type:=[real,[i,10],false]$FortranType\bound{type }}
\indentrel{3}\begin{verbatim}
   (11)  REAL
             (i,10)
                                         Type: FortranType
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{generalFortranPageEmpty12}
\begin{paste}{generalFortranPageEmpty12}{generalFortranPagePatch12}
\pastebutton{generalFortranPageEmpty12}{\showpaste}
\tab{5}\spadcommand{type:=[real,[i,10],false]$FortranType\bound{type }}
\end{paste}\end{patch}

\begin{patch}{generalFortranPagePatch13}
\begin{paste}{generalFortranPageFull13}{generalFortranPageEmpty13}
\pastebutton{generalFortranPageFull13}{\hidepaste}
\tab{5}\spadcommand{scalarTypeOf type\free{type }}
\indentrel{3}\begin{verbatim}
   (12)  REAL
                Type: Union(fst: FortranScalarType,...)
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{generalFortranPageEmpty13}
\begin{paste}{generalFortranPageEmpty13}{generalFortranPagePatch13}
\pastebutton{generalFortranPageEmpty13}{\showpaste}
\tab{5}\spadcommand{scalarTypeOf type\free{type }}
\end{paste}\end{patch}

\begin{patch}{generalFortranPagePatch14}
\begin{paste}{generalFortranPageFull14}{generalFortranPageEmpty14}
\pastebutton{generalFortranPageFull14}{\hidepaste}
\tab{5}\spadcommand{dimensionsOf type\free{type }}
\indentrel{3}\begin{verbatim}
   (13)  [i,10]
                           Type: List Polynomial Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{generalFortranPageEmpty14}
\begin{paste}{generalFortranPageEmpty14}{generalFortranPagePatch14}
\pastebutton{generalFortranPageEmpty14}{\showpaste}
```

```
\tab{5}\spadcommand{dimensionsOf type\free{type }}
\end{paste}\end{patch}

\begin{patch}{generalFortranPagePatch15}
\begin{paste}{generalFortranPageFull15}{generalFortranPageEmpty15}
\pastebutton{generalFortranPageFull15}{\hidepaste}
\tab{5}\spadcommand{external? type\free{type }}
\indentrel{3}\begin{verbatim}
   (14)  false
                                           Type: Boolean
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{generalFortranPageEmpty15}
\begin{paste}{generalFortranPageEmpty15}{generalFortranPagePatch15}
\pastebutton{generalFortranPageEmpty15}{\showpaste}
\tab{5}\spadcommand{external? type\free{type }}
\end{paste}\end{patch}

\begin{patch}{generalFortranPagePatch16}
\begin{paste}{generalFortranPageFull16}{generalFortranPageEmpty16}
\pastebutton{generalFortranPageFull16}{\hidepaste}
\tab{5}\spadcommand{fortranLogical()}
\indentrel{3}\begin{verbatim}
   (15)  LOGICAL
                                       Type: FortranType
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{generalFortranPageEmpty16}
\begin{paste}{generalFortranPageEmpty16}{generalFortranPagePatch16}
\pastebutton{generalFortranPageEmpty16}{\showpaste}
\tab{5}\spadcommand{fortranLogical()}
\end{paste}\end{patch}

\begin{patch}{generalFortranPagePatch17}
\begin{paste}{generalFortranPageFull17}{generalFortranPageEmpty17}
\pastebutton{generalFortranPageFull17}{\hidepaste}
\tab{5}\spadcommand{construct(integer,[],true)$FortranType}
\indentrel{3}\begin{verbatim}
   (16)  EXTERNAL INTEGER
                                       Type: FortranType
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{generalFortranPageEmpty17}
\begin{paste}{generalFortranPageEmpty17}{generalFortranPagePatch17}
\pastebutton{generalFortranPageEmpty17}{\showpaste}
\tab{5}\spadcommand{construct(integer,[],true)$FortranType}
\end{paste}\end{patch}
```

```
\begin{patch}{generalFortranPagePatch18}
\begin{paste}{generalFortranPageFull18}{generalFortranPageEmpty18}
\pastebutton{generalFortranPageFull18}{\hidepaste}
\tab{5}\spadcommand{symbols := empty()$SymbolTable\bound{symbols }}
\indentrel{3}\begin{verbatim}
   (17)  table()
                                         Type: SymbolTable
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{generalFortranPageEmpty18}
\begin{paste}{generalFortranPageEmpty18}{generalFortranPagePatch18}
\pastebutton{generalFortranPageEmpty18}{\showpaste}
\tab{5}\spadcommand{symbols := empty()$SymbolTable\bound{symbols }}
\end{paste}\end{patch}

\begin{patch}{generalFortranPagePatch19}
\begin{paste}{generalFortranPageFull19}{generalFortranPageEmpty19}
\pastebutton{generalFortranPageFull19}{\hidepaste}
\tab{5}\spadcommand{declare!(X,fortranReal(),symbols)\free{symbols }}
\indentrel{3}\begin{verbatim}
   (18)  REAL
                                         Type: FortranType
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{generalFortranPageEmpty19}
\begin{paste}{generalFortranPageEmpty19}{generalFortranPagePatch19}
\pastebutton{generalFortranPageEmpty19}{\showpaste}
\tab{5}\spadcommand{declare!(X,fortranReal(),symbols)\free{symbols }}
\end{paste}\end{patch}

\begin{patch}{generalFortranPagePatch20}
\begin{paste}{generalFortranPageFull20}{generalFortranPageEmpty20}
\pastebutton{generalFortranPageFull20}{\hidepaste}
\tab{5}\spadcommand{declare!(M,construct(real,[i,j],false)$FortranType,symbols)\free{symbols }}
\indentrel{3}\begin{verbatim}
   (19)  REAL
             (i,j)
                                         Type: FortranType
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{generalFortranPageEmpty20}
\begin{paste}{generalFortranPageEmpty20}{generalFortranPagePatch20}
\pastebutton{generalFortranPageEmpty20}{\showpaste}
\tab{5}\spadcommand{declare!(M,construct(real,[i,j],false)$FortranType,symbols)\free{symbols }}
\end{paste}\end{patch}
```

```
\begin{patch}{generalFortranPagePatch21}
\begin{paste}{generalFortranPageFull21}{generalFortranPageEmpty21}
\pastebutton{generalFortranPageFull21}{\hidepaste}
\tab{5}\spadcommand{declare!([i,j],fortranInteger(),symbols)\free{symbols }}
\indentrel{3}\begin{verbatim}
   (20)  INTEGER
                                     Type: FortranType
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{generalFortranPageEmpty21}
\begin{paste}{generalFortranPageEmpty21}{generalFortranPagePatch21}
\pastebutton{generalFortranPageEmpty21}{\showpaste}
\tab{5}\spadcommand{declare!([i,j],fortranInteger(),symbols)\free{symbols }}
\end{paste}\end{patch}

\begin{patch}{generalFortranPagePatch22}
\begin{paste}{generalFortranPageFull22}{generalFortranPageEmpty22}
\pastebutton{generalFortranPageFull22}{\hidepaste}
\tab{5}\spadcommand{symbols\free{symbols }}
\indentrel{3}\begin{verbatim}
   (21)
   table(X= REAL,M= REAL     ,i= INTEGER,j= INTEGER)
                        (i,j)
                                     Type: SymbolTable
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{generalFortranPageEmpty22}
\begin{paste}{generalFortranPageEmpty22}{generalFortranPagePatch22}
\pastebutton{generalFortranPageEmpty22}{\showpaste}
\tab{5}\spadcommand{symbols\free{symbols }}
\end{paste}\end{patch}

\begin{patch}{generalFortranPagePatch23}
\begin{paste}{generalFortranPageFull23}{generalFortranPageEmpty23}
\pastebutton{generalFortranPageFull23}{\hidepaste}
\tab{5}\spadcommand{fortranTypeOf(i,symbols)\free{symbols }}
\indentrel{3}\begin{verbatim}
   (22)  INTEGER
                                     Type: FortranType
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{generalFortranPageEmpty23}
\begin{paste}{generalFortranPageEmpty23}{generalFortranPagePatch23}
\pastebutton{generalFortranPageEmpty23}{\showpaste}
\tab{5}\spadcommand{fortranTypeOf(i,symbols)\free{symbols }}
\end{paste}\end{patch}
```

```
\begin{patch}{generalFortranPagePatch24}
\begin{paste}{generalFortranPageFull24}{generalFortranPageEmpty24}
\pastebutton{generalFortranPageFull24}{\hidepaste}
\tab{5}\spadcommand{typeList(real,symbols)\free{symbols }}
\indentrel{3}\begin{verbatim}
   (23)  [X,[M,i,j]]
Type: List Union(name: Symbol,bounds: List Union(S: Symbol,P: Polynomial Integer))
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{generalFortranPageEmpty24}
\begin{paste}{generalFortranPageEmpty24}{generalFortranPagePatch24}
\pastebutton{generalFortranPageEmpty24}{\showpaste}
\tab{5}\spadcommand{typeList(real,symbols)\free{symbols }}
\end{paste}\end{patch}

\begin{patch}{generalFortranPagePatch25}
\begin{paste}{generalFortranPageFull25}{generalFortranPageEmpty25}
\pastebutton{generalFortranPageFull25}{\hidepaste}
\tab{5}\spadcommand{printTypes symbols\free{symbols }}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{generalFortranPageEmpty25}
\begin{paste}{generalFortranPageEmpty25}{generalFortranPagePatch25}
\pastebutton{generalFortranPageEmpty25}{\showpaste}
\tab{5}\spadcommand{printTypes symbols\free{symbols }}
\end{paste}\end{patch}

\begin{patch}{generalFortranPagePatch26}
\begin{paste}{generalFortranPageFull26}{generalFortranPageEmpty26}
\pastebutton{generalFortranPageFull26}{\hidepaste}
\tab{5}\spadcommand{newSubProgram F}
\indentrel{3}\begin{verbatim}
                                             Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{generalFortranPageEmpty26}
\begin{paste}{generalFortranPageEmpty26}{generalFortranPagePatch26}
\pastebutton{generalFortranPageEmpty26}{\showpaste}
\tab{5}\spadcommand{newSubProgram F}
\end{paste}\end{patch}

\begin{patch}{generalFortranPagePatch27}
\begin{paste}{generalFortranPageFull27}{generalFortranPageEmpty27}
\pastebutton{generalFortranPageFull27}{\hidepaste}
\tab{5}\spadcommand{endSubProgram()}
```

```
\indentrel{3}\begin{verbatim}
   (26)  MAIN
                                          Type: Symbol
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{generalFortranPageEmpty27}
\begin{paste}{generalFortranPageEmpty27}{generalFortranPagePatch27}
\pastebutton{generalFortranPageEmpty27}{\showpaste}
\tab{5}\spadcommand{endSubProgram()}
\end{paste}\end{patch}

\begin{patch}{generalFortranPagePatch28}
\begin{paste}{generalFortranPageFull28}{generalFortranPageEmpty28}
\pastebutton{generalFortranPageFull28}{\hidepaste}
\tab{5}\spadcommand{newSubProgram F\bound{forPleasure }}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{generalFortranPageEmpty28}
\begin{paste}{generalFortranPageEmpty28}{generalFortranPagePatch28}
\pastebutton{generalFortranPageEmpty28}{\showpaste}
\tab{5}\spadcommand{newSubProgram F\bound{forPleasure }}
\end{paste}\end{patch}

\begin{patch}{generalFortranPagePatch29}
\begin{paste}{generalFortranPageFull29}{generalFortranPageEmpty29}
\pastebutton{generalFortranPageFull29}{\hidepaste}
\tab{5}\spadcommand{argumentList!(F,[X])\free{forPleasure }}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{generalFortranPageEmpty29}
\begin{paste}{generalFortranPageEmpty29}{generalFortranPagePatch29}
\pastebutton{generalFortranPageEmpty29}{\showpaste}
\tab{5}\spadcommand{argumentList!(F,[X])\free{forPleasure }}
\end{paste}\end{patch}

\begin{patch}{generalFortranPagePatch30}
\begin{paste}{generalFortranPageFull30}{generalFortranPageEmpty30}
\pastebutton{generalFortranPageFull30}{\hidepaste}
\tab{5}\spadcommand{returnType!(F,real)\free{forPleasure }}
\indentrel{3}\begin{verbatim}
                                            Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{generalFortranPageEmpty30}
\begin{paste}{generalFortranPageEmpty30}{generalFortranPagePatch30}
\pastebutton{generalFortranPageEmpty30}{\showpaste}
\tab{5}\spadcommand{returnType!(F,real)\free{forPleasure }}
\end{paste}\end{patch}

\begin{patch}{generalFortranPagePatch31}
\begin{paste}{generalFortranPageFull31}{generalFortranPageEmpty31}
\pastebutton{generalFortranPageFull31}{\hidepaste}
\tab{5}\spadcommand{declare!(X,fortranReal(),F)\free{forPleasure }}
\indentrel{3}\begin{verbatim}
   (30)  REAL
                                       Type: FortranType
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{generalFortranPageEmpty31}
\begin{paste}{generalFortranPageEmpty31}{generalFortranPagePatch31}
\pastebutton{generalFortranPageEmpty31}{\showpaste}
\tab{5}\spadcommand{declare!(X,fortranReal(),F)\free{forPleasure }}
\end{paste}\end{patch}

\begin{patch}{generalFortranPagePatch32}
\begin{paste}{generalFortranPageFull32}{generalFortranPageEmpty32}
\pastebutton{generalFortranPageFull32}{\hidepaste}
\tab{5}\spadcommand{printHeader F\free{forPleasure }}
\indentrel{3}\begin{verbatim}
                                              Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{generalFortranPageEmpty32}
\begin{paste}{generalFortranPageEmpty32}{generalFortranPagePatch32}
\pastebutton{generalFortranPageEmpty32}{\showpaste}
\tab{5}\spadcommand{printHeader F\free{forPleasure }}
\end{paste}\end{patch}

\begin{patch}{generalFortranPagePatch33}
\begin{paste}{generalFortranPageFull33}{generalFortranPageEmpty33}
\pastebutton{generalFortranPageFull33}{\hidepaste}
\tab{5}\spadcommand{LT(x,y)}
\indentrel{3}\begin{verbatim}
   (32)  x < y
                                            Type: Switch
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{generalFortranPageEmpty33}
\begin{paste}{generalFortranPageEmpty33}{generalFortranPagePatch33}
```

```
\pastebutton{generalFortranPageEmpty33}{\showpaste}
\tab{5}\spadcommand{LT(x,y)}
\end{paste}\end{patch}

\begin{patch}{generalFortranPagePatch34}
\begin{paste}{generalFortranPageFull34}{generalFortranPageEmpty34}
\pastebutton{generalFortranPageFull34}{\hidepaste}
\tab{5}\spadcommand{c := cond(LT(X,Y),assign(F,X),cond(GT(Y,Z),assign(F,Y),assign(F,Z)))\bound{c }}
\indentrel{3}\begin{verbatim}
   (33)  conditional
                                    Type: FortranCode
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{generalFortranPageEmpty34}
\begin{paste}{generalFortranPageEmpty34}{generalFortranPagePatch34}
\pastebutton{generalFortranPageEmpty34}{\showpaste}
\tab{5}\spadcommand{c := cond(LT(X,Y),assign(F,X),cond(GT(Y,Z),assign(F,Y),assign(F,Z)))\bound{c }}
\end{paste}\end{patch}

\begin{patch}{generalFortranPagePatch35}
\begin{paste}{generalFortranPageFull35}{generalFortranPageEmpty35}
\pastebutton{generalFortranPageFull35}{\hidepaste}
\tab{5}\spadcommand{printCode c\free{c }}
\indentrel{3}\begin{verbatim}
                                           Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{generalFortranPageEmpty35}
\begin{paste}{generalFortranPageEmpty35}{generalFortranPagePatch35}
\pastebutton{generalFortranPageEmpty35}{\showpaste}
\tab{5}\spadcommand{printCode c\free{c }}
\end{paste}\end{patch}

\begin{patch}{generalFortranPagePatch36}
\begin{paste}{generalFortranPageFull36}{generalFortranPageEmpty36}
\pastebutton{generalFortranPageFull36}{\hidepaste}
\tab{5}\spadcommand{symbols := empty()$SymbolTable\bound{symbols }}
\indentrel{3}\begin{verbatim}
   (35)  table()
                                    Type: SymbolTable
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{generalFortranPageEmpty36}
\begin{paste}{generalFortranPageEmpty36}{generalFortranPagePatch36}
\pastebutton{generalFortranPageEmpty36}{\showpaste}
\tab{5}\spadcommand{symbols := empty()$SymbolTable\bound{symbols }}
\end{paste}\end{patch}
```

```
\begin{patch}{generalFortranPagePatch37}
\begin{paste}{generalFortranPageFull37}{generalFortranPageEmpty37}
\pastebutton{generalFortranPageFull37}{\hidepaste}
\tab{5}\spadcommand{declare!([X,Y],fortranReal(),symbols)\free{symbols }}
\indentrel{3}\begin{verbatim}
   (36)  REAL
                                      Type: FortranType
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{generalFortranPageEmpty37}
\begin{paste}{generalFortranPageEmpty37}{generalFortranPagePatch37}
\pastebutton{generalFortranPageEmpty37}{\showpaste}
\tab{5}\spadcommand{declare!([X,Y],fortranReal(),symbols)\free{symbols }}
\end{paste}\end{patch}

\begin{patch}{generalFortranPagePatch38}
\begin{paste}{generalFortranPageFull38}{generalFortranPageEmpty38}
\pastebutton{generalFortranPageFull38}{\hidepaste}
\tab{5}\spadcommand{FP := FortranProgram(F,real,[X,Y],symbols)\free{symbols }\bound{FP }}
\indentrel{3}\begin{verbatim}
   (37)  FortranProgram(F,REAL,[X,Y],table(...,...))
                                           Type: Domain
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{generalFortranPageEmpty38}
\begin{paste}{generalFortranPageEmpty38}{generalFortranPagePatch38}
\pastebutton{generalFortranPageEmpty38}{\showpaste}
\tab{5}\spadcommand{FP := FortranProgram(F,real,[X,Y],symbols)\free{symbols }\bound{FP }}
\end{paste}\end{patch}

\begin{patch}{generalFortranPagePatch39}
\begin{paste}{generalFortranPageFull39}{generalFortranPageEmpty39}
\pastebutton{generalFortranPageFull39}{\hidepaste}
\tab{5}\spadcommand{asp := X*sin(Y)\bound{asp }}
\indentrel{3}\begin{verbatim}
   (38)  X sin(Y)
                               Type: Expression Integer
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{generalFortranPageEmpty39}
\begin{paste}{generalFortranPageEmpty39}{generalFortranPagePatch39}
\pastebutton{generalFortranPageEmpty39}{\showpaste}
\tab{5}\spadcommand{asp := X*sin(Y)\bound{asp }}
\end{paste}\end{patch}

\begin{patch}{generalFortranPagePatch40}
```

```
\begin{paste}{generalFortranPageFull40}{generalFortranPageEmpty40}
\pastebutton{generalFortranPageFull40}{\hidepaste}
\tab{5}\spadcommand{outputAsFortran(asp::FP)\free{FP asp }}
\indentrel{3}\begin{verbatim}
                                              Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{generalFortranPageEmpty40}
\begin{paste}{generalFortranPageEmpty40}{generalFortranPagePatch40}
\pastebutton{generalFortranPageEmpty40}{\showpaste}
\tab{5}\spadcommand{outputAsFortran(asp::FP)\free{FP asp }}
\end{paste}\end{patch}

\begin{patch}{generalFortranPagePatch41}
\begin{paste}{generalFortranPageFull41}{generalFortranPageEmpty41}
\pastebutton{generalFortranPageFull41}{\hidepaste}
\tab{5}\spadcommand{declare!(Z,fortranReal(),symbols)\free{symbols }\bound{Z }}
\indentrel{3}\begin{verbatim}
   (40)  REAL
                                       Type: FortranType
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{generalFortranPageEmpty41}
\begin{paste}{generalFortranPageEmpty41}{generalFortranPagePatch41}
\pastebutton{generalFortranPageEmpty41}{\showpaste}
\tab{5}\spadcommand{declare!(Z,fortranReal(),symbols)\free{symbols }\bound{Z }}
\end{paste}\end{patch}

\begin{patch}{generalFortranPagePatch42}
\begin{paste}{generalFortranPageFull42}{generalFortranPageEmpty42}
\pastebutton{generalFortranPageFull42}{\hidepaste}
\tab{5}\spadcommand{outputAsFortran([c,returns()]::FP)\free{FP c Z }}
\indentrel{3}\begin{verbatim}
                                              Type: Void
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{generalFortranPageEmpty42}
\begin{paste}{generalFortranPageEmpty42}{generalFortranPagePatch42}
\pastebutton{generalFortranPageEmpty42}{\showpaste}
\tab{5}\spadcommand{outputAsFortran([c,returns()]::FP)\free{FP c Z }}
\end{paste}\end{patch}
```

Some technical information

— ug15.ht —

```
\begin{page}{nagTechnicalPage}{15.3.5. Some technical information}
\beginscroll

The model adopted for the link is a server-client configuration --
Axiom acting as a client via a local agent (a process called {\tt
nagman}). The server side is implemented by the {\tt nagd} daemon
process which may run on a different host.  The {\tt nagman} local
agent is started by default whenever you start Axiom. The {\tt nagd}
server must be started separately.  Instructions for installing and
running the server are supplied in the NAG documentation.  Use the
\spadcmd{)set naglink host} system command to point your local agent
to a server in your network.

On the Axiom side, one sees a set of {\em packages}
(ask \Browse{} for {\em Nag*}) for each chapter, each exporting
operations with the same name as a routine in the \naglib{}.
The arguments and return value of each operation belong to
standard Axiom types.

The {\tt man} pages for the \naglib{} are accessible via the description
of each operation in \Browse{} (among other places).

In the implementation of each operation, the set of inputs is passed
to the local agent {\tt nagman}, which makes a Remote Procedure Call
(RPC) to the remote {\tt nagd} daemon process.  The local agent
receives the RPC results and forwards them to the Axiom workspace
where they are interpreted appropriately.

How are Fortran subroutines turned into RPC calls?  For each Fortran
routine in the \naglib{}, a C main() routine is supplied.  Its job is
to assemble the RPC input (numeric) data stream into the appropriate
Fortran data structures for the routine, call the Fortran routine from
C and serialize the results into an RPC output data stream.

Many \naglib{} routines accept ASPs (Argument Subprogram Parameters).
These specify user-supplied Fortran routines (e.g. a routine to
supply values of a function is required for numerical integration).
How are they handled? There are new facilities in Axiom to help.
A set of Axiom domains has been provided to turn values in standard
 Axiom types (such as Expression Integer) into the appropriate
piece of Fortran for each case (a filename pointing to Fortran source
for the ASP can always be supplied instead).
Ask \Browse{} for {\em Asp*} to see these domains. The Fortran fragments
are included in the outgoing RPC stream, but {\tt nagd} intercepts them,
compiles them, and links them with the main() C program before executing
```

```
the resulting program on the numeric part of the RPC stream.

\endscroll
\autobuttons
\end{page}
```

———-

Interactive Front-end and Language

⇒ "notitle" (ugLangLoopsBreakPage) 9 on page 1770
⇒ "notitle" (ugLangBlocksPage) 9 on page 1753

— **ug15.ht** —

```
\begin{page}{ugWhatsNewLanguagePage}
{15.4. Interactive Front-end and Language}
\beginscroll

The \axiom{leave} keyword has been replaced by the \axiom{break}
keyword for compatibility with the new Axiom extension language.  See
section \downlink{``break in Loops''}{ugLangLoopsBreakPage} in
Section 5.4.3\ignore{ugLangLoopsBreak} for more
information.

Curly braces are no longer used to create sets. Instead, use
\axiomFun{set} followed by a bracketed expression. For example,
\xtc{
}{
\spadpaste{set [1,2,3,4]}
}

Curly braces are now used to enclose a block (see section
\downlink{``Blocks''}{ugLangBlocksPage} in Section
5.2\ignore{ugLangBlocks} for more information). For
compatibility, a block can still be enclosed by parentheses as well.

New coercions to and from type \axiomType{Expression} have been
added. For example, it is now possible to map a polynomial
represented as an expression to an appropriate polynomial type.

Various messages have been added or rewritten for clarity.

\endscroll
\autobuttons
\end{page}
```

```
\begin{patch}{ugWhatsNewLanguagePagePatch1}
\begin{paste}{ugWhatsNewLanguagePageFull1}{ugWhatsNewLanguagePageEmpty1}
\pastebutton{ugWhatsNewLanguagePageFull1}{\hidepaste}
\tab{5}\spadcommand{set [1,2,3,4]}
\indentrel{3}\begin{verbatim}
   (1)  {1,2,3,4}
                              Type: Set PositiveInteger
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{ugWhatsNewLanguagePageEmpty1}
\begin{paste}{ugWhatsNewLanguagePageEmpty1}{ugWhatsNewLanguagePagePatch1}
\pastebutton{ugWhatsNewLanguagePageEmpty1}{\showpaste}
\tab{5}\spadcommand{set [1,2,3,4]}
\end{paste}\end{patch}
```

———-

Library

⇒ "notitle" (FullPartialFracExpansionXmpPage) 3.46 on page 549

— **ug15.ht** —

```
\begin{page}{ugWhatsNewLibraryPage}{15.5. Library}
\beginscroll

The \axiomType{FullPartialFracExpansion}
domain has been added. This domain computes factor-free full
partial fraction expansions.
See section
\downlink{`FullPartialFracExpansion'}
{FullPartialFracExpansionXmpPage}
\ignore{FullPartialFracExpansion}
for examples.

We have implemented the Bertrand/Cantor algorithm for integrals of
hyperelliptic functions. This brings a major speedup for some
classes of algebraic integrals.

We have implemented a new (direct) algorithm for integrating
trigonometric functions. This brings a speedup and an improvement in
the answer quality.

The {\sf SmallFloat} domain has been renamed
\axiomType{DoubleFloat} and {\sf SmallInteger} has been renamed
\axiomType{SingleInteger}. The new abbreviations as
```

```
\axiomType{DFLOAT} and \axiomType{SINT}, respectively.
We have defined the macro {\sf SF}, the old abbreviation for {\sf
SmallFloat}, to expand to \axiomType{DoubleFloat} and modified
the documentation and input file examples to use the new names
and abbreviations. You should do the same in any private Axiom
files you have.

There are many new categories, domains and packages related to the
NAG Library Link facility. See the file

\unixcommand{\env{AXIOM}/../../src/algebra/exposed.lsp}
{xterm\ -e\ vi\ +"/naglink"\ \env{AXIOM}/../../src/algebra/exposed.lsp}

for a list of constructors in the {\bf naglink} Axiom exposure group.

We have made improvements to the differential equation solvers
and there is a new facility for solving systems of first-order
linear differential equations.
In particular, an important fix was made to the solver for
inhomogeneous linear ordinary differential equations that
corrected the calculation of particular solutions.
We also made improvements to the polynomial
and transcendental equation solvers including the
ability to solve some classes of systems of transcendental
equations.

The efficiency of power series have been improved and left and right
expansions of \spad{tan(f(x))} at \spad{x =} a pole of \spad{f(x)}
can now be computed.
A number of power series bugs were fixed and the
\axiomType{GeneralUnivariatePowerSeries}
domain was added.
The power series variable can appear in the coefficients and when this
happens, you cannot differentiate or integrate the series.
Differentiation and integration with respect to other variables is
supported.

A domain was added for representing asymptotic expansions of a
function at an exponential singularity.

For limits, the main new feature is the exponential expansion domain
used to treat certain exponential singularities.  Previously, such
singularities were treated in an {\it ad hoc} way and only a few cases
were covered.  Now Axiom can do things like

\begin{verbatim}
limit( (x+1)**(x+1)/x**x - x**x/(x-1)**(x-1), x = %plusInfinity)
\end{verbatim}

in a systematic way.  It only does one level of nesting, though.  In
```

```
other words, we can handle \spad{exp(} some function with a pole
\spad{)}, but not \texht{\linebreak}{} \spad{exp(exp(} some function
with a pole \spad{)).}

The computation of integral bases has been improved through careful
use of Hermite row reduction. A P-adic algorithm for function fields
of algebraic curves in finite characteristic has also been developed.

Miscellaneous: There is improved conversion of definite and indefinite
integrals to \axiomType{InputForm}; binomial coefficients are
displayed in a new way; some new simplifications of radicals have been
implemented; the operation \spadfun{complexForm} for converting to
rectangular coordinates has been added; symmetric product operations
have been added to \axiomType{LinearOrdinaryDifferentialOperator}.

\endscroll
\autobuttons
\end{page}
```

———-

HyperDoc

⇒ "notitle" (ugHyperKeysPage) 8 on page 1709

— ug15.ht —

```
\begin{page}{ugWhatsNewHyperDocPage}{15.6. \HyperName}
\beginscroll

The buttons on the titlebar and scrollbar have been replaced
with ones which have a 3D effect. You can change the foreground and
background colors of these ``controls'' by including and modifying
the following lines in your {\bf .Xdefaults} file.
\begin{verbatim}
Axiom.hyperdoc.ControlBackground: White
Axiom.hyperdoc.ControlForeground: Black
\end{verbatim}

For various reasons, Hyperdoc sometimes displays a
secondary window. You can control the size and placement of this
window by including and modifying
the following line in your {\bf .Xdefaults} file.
%
\begin{verbatim}
Axiom.hyperdoc.FormGeometry: =950x450+100+0
\end{verbatim}
%
```

```
This setting is a standard X Window System geometry specification:
you are requesting a window 950 pixels wide by 450 deep and placed in
the upper left corner.

Some key definitions have been changed to conform more closely with
the CUA guidelines. Press \texht{F9}{\downlink{F9}{ugHyperKeysPage}}
to see the current definitions.

Input boxes (for example, in the Browser) now accept paste-ins from
the X Window System. Use the second button to paste in something
you have previously copied or cut. An example of how you can use this
is that you can paste the type from an Axiom computation
into the main Browser input box.


\endscroll
\autobuttons
\end{page}
```

———-

Documentation

⇒ "notitle" (ugGraphTwoDbuildPage) 11 on page 1995
⇒ "notitle" (ugGraphTwoDappendPage) 11 on page 2015
⇒ "notitle" (ugIntroCallFunPage) 6 on page 1505
⇒ "notitle" (ugUserRulesPage) 10 on page 1949

— **ug15.ht** —

```
\begin{page}{ugWhatsNewDocumentationPage}{15.7. Documentation}
\beginscroll
\texht{
We describe here a few additions to the on-line
version of the Axiom book which you can read with
HyperDoc.
}{}

A section has been added to the graphics chapter, describing how to
build \twodim{} graphs from lists of points. An example is given
showing how to read the points from a file.  See section
\downlink{``Building Two-Dimensional Graphs''}
{ugGraphTwoDbuildPage} in Section
7.1.9\ignore{ugGraphTwoDbuild} for details.

A further section has been added to that same chapter, describing how
to add a \twodim{} graph to a viewport which already contains other
```

```
graphs.  See section
\downlink{``Appending a Graph to a Viewport
Window Containing a Graph''}{ugGraphTwoDappendPage} in
Section 7.1.10\ignore{ugGraphTwoDappend} for
details.

Chapter 3
and the on-line Hyperdoc help have been unified.

An explanation of operation names ending in ``?'' and ``!'' has been
added to the first chapter.  See the end of the section
\downlink{``Calling Functions''}{ugIntroCallFunPage} in Section
1.3.6\ignore{ugIntroCallFun} for details.

An expanded explanation of using predicates has been added to the
sixth chapter. See the example involving \userfun{evenRule} in the
middle of the section
\downlink{``Rules and Pattern Matching''}{ugUserRulesPage} in Section
6.21\ignore{ugUserRules} for details.

\endscroll
\autobuttons
\end{page}
```

Chapter 19

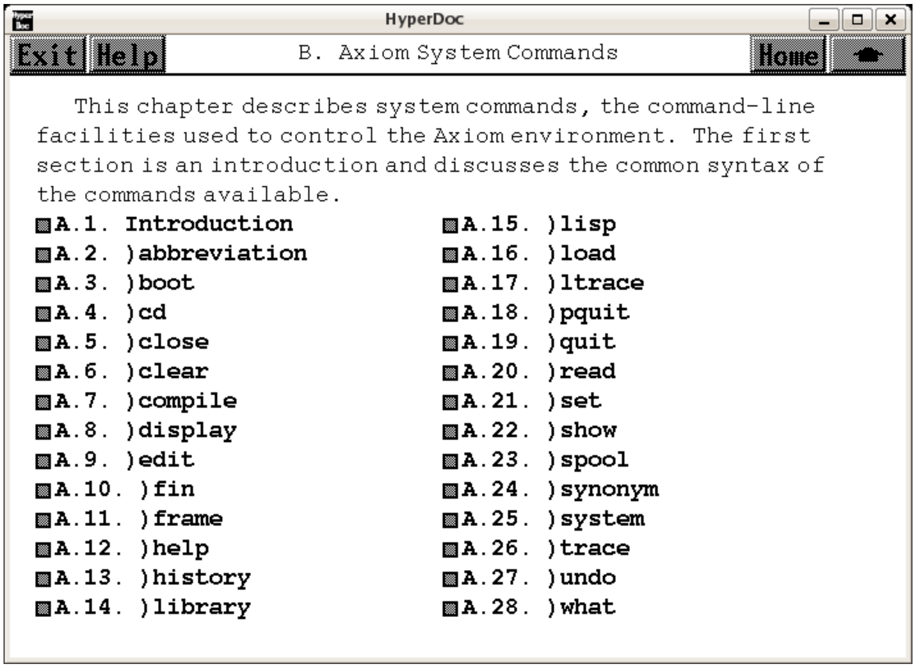
Users Guide Chapter 16 (ug16.ht)

— **ug16.ht** —

```
\newcommand{\lanb}{{\tt [}}
\newcommand{\ranb}{{\tt ]}}
\newcommand{\vertline}{\texht{$|$}{{\tt |}}}
```

————

Axiom System Commands



⇐ “Reference” (TopReferencePage) 3.1 on page 123
⇒ “Introduction” (ugSysCmdOverviewPage) 19 on page 2540
⇒ “)abbreviation” (ugSysCmdabbreviationPage) 19 on page 2542
⇒ “)boot” (ugSysCmdbootPage) 19 on page 2544
⇒ “)cd” (ugSysCmdcdPage) 19 on page 2545
⇒ “)close” (ugSysCmdclosePage) 19 on page 2546
⇒ “)clear” (ugSysCmdclearPage) 19 on page 2547
⇒ “)compile” (ugSysCmdcompilePage) 19 on page 2549
⇒ “)display” (ugSysCmddisplayPage) 19 on page 2552
⇒ “)edit” (ugSysCmdeditPage) 19 on page 2554
⇒ “)fin” (ugSysCmdfinPage) 19 on page 2555
⇒ “)frame” (ugSysCmdframePage) 19 on page 2556
⇒ “)help” (ugSysCmdhelpPage) 19 on page 2558
⇒ “)history” (ugSysCmdhistoryPage) 19 on page 2559
⇒ “)library” (ugSysCmdlibraryPage) 19 on page 2563
⇒ “)lisp” (ugSysCmdlispPage) 19 on page 2565
⇒ “)load” (ugSysCmdloadPage) 19 on page 2566
⇒ “)ltrace” (ugSysCmdltracePage) 19 on page 2566
⇒ “)pquit” (ugSysCmdpquitPage) 19 on page 2567
⇒ “)quit” (ugSysCmdquitPage) 19 on page 2569
⇒ “)read” (ugSysCmdreadPage) 19 on page 2570
⇒ “)set” (ugSysCmdsetPage) 19 on page 2571

⇒ ")show" (ugSysCmdshowPage) 19 on page 2573
⇒ ")spool" (ugSysCmdspoolPage) 19 on page 2574
⇒ ")synonym" (ugSysCmdsynonymPage) 19 on page 2575
⇒ ")system" (ugSysCmdsystemPage) 19 on page 2576
⇒ ")trace" (ugSysCmdtracePage) 19 on page 2578
⇒ ")undo" (ugSysCmdundoPage) 19 on page 2584
⇒ ")what" (ugSysCmdwhatPage) 19 on page 2586

— **ug16.ht** —

```
\begin{page}{ugSysCmdPage}{B. Axiom System Commands}
\beginscroll

\texht{\bgroup\baselineskip 10pt\ixpt{}\def\Isize{\SIsize}}{}

This chapter describes system commands, the command-line
facilities used to control the Axiom environment.
The first section is an introduction and discusses the common
syntax of the commands available.

\table{
  { \downlink{\menuitemstyle{A.1. Introduction}}{ugSysCmdOverviewPage} }
  { \downlink{\menuitemstyle{A.2. )abbreviation}}
{ugSysCmdabbreviationPage} }
  { \downlink{\menuitemstyle{A.3. )boot}}{ugSysCmdbootPage} }
  { \downlink{\menuitemstyle{A.4. )cd}}{ugSysCmdcdPage} }
  { \downlink{\menuitemstyle{A.5. )close}}{ugSysCmdclosePage} }
  { \downlink{\menuitemstyle{A.6. )clear}}{ugSysCmdclearPage} }
  { \downlink{\menuitemstyle{A.7. )compile}}{ugSysCmdcompilePage} }
  { \downlink{\menuitemstyle{A.8. )display}}{ugSysCmddisplayPage} }
  { \downlink{\menuitemstyle{A.9. )edit}}{ugSysCmdeditPage} }
  { \downlink{\menuitemstyle{A.10. )fin}}{ugSysCmdfinPage} }
  { \downlink{\menuitemstyle{A.11. )frame}}{ugSysCmdframePage} }
  { \downlink{\menuitemstyle{A.12. )help}}{ugSysCmdhelpPage} }
  { \downlink{\menuitemstyle{A.13. )history}}{ugSysCmdhistoryPage} }
  { \downlink{\menuitemstyle{A.14. )library}}{ugSysCmdlibraryPage} }
  { \downlink{\menuitemstyle{A.15. )lisp}}{ugSysCmdlispPage} }
  { \downlink{\menuitemstyle{A.16. )load}}{ugSysCmdloadPage} }
  { \downlink{\menuitemstyle{A.17. )ltrace}}{ugSysCmdltracePage} }
  { \downlink{\menuitemstyle{A.18. )pquit}}{ugSysCmdpquitPage} }
  { \downlink{\menuitemstyle{A.19. )quit}}{ugSysCmdquitPage} }
  { \downlink{\menuitemstyle{A.20. )read}}{ugSysCmdreadPage} }
  { \downlink{\menuitemstyle{A.21. )set}}{ugSysCmdsetPage} }
  { \downlink{\menuitemstyle{A.22. )show}}{ugSysCmdshowPage} }
  { \downlink{\menuitemstyle{A.23. )spool}}{ugSysCmdspoolPage} }
  { \downlink{\menuitemstyle{A.24. )synonym}}{ugSysCmdsynonymPage} }
  { \downlink{\menuitemstyle{A.25. )system}}{ugSysCmdsystemPage} }
  { \downlink{\menuitemstyle{A.26. )trace}}{ugSysCmdtracePage} }
  { \downlink{\menuitemstyle{A.27. )undo}}{ugSysCmdundoPage} }
  { \downlink{\menuitemstyle{A.28. )what}}{ugSysCmdwhatPage} }
```

```
}
\endscroll
\autobuttons
\end{page}
```

———-

Introduction

⇒ “notitle” (ugSysCmdsetPage) 19 on page 2571
⇒ “notitle” (ugSysCmdcompilePage) 19 on page 2549

— **ug16.ht** —

```
\begin{page}{ugSysCmdOverviewPage}{B.1. Introduction}
\beginscroll

System commands are used to perform Axiom environment management.
Among the commands are those that display what has been defined or
computed, set up multiple logical Axiom environments (frames), clear
definitions, read files of expressions and commands, show what
functions are available, and terminate Axiom.

Some commands are restricted: the commands
\begin{verbatim}
)set userlevel interpreter
)set userlevel compiler
)set userlevel development
\end{verbatim}
set the user-access level to the three possible choices.  All commands
are available at {\tt development} level and the fewest are available
at {\tt interpreter} level.  The default user-level is {\tt
interpreter}.  In addition to the \spadcmd{)set} command (discussed in
\downlink{``)set''}{ugSysCmdsetPage} in Section
B.21\ignore{ugSysCmdset}) you can use the Hyperdoc
settings facility to change the {\it user-level.}  \texht{}{Click on
\lispmemolink{Settings}{(|htSystemVariables|)} here to immediately go
to the settings facility.}

Each command listing begins with one or more syntax pattern
descriptions plus examples of related commands.  The syntax
descriptions are intended to be easy to read and do not necessarily
represent the most compact way of specifying all possible arguments
and options; the descriptions may occasionally be redundant.

All system commands begin with a right parenthesis which should be in
the first available column of the input line (that is, immediately
```

```
after the input prompt, if any).  System commands may be issued
directly to Axiom or be included in {\bf .input} files.

A system command {\it argument} is a word that directly follows the
command name and is not followed or preceded by a right parenthesis.
A system command {\it option} follows the system command and is
directly preceded by a right parenthesis.  Options may have arguments:
they directly follow the option.  This example may make it easier to
remember what is an option and what is an argument:

\centerline{{{\tt )syscmd {\it arg1 arg2} )opt1
{\it opt1arg1 opt1arg2} )opt2 {\it opt2arg1} ...}}}

In the system command descriptions, optional arguments and options are
enclosed in brackets (``\lanb'' and ``\ranb'').  If an argument or
option name is in italics, it is meant to be a variable and must have
some actual value substituted for it when the system command call is
made.  For example, the syntax pattern description

\noindent
{\tt )read} {\it fileName} {\tt \lanb{})quietly\ranb{}}

\noindent
would imply that you must provide an actual file name for
{\it fileName} but need not use the {\tt )quietly} option.
Thus
\begin{verbatim}
)read matrix.input
\end{verbatim}
is a valid instance of the above pattern.

System command names and options may be abbreviated and may be in
upper or lower case.
The case of actual arguments may be significant, depending on the
particular situation (such as in file names).
System command names and options may be abbreviated to the minimum
number of starting letters so that the name or option is unique.
Thus
\begin{verbatim}
)s Integer
\end{verbatim}
is not a valid abbreviation for the {\tt )set} command,
because both {\tt )set} and {\tt )show}
begin with the letter ``s''.
Typically, two or three letters are sufficient for disambiguating names.
In our descriptions of the commands, we have used no abbreviations for
either command names or options.

In some syntax descriptions we use a vertical line ``\vertline'' to
indicate that you must specify one of the listed choices.  For
```

```
example, in \begin{verbatim} )set output fortran on | off
\end{verbatim} only {\tt on} and {\tt off} are acceptable words for
following {\tt boot}.  We also sometimes use ‘‘...’’ to indicate that
additional arguments or options of the listed form are allowed.
Finally, in the syntax descriptions we may also list the syntax of
related commands.

\endscroll
\autobuttons
\end{page}
```

———-

)abbreviation

— ug16.ht —

```
\begin{page}{ugSysCmdabbreviationPage}{B.2. )abbreviation}
\beginscroll


\par\noindent{\bf User Level Required:} compiler

\par\noindent{\bf Command Syntax:}
\begin{items}
\item {\tt )abbreviation query  \lanb{}{\it nameOrAbbrev}\ranb{}}
\item {\tt )abbreviation category  {\it abbrev  fullname} \lanb{}
)quiet\ranb{}}
\item {\tt )abbreviation domain  {\it abbrev  fullname}   \lanb{}
)quiet\ranb{}}
\item {\tt )abbreviation package  {\it abbrev  fullname}  \lanb{}
)quiet\ranb{}}
\item {\tt )abbreviation remove  {\it nameOrAbbrev}}
\end{items}

\par\noindent{\bf Command Description:}

This command is used to query, set and remove abbreviations for category,
domain and package constructors.
Every constructor must have a unique abbreviation.
This abbreviation is part of the name of the subdirectory
under which the components of the compiled constructor are
stored.
Furthermore, by issuing this command you
let the system know what file to load automatically if you use a new
constructor.
Abbreviations must start with a letter and then be followed by
```

```
up to seven letters or digits.
Any letters appearing in the abbreviation must be in uppercase.

When used with the {\tt query} argument,
this command may be used to list the name
associated with a  particular abbreviation or the  abbreviation for a
constructor.
If no abbreviation or name is given, the names and corresponding
abbreviations for {\it all} constructors are listed.

The following shows the abbreviation for the constructor \spadtype{List}:
\begin{verbatim}
)abbreviation query List
\end{verbatim}
The following shows the constructor name corresponding to the
abbreviation \spadtype{NNI}:
\begin{verbatim}
)abbreviation query NNI
\end{verbatim}
The following lists all constructor names and their abbreviations.
\begin{verbatim}
)abbreviation query
\end{verbatim}

To add an abbreviation for a constructor, use this command with
{\tt category}, {\tt domain} or {\tt package}.
The following add abbreviations to the system for a
category, domain and package, respectively:
\begin{verbatim}
)abbreviation domain   SET Set
)abbreviation category COMPCAT  ComplexCategory
)abbreviation package  LIST2MAP ListToMap
\end{verbatim}
If the {\tt )quiet} option is used,
no output is displayed from this command.
You would normally only define an abbreviation in a library source file.
If this command is issued for a constructor that has already been loaded, the
constructor will be reloaded next time it is referenced.  In particular, you
can use this command to force the automatic reloading of constructors.

To remove an abbreviation, the {\tt remove} argument is used.
This is usually
only used to correct a previous command that set an abbreviation for a
constructor name.
If, in fact, the abbreviation does exist, you are prompted
for confirmation of the removal request.
Either of the following commands
will remove the abbreviation \spadtype{VECTOR2} and the
constructor name \spadtype{VectorFunctions2} from the system:
\begin{verbatim}
```

```
)abbreviation remove VECTOR2
)abbreviation remove VectorFunctions2
\end{verbatim}

\par\noindent{\bf Also See:}
\downlink{``)compile''}{ugSysCmdcompilePage} in section B.7

\endscroll
\autobuttons
\end{page}
```

———-

)boot

⇒ “notitle” (ugSysCmdfinPage) 19 on page 2555
⇒ “notitle” (ugSysCmdlispPage) 19 on page 2565
⇒ “notitle” (ugSysCmdsetPage) 19 on page 2571
⇒ “notitle” (ugSysCmdsystemPage) 19 on page 2576

— **ug16.ht** —

```
\begin{page}{ugSysCmdbootPage}{B.3. )boot}
\beginscroll


\par\noindent{\bf User Level Required:} development

\par\noindent{\bf Command Syntax:}
\begin{items}
\item {\tt )boot} {\it bootExpression}
\end{items}

\par\noindent{\bf Command Description:}

This command is used by Axiom system developers to execute
expressions written in the BOOT language.
For example,
\begin{verbatim}
)boot times3(x) == 3*x
\end{verbatim}
creates and compiles the \Lisp{} function ``times3''
obtained by translating the BOOT code.

\par\noindent{\bf Also See:}
\downlink{``)fin''}{ugSysCmdfinPage} in section B.10
\downlink{``)lisp''}{ugSysCmdlispPage} in section B.15
```

```
\downlink{``)set''}{ugSysCmdsetPage} in section B.21
\downlink{``)system''}{ugSysCmdsystemPage} in section B.25

\endscroll
\autobuttons
\end{page}
```

———-

)cd

⇒ "notitle" (ugSysCmdcompilePage) 19 on page 2549
⇒ "notitle" (ugSysCmdeditPage) 19 on page 2554
⇒ "notitle" (ugSysCmdhistoryPage) 19 on page 2559
⇒ "notitle" (ugSysCmdlibraryPage) 19 on page 2563
⇒ "notitle" (ugSysCmdreadPage) 19 on page 2570
⇒ "notitle" (ugSysCmdspoolPage) 19 on page 2574

— **ug16.ht** —

```
\begin{page}{ugSysCmdcdPage}{B.4. )cd}
\beginscroll


\par\noindent{\bf User Level Required:} interpreter

\par\noindent{\bf Command Syntax:}
\begin{items}
\item {\tt )cd} {\it directory}
\end{items}

\par\noindent{\bf Command Description:}

This command sets the Axiom working current directory.
The current directory is used for looking for
input files (for {\tt )read}),
Axiom library source files (for {\tt )compile}),
saved history environment files (for {\tt )history )restore}),
compiled Axiom library files (for \spadcmd{)library}), and
files to edit (for {\tt )edit}).
It is also used for writing
spool files (via {\tt )spool}),
writing history input files (via {\tt )history )write}) and
history environment files (via {\tt )history )save}),and
compiled Axiom library files (via {\tt )compile}).

If issued with no argument, this command sets the Axiom
```

```
current directory to your home directory.
If an argument is used, it must be a valid directory name.
Except for the ``{\tt )}'' at the beginning of the command,
this has the same syntax as the operating system {\tt cd} command.

\par\noindent{\bf Also See:}
\downlink{``)compile''}{ugSysCmdcompilePage} in section B.7
\downlink{``)edit''}{ugSysCmdeditPage} in section B.9
\downlink{``)history''}{ugSysCmdhistoryPage} in section B.13
\downlink{``)library''}{ugSysCmdlibraryPage} in section B.14
\downlink{``)read''}{ugSysCmdreadPage} in section B.20
\downlink{``)spool''}{ugSysCmdspoolPage} in section B.32

\endscroll
\autobuttons
\end{page}
```

———-

)close

⇒ "notitle" (ugSysCmdquitPage) 19 on page 2569

— **ug16.ht** —

```
\begin{page}{ugSysCmdclosePage}{B.5. )close}
\beginscroll


\par\noindent{\bf User Level Required:} interpreter

\par\noindent{\bf Command Syntax:}
\begin{items}
\item{\tt )close}
\item{\tt )close )quietly}
\end{items}
\par\noindent{\bf Command Description:}

This command is used to close down interpreter client processes.  Such
processes are started by Hyperdoc to run Axiom examples when you click
on their text. When you have finished examining or modifying the
example and you do not want the extra window around anymore, issue
\begin{verbatim}
)close
\end{verbatim}
to the Axiom prompt in the window.

If you try to close down the last remaining interpreter client
```

```
process, Axiom will offer to close down the entire Axiom
session and return you to the operating system by displaying something
like
\begin{verbatim}
   This is the last Axiom session. Do you want to kill Axiom?
\end{verbatim}
Type "y" (followed by the Return key) if this is what you had in mind.
Type "n" (followed by the Return key) to cancel the command.

You can use the {\tt )quietly} option to force Axiom to
close down the interpreter client process without closing down
the entire Axiom session.

\par\noindent{\bf Also See:}
\downlink{``)quit''}{ugSysCmdquitPage} in section B.19

\endscroll
\autobuttons
\end{page}
```

———-

)clear

⇒ “notitle” (ugSysCmddisplayPage) 19 on page 2552
⇒ “notitle” (ugSysCmdhistoryPage) 19 on page 2559
⇒ “notitle” (ugSysCmdundoPage) 19 on page 2584

— **ug16.ht** —

```
\begin{page}{ugSysCmdclearPage}{B.6. )clear}
\beginscroll


\par\noindent{\bf User Level Required:} interpreter

\par\noindent{\bf Command Syntax:}
\begin{items}
\item{\tt )clear all}
\item{\tt )clear completely}
\item{\tt )clear properties all}
\item{\tt )clear properties}  {\it obj1 \lanb{}obj2 ...\ranb{}}
\item{\tt )clear value      all}
\item{\tt )clear value}     {\it obj1 \lanb{}obj2 ...\ranb{}}
\item{\tt )clear mode       all}
\item{\tt )clear mode}      {\it obj1 \lanb{}obj2 ...\ranb{}}
\end{items}
```

```
\par\noindent{\bf Command Description:}

This command is used to remove function and variable declarations,
definitions and values from the workspace.  To empty the entire
workspace and reset the step counter to 1, issue
\begin{verbatim}
)clear all
\end{verbatim}
To remove everything in the workspace but not reset the step counter,
issue
\begin{verbatim}
)clear properties all
\end{verbatim}
To remove everything about the object {\tt x}, issue
\begin{verbatim}
)clear properties x
\end{verbatim}
To remove everything about the objects {\tt x, y} and {\tt f}, issue
\begin{verbatim}
)clear properties x y f
\end{verbatim}

The word {\tt properties} may be abbreviated to the single letter
``{\tt p}''.
\begin{verbatim}
)clear p all
)clear p x
)clear p x y f
\end{verbatim}
All definitions of functions and values of variables may be removed
by either
\begin{verbatim}
)clear value all
)clear v all
\end{verbatim}
This retains whatever declarations the objects had.  To remove
definitions and
values for the specific objects {\tt x, y} and {\tt f}, issue
\begin{verbatim}
)clear value x y f
)clear v x y f
\end{verbatim}
To remove  the declarations  of everything while  leaving the
definitions and
values, issue
\begin{verbatim}
)clear mode  all
)clear m all
\end{verbatim}
To remove declarations for the specific objects {\tt x, y} and
```

```
{\tt f}, issue
\begin{verbatim}
)clear mode x y f
)clear m x y f
\end{verbatim}
The {\tt )display names} and {\tt )display properties} commands
may be used
to see what is currently in the workspace.

The command
\begin{verbatim}
)clear completely
\end{verbatim}
does everything that {\tt )clear all} does, and also clears the internal
system function and constructor caches.

\par\noindent{\bf Also See:}
\downlink{``)display''}{ugSysCmddisplayPage} in section B.8
\downlink{``)history''}{ugSysCmdhistoryPage} in section B.13
\downlink{``)undo''}{ugSysCmdundoPage} in section B.27

\endscroll
\autobuttons
\end{page}
```

———-

)compile

⇒ "notitle" (ugSysCmdcdPage) 19 on page 2545
⇒ "notitle" (ugSysCmdtracePage) 19 on page 2578
⇒ "notitle" (ugSysCmdabbreviationPage) 19 on page 2542
⇒ "notitle" (ugSysCmdeditPage) 19 on page 2554
⇒ "notitle" (ugSysCmdlibraryPage) 19 on page 2563

— **ug16.ht** —

```
\begin{page}{ugSysCmdcompilePage}{B.7. )compile}
\beginscroll


\par\noindent{\bf User Level Required:} compiler

\par\noindent{\bf Command Syntax:}

\begin{items}
\item {\tt )compile}
```

```
\item {\tt )compile {\it fileName}}
\item {\tt )compile {\it fileName}.spad}
\item {\tt )compile {\it directory/fileName}.spad}
\item {\tt )compile {\it fileName} )quiet}
\item {\tt )compile {\it fileName} )noquiet}
\item {\tt )compile {\it fileName} )break}
\item {\tt )compile {\it fileName} )nobreak}
\item {\tt )compile {\it fileName} )library}
\item {\tt )compile {\it fileName} )nolibrary}
\item {\tt )compile {\it fileName} )vartrace}
\item {\tt )compile {\it fileName} )constructor} {\it nameOrAbbrev}
\end{items}

\par\noindent{\bf Command Description:}

You use this command to invoke the Axiom library compiler.  This
compiles files with file extension {\tt .spad} with the Axiom system
compiler. The command first looks in the standard system directories
for files with extension {\tt .spad}.

Should you not want the {\tt )library} command automatically invoked,
call {\tt )compile} with the {\tt )nolibrary} option. For example,
\begin{verbatim}
)compile mycode )nolibrary
\end{verbatim}
By default, the {\tt )library} system command exposes all domains and
categories it processes. This means that the Axiom intepreter will consider
those domains and categories when it is trying to resolve a reference to a
function.
Sometimes domains and categories should not be exposed. For example, a domain
may just be used privately by another domain and may not be meant for
top-level use. The {\tt )library} command should still be used, though, so
that the code will be loaded on demand. In this case, you should use the
{\tt )nolibrary} option on {\tt )compile} and the {\tt )noexpose} option in
the {\tt )library} command. For
example,
\begin{verbatim}
)compile mycode.spad )nolibrary
)library mycode )noexpose
\end{verbatim}
Once you have established your own collection of compiled code, you may find
it handy to use the )dir option on the )library command. This causes )library
to process all compiled code in the specified directory. For example,
\begin{verbatim}
)library )dir /u/jones/quantum
\end{verbatim}
You must give an explicit directory after )dir, even if you want all compiled
code in the current working directory processed.
\begin{verbatim}
)library )dir .
```

```
\end{verbatim}
You can compile category, domain, and package constructors contained in files
with file extension {\tt .spad}. You can compile individual constructors or
every constructor in a file.

The full filename is remembered between invocations of this command and
{\tt )edit} commands. The sequence of commands
\begin{verbatim}
)compile matrix.spad
)edit
)compile
\end{verbatim}
will call the compiler, edit, and then call the compiler again on the file
matrix.spad. If you do not specify a directory, the working current directory
(see description of command )cd ) is searched for the file. If the file is
not found, the standard system directories are searched.

If you do not give any options, all constructors within a file are compiled.
Each constructor should have an {\tt )abbreviation} command in the file in
which it is defined. We suggest that you place the {\tt )abbreviation}
commands at the top of the file in the order in which the constructors are
defined. The list of commands serves as a table of contents for the file.

The {\tt )library} option causes directories containing the compiled code
for each constructor to be created in the working current directory. The
name of such a directory consists of the constructor abbreviation and the
{\tt .nrlib} file extension. For example, the directory containing the
compiled code for the {\tt MATRIX} constructor is called {\bf MATRIX.nrlib}.
The {\tt )nolibrary} option says that such files should not be created.

The {\tt )vartrace} option causes the compiler to generate extra code for the
constructor to support conditional tracing of variable assignments.
Without this option, this code is suppressed
and one cannot use the )vars option for the trace command.

The {\tt )constructor} option is used to specify a particular constructor to
compile. All other constructors in the file are ignored. The constructor name
or abbreviation follows {\tt )constructor}. Thus either

\begin{verbatim}
)compile matrix.spad )constructor RectangularMatrix
\end{verbatim}
or
\begin{verbatim}
)compile matrix.spad )constructor RMATRIX
\end{verbatim}
compiles the {\tt RectangularMatrix} constructor defined in {\bf matrix.spad}.

The {\tt )break} and {\tt )nobreak} options determine what the compiler does
when it encounters an error. {\tt )break} is the default and it indicates that
```

```
processing should stop at the first error. The value of the {\tt )set break}
variable then controls what happens.

\par\noindent{\bf Also See:}
\downlink{``)abbreviation''}{ugSysCmdabbreviationPage} in section B.2
\downlink{``)edit''}{ugSysCmdeditPage} in section B.9
\downlink{``)library''}{ugSysCmdlibraryPage} in section B.14

\endscroll
\autobuttons
\end{page}
```

———-

)display

⇒ "notitle" (ugSysCmdclearPage) 19 on page 2547
⇒ "notitle" (ugSysCmdhistoryPage) 19 on page 2559
⇒ "notitle" (ugSysCmdsetPage) 19 on page 2571
⇒ "notitle" (ugSysCmdshowPage) 19 on page 2573
⇒ "notitle" (ugSysCmdwhatPage) 19 on page 2586

— **ug16.ht** —

```
\begin{page}{ugSysCmddisplayPage}{B.8. )display}
\beginscroll


\par\noindent{\bf User Level Required:} interpreter

\par\noindent{\bf Command Syntax:}
\begin{items}
\item {\tt )display all}
\item {\tt )display properties}
\item {\tt )display properties all}
\item {\tt )display properties} {\it \lanb{}obj1 \lanb{}obj2 ...
\ranb{}\ranb{}}
\item {\tt )display value all}
\item {\tt )display value} {\it \lanb{}obj1 \lanb{}obj2 ...\ranb{}\ranb{}}
\item {\tt )display mode all}
\item {\tt )display mode} {\it \lanb{}obj1 \lanb{}obj2 ...\ranb{}\ranb{}}
\item {\tt )display names}
\item {\tt )display operations} {\it opName}
\end{items}
\par\noindent{\bf Command Description:}

This command is  used to display the contents of  the workspace and
```

```
signatures of functions  with a  given  name.\footnote{A
\spadgloss{signature} gives the argument and return types of a
function.}

The command
\begin{verbatim}
)display names
\end{verbatim}
lists the names of all user-defined objects in the workspace.  This is
useful if you do not wish to see everything about the objects and need
only be reminded of their names.

The commands
\begin{verbatim}
)display all
)display properties
)display properties all
\end{verbatim}
all do the same thing: show the values and types and declared modes of
all variables in the workspace.  If you have defined functions, their
signatures and definitions will also be displayed.

To show all information about a  particular variable or user functions,
for example, something named {\tt d}, issue
\begin{verbatim}
)display properties d
\end{verbatim}
To just show the value (and the type) of {\tt d}, issue
\begin{verbatim}
)display value d
\end{verbatim}
To just show the declared mode of {\tt d}, issue
\begin{verbatim}
)display mode d
\end{verbatim}

All modemaps for a given operation may be displayed by using {\tt
)display operations}.  A \spadgloss{modemap} is a collection of
information about a particular reference to an operation.  This
includes the types of the arguments and the return value, the location
of the implementation and any conditions on the types.  The modemap
may contain patterns.  The following displays the modemaps for the
operation \spadfunFrom{complex}{ComplexCategory}:
\begin{verbatim}
)d op complex
\end{verbatim}

\par\noindent{\bf Also See:}
\downlink{``)clear''}{ugSysCmdclearPage} in section B.6
\downlink{``)history''}{ugSysCmdhistoryPage} in section B.13
```

```
\downlink{``)set''}{ugSysCmdsetPage} in section B.21
\downlink{``)show''}{ugSysCmdshowPage} in section B.22
\downlink{``)what''}{ugSysCmdwhatPage} in section B.28

\endscroll
\autobuttons
\end{page}
```

———-

)edit

⇒ “notitle” (ugSysCmdsystemPage) 19 on page 2576
⇒ “notitle” (ugSysCmdcompilePage) 19 on page 2549
⇒ “notitle” (ugSysCmdreadPage) 19 on page 2570

— **ug16.ht** —

```
\begin{page}{ugSysCmdeditPage}{B.9. )edit}
\beginscroll


\par\noindent{\bf User Level Required:} interpreter

\par\noindent{\bf Command Syntax:}
\begin{items}
\item{\tt )edit} \lanb{}{\it filename}\ranb{}
\end{items}
\par\noindent{\bf Command Description:}

This command is  used to edit files.
It works in conjunction  with the {\tt )read}
and {\tt )compile} commands to remember the name
of the file on which you are working.
By specifying the name fully, you  can edit any file you wish.
Thus
\begin{verbatim}
)edit /u/julius/matrix.input
\end{verbatim}
will place  you in an editor looking at the  file
{\tt /u/julius/matrix.input}.
By default, the editor is {\tt vi},
but if you have an EDITOR shell environment variable defined, that editor
will be used.
When Axiom is running under the X Window System,
it will try to open a separate {\tt xterm} running your editor if
it thinks one is necessary.
```

```
For example, under the Korn shell, if you issue
\begin{verbatim}
export EDITOR=emacs
\end{verbatim}
then the emacs
editor will be used by \spadcmd{)edit}.

If you do not specify a file name, the last file you edited,
read or compiled will be used.
If there is no ``last file'' you will be placed in the editor editing
an empty unnamed file.

It is possible to use the {\tt )system} command to edit a file directly.
For example,
\begin{verbatim}
)system emacs /etc/rc.tcpip
\end{verbatim}
calls {\tt emacs} to edit the file.

\par\noindent{\bf Also See:}
\downlink{``)system''}{ugSysCmdsystemPage} in section B.25
\downlink{``)compile''}{ugSysCmdcompilePage} in section B.7
\downlink{``)read''}{ugSysCmdreadPage} in section B.20

\endscroll
\autobuttons
\end{page}
```

——————

)fin

⇒ "notitle" (ugSysCmdpquitPage) 19 on page 2567

— **ug16.ht** —

```
\begin{page}{ugSysCmdfinPage}{B.10. )fin}
\beginscroll


\par\noindent{\bf User Level Required:} development

\par\noindent{\bf Command Syntax:}
\begin{items}
\item {\tt )fin}
\end{items}
\par\noindent{\bf Command Description:}
```

```
This command is used by Axiom
developers to leave the Axiom system and return
to the underlying \Lisp{} system.
To return to Axiom, issue the
``{\tt (\vertline{}spad\vertline{})}''
function call to \Lisp{}.

\par\noindent{\bf Also See:}
\downlink{``)pquit''}{ugSysCmdpquitPage} in section B.18

\endscroll
\autobuttons
\end{page}
```

———-

)frame

⇒ "notitle" (ugSysCmdhistoryPage) 19 on page 2559
⇒ "notitle" (ugSysCmdsetPage) 19 on page 2571

— **ug16.ht** —

```
\begin{page}{ugSysCmdframePage}{B.11. )frame}
\beginscroll


\par\noindent{\bf User Level Required:} interpreter

\par\noindent{\bf Command Syntax:}
\begin{items}
\item{\tt )frame  new  {\it frameName}}
\item{\tt )frame  drop  {\it \lanb{}frameName\ranb{}}}
\item{\tt )frame  next}
\item{\tt )frame  last}
\item{\tt )frame  names}
\item{\tt )frame  import {\it frameName}
{\it \lanb{}objectName1 \lanb{}objectName2 ...\ranb{}\ranb{}}}
\item{\tt )set message frame on \vertline{} off}
\item{\tt )set message prompt frame}
\end{items}

\par\noindent{\bf Command Description:}

A {\it frame} can be thought of as a logical session within the
physical session that you get when you start the system.  You can
have as many frames as you want, within the limits of your computer's
```

```
storage, paging space, and so on.
Each frame has its own {\it step number}, {\it environment} and {\it
history.}  You can have a variable named {\tt a} in one frame and it
will have nothing to do with anything that might be called {\tt a} in
any other frame.

Some frames are created by the Hyperdoc program and these can
have pretty strange names, since they are generated automatically.
To find out the names
of all frames, issue
\begin{verbatim}
)frame names
\end{verbatim}
It will indicate the name of the current frame.

You create a new frame
``{\bf quark}'' by issuing
\begin{verbatim}
)frame new quark
\end{verbatim}
The history facility can be turned on by issuing either
{\tt )set history on} or {\tt )history )on}.
If the history facility is on and you are saving history information
in a file rather than in the Axiom environment
then a history file with filename {\bf quark.axh} will
be created as you enter commands.
If you wish to go back to what
you were doing in the
``{\bf initial}'' frame, use
\begin{verbatim}
)frame next
\end{verbatim}
or
\begin{verbatim}
)frame last
\end{verbatim}
to cycle through the ring of available frames to get back to
``{\bf initial}''.

If you want to throw
away a frame (say ``{\bf quark}''), issue
\begin{verbatim}
)frame drop quark
\end{verbatim}
If you omit the name, the current frame is dropped.

If you do use frames with the history facility on and writing to a file,
you may want to delete some of the older history files.
These are directories, so you may want to issue a command like
{\tt rm -r quark.axh} to the operating system.
```

```
You can bring things from another frame by using
{\tt )frame import}.
For example, to bring the {\tt f} and {\tt g} from the frame ``{\bf quark}''
to the current frame, issue
\begin{verbatim}
)frame import quark f g
\end{verbatim}
If you want everything from the frame ``{\bf quark}'', issue
\begin{verbatim}
)frame import quark
\end{verbatim}
You will be asked to verify that you really want everything.

There are two {\tt )set} flags
to make it easier to tell where you are.
\begin{verbatim}
)set message frame on | off
\end{verbatim}
will print more messages about frames when it is set on.
By default, it is off.
\begin{verbatim}
)set message prompt frame
\end{verbatim}
will give a prompt
that looks like
\begin{verbatim}
initial (1) ->
\end{verbatim}
when you start up. In this case, the frame name and step make up the
prompt.

\par\noindent{\bf Also See:}
\downlink{``)history''}{ugSysCmdhistoryPage} in section B.13
\downlink{``)set''}{ugSysCmdsetPage} in section B.21

\endscroll
\autobuttons
\end{page}
```

———-

)help

— ug16.ht —

```
\begin{page}{ugSysCmdhelpPage}{B.12. )help}
```

```
\beginscroll


\par\noindent{\bf User Level Required:} interpreter

\par\noindent{\bf Command Syntax:}
\begin{items}
\item{\tt )help}
\item{\tt )help} {\it commandName}
\end{items}

\par\noindent{\bf Command Description:}

This command displays help information about system commands.
If you issue
\begin{verbatim}
)help
\end{verbatim}
then this very text will be shown.
You can also give the name or abbreviation of a system command
to display information about it.
For example,
\begin{verbatim}
)help clear
\end{verbatim}
will display the description of the {\tt )clear} system command.

All this material is available in the Axiom User Guide
and in Hyperdoc.
In Hyperdoc, choose the {\bf Commands} item from the
{\bf Reference} menu.



\endscroll
\autobuttons
\end{page}
```

———-

)history

⇒ “notitle” (ugSysCmdframePage) 19 on page 2556
⇒ “notitle” (ugSysCmdcdPage) 19 on page 2545
⇒ “notitle” (ugSysCmdreadPage) 19 on page 2570
⇒ “notitle” (ugSysCmdsetPage) 19 on page 2571

⇒ "notitle" (ugSysCmdundoPage) 19 on page 2584

— ug16.ht —

```
\begin{page}{ugSysCmdhistoryPage}{B.13. )history}
\beginscroll


\par\noindent{\bf User Level Required:} interpreter

\par\noindent{\bf Command Syntax:}
\begin{items}
\item{\tt )history )on}
\item{\tt )history )off}
\item{\tt )history )write} {\it historyInputFileName}
\item{\tt )history )show \lanb{}{\it n}\ranb{} \lanb{}both\ranb{}}
\item{\tt )history )save} {\it savedHistoryName}
\item{\tt )history )restore} \lanb{}{\it savedHistoryName}\ranb{}
\item{\tt )history )reset}
\item{\tt )history )change} {\it n}
\item{\tt )history )memory}
\item{\tt )history )file}
\item{\tt \%}
\item{\tt \%\%({\it n})}
\item{\tt )set history on \vertline{} off}
\end{items}

\par\noindent{\bf Command Description:}

The {\it history} facility within Axiom allows you to restore your
environment to that of another session and recall previous
computational results.
Additional commands allow you to review previous
input lines and to create an {\bf .input} file of the lines typed to
Axiom.

Axiom saves your input and output if the history facility is
turned on (which is the default).
This information is saved if either of
\begin{verbatim}
)set history on
)history )on
\end{verbatim}
has been issued.
Issuing either
\begin{verbatim}
)set history off
)history )off
\end{verbatim}
will discontinue the recording of information.
```

```
Whether the facility is disabled or not, the value of \spadSyntax{\%}
in Axiom always refers to the result of the last computation.  If you
have not yet entered anything, \spadSyntax{\%} evaluates to an object
of type \spadtype{Variable('\%)}.  The function \spadSyntax{\%\%} may
be used to refer to other previous results if the history facility is
enabled.  In that case, {\tt \%\%(n)} is the output from step {\tt n}
if {\tt n > 0}.  If {\tt n < 0}, the step is computed relative to the
current step.  Thus {\tt \%\%(-1)} is also the previous step, {\tt
\%\%(-2)}, is the step before that, and so on.  If an invalid step
number is given, Axiom will signal an error.

The {\it environment} information can either be saved in a file or
entirely in memory (the default).  Each frame
(\downlink{``)frame''}{ugSysCmdframePage} in Section
B.11\ignore{ugSysCmdframe}) has its own history
database.  When it is kept in a file, some of it may also be kept in
memory for efficiency.  When the information is saved in a file, the
name of the file is of the form {\bf FRAME.axh} where ``{\bf FRAME}''
is the name of the current frame.  The history file is placed in the
current working directory (see
\downlink{``)cd''}{ugSysCmdcdPage} in Section
B.4\ignore{ugSysCmdcd}).  Note that these history
database files are not text files (in fact, they are directories
themselves), and so are not in human-readable format.

The options to the {\tt )history} command are as follows:

\indent{0}
\beginitems
\item[{\tt )change} {\it n}]
will set the number of steps that are saved in memory to {\it n}.
This option only has effect when the history data is maintained in a
file.
If you have issued {\tt )history )memory} (or not changed the default)
there is no need to use {\tt )history )change}.

\item[{\tt )on}]
will start the recording of information.
If the workspace is not empty, you will be asked to confirm this
request.
If you do so, the workspace will be cleared and history data will begin
being saved.
You can also turn the facility on by issuing {\tt )set history on}.

\item[{\tt )off}]
will stop the recording of information.
The {\tt )history )show} command will not work after issuing this
command.
Note that this command may be issued to save time, as there is some
performance penalty paid for saving the environment data.
```

```
You can also turn the facility off by issuing {\tt )set history off}.

\item[{\tt )file}]
indicates that history data should be saved in an external file on disk.

\item[{\tt )memory}]
indicates that all history data should be kept in memory rather than
saved in a file.
Note that if you are computing with very large objects it may not be
practical to kept this data in memory.

\item[{\tt )reset}]
will flush the internal list of the most recent workspace calculations
so that the data structures may be garbage collected by the underlying
\Lisp{} system.
Like {\tt )history )change}, this option only has real effect when
history data is being saved in a file.

\item[{\tt )restore} \lanb{}{\it savedHistoryName}\ranb{}]
completely clears the environment and restores it to a saved session, if
possible.
The {\tt )save} option below allows you to save a session to a file
with a given name. If you had issued
{\tt )history )save jacobi}
the command
{\tt )history )restore jacobi}
would clear the current workspace and load the contents of the named
saved session. If no saved session name is specified, the system looks
for a file called {\bf last.axh}.

\item[{\tt )save} {\it savedHistoryName}]
is used to save  a snapshot of the environment in a file.
This file is placed in the current working directory
(see \downlink{``)cd''}{ugSysCmdcdPage}
in Section B.4\ignore{ugSysCmdcd}).
Use {\tt )history )restore} to restore the environment to the state
preserved in the file.
This option also creates an input file containing all the lines of input
since you created the workspace frame (for example, by starting your
Axiom session) or last did a \spadcmd{)clear all} or
\spadcmd{)clear completely}.

\item[{\tt )show} \lanb{}{\it n}\ranb{} \lanb{}{\tt both}\ranb{}]
can show previous input lines and output results.
{\tt )show} will display up to twenty of the last input lines
(fewer if you haven't typed in twenty lines).
{\tt )show} {\it n} will display up to {\it n} of the last input lines.
{\tt )show both} will display up to five of the last input lines and
output results.
{\tt )show} {\it n} {\tt both} will display up to {\it n} of the last
```

```
input lines and output results.

\item[{\tt )write} {\it historyInputFile}]
creates an {\bf .input} file with the input lines typed since the start
of the session/frame or the last {\tt )clear all} or {\tt )clear
completely}.
If {\it historyInputFileName} does not
contain a period (``.'') in the filename,
{\bf .input} is appended to it.
For example,
{\tt )history )write chaos}
and
{\tt )history )write chaos.input}
both write the input lines to a file called {\bf chaos.input} in your
current working directory.
If you issued one or more {\tt )undo} commands,
{\tt )history )write}
eliminates all
input lines backtracked over as a result of {\tt )undo}.
You can edit this file and then use {\tt )read} to have Axiom process
the contents.
\enditems
\indent{0}

\par\noindent{\bf Also See:}
\downlink{``)frame''}{ugSysCmdframePage} in section B.11
\downlink{``)read''}{ugSysCmdreadPage} in section B.20
\downlink{``)set''}{ugSysCmdsetPage} in section B.21
\downlink{``)undo''}{ugSysCmdundoPage} in section B.27

\endscroll
\autobuttons
\end{page}
```

———-

)library

⇒ “notitle” (ugSysCmdcdPage) 19 on page 2545
⇒ “notitle” (ugSysCmdcompilePage) 19 on page 2549
⇒ “notitle” (ugSysCmdframePage) 19 on page 2556
⇒ “notitle” (ugSysCmdsetPage) 19 on page 2571

— **ug16.ht** —

```
\begin{page}{ugSysCmdlibraryPage}{B.14. )library}
\beginscroll
```

```
\par\noindent{\bf User Level Required:} interpreter

\par\noindent{\bf Command Syntax:}
\begin{items}
\item{\tt )library {\it libName1  \lanb{}libName2 ...\ranb{}}}
\item{\tt )library )dir {\it dirName}}
\item{\tt )library )only {\it objName1  \lanb{}objlib2 ...\ranb{}}}
\item{\tt )library )noexpose}
\end{items}

\par\noindent{\bf Command Description:}

This command replaces the {\tt )load} system command that
was available in Axiom releases before version 2.0.
The \spadcmd{)library} command makes available to Axiom the compiled
objects in the libraries listed.

For example, if you {\tt )compile dopler.as} in your home
directory, issue {\tt )library dopler} to have Axiom look
at the library, determine the category and domain constructors present,
update the internal database with various properties of the
constructors, and arrange for the constructors to be
automatically loaded when needed.
If the {\tt )noexpose} option has not been given, the
constructors will be exposed (that is, available) in the current
frame.

If you compiled a file with the old system compiler, you will
have an {\it nrlib} present, for example, {\it DOPLER.nrlib,}
where {\tt DOPLER} is a constructor abbreviation.
The command {\tt )library DOPLER} will then do the analysis and
database updates as above.

To tell the system about all libraries in a directory, use
{\tt )library )dir dirName} where {\tt dirName} is an explicit
directory.
You may specify ``.'' as the directory, which means the current
directory from which you started the system or the one you set
via the \spadcmd{)cd} command. The directory name is required.

You may only want to tell the system about particular
constructors within a library. In this case, use the {\tt )only}
option. The command {\tt )library dopler )only Test1} will only
cause the {\sf Test1} constructor to be analyzed, autoloaded,
etc..

Finally, each constructor in a library are usually automatically
exposed when the \spadcmd{)library} command is used. Use the {\tt
```

```
)noexpose} option if you not want them exposed. At a later time you
can use {\tt )set expose add constructor} to expose any hidden
constructors.

{\bf Note for Axiom beta testers:} At various times this
command was called {\tt )local} and {\tt )with} before the name
{\tt )library} became the official name.

\par\noindent{\bf Also See:}
\downlink{``)cd''}{ugSysCmdcdPage} in section B.4
\downlink{``)compile''}{ugSysCmdcompilePage} in section B.7
\downlink{``)frame''}{ugSysCmdframePage} in section B.11
\downlink{``)set''}{ugSysCmdsetPage} in section B.21

\endscroll
\autobuttons
\end{page}
```

———-

)lisp

⇒ "notitle" (ugSysCmdsystemPage) 19 on page 2576
⇒ "notitle" (ugSysCmdbootPage) 19 on page 2544
⇒ "notitle" (ugSysCmdfinPage) 19 on page 2555

— **ug16.ht** —

```
\begin{page}{ugSysCmdlispPage}{B.15. )lisp}
\beginscroll


\par\noindent{\bf User Level Required:} development

\par\noindent{\bf Command Syntax:}
\begin{items}
\item {\tt )lisp} {\it\lanb{}lispExpression\ranb{}}
\end{items}

\par\noindent{\bf Command Description:}

This command is used by Axiom system developers to have single
expressions evaluated by the \Lisp{} system on which Axiom is built.
The {\it lispExpression} is read by the \Lisp{} reader and evaluated.
If this expression is not complete (unbalanced parentheses, say), the
reader will wait until a complete expression is entered.
```

```
Since this command is only useful for evaluating single expressions,
the {\tt )fin} command may be used to drop out of Axiom into \Lisp{}.

\par\noindent{\bf Also See:}
\downlink{``)system''}{ugSysCmdsystemPage} in section B.25
\downlink{``)boot''}{ugSysCmdbootPage} in section B.3
\downlink{``)fin''}{ugSysCmdfinPage} in section B.10

\endscroll
\autobuttons
\end{page}
```

———-

)load

— **ug16.ht** —

```
\begin{page}{ugSysCmdloadPage}{B.16. )load}
\beginscroll


\par\noindent{\bf User Level Required:} interpreter

\par\noindent{\bf Command Description:}

This command is obsolete. Use \spadcmd{)library} instead.

\endscroll
\autobuttons
\end{page}
```

———-

)ltrace

⇒ "notitle" (ugSysCmdbootPage) 19 on page 2544
⇒ "notitle" (ugSysCmdlispPage) 19 on page 2565
⇒ "notitle" (ugSysCmdtracePage) 19 on page 2578

— **ug16.ht** —

```
\begin{page}{ugSysCmdltracePage}{B.17. )ltrace}
\beginscroll
```

```
\par\noindent{\bf User Level Required:} development

\par\noindent{\bf Command Syntax:}

This command has the same arguments as options as the
\spadcmd{)trace} command.

\par\noindent{\bf Command Description:}

This command is used by Axiom system developers to trace
\Lisp{} or
BOOT functions.
It is not supported for general use.

\par\noindent{\bf Also See:}
\downlink{``)boot''}{ugSysCmdbootPage} in section B.3
\downlink{``)lisp''}{ugSysCmdlispPage} in section B.15
\downlink{``)trace''}{ugSysCmdtracePage} in section B.26

\endscroll
\autobuttons
\end{page}
```

———-

)pquit

⇒ "notitle" (ugSysCmdfinPage) 19 on page 2555
⇒ "notitle" (ugSysCmdhistoryPage) 19 on page 2559
⇒ "notitle" (ugSysCmdclosePage) 19 on page 2546
⇒ "notitle" (ugSysCmdquitPage) 19 on page 2569
⇒ "notitle" (ugSysCmdsystemPage) 19 on page 2576

— **ug16.ht** —

```
\begin{page}{ugSysCmdpquitPage}{B.18. )pquit}
\beginscroll


\par\noindent{\bf User Level Required:} interpreter

\par\noindent{\bf Command Syntax:}
\begin{items}
\item{\tt )pquit}
\end{items}
```

```
\par\noindent{\bf Command Description:}

This command is used to terminate Axiom  and return to the
operating system.
Other than by redoing all your computations or by
using the {\tt )history )restore}
command to try to restore your working environment,
you cannot return to Axiom in the same state.

{\tt )pquit} differs from the {\tt )quit} in that it always asks for
confirmation that you want to terminate Axiom (the ‘‘p’’ is for
‘‘protected’’).
When you enter the {\tt )pquit} command, Axiom responds
%
\centerline{{Please enter {\bf y} or {\bf yes}
if you really want to leave the interactive }}
\centerline{{environment and return to the operating system:}}
%
If you respond with {\tt y} or {\tt yes}, you will see the message
%
\centerline{{You are now leaving the Axiom interactive environment. }}
\centerline{{Issue the command {\bf axiom} to
the operating system to start a new session.}}
%
and Axiom will terminate and return you to the operating
system (or the environment from which you invoked the system).
If you responded with something other than {\tt y} or {\tt yes}, then
the message
%
\centerline{{You have chosen to remain in the
Axiom interactive environment.}}
%
will be displayed and, indeed, Axiom would still be running.

\par\noindent{\bf Also See:}
\downlink{‘‘)fin’’}{ugSysCmdfinPage} in section B.10
\downlink{‘‘)history’’}{ugSysCmdhistoryPage} in section B.13
\downlink{‘‘)close’’}{ugSysCmdclosePage} in section B.5
\downlink{‘‘)quit’’}{ugSysCmdquitPage} in section B.19
\downlink{‘‘)system’’}{ugSysCmdsystemPage} in section B.25

\endscroll
\autobuttons
\end{page}
```

———-

)quit

⇒ “notitle” (ugSysCmdfinPage) 19 on page 2555
⇒ “notitle” (ugSysCmdhistoryPage) 19 on page 2559
⇒ “notitle” (ugSysCmdclosePage) 19 on page 2546
⇒ “notitle” (ugSysCmdpquitPage) 19 on page 2567
⇒ “notitle” (ugSysCmdsystemPage) 19 on page 2576

— **ug16.ht** —

```
\begin{page}{ugSysCmdquitPage}{B.19. )quit}
\beginscroll


\par\noindent{\bf User Level Required:} interpreter

\par\noindent{\bf Command Syntax:}
\begin{items}
\item{\tt )quit}
\item{\tt )set quit protected \vertline{} unprotected}
\end{items}

\par\noindent{\bf Command Description:}

This command is used to terminate Axiom  and return to the
operating system.
Other than by redoing all your computations or by
using the {\tt )history )restore}
command to try to restore your working environment,
you cannot return to Axiom in the same state.

{\tt )quit} differs from the {\tt )pquit} in that it asks for
confirmation only if the command
\begin{verbatim}
)set quit protected
\end{verbatim}
has been issued.
Otherwise, {\tt )quit} will make Axiom terminate and return you
to the operating system (or the environment from which you invoked the
system).

The default setting is {\tt )set quit protected} so that {\tt )quit}
and {\tt )pquit} behave in the same way.
If you do issue
\begin{verbatim}
)set quit unprotected
\end{verbatim}
we
suggest that you do not (somehow) assign {\tt )quit} to be
```

```
executed when you press, say, a function key.

\par\noindent{\bf Also See:}
\downlink{``)fin''}{ugSysCmdfinPage} in section B.10
\downlink{``)history''}{ugSysCmdhistoryPage} in section B.13
\downlink{``)close''}{ugSysCmdclosePage} in section B.5
\downlink{``)quit''}{ugSysCmdpquitPage} in section B.19
\downlink{``)system''}{ugSysCmdsystemPage} in section B.25

\endscroll
\autobuttons
\end{page}
```

———-

)read

⇒ "notitle" (ugInOutInPage) 9 on page 1722
⇒ "notitle" (ugSysCmdcompilePage) 19 on page 2549
⇒ "notitle" (ugSysCmdeditPage) 19 on page 2554
⇒ "notitle" (ugSysCmdhistoryPage) 19 on page 2559

— **ug16.ht** —

```
\begin{page}{ugSysCmdreadPage}{B.20. )read}
\beginscroll


\par\noindent{\bf User Level Required:} interpreter

\par\noindent{\bf Command Syntax:}
\begin{items}
\item {\tt )read} {\it \lanb{}fileName\ranb{}}
\item {\tt )read} {\it \lanb{}fileName\ranb{}}
\lanb{}{\tt )quiet}\ranb{} \lanb{}{\tt )ifthere}\ranb{}
\end{items}
\par\noindent{\bf Command Description:}

This command is used to read {\bf .input} files into Axiom.
The command
\begin{verbatim}
)read matrix.input
\end{verbatim}
will read the contents of the file {\bf matrix.input} into Axiom.  The
``.input'' file extension is optional.  See
\downlink{``Input Files and Output Styles''}{ugInOutInPage} in Section 4
\ignore{ugInOutIn} for more information about {\bf
```

```
.input} files.

This command remembers the previous file you edited, read or compiled.
If you do not specify a file name, the previous file will be read.

The {\tt )ifthere} option checks to see whether the {\bf .input} file
exists.  If it does not, the {\tt )read} command does nothing.  If you
do not use this option and the file does not exist, you are asked to
give the name of an existing {\bf .input} file.

The {\tt )quiet} option suppresses output while the file is being read.

\par\noindent{\bf Also See:}
\downlink{``)compile''}{ugSysCmdcompilePage} in section B.7
\downlink{``)edit''}{ugSysCmdeditPage} in section B.9
\downlink{``)history''}{ugSysCmdhistoryPage}  section B.13

\endscroll
\autobuttons
\end{page}
```

———-

)set

⇒ "notitle" (ugSysCmdquitPage) 19 on page 2569

— **ug16.ht** —

```
\begin{page}{ugSysCmdsetPage}{B.21. )set}
\beginscroll


\par\noindent{\bf User Level Required:} interpreter

\par\noindent{\bf Command Syntax:}
\begin{items}
\item {\tt )set}
\item {\tt )set} {\it label1 \lanb{}... labelN\ranb{}}
\item {\tt )set} {\it label1 \lanb{}... labelN\ranb{} newValue}
\end{items}
\par\noindent{\bf Command Description:}

The {\tt )set} command is used to view or set system variables that
control what messages are displayed, the type of output desired, the
status of the history facility, the way Axiom user functions are
cached, and so on.
Since this collection is very large, we will not discuss them here.
```

```
Rather, we will show how the facility is used.
We urge you to explore the {\tt )set} options to familiarize yourself
with how you can modify your Axiom working environment.
There is a Hyperdoc version of this same facility available from the
main Hyperdoc menu.
\texht{}{Click \lispmemolink{here}{(|htSystemVariables|)} to go to it.}

The {\tt )set} command is command-driven with a menu display.
It is tree-structured.
To see all top-level nodes, issue {\tt )set} by itself.
\begin{verbatim}
)set
\end{verbatim}
Variables with values have them displayed near the right margin.
Subtrees of selections have ``{\tt ...}''
displayed in the value field.
For example, there are many kinds of messages, so issue
{\tt )set message} to see the choices.
\begin{verbatim}
)set message
\end{verbatim}
The current setting  for the variable that displays
whether computation times
are displayed is visible in the menu displayed by the last command.
To see more information, issue
\begin{verbatim}
)set message time
\end{verbatim}
This shows that time printing is on now.
To turn it off, issue
\begin{verbatim}
)set message time off
\end{verbatim}

As noted above, not all settings have so many qualifiers.
For example, to change the {\tt )quit} command to being unprotected
(that is, you will not be prompted for verification), you need only issue
\begin{verbatim}
)set quit unprotected
\end{verbatim}

\par\noindent{\bf Also See:}
\downlink{``)quit''}{ugSysCmdquitPage} in section B.19

\endscroll
\autobuttons
\end{page}
```

)show

⇒ “notitle” (ugSysCmddisplayPage) 19 on page 2552
⇒ “notitle” (ugSysCmdsetPage) 19 on page 2571
⇒ “notitle” (ugSysCmdwhatPage) 19 on page 2586

— **ug16.ht** —

```
\begin{page}{ugSysCmdshowPage}{B.22. )show}
\beginscroll


\par\noindent{\bf User Level Required:} interpreter

\par\noindent{\bf Command Syntax:}
\begin{items}
\item{\tt )show {\it nameOrAbbrev}}
\item{\tt )show {\it nameOrAbbrev} )operations}
\item{\tt )show {\it nameOrAbbrev} )attributes}
\end{items}

\par\noindent{\bf Command Description:}
This command displays information about Axiom
domain, package and category {\it constructors}.
If no options are given, the {\tt )operations} option is assumed.
For example,
\begin{verbatim}
)show POLY
)show POLY )operations
)show Polynomial
)show Polynomial )operations
\end{verbatim}
each display basic information about the
\spadtype{Polynomial} domain constructor and then provide a
listing of operations.
Since \spadtype{Polynomial} requires a \spadtype{Ring} (for example,
\spadtype{Integer}) as argument, the above commands all refer
to a unspecified ring {\tt R}.
In the list of operations, \spadSyntax{\$} means
\spadtype{Polynomial(R)}.

The basic information displayed includes the {\it signature}
of the constructor (the name and arguments), the constructor
{\it abbreviation}, the {\it exposure status} of the constructor, and the
name of the {\it library source file} for the constructor.

If operation information about a specific domain is wanted,
the full or abbreviated domain name may be used.
For example,
```

```
\begin{verbatim}
)show POLY INT
)show POLY INT )operations
)show Polynomial Integer
)show Polynomial Integer )operations
\end{verbatim}
are among  the combinations that will
display the operations exported  by the
domain \spadtype{Polynomial(Integer)} (as opposed to the general
{\it domain constructor} \spadtype{Polynomial}).
Attributes may be listed by using the {\tt )attributes} option.

\par\noindent{\bf Also See:}
\downlink{``)display''}{ugSysCmddisplayPage} in section B.8
\downlink{``)set''}{ugSysCmdsetPage} in section B.21
\downlink{``)what''}{ugSysCmdwhatPage} in section B.28

\endscroll
\autobuttons
\end{page}
```

———

)spool

⇒ "notitle" (ugSysCmdcdPage) 19 on page 2545

— **ug16.ht** —

```
\begin{page}{ugSysCmdspoolPage}{B.23. )spool}
\beginscroll


\par\noindent{\bf User Level Required:} interpreter

\par\noindent{\bf Command Syntax:}
\begin{items}
\item{\tt )spool} \lanb{}{\it fileName}\ranb{}
\item{\tt )spool}
\end{items}

\par\noindent{\bf Command Description:}

This command is used to save {\it (spool)} all Axiom input and output
into a file, called a {\it spool file.}
You can only have one spool file active at a time.
To start spool, issue this command with a filename. For example,
\begin{verbatim}
```

```
)spool integrate.out
\end{verbatim}
To stop spooling, issue {\tt )spool} with no filename.

If the filename is qualified with a directory, then the output will be
placed in that directory.  If no directory information is given, the
spool file will be placed in the {\it current directory.}  The current
directory is the directory from which you started Axiom or is the
directory you specified using the {\tt )cd} command.

\par\noindent{\bf Also See:}
\downlink{``)cd''}{ugSysCmdcdPage} in section B.4
\ugSysCmdcdNumber

\endscroll
\autobuttons
\end{page}
```

———-

)synonym

⇒ "notitle" (ugSysCmdsetPage) 19 on page 2571
⇒ "notitle" (ugSysCmdwhatPage) 19 on page 2586

— **ug16.ht** —

```
\begin{page}{ugSysCmdsynonymPage}{B.24. )synonym}
\beginscroll


\par\noindent{\bf User Level Required:} interpreter

\par\noindent{\bf Command Syntax:}
\begin{items}
\item{\tt )synonym}
\item{\tt )synonym} {\it synonym fullCommand}
\item{\tt )what synonyms}
\end{items}

\par\noindent{\bf Command Description:}

This command is used to create short synonyms for system command
expressions.  For example, the following synonyms might simplify
commands you often use.
\begin{verbatim}
)synonym save         history )save
```

```
)synonym restore      history )restore
)synonym mail         system mail
)synonym ls           system ls
)synonym fortran      set output fortran
\end{verbatim}
Once defined, synonyms can be
used in place of the longer  command expressions.
Thus
\begin{verbatim}
)fortran on
\end{verbatim}
is the same as the longer
\begin{verbatim}
)set fortran output on
\end{verbatim}
To list all defined synonyms, issue either of
\begin{verbatim}
)synonyms
)what synonyms
\end{verbatim}
To list, say, all synonyms that contain the substring
``{\tt ap}'', issue
\begin{verbatim}
)what synonyms ap
\end{verbatim}

\par\noindent{\bf Also See:}
\downlink{``)set''}{ugSysCmdsetPage} in section B.21
\downlink{``)what''}{ugSysCmdwhatPage} in section B.28

\endscroll
\autobuttons
\end{page}
```

———-

)system

⇒ "notitle" (ugSysCmdbootPage) 19 on page 2544
⇒ "notitle" (ugSysCmdfinPage) 19 on page 2555
⇒ "notitle" (ugSysCmdlispPage) 19 on page 2565
⇒ "notitle" (ugSysCmdpquitPage) 19 on page 2567
⇒ "notitle" (ugSysCmdquitPage) 19 on page 2569

— **ug16.ht** —

```
\begin{page}{ugSysCmdsystemPage}{B.25. )system}
```

```
\beginscroll

\par\noindent{\bf User Level Required:} interpreter

\par\noindent{\bf Command Syntax:}
\begin{items}
\item{\tt )system} {\it cmdExpression}
\end{items}

\par\noindent{\bf Command Description:}

This command may be used to issue commands to the operating system
while remaining in Axiom.  The {\it cmdExpression} is passed to the
operating system for execution.

To get an operating system shell, issue, for example,
\spadcmd{)system sh}.
When you enter the key combination,
\texht{\fbox{\bf Ctrl}--\fbox{\bf D}}{{\bf Ctrl-D}}
(pressing and holding the
\texht{\fbox{\bf Ctrl}}{{\bf Ctrl}} key and then pressing the
\texht{\fbox{\bf D}}{{\bf D}} key)
the shell will terminate and you will return to Axiom.
We do not recommend this way of creating a shell because
\Lisp{} may field some interrupts instead of the shell.
If possible, use a shell running in another window.

If you execute programs that misbehave you may not be able to return
to Axiom.  If this happens, you may have no other choice than to
restart Axiom and restore the environment via {\tt )history )restore},
if possible.

\par\noindent{\bf Also See:}
\downlink{``)boot''}{ugSysCmdbootPage} in section B.3
\downlink{``)fin''}{ugSysCmdfinPage} in section B.10
\downlink{``)lisp''}{ugSysCmdlispPage} in section B.15
\downlink{``)quit''}{ugSysCmdpquitPage} in section B.18
\downlink{``)quit''}{ugSysCmdquitPage} in section B.19

\endscroll
\autobuttons
\end{page}
```

)trace

⇒ "notitle" (ugSysCmdcompilePage) 19 on page 2549
⇒ "notitle" (ugSysCmdbootPage) 19 on page 2544
⇒ "notitle" (ugSysCmdlispPage) 19 on page 2565
⇒ "notitle" (ugSysCmdltracePage) 19 on page 2566

— ug16.ht —

```
\begin{page}{ugSysCmdtracePage}{B.26. )trace}
\beginscroll


\par\noindent{\bf User Level Required:} interpreter

\par\noindent{\bf Command Syntax:}
\begin{items}
\item{\tt )trace}
\item{\tt )trace )off}

\item{\tt )trace} {\it function \lanb{}options\ranb{}}
\item{\tt )trace} {\it constructor \lanb{}options\ranb{}}
\item{\tt )trace} {\it domainOrPackage \lanb{}options\ranb{}}
\end{items}
%
where options can be one or more of
%
\begin{items}
\item{\tt )after} {\it S-expression}
\item{\tt )before} {\it S-expression}
\item{\tt )break after}
\item{\tt )break before}
\item{\tt )cond} {\it S-expression}
\item{\tt )count}
\item{\tt )count} {\it n}
\item{\tt )depth} {\it n}
\item{\tt )local} {\it op1 \lanb{}... opN\ranb{}}
\item{\tt )nonquietly}
\item{\tt )nt}
\item{\tt )off}
\item{\tt )only} {\it listOfDataToDisplay}
\item{\tt )ops}
\item{\tt )ops} {\it op1 \lanb{}... opN \ranb{}}
\item{\tt )restore}
\item{\tt )stats}
\item{\tt )stats reset}
\item{\tt )timer}
\item{\tt )varbreak}
\item{\tt )varbreak} {\it var1 \lanb{}... varN \ranb{}}
```

```
\item{\tt )vars}
\item{\tt )vars} {\it var1 \lanb{}... varN \ranb{}}
\item{\tt )within} {\it executingFunction}
\end{items}

\par\noindent{\bf Command Description:}

This command is used to trace the execution of functions that make
up the Axiom system, functions defined by users,
and functions from the system library.
Almost all options are available for each type of function but
exceptions will be noted below.

To list all functions, constructors, domains and packages that are
traced, simply issue
\begin{verbatim}
)trace
\end{verbatim}
To untrace everything that is traced, issue
\begin{verbatim}
)trace )off
\end{verbatim}
When a function is traced, the default system action is to display
the arguments to the function and the return value when the
function is exited.
Note that if a function is left via an action such as a {\tt THROW}, no
return value will be displayed.
Also, optimization of tail recursion may decrease the number of
times a function is actually invoked and so may cause less trace
information to be displayed.
Other information can be displayed or collected when a function is
traced and this is controlled by the various options.
Most options will be of interest only to Axiom system
developers.
If a domain or package is traced, the default action is to trace
all functions exported.

Individual interpreter, lisp or boot
functions can be traced by listing their names after
{\tt )trace}.
Any options that are present must follow the functions to be
traced.
\begin{verbatim}
)trace f
\end{verbatim}
traces the function {\tt f}.
To untrace {\tt f}, issue
\begin{verbatim}
)trace f )off
\end{verbatim}
```

```
Note that if a function name contains a special character, it will
be necessary to escape the character with an underscore
%
\begin{verbatim}
)trace _/D_,1
\end{verbatim}

To trace all domains or packages that are or will be created from a
particular constructor, give the constructor name or abbreviation
after {\tt )trace}.

\begin{verbatim}
)trace MATRIX
)trace List Integer
\end{verbatim}

The first command traces all domains currently instantiated with
\spadtype{Matrix}.  If additional domains are instantiated with this
constructor (for example, if you have used \spadtype{Matrix(Integer)}
and \spadtype{Matrix(Float)}), they will be automatically traced.  The
second command traces \spadtype{List(Integer)}.  It is possible to
trace individual functions in a domain or package.  See the {\tt )ops}
option below.

The following are the general options for the {\tt )trace}
command.

%!! system command parser doesn't treat general s-expressions correctly,
%!! I recommand not documenting )after )before and )cond
\indent{0}
\beginitems
%\item[{\tt )after} {\it S-expression}]
%causes the given \Lisp{} {\it S-expression} to be
%executed after exiting the traced function.

%\item[{\tt )before} {\it S-expression}]
%causes the given \Lisp{} {\it S-expression} to be
%executed before entering the traced function.

\item[{\tt )break after}]
causes a \Lisp{} break loop to be entered after
exiting the traced function.

\item[{\tt )break before}]
causes a \Lisp{} break loop to be entered before
entering the traced function.

\item[{\tt )break}]
is the same as \spadcmd{)break before}.
```

```
%\item[{\tt )cond} {\it S-expression}]
%causes trace information to be shown only if the given
%\Lisp{} {\it S-expression} evaluates to non-NIL.  For
%example, the following command causes the system function
%{\tt resolveTT} to be traced but to have the information
%displayed only if the value of the variable
%{\tt \$reportBottomUpFlag} is non-NIL.
%\begin{verbatim}
%)trace resolveTT )cond \_\$reportBottomUpFlag}
%\end{verbatim}

\item[{\tt )count}]
causes the system to keep a count of the number of times the
traced function is entered.  The total can be displayed with
{\tt )trace )stats} and cleared with {\tt )trace )stats reset}.

\item[{\tt )count} {\it n}]
causes information about the traced function to be displayed for
the first {\it n} executions.  After the \eth{\it n} execution, the
function is untraced.

\item[{\tt )depth} {\it n}]
causes trace information to be shown for only {\it n} levels of
recursion of the traced function.  The command
\begin{verbatim}
)trace fib )depth 10
\end{verbatim}
will cause the display of only 10 levels of trace information for
the recursive execution of a user function \userfun{fib}.

\item[{\tt )math}]
causes the function arguments and return value to be displayed in the
Axiom monospace two-dimensional math format.

\item[{\tt )nonquietly}]
causes the display of additional messages when a function is
traced.

\item[{\tt )nt}]
This suppresses all normal trace information.  This option is
useful if the {\tt )count} or {\tt )timer} options are used and
you are interested in the statistics but not the function calling
information.

\item[{\tt )off}]
causes untracing of all or specific functions.  Without an
argument, all functions, constructors, domains and packages are
untraced.  Otherwise, the given functions and other objects
are untraced.  To
immediately retrace the untraced functions, issue {\tt )trace
```

```
)restore}.

\item[{\tt )only} {\it listOfDataToDisplay}]
causes only specific trace information to be shown.  The items are
listed by using the following abbreviations:
\indent{0}
\beginitems
\item[a]        display all arguments
\item[v]        display return value
\item[1]        display first argument
\item[2]        display second argument
\item[15]       display the 15th argument, and so on
\enditems
\indent{0}
\enditems
\indent{0}
\indent{0}
\beginitems

\item[{\tt )restore}]
causes the last untraced functions to be retraced.  If additional
options are present, they are added to those previously in effect.

\item[{\tt )stats}]
causes the display of statistics collected by the use of the
{\tt )count} and {\tt )timer} options.

\item[{\tt )stats reset}]
resets to 0 the statistics collected by the use of the
{\tt )count} and {\tt )timer} options.

\item[{\tt )timer}]
causes the system to keep a count of execution times for the
traced function.  The total can be displayed with {\tt )trace
)stats} and cleared with {\tt )trace )stats reset}.

%!! only for lisp, boot, may not work in any case, recommend removing
%\item[{\tt )varbreak}]
%causes a \Lisp{} break loop to be entered after
%the assignment to any variable in the traced function.

\item[{\tt )varbreak} {\it var1 \lanb{}... varN\ranb{}}]
causes a \Lisp{} break loop to be entered after
the assignment to any of the listed variables in the traced
function.

\item[{\tt )vars}]
causes the display of the value of any variable after it is assigned
in the traced function.  Note that library code must have been
compiled (see
```

```
\downlink{``)compile''}{ugSysCmdcompilePage} in Section
B.7\ignore{ugSysCmdcompile}) using the {\tt
)vartrace} option in order to support this option.

\item[{\tt )vars} {\it var1 \lanb{}... varN\ranb{}}]
causes the display of the value of any of the specified variables
after they are assigned in the traced function.  Note that library
code must have been compiled (see
\downlink{``)compile''}{ugSysCmdcompilePage} in Section
B.7\ignore{ugSysCmdcompile}) using the {\tt
)vartrace} option in order to support this option.

\item[{\tt )within} {\it executingFunction}]
causes the display of trace information only if the traced
function is called when the given {\it executingFunction} is running.
\enditems
\indent{0}

The following are the options for tracing constructors, domains
and packages.

\indent{0}
\beginitems
\item[{\tt )local} {\it \lanb{}op1 \lanb{}... opN\ranb{}\ranb{}}]
causes local functions of the constructor to be traced.  Note that
to untrace an individual local function, you must use the fully
qualified internal name, using the escape character
\spadSyntax{\_} before the semicolon.
\begin{verbatim}
)trace FRAC )local
)trace FRAC_;cancelGcd )off
\end{verbatim}

\item[{\tt )ops} {\it op1 \lanb{}... opN\ranb{}}]
By default, all operations from a domain or package are traced
when the domain or package is traced.  This option allows you to
specify that only particular operations should be traced.  The
command
%
\begin{verbatim}
)trace Integer )ops min max _+ _-
\end{verbatim}
%
traces four operations from the domain \spadtype{Integer}.  Since
{\tt +} and {\tt -} are special
characters, it is necessary
to escape them with an underscore.
\enditems
\indent{0}
```

```
\par\noindent{\bf Also See:}
\downlink{``)boot''}{ugSysCmdbootPage} in section B.3
\downlink{``)lisp''}{ugSysCmdlispPage} in section B.15
\downlink{``)trace''}{ugSysCmdltracePage} in section B.26

\endscroll
\autobuttons
\end{page}
```

———-

)undo

⇒ "notitle" (ugSysCmdhistoryPage) 19 on page 2559

— **ug16.ht** —

```
\begin{page}{ugSysCmdundoPage}{B.27. )undo}
\beginscroll


\par\noindent{\bf User Level Required:} interpreter

\par\noindent{\bf Command Syntax:}
\begin{items}
\item{\tt )undo}
\item{\tt )undo} {\it integer}
\item{\tt )undo} {\it integer \lanb{}option\ranb{}}
\item{\tt )undo} {\tt )redo}
\end{items}
%
where {\it option} is one of
%
\begin{items}
\item{\tt )after}
\item{\tt )before}
\end{items}

\par\noindent{\bf Command Description:}

This command is used to
restore the state of the user environment to an earlier
point in the interactive session.
The argument of an {\tt )undo} is an integer which must designate some
step number in the interactive session.

\begin{verbatim}
)undo n
```

```
)undo n )after
\end{verbatim}
These commands return the state of the interactive
environment to that immediately after step {\tt n}.
If {\tt n} is a positive number, then {\tt n} refers to step nummber
{\tt n}. If {\tt n} is a negative number, it refers to the \eth{\tt n}
previous command (that is, undoes the effects of the last \smath{-n}
commands).

A {\tt )clear all} resets the {\tt )undo} facility.
Otherwise, an {\tt )undo} undoes the effect of {\tt )clear} with
options {\tt properties}, {\tt value}, and {\tt mode}, and
that of a previous {\tt undo}.
If any such system commands are given between steps \smath{n} and
\smath{n + 1} (\smath{n > 0}), their effect is undone
for {\tt )undo m} for any \texht{\smath{0 < m \leq n}.}{0 < m <= n}.

The command {\tt )undo} is equivalent to {\tt )undo -1} (it undoes
the effect of the previous user expression).
The command {\tt )undo 0} undoes any of the above system commands
issued since the last user expression.

\begin{verbatim}
)undo n )before
\end{verbatim}
This command returns the state of the interactive
environment to that immediately before step {\tt n}.
Any {\tt )undo} or {\tt )clear} system commands
given before step {\tt n} will not be undone.

\begin{verbatim}
)undo )redo
\end{verbatim}
This command reads the file {\tt redo.input}.
created by the last {\tt )undo} command.
This file consists of all user input lines, excluding those
backtracked over due to a previous {\tt )undo}.

\par\noindent{\bf Also See:}
\downlink{``)history''}{ugSysCmdhistoryPage} in section B.13
The command {\tt )history )write} will eliminate the ``undone'' command
lines of your program.

\endscroll
\autobuttons
\end{page}
```

)what

⇒ "notitle" (ugSysCmddisplayPage) 19 on page 2552
⇒ "notitle" (ugSysCmdsetPage) 19 on page 2571
⇒ "notitle" (ugSysCmdshowPage) 19 on page 2573

— **ug16.ht** —

```
\begin{page}{ugSysCmdwhatPage}{B.28. )what}
\beginscroll


\par\noindent{\bf User Level Required:} interpreter

\par\noindent{\bf Command Syntax:}
\begin{items}
\item{\tt )what categories} {\it pattern1} \lanb{}
{\it pattern2 ...\ranb{}}
\item{\tt )what commands  } {\it pattern1} \lanb{}
{\it pattern2 ...\ranb{}}
\item{\tt )what domains   } {\it pattern1} \lanb{}
{\it pattern2 ...\ranb{}}
\item{\tt )what operations} {\it pattern1} \lanb{}
{\it pattern2 ...\ranb{}}
\item{\tt )what packages  } {\it pattern1} \lanb{}
{\it pattern2 ...\ranb{}}
\item{\tt )what synonym   } {\it pattern1} \lanb{}
{\it pattern2 ...\ranb{}}
\item{\tt )what things    } {\it pattern1} \lanb{}
{\it pattern2 ...\ranb{}}
\item{\tt )apropos        } {\it pattern1} \lanb{}
{\it pattern2 ...\ranb{}}
\end{items}

\par\noindent{\bf Command Description:}

This command is used to display lists of things in the system.  The
patterns are all strings and, if present, restrict the contents of the
lists.  Only those items that contain one or more of the strings as
substrings are displayed.  For example,
\begin{verbatim}
)what synonym
\end{verbatim}
displays all command synonyms,
\begin{verbatim}
)what synonym ver
\end{verbatim}
displays all command synonyms containing the substring ``{\tt ver}'',
\begin{verbatim}
```

```
)what synonym ver pr
\end{verbatim}
displays all command synonyms
containing the substring  ``{\tt ver}'' or  the substring
``{\tt pr}''.
Output similar to the following will be displayed
\begin{verbatim}
---------------- System Command Synonyms -----------------

user-defined synonyms satisfying patterns:
      ver pr

  )apr ........................... )what things
  )apropos ....................... )what things
  )prompt ........................ )set message prompt
  )version ....................... )lisp *yearweek*
\end{verbatim}

Several other things can be listed with the {\tt )what} command:

\indent{0}
\beginitems
\item[{\tt categories}] displays a list of category constructors.
\item[{\tt commands}]  displays a list of  system commands available
at your user-level.
Your user-level
is set via the  {\tt )set userlevel} command.
To get a description of a particular command, such as ``{\tt )what}'',
issue {\tt )help what}.
\item[{\tt domains}]   displays a list of domain constructors.
\item[{\tt operations}] displays a list of operations in  the system
library. It  is recommended that you  qualify this command with one or
more patterns, as there are thousands of operations available.  For
example, say you are looking for functions that involve computation of
eigenvalues.  To find their names, try {\tt )what operations eig}.
A rather large list of operations  is loaded into the workspace when
this command  is first issued.  This  list will be deleted  when you
clear the workspace  via {\tt )clear all} or {\tt )clear completely}.
It will be re-created if it is needed again.
\item[{\tt packages}]  displays a list of package constructors.
\item[{\tt synonym}]  lists system command synonyms.
\item[{\tt things}]    displays all  of the  above types for  items
containing the pattern strings as  substrings.
The command synonym  {\tt )apropos} is equivalent to
{\tt )what things}.
\enditems
\indent{0}

\par\noindent{\bf Also See:}
\downlink{``)display''}{ugSysCmddisplayPage} in section B.8
```

```
\downlink{‘‘)set’’}{ugSysCmdsetPage} in section B.21
\downlink{‘‘)show’’}{ugSysCmdshowPage} in section B.22

\texht{\egroup}{}
\endscroll
\autobuttons
\end{page}
```

Chapter 20

Users Guide Chapter 21 (ug21.ht)

Programs for Axiom Images

— **ug21.ht** —

```
\begin{page}{ugAppGraphicsPage}{G. Programs for Axiom Images}
\beginscroll
%
This appendix contains the Axiom programs used to generate
the images in the \Gallery{} color insert of this book.
All these input files are included
with the Axiom system.
To produce the images
on page 6 of the \Gallery{} insert, for example, issue the command:
\begin{verbatim}
)read images6
\end{verbatim}

These images were produced on an IBM RS/6000 model 530 with a
standard color graphics adapter.  The smooth shaded images
were made from X Window System screen dumps.
The remaining images were produced with Axiom-generated
PostScript output.  The images were reproduced from slides made on an Agfa
ChromaScript PostScript interpreter with a Matrix Instruments QCR camera.

\beginmenu
    \menudownlink{{F.1. images1.input}}{ugFimagesOnePage}
    \menudownlink{{F.2. images2.input}}{ugFimagesTwoPage}
    \menudownlink{{F.3. images3.input}}{ugFimagesThreePage}
```

```
    \menudownlink{{F.4. images5.input}}{ugFimagesFivePage}
    \menudownlink{{F.5. images6.input}}{ugFimagesSixPage}
    \menudownlink{{F.6. images7.input}}{ugFimagesSevenPage}
    \menudownlink{{F.7. images8.input}}{ugFimagesEightPage}
    \menudownlink{{F.8. conformal.input}}{ugFconformalPage}
    \menudownlink{{F.9. tknot.input}}{ugFtknotPage}
    \menudownlink{{F.10. ntube.input}}{ugFntubePage}
    \menudownlink{{F.11. dhtri.input}}{ugFdhtriPage}
    \menudownlink{{F.12. tetra.input}}{ugFtetraPage}
    \menudownlink{{F.13. antoine.input}}{ugFantoinePage}
    \menudownlink{{F.14. scherk.input}}{ugFscherkPage}
\endmenu
\endscroll
\autobuttons
\end{page}
```

——–

images1.input

— ug21.ht —

```
\begin{page}{ugFimagesOnePage}{G.1. images1.input}
\beginscroll

\labelSpace{3pc}


\noindent
{\tt 1.\ \ \ )read\ tknot}\newline
{\tt 2.\ \ \ }\newline
{\tt 3.\ \ \ torusKnot(15,17,\ 0.1,\ 6,\ 700)}\newline

\noindent

\newpage
\endscroll
\autobuttons
\end{page}
```

——–

images2.input

— ug21.ht —

```
\begin{page}{ugFimagesTwoPage}{G.2. images2.input}
\beginscroll

These images illustrate how Newton's method converges when computing
the complex cube roots of 2.  Each point in the \smath{(x,y)}-plane
represents the complex number \smath{x + iy,} which is given as a
starting point for Newton's method.  The poles in these images
represent bad starting values.  The flat areas are the regions of
convergence to the three roots.


\noindent
{\tt 1.\ \ \ )read\ newton}\newline
{\tt 2.\ \ \ )read\ vectors}\newline
{\tt 3.\ \ \ f\ :=\ newtonStep(x**3\ -\ 2)}\newline
{\tt 4.\ \ \ }\newline

\noindent

The function \texht{$f^n$}{f**n} computes $n$ steps of Newton's method.


\noindent
{\tt 1.\ \ \ clipValue\ :=\ 4}\newline
{\tt 2.\ \ \ drawComplexVectorField(f**3,\ -3..3,\ -3..3)}\newline
{\tt 3.\ \ \ drawComplex(f**3,\ -3..3,\ -3..3)}\newline
{\tt 4.\ \ \ drawComplex(f**4,\ -3..3,\ -3..3)}\newline

\noindent

\endscroll
\autobuttons
\end{page}
```

————

images3.input

— ug21.ht —

```
\begin{page}{ugFimagesThreePage}{G.3. images3.input}
```

```
\beginscroll


\noindent
{\tt 1.\ \ \ )r\ tknot}\newline
{\tt 2.\ \ \ for\ i\ in\ 0..4\ repeat\
torusKnot(2,\ 2\ +\ i/4,\ 0.5,\ 25,\ 250)}\newline

\noindent

\endscroll
\autobuttons
\end{page}
```

———-

images5.input

— ug21.ht —

```
\begin{page}{ugFimagesFivePage}{G.4. images5.input}
\beginscroll

The parameterization of the Etruscan Venus is due to George Frances.


\noindent
{\tt 1.\ \ \ venus(a,r,steps)\ ==}\newline
{\tt 2.\ \ \ \ \ surf\ :=\
(u:DFLOAT,\ v:DFLOAT):\ Point\ DFLOAT\ +->}\newline
{\tt 3.\ \ \ \ \ \ \ cv\ :=\ cos(v)}\newline
{\tt 4.\ \ \ \ \ \ \ sv\ :=\ sin(v)}\newline
{\tt 5.\ \ \ \ \ \ \ cu\ :=\ cos(u)}\newline
{\tt 6.\ \ \ \ \ \ \ su\ :=\ sin(u)}\newline
{\tt 7.\ \ \ \ \ \ \ x\ :=\ r\ *\ cos(2*u)\ *\ cv\ +\ sv\ *\ cu}\newline
{\tt 8.\ \ \ \ \ \ \ y\ :=\ r\ *\ sin(2*u)\ *\ cv\ -\ sv\ *\ su}\newline
{\tt 9.\ \ \ \ \ \ \ z\ :=\ a\ *\ cv}\newline
{\tt 10.\ \ \ \ \ \ point\ [x,y,z]}\newline
{\tt 11.\ \ \ \
draw(surf,\ 0..\%pi,\ -\%pi..\%pi,\ var1Steps==steps,}\newline
{\tt 12.\ \ \ \ \ \ \ \ \
var2Steps==steps,\ title\ ==\ "Etruscan\ Venus")}\newline
{\tt 13.\ \ }\newline
{\tt 14.\ \ venus(5/2,\ 13/10,\ 50)}\newline

\noindent
```

```
The Figure-8 Klein Bottle
parameterization is from
‘‘Differential Geometry and Computer Graphics’’ by Thomas Banchoff,
in {\it Perspectives in Mathematics,} Anniversary of Oberwolfasch 1984,
Birkh\"{a}user-Verlag, Basel, pp. 43-60.


\noindent
{\tt 1.\ \ \ klein(x,y)\ ==}\newline
{\tt 2.\ \ \ \ \ cx\ :=\ cos(x)}\newline
{\tt 3.\ \ \ \ \ cy\ :=\ cos(y)}\newline
{\tt 4.\ \ \ \ \ sx\ :=\ sin(x)}\newline
{\tt 5.\ \ \ \ \ sy\ :=\ sin(y)}\newline
{\tt 6.\ \ \ \ \ sx2\ :=\ sin(x/2)}\newline
{\tt 7.\ \ \ \ \ cx2\ :=\ cos(x/2)}\newline
{\tt 8.\ \ \ \ \ sq2\ :=\ sqrt(2.0@DFLOAT)}\newline
{\tt 9.\ \ \ \ \
point\ [cx\ *\ (cx2\ *\ (sq2\ +\ cy)\ +\ (sx2\ *\ sy\ *\ cy)),\ \_}
\newline
{\tt 10.\ \ \ \ \ \ \ \ \ \ \
sx\ *\ (cx2\ *\ (sq2\ +\ cy)\ +\ (sx2\ *\ sy\ *\ cy)),\ \_}\newline
{\tt 11.\ \ \ \ \ \ \ \ \ \ \
-sx2\ *\ (sq2\ +\ cy)\ +\ cx2\ *\ sy\ *\ cy]}\newline
{\tt 12.\ \ }\newline
{\tt 13.\ \ draw(klein,\ 0..4*\%pi,\ 0..2*\%pi,\ var1Steps==50,}\newline
{\tt 14.\ \ \ \ \ \ \
var2Steps==50,title=="Figure\ Eight\ Klein\ Bottle")}\newline

\noindent

The next two images are examples of generalized tubes.


\noindent
{\tt 15.\ \ )read\ ntube}\newline
{\tt 16.\ \ rotateBy(p,\ theta)\ ==}\newline
{\tt 17.\ \ \ \ c\ :=\ cos(theta)}\newline
{\tt 18.\ \ \ \ s\ :=\ sin(theta)}\newline
{\tt 19.\ \ \ \ point\ [p.1*c\ -\ p.2*s,\ p.1*s\ +\ p.2*c]}\newline
{\tt 20.\ \ }\newline
{\tt 21.\ \ bcircle\ t\ ==\ }\newline
{\tt 22.\ \ \ \ point\ [3*cos\ t,\ 3*sin\ t,\ 0]}\newline
{\tt 23.\ \ \ }\newline
{\tt 24.\ \ twist(u,\ t)\ ==}\newline
{\tt 25.\ \ \ \ theta\ :=\ 4*t}\newline
{\tt 26.\ \ \ \ p\ :=\ point\ [sin\ u,\ cos(u)/2]}\newline
{\tt 27.\ \ \ \ rotateBy(p,\ theta)}\newline
{\tt 28.\ \ \ }\newline
{\tt 29.\ \ ntubeDrawOpt(bcircle,\ twist,\ 0..2*\%pi,\ 0..2*\%pi,}
\newline
```

```
{\tt 30.\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
var1Steps\ ==\ 70,\ var2Steps\ ==\ 250)}\newline
{\tt 31.\ \ }\newline
{\tt 32.\ \ twist2(u,\ t)\ ==}\newline
{\tt 33.\ \ \ \ theta\ :=\ t}\newline
{\tt 34.\ \ \ \ p\ :=\ point\ [sin\ u,\ cos(u)]}\newline
{\tt 35.\ \ \ \ rotateBy(p,\ theta)}\newline
{\tt 36.\ \ }\newline
{\tt 37.\ \ cf(u,v)\ ==\ sin(21*u)}\newline
{\tt 38.\ \ }\newline
{\tt 39.\ \ ntubeDrawOpt(bcircle,\ twist2,\ 0..2*\%pi,\ 0..2*\%pi,}
\newline
{\tt 40.\ \ \ \ colorFunction\ ==\ cf,\ var1Steps\ ==\ 168,}\newline
{\tt 41.\ \ \ \ var2Steps\ ==\ 126)}\newline

\noindent

\endscroll
\autobuttons
\end{page}
```

———-

images6.input

— ug21.ht —

```
\begin{page}{ugFimagesSixPage}{G.5. images6.input}
\beginscroll

\labelSpace{3pc}

\noindent
{\tt 1.\ \ \ gam(x,y)\ ==\ }\newline
{\tt 2.\ \ \ \ \ g\ :=\ Gamma\ complex(x,y)}\newline
{\tt 3.\ \ \ \ \
point\ [x,y,max(min(real\ g,\ 4),\ -4),\ argument\ g]}\newline
{\tt 4.\ \ \ }\newline
{\tt 5.\ \ \ }\newline
{\tt 6.\ \ \ draw(gam,\ -\%pi..\%pi,\ -\%pi..\%pi,\ }\newline
{\tt 7.\ \ \ \ \ \ \ \ title\ ==\ "Gamma(x\ +\ \%i*y)",\ \_}\newline
{\tt 8.\ \ \ \ \ \ \ \ var1Steps\ ==\ 100,\ var2Steps\ ==\ 100)}\newline
{\tt 9.\ \ \ }\newline
{\tt 10.\ \ b(x,y)\ ==\ Beta(x,y)}\newline
{\tt 11.\ \ }\newline
{\tt 12.\ \ draw(b,\ -3.1..3,\ -3.1\ ..\ 3,\ title\ ==\ "Beta(x,y)")}
\newline
```

```
{\tt 13.\ \ }\newline
{\tt 14.\ \ atf(x,y)\ ==\ }\newline
{\tt 15.\ \ \ \ a\ :=\ atan\ complex(x,y)}\newline
{\tt 16.\ \ \ \ point\ [x,y,real\ a,\ argument\ a]}\newline
{\tt 17.\ \ }\newline
{\tt 18.\ \ draw(atf,\ -3.0..\%pi,\ -3.0..\%pi)}\newline

\noindent


\endscroll
\autobuttons
\end{page}
```

———-

images7.input

— ug21.ht —

```
\begin{page}{ugFimagesSevenPage}{G.6. images7.input}
\beginscroll

First we look at the conformal
map \texht{$z \mapsto z + 1/z$}{z +-> z + 1/z}.
\labelSpace{2pc}

\noindent
{\tt 1.\ \ \ )read\ conformal}\newline
{\tt 2.\ \ \ }\newline
{\tt 3.\ \ \ }\newline
{\tt 4.\ \ \ f\ z\ ==\ z}\newline
{\tt 5.\ \ \ }\newline
{\tt 6.\ \ conformalDraw(f,\ -2..2,\ -2..2,\ 9,\ 9,\ "cartesian")}
\newline
{\tt 7.\ \ \ }\newline
{\tt 8.\ \ \ f\ z\ ==\ z\ +\ 1/z}\newline
{\tt 9.\ \ \ \ }\newline
{\tt 10.\ \ conformalDraw(f,\ -2..2,\ -2..2,\ 9,\ 9,\ "cartesian")}
\newline

\noindent

The map \texht{$z \mapsto -(z+1)/(z-1)$}{z +-> -(z+1)/(z-1)} maps
the unit disk to the right half-plane, as shown
on the Riemann sphere.
```

```
\noindent
{\tt 1.\ \ \ f\ z\ ==\ z}\newline
{\tt 2.\ \ \ }\newline
{\tt 3.\ \ \ riemannConformalDraw(f,0.1..0.99,0..2*\%pi,7,11,"polar")}
\newline
{\tt 4.\ \ \ }\newline
{\tt 5.\ \ \ f\ z\ ==\ -(z+1)/(z-1)}\newline
{\tt 6.\ \ \ }\newline
{\tt 7.\ \ \ riemannConformalDraw(f,0.1..0.99,0..2*\%pi,7,11,"polar")}
\newline
{\tt 8.\ \ \ }\newline
{\tt 9.\ \ \ riemannSphereDraw(-4..4,\ -4..4,\ 7,\ 7,\ "cartesian")}
\newline

\noindent

\endscroll
\autobuttons
\end{page}
```

---

images8.input

— ug21.ht —

```
\begin{page}{ugFimagesEightPage}{G.7. images8.input}
\beginscroll

\labelSpace{1pc}

\noindent
{\tt 1.\ \ \ )read\ dhtri}\newline
{\tt 2.\ \ \ )read\ tetra}\newline
{\tt 3.\ \ \ drawPyramid\ 4}\newline
{\tt 4.\ \ \ }\newline
{\tt 5.\ \ \ )read\ antoine}\newline
{\tt 6.\ \ \ drawRings\ 2}\newline
{\tt 7.\ \ \ }\newline
{\tt 8.\ \ \ )read\ scherk}\newline
{\tt 9.\ \ \ drawScherk(3,3)}\newline
{\tt 10.\ \ }\newline
{\tt 11.\ \ )read\ ribbonsnew}\newline
{\tt 12.\ \ drawRibbons([x**i\ for\ i\ in\ 1..5],\ x=-1..1,\ y=0..2)}\newline

\noindent
```

```
%\input{gallery/conformal.htex}
\endscroll
\autobuttons
\end{page}
```

———-

conformal.input

— **ug21.ht** —

```
\begin{page}{ugFconformalPage}{G.8. conformal.input}
\beginscroll
%
The functions in this section draw conformal maps both on the
plane and on the Riemann sphere.

%-- Compile, don't interpret functions.
%\xmpLine{)set fun comp on}{}

\noindent
{\tt 1.\ \ \ C\ :=\ Complex\ DoubleFloat}\newline
{\tt 2.\ \ \ S\ :=\ Segment\ DoubleFloat}\newline
{\tt 3.\ \ \ R3\ :=\ Point\ DFLOAT}\newline
{\tt 4.\ \ \ \ }\newline

\noindent

\userfun{conformalDraw}{\it (f, rRange, tRange, rSteps, tSteps,
coord)} draws the image of the coordinate grid under {\it f} in the
complex plane.  The grid may be given in either polar or Cartesian
coordinates.  Argument {\it f} is the function to draw; {\it rRange}
is the range of the radius (in polar) or real (in Cartesian);
{\it tRange} is the range of \texht{$\theta$}{\theta} (in polar) or
imaginary (in Cartesian); {\it tSteps, rSteps}, are the number of
intervals in the {\it r} and \texht{$\theta$}{\theta} directions; and
{\it coord} is the coordinate system to use (either {\tt "polar"} or
{\tt "cartesian"}).


\noindent
{\tt 1.\ \ \
conformalDraw:\ (C\ ->\ C,\ S,\ S,\ PI,\ PI,\ String)\ ->\ VIEW3D}\newline
{\tt 2.\ \ \
conformalDraw(f,rRange,tRange,rSteps,tSteps,coord)\ ==}\newline
```

```
{\tt 3.\ \ \ \ \ transformC\ :=}\newline
{\tt 4.\ \ \ \ \ \ \ coord\ =\ "polar"\ =>\ polar2Complex}\newline
{\tt 5.\ \ \ \ \ \ \ cartesian2Complex}\newline
{\tt 6.\ \ \ \ \ cm\ :=\ makeConformalMap(f,\ transformC)}\newline
{\tt 7.\ \ \ \ \ sp\ :=\ createThreeSpace()}\newline
{\tt 8.\ \ \ \ \
adaptGrid(sp,\ cm,\ rRange,\ tRange,\ rSteps,\ tSteps)}\newline
{\tt 9.\ \ \ \ \ makeViewport3D(sp,\ "Conformal\ Map")}\newline

\noindent

\userfun{riemannConformalDraw}{\it (f, rRange, tRange, rSteps, tSteps,
coord)} draws the image of the coordinate grid under {\it f} on the
Riemann sphere.  The grid may be given in either polar or Cartesian
coordinates.  Its arguments are the same as those for
\userfun{conformalDraw}.

\noindent
{\tt 10.\ \ riemannConformalDraw:(C->C,S,S,PI,PI,String)->VIEW3D}\newline
{\tt 11.\ \ riemannConformalDraw(f,\ rRange,\ tRange,}\newline
{\tt 12.\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
rSteps,\ tSteps,\ coord)\ ==}\newline
{\tt 13.\ \ \ \ transformC\ :=}\newline
{\tt 14.\ \ \ \ \ \ coord\ =\ "polar"\ =>\ polar2Complex}\newline
{\tt 15.\ \ \ \ \ \ cartesian2Complex}\newline
{\tt 16.\ \ \ \ sp\ :=\ createThreeSpace()}\newline
{\tt 17.\ \ \ \ cm\ :=\ makeRiemannConformalMap(f,\ transformC)}\newline
{\tt 18.\ \ \ \
adaptGrid(sp,\ cm,\ rRange,\ tRange,\ rSteps,\ tSteps)}\newline
{\tt 19.\ \ \ \
curve(sp,[point\ [0,0,2.0@DFLOAT,0],point\ [0,0,2.0@DFLOAT,0]])}\newline
{\tt 20.\ \ \ \
makeViewport3D(sp,"Map\ on\ the\ Riemann\ Sphere")}\newline
{\tt 21.\ \ }\newline
{\tt 22.\ \
adaptGrid(sp,\ f,\ uRange,\ vRange,\ \ uSteps,\ vSteps)\ ==}\newline
{\tt 23.\ \ \ \ delU\ :=\ (hi(uRange)\ -\ lo(uRange))/uSteps}\newline
{\tt 24.\ \ \ \ delV\ :=\ (hi(vRange)\ -\ lo(vRange))/vSteps}\newline
{\tt 25.\ \ \ \ uSteps\ :=\ uSteps\ +\ 1;\ vSteps\ :=\ vSteps\ +\ 1}\newline
{\tt 26.\ \ \ \ u\ :=\ lo\ uRange}\newline
{\tt 27.\ \ \ \ for\ i\ in\ 1..uSteps\ repeat}\newline
{\tt 28.\ \ \ \ \ \ c\ :=\ curryLeft(f,u)}\newline
{\tt 29.\ \ \ \ \ \ cf\ :=\ (t:DFLOAT):DFLOAT\ +->\ 0}\newline
{\tt 30.\ \ \ \ \ \
makeObject(c,vRange::SEG\ Float,colorFunction==cf,}\newline
{\tt 31.\ \ \ \ \ \ \ \
space\ ==\ sp,\ tubeRadius\ ==\ .02,\ tubePoints\ ==\ 6)}\newline
{\tt 32.\ \ \ \ \ \ u\ :=\ u\ +\ delU}\newline
{\tt 33.\ \ \ \ v\ :=\ lo\ vRange}\newline
{\tt 34.\ \ \ \ for\ i\ in\ 1..vSteps\ repeat}\newline
```

```
{\tt 35.\ \ \ \ \ \ c\ :=\ curryRight(f,v)}\newline
{\tt 36.\ \ \ \ \ \ cf\ :=\ (t:DFLOAT):DFLOAT\ +->\ 1}\newline
{\tt 37.\ \ \ \ \ \
makeObject(c,uRange::SEG\ Float,colorFunction==cf,}\newline
{\tt 38.\ \ \ \ \ \ \ \
space\ ==\ sp,\ tubeRadius\ ==\ .02,\ tubePoints\ ==\ 6)}\newline
{\tt 39.\ \ \ \ \ \ v\ :=\ v\ +\ delV}\newline
{\tt 40.\ \ \ \ void()}\newline
{\tt 41.\ \ }\newline
{\tt 42.\ \ riemannTransform(z)\ ==}\newline
{\tt 43.\ \ \ \ r\ :=\ sqrt\ norm\ z}\newline
{\tt 44.\ \ \ \ cosTheta\ :=\ (real\ z)/r}\newline
{\tt 45.\ \ \ \ sinTheta\ :=\ (imag\ z)/r}\newline
{\tt 46.\ \ \ \ cp\ :=\ 4*r/(4+r**2)}\newline
{\tt 47.\ \ \ \ sp\ :=\ sqrt(1-cp*cp)}\newline
{\tt 48.\ \ \ \ if\ r>2\ then\ sp\ :=\ -sp}\newline
{\tt 49.\ \ \ \ point\ [cosTheta*cp,\ sinTheta*cp,\ -sp\ +\ 1]}\newline
{\tt 50.\ \ \ }\newline
{\tt 51.\ \ cartesian2Complex(r:DFLOAT,\ i:DFLOAT):C\ ==}\newline
{\tt 52.\ \ \ \ complex(r,\ i)}\newline
{\tt 53.\ \ }\newline
{\tt 54.\ \ polar2Complex(r:DFLOAT,\ th:DFLOAT):C\ ==\ }\newline
{\tt 55.\ \ \ \ complex(r*cos(th),\ r*sin(th))}\newline
{\tt 56.\ \ }\newline
{\tt 57.\ \ makeConformalMap(f,\ transformC)\ ==}\newline
{\tt 58.\ \ \ \ (u:DFLOAT,v:DFLOAT):R3\ +->\ }\newline
{\tt 59.\ \ \ \ \ \ z\ :=\ f\ transformC(u,\ v)}\newline
{\tt 60.\ \ \ \ \ \ point\ [real\ z,\ imag\ z,\ 0.0@DFLOAT]}\newline
{\tt 61.\ \ \ }\newline
{\tt 62.\ \ makeRiemannConformalMap(f,\ transformC)\ ==}\newline
{\tt 63.\ \ \ \ (u:DFLOAT,\ v:DFLOAT):R3\ +->}\newline
{\tt 64.\ \ \ \ \ \ riemannTransform\ f\ transformC(u,\ v)}\newline
{\tt 65.\ \ }\newline
{\tt 66.\ \
riemannSphereDraw:\ (S,\ S,\ PI,\ PI,\ String)\ ->\ VIEW3D}\newline
{\tt 67.\ \
riemannSphereDraw(rRange,tRange,rSteps,tSteps,coord)\ ==}\newline
{\tt 68.\ \ \ \ transformC\ :=}\newline
{\tt 69.\ \ \ \ \ \ coord\ =\ "polar"\ =>\ polar2Complex}\newline
{\tt 70.\ \ \ \ \ \ cartesian2Complex}\newline
{\tt 71.\ \ \ \ grid\ :=\ (u:DFLOAT,\ v:DFLOAT):\ R3\ +->\ }\newline
{\tt 72.\ \ \ \ \ \ z1\ :=\ transformC(u,\ v)}\newline
{\tt 73.\ \ \ \ \ \ point\ [real\ z1,\ imag\ z1,\ 0]}\newline
{\tt 74.\ \ \ \ sp\ :=\ createThreeSpace()}\newline
{\tt 75.\ \ \ \
adaptGrid(sp,\ grid,\ rRange,\ tRange,\ rSteps,\ tSteps)}\newline
{\tt 76.\ \ \ \
connectingLines(sp,grid,rRange,tRange,rSteps,tSteps)}\newline
{\tt 77.\ \ \ \
makeObject(riemannSphere,0..2*\%pi,0..\%pi,space==sp)}\newline
```

```
{\tt 78.\ \ \ \ f\ :=\ (z:C):C\ +->\ z}\newline
{\tt 79.\ \ \ \
cm\ :=\ makeRiemannConformalMap(f,\ transformC)}\newline
{\tt 80.\ \ \ \
adaptGrid(sp,\ cm,\ rRange,\ tRange,\ rSteps,\ tSteps)}\newline
{\tt 81.\ \ \ \ makeViewport3D(sp,\ "Riemann\ Sphere")}\newline
{\tt 82.\ \ \ }\newline
{\tt 83.\ \
connectingLines(sp,f,uRange,vRange,uSteps,vSteps)\ ==}\newline
{\tt 84.\ \ \ \ delU\ :=\ (hi(uRange)\ -\ lo(uRange))/uSteps}\newline
{\tt 85.\ \ \ \ delV\ :=\ (hi(vRange)\ -\ lo(vRange))/vSteps}\newline
{\tt 86.\ \ \ \
uSteps\ :=\ uSteps\ +\ 1;\ vSteps\ :=\ vSteps\ +\ 1}\newline
{\tt 87.\ \ \ \ u\ :=\ lo\ uRange}\newline
{\tt 88.\ \ \ \ for\ i\ in\ 1..uSteps\ repeat}\newline
{\tt 89.\ \ \ \ \ \ v\ :=\ lo\ vRange}\newline
{\tt 90.\ \ \ \ \ \ for\ j\ in\ 1..vSteps\ repeat}\newline
{\tt 91.\ \ \ \ \ \ \ \ p1\ :=\ f(u,v)}\newline
{\tt 92.\ \ \ \ \ \ \ \
p2\ :=\ riemannTransform\ complex(p1.1,\ p1.2)}\newline
{\tt 93.\ \ \ \ \ \ \ \ fun\ :=\ lineFromTo(p1,p2)}\newline
{\tt 94.\ \ \ \ \ \ \ \ cf\ :=\ (t:DFLOAT):DFLOAT\ +->\ 3}\newline
{\tt 95.\ \ \ \ \ \ \ \
makeObject(fun,\ 0..1,space==sp,tubePoints==4,}\newline
{\tt 96.\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
tubeRadius==0.01,colorFunction==cf)}\newline
{\tt 97.\ \ \ \ \ \ \ \ v\ :=\ v\ +\ delV}\newline
{\tt 98.\ \ \ \ \ \ u\ :=\ u\ +\ delU}\newline
{\tt 99.\ \ \ \ void()}\newline
{\tt 100.\ \ }\newline
{\tt 101.\ riemannSphere(u,v)\ ==\ }\newline
{\tt 102.\ \ \ sv\ :=\ sin(v)}\newline
{\tt 103.\ \ \
0.99@DFLOAT*(point\ [cos(u)*sv,sin(u)*sv,cos(v),0.0@DFLOAT])+}\newline
{\tt 104.\ \ \ \ \
point\ [0.0@DFLOAT,\ 0.0@DFLOAT,\ 1.0@DFLOAT,\ 4.0@DFLOAT]}\newline
{\tt 105.\ \ }\newline
{\tt 106.\ lineFromTo(p1,\ p2)\ ==}\newline
{\tt 107.\ \ \ d\ :=\ p2\ -\ p1}\newline
{\tt 108.\ \ \ (t:DFLOAT):Point\ DFLOAT\ +->}\newline
{\tt 109.\ \ \ \ \ p1\ +\ t*d}\newline

\noindent

%\input{gallery/tknot.htex}
\endscroll
\autobuttons
\end{page}
```

———-

tknot.input

— ug21.ht —

```
\begin{page}{ugFtknotPage}{G.9. tknot.input}
\beginscroll
%
Create a $(p,q)$ torus-knot with radius $r$ around the curve.
The formula was derived by Larry Lambe.


\noindent
{\tt 1.\ \ \ )read\ ntube}\newline
{\tt 2.\ \ \
torusKnot:\ (DFLOAT,\ DFLOAT,\ DFLOAT,\ PI,\ PI)\ ->\ VIEW3D}\newline
{\tt 3.\ \ \ torusKnot(p,\ q\ ,r,\ uSteps,\ tSteps)\ ==}\newline
{\tt 4.\ \ \ \ \ knot\ :=\ (t:DFLOAT):Point\ DFLOAT\ +->\ }\newline
{\tt 5.\ \ \ \ \ \ \ fac\ :=\ 4/(2.2@DFLOAT-sin(q*t))}\newline
{\tt 6.\ \ \ \ \ \ \
fac\ *\ point\ [cos(p*t),\ sin(p*t),\ cos(q*t)]}\newline
{\tt 7.\ \ \ \ \
circle\ :=\ (u:DFLOAT,\ t:DFLOAT):\ Point\ DFLOAT\ +->}\newline
{\tt 8.\ \ \ \ \ \ \ r\ *\ point\ [cos\ u,\ sin\ u]}\newline
{\tt 9.\ \ \ \ \
ntubeDrawOpt(knot,\ circle,\ 0..2*\%pi,\ 0..2*\%pi,}\newline
{\tt 10.\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
var1Steps\ ==\ uSteps,\ var2Steps\ ==\ tSteps)}\newline
{\tt 11.\ \ }\newline

\noindent

%\input{gallery/ntube.htex}
\endscroll
\autobuttons
\end{page}
```

———-

ntube.input

⇒ "notitle" (ugFimagesFivePage) 20 on page 2592

— ug21.ht —

```
\begin{page}{ugFntubePage}{G.10. ntube.input}
\beginscroll

The functions in this file create generalized tubes (also known as
generalized cylinders).  These functions draw a 2-d curve in the
normal planes around a 3-d curve.


\noindent
{\tt 1.\ \ \ R3\ :=\ Point\ DFLOAT}\newline
{\tt 2.\ \ \ R2\ :=\ Point\ DFLOAT}\newline
{\tt 3.\ \ \ S\ :=\ Segment\ Float}\newline
{\tt 4.\ \ \ }\newline
{\tt 5.\ \ \ ThreeCurve\ :=\ DFLOAT\ ->\ R3}\newline
{\tt 6.\ \ \ TwoCurve\ :=\ (DFLOAT,\ DFLOAT)\ ->\ R2}\newline
{\tt 7.\ \ \ Surface\ :=\ (DFLOAT,\ DFLOAT)\ ->\ R3}\newline
{\tt 8.\ \ \ }\newline
{\tt 9.\ \ \ FrenetFrame\ :=\ }\newline
{\tt 10.\ \ \ \ \
Record(value:R3,tangent:R3,normal:R3,binormal:R3)}\newline
{\tt 11.\ \ frame:\ FrenetFrame}\newline
{\tt 12.\ \ }\newline

\noindent

\userfun{ntubeDraw}{\it (spaceCurve, planeCurve,}
$u_0 .. u_1,$ $t_0 .. t_1)$ draws
{\it planeCurve} in the normal planes of {\it
spaceCurve.}  The parameter $u_0 .. u_1$ specifies the parameter range
for {\it planeCurve} and $t_0 .. t_1$ specifies the parameter range
for {\it spaceCurve}.  Additionally, the plane curve function takes a
second parameter: the current parameter of {\it spaceCurve}.  This
allows the plane curve to change shape as it goes around the space
curve.


\noindent
{\tt 1.\ \ \
ntubeDraw:\ (ThreeCurve,TwoCurve,S,S)\ ->\ VIEW3D}\newline
{\tt 2.\ \ \
ntubeDraw(spaceCurve,planeCurve,uRange,tRange)\ ==}\newline
{\tt 3.\ \ \ \ \
ntubeDrawOpt(spaceCurve,\ planeCurve,\ uRange,\ \_}\newline
{\tt 4.\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
tRange,\ []\$List\ DROPT)}\newline
{\tt 5.\ \ \ \ }\newline
{\tt 6.\ \ \
ntubeDrawOpt:\ (ThreeCurve,TwoCurve,S,S,List\ DROPT)}\newline
{\tt 7.\ \ \ \ \ \ \ ->\ VIEW3D}\newline
{\tt 8.\ \ \
```

```
ntubeDrawOpt(spaceCurve,planeCurve,uRange,tRange,l)\ ==}\newline
{\tt 9.\ \ \ \ \ \ \ \ \ \ \ \ \ }\newline
{\tt 10.\ \ \ \
delT:DFLOAT\ :=\ (hi(tRange)\ -\ lo(tRange))/10000}\newline
{\tt 11.\ \ \ \ oldT:DFLOAT\ :=\ lo(tRange)\ -\ 1}\newline
{\tt 12.\ \ \ \
fun\ :=\ ngeneralTube(spaceCurve,planeCurve,delT,oldT)}\newline
{\tt 13.\ \ \ \ draw(fun,\ uRange,\ tRange,\ l)}\newline
{\tt 14.\ \ }\newline

\noindent

\userfun{nfrenetFrame}{\it (c, t, delT)}
numerically computes the Frenet frame
about the curve {\it c} at {\it t}.
Parameter {\it delT} is a small number used to
compute derivatives.

\noindent
{\tt 15.\ \ nfrenetFrame(c,\ t,\ delT)\ ==}\newline
{\tt 16.\ \ \ \ f0\ :=\ c(t)}\newline
{\tt 17.\ \ \ \ f1\ :=\ c(t+delT)}\newline
{\tt 18.\ \ \ \ t0\ :=\ f1\ -\ f0}\newline
{\tt 19.\ \ \ \
n0\ :=\ f1\ +\ f0\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ }\newline
{\tt 20.\ \ \ \ b\ :=\ cross(t0,\ n0)}\newline
{\tt 21.\ \ \ \ n\ :=\ cross(b,t0)}\newline
{\tt 22.\ \ \ \ ln\ :=\ length\ n}\newline
{\tt 23.\ \ \ \ lb\ :=\ length\ b}\newline
{\tt 24.\ \ \ \ ln\ =\ 0\ or\ lb\ =\ 0\ =>}\newline
{\tt 25.\ \ \ \ \ \ \ \
error\ "Frenet\ Frame\ not\ well\ defined"}\newline
{\tt 26.\ \ \ \ n\ :=\ (1/ln)*n}\newline
{\tt 27.\ \ \ \ b\ :=\ (1/lb)*b}\newline
{\tt 28.\ \ \ \ [f0,\ t0,\ n,\ b]\$FrenetFrame}\newline

\noindent

\userfun{ngeneralTube}{\it (spaceCurve, planeCurve,}{\it  delT, oltT)}
creates a function that can be passed to the system axiomFun{draw} command.
The function is a parameterized surface for the general tube
around {\it spaceCurve}.  {\it delT} is a small number used to compute
derivatives. {\it oldT} is used to hold the current value of the
{\it t} parameter for {\it spaceCurve.}  This is an efficiency measure
to ensure that frames are only computed once for each value of {\it t}.

\noindent
{\tt 29.\ \
ngeneralTube:\ (ThreeCurve,\ TwoCurve,\ DFLOAT,\ DFLOAT)\ ->\ Surface}
\newline
```

```
{\tt 30.\ \
ngeneralTube(spaceCurve,\ planeCurve,\ delT,\ oldT)\ ==}\newline
{\tt 31.\ \ \ \ free\ frame}\newline
{\tt 32.\ \ \ \ (v:DFLOAT,\ t:\ DFLOAT):\ R3\ +->}\newline
{\tt 33.\ \ \ \ \ \ if\ (t\ \texht{$\sim$}{~}=\ oldT)\ then}\newline
{\tt 34.\ \ \ \ \ \ \ \
frame\ :=\ nfrenetFrame(spaceCurve,\ t,\ delT)}\newline
{\tt 35.\ \ \ \ \ \ \ \ oldT\ :=\ t}\newline
{\tt 36.\ \ \ \ \ \ p\ :=\ planeCurve(v,\ t)}\newline
{\tt 37.\ \ \ \ \ \ frame.value\ +\ p.1*frame.normal\ +\ p.2*frame.binormal}\newline

\noindent

%\input{gallery/dhtri.htex}
\endscroll
\autobuttons
\end{page}
```

———-

dhtri.input

— ug21.ht —

```
\begin{page}{ugFdhtriPage}{G.11. dhtri.input}
\beginscroll
%
Create affine transformations (DH matrices) that transform
a given triangle into another.


\noindent
{\tt 1.\ \ \ tri2tri:\ (List\ Point\ DFLOAT,
\ List\ Point\ DFLOAT)\ ->\ DHMATRIX(DFLOAT)}\newline
{\tt 2.\ \ \ tri2tri(t1,\ t2)\ ==}\newline
{\tt 3.\ \ \ \ \ n1\ :=\ triangleNormal(t1)}\newline
{\tt 4.\ \ \ \ \ n2\ :=\ triangleNormal(t2)}\newline
{\tt 5.\ \ \ \ \ tet2tet(concat(t1,\ n1),\ concat(t2,\ n2))}\newline
{\tt 6.\ \ \ \ }\newline
{\tt 7.\ \ \ tet2tet:\ (List\ Point\ DFLOAT,
\ List\ Point\ DFLOAT)\ ->\ DHMATRIX(DFLOAT)}\newline
{\tt 8.\ \ \ tet2tet(t1,\ t2)\ ==}\newline
{\tt 9.\ \ \ \ \ m1\ :=\ makeColumnMatrix\ t1}\newline
{\tt 10.\ \ \ \ m2\ :=\ makeColumnMatrix\ t2}\newline
{\tt 11.\ \ \ \ m2\ *\ inverse(m1)}\newline
{\tt 12.\ \ \ }\newline
{\tt 13.\ \ makeColumnMatrix(t)\ ==}\newline
```

```
{\tt 14.\ \ \ \ m\ :=\ new(4,4,0)\$DHMATRIX(DFLOAT)}\newline
{\tt 15.\ \ \ \ for\ x\ in\ t\ for\ i\ in\ 1..repeat}\newline
{\tt 16.\ \ \ \ \ \ for\ j\ in\ 1..3\ repeat}\newline
{\tt 17.\ \ \ \ \ \ \ \ m(j,i)\ :=\ x.j}\newline
{\tt 18.\ \ \ \ \ \ m(4,i)\ :=\ 1}\newline
{\tt 19.\ \ \ \ m}\newline
{\tt 20.\ \ \ }\newline
{\tt 21.\ \ triangleNormal(t)\ ==}\newline
{\tt 22.\ \ \ \ a\ :=\ triangleArea\ t}\newline
{\tt 23.\ \ \ \ p1\ :=\ t.2\ -\ t.1}\newline
{\tt 24.\ \ \ \ p2\ :=\ t.3\ -\ t.2}\newline
{\tt 25.\ \ \ \ c\ :=\ cross(p1,\ p2)}\newline
{\tt 26.\ \ \ \ len\ :=\ length(c)}\newline
{\tt 27.\ \ \ \ len\ =\ 0\ =>\ error\ "degenerate\ triangle!"}\newline
{\tt 28.\ \ \ \ c\ :=\ (1/len)*c}\newline
{\tt 29.\ \ \ \ t.1\ +\ sqrt(a)\ *\ c}\newline
{\tt 30.\ \ \ }\newline
{\tt 31.\ \ triangleArea\ t\ ==}\newline
{\tt 32.\ \ \ \ a\ :=\ length(t.2\ -\ t.1)}\newline
{\tt 33.\ \ \ \ b\ :=\ length(t.3\ -\ t.2)}\newline
{\tt 34.\ \ \ \ c\ :=\ length(t.1\ -\ t.3)}\newline
{\tt 35.\ \ \ \ s\ :=\ (a+b+c)/2}\newline
{\tt 36.\ \ \ \ sqrt(s*(s-a)*(s-b)*(s-c))}\newline

\noindent


\endscroll
\autobuttons
\end{page}
```

——————-

tetra.input

— **ug21.ht** —

```
\begin{page}{ugFtetraPage}{G.12. tetra.input}
\beginscroll
%
%\input{gallery/tetra.htex}
%\outdent{Sierpinsky's Tetrahedron}

\labelSpace{3pc}


\noindent
```

```
{\tt 1.\ \ \ )set\ expose\ add\ con\ DenavitHartenbergMatrix}\newline
{\tt 2.\ \ \ }\newline
{\tt 3.\ \ \ x1:DFLOAT\ :=\ sqrt(2.0@DFLOAT/3.0@DFLOAT)}\newline
{\tt 4.\ \ \ x2:DFLOAT\ :=\ sqrt(3.0@DFLOAT)/6}\newline
{\tt 5.\ \ \ }\newline
{\tt 6.\ \ \ p1\ :=\ point\ [-0.5@DFLOAT,\ -x2,\ 0.0@DFLOAT]}\newline
{\tt 7.\ \ \ p2\ :=\ point\ [0.5@DFLOAT,\ -x2,\ 0.0@DFLOAT]}\newline
{\tt 8.\ \ \ p3\ :=\ point\ [0.0@DFLOAT,\ 2*x2,\ 0.0@DFLOAT]}\newline
{\tt 9.\ \ \ p4\ :=\ point\ [0.0@DFLOAT,\ 0.0@DFLOAT,\ x1]}\newline
{\tt 10.\ \ \ }\newline
{\tt 11.\ \ baseTriangle\ \ :=\ [p2,\ p1,\ p3]}\newline
{\tt 12.\ \ }\newline
{\tt 13.\ \ mt\ \ :=\
[0.5@DFLOAT*(p2+p1),\ 0.5@DFLOAT*(p1+p3),\ 0.5@DFLOAT*(p3+p2)]}\newline
{\tt 14.\ \ }\newline
{\tt 15.\ \ bt1\ :=\ [mt.1,\ p1,\ mt.2]}\newline
{\tt 16.\ \ bt2\ :=\ [p2,\ mt.1,\ mt.3]}\newline
{\tt 17.\ \ bt3\ :=\ [mt.2,\ p3,\ mt.3]}\newline
{\tt 18.\ \ bt4\ :=\
[0.5@DFLOAT*(p2+p4),\ 0.5@DFLOAT*(p1+p4),\ 0.5@DFLOAT*(p3+p4)]}\newline
{\tt 19.\ \ }\newline
{\tt 20.\ \ tt1\ :=\ tri2tri(baseTriangle,\ bt1)}\newline
{\tt 21.\ \ tt2\ :=\ tri2tri(baseTriangle,\ bt2)}\newline
{\tt 22.\ \ tt3\ :=\ tri2tri(baseTriangle,\ bt3)}\newline
{\tt 23.\ \ tt4\ :=\ tri2tri(baseTriangle,\ bt4)}\newline
{\tt 24.\ \ }\newline
{\tt 25.\ \ drawPyramid(n)\ ==}\newline
{\tt 26.\ \ \ \ s\ :=\ createThreeSpace()}\newline
{\tt 27.\ \ \ \ dh\ :=\ rotatex(0.0@DFLOAT)}\newline
{\tt 28.\ \ \ \ drawPyramidInner(s,\ n,\ dh)}\newline
{\tt 29.\ \ \ \ makeViewport3D(s,\ "Sierpinsky\ Tetrahedron")}\newline
{\tt 30.\ \ }\newline
{\tt 31.\ \ drawPyramidInner(s,\ n,\ dh)\ ==}\newline
{\tt 32.\ \ \ \ n\ =\ 0\ =>\ makeTetrahedron(s,\ dh,\ n)}\newline
{\tt 33.\ \ \ \ drawPyramidInner(s,\ n-1,\ dh\ *\ tt1)}\newline
{\tt 34.\ \ \ \ drawPyramidInner(s,\ n-1,\ dh\ *\ tt2)}\newline
{\tt 35.\ \ \ \ drawPyramidInner(s,\ n-1,\ dh\ *\ tt3)}\newline
{\tt 36.\ \ \ \ drawPyramidInner(s,\ n-1,\ dh\ *\ tt4)}\newline
{\tt 37.\ \ }\newline
{\tt 38.\ \ makeTetrahedron(sp,\ dh,\ color)\ ==}\newline
{\tt 39.\ \ \ \ w1\ :=\ dh*p1}\newline
{\tt 40.\ \ \ \ w2\ :=\ dh*p2}\newline
{\tt 41.\ \ \ \ w3\ :=\ dh*p3}\newline
{\tt 42.\ \ \ \ w4\ :=\ dh*p4}\newline
{\tt 43.\ \ \ \ polygon(sp,\ [w1,\ w2,\ w4])}\newline
{\tt 44.\ \ \ \ polygon(sp,\ [w1,\ w3,\ w4])}\newline
{\tt 45.\ \ \ \ polygon(sp,\ [w2,\ w3,\ w4])}\newline
{\tt 46.\ \ \ \ void()}\newline

\noindent
```

```
%\input{gallery/antoine.htex}
\endscroll
\autobuttons
\end{page}
```

———-

antoine.input

— **ug21.ht** —

```
\begin{page}{ugFantoinePage}{G.13. antoine.input}
\beginscroll
%
Draw Antoine's Necklace.
Thank you to Matthew Grayson at IBM's T.J Watson Research Center for the idea.


\noindent
{\tt 1.\ \ \ )set\ expose\ add\ con\ DenavitHartenbergMatrix}\newline
{\tt 2.\ \ \ }\newline
{\tt 3.\ \ \ torusRot:\ DHMATRIX(DFLOAT)}\newline
{\tt 4.\ \ \ }\newline
{\tt 5.\ \ \ }\newline
{\tt 6.\ \ \ drawRings(n)\ ==}\newline
{\tt 7.\ \ \ \ \ s\ :=\ createThreeSpace()}\newline
{\tt 8.\ \ \ \ \ dh:DHMATRIX(DFLOAT)\ :=\ identity()}\newline
{\tt 9.\ \ \ \ \ drawRingsInner(s,\ n,\ dh)}\newline
{\tt 10.\ \ \ \ makeViewport3D(s,\ "Antoine's\ Necklace")}\newline
{\tt 11.\ \ }\newline

\noindent

In order to draw Antoine rings, we take one ring, scale it down to a
smaller size, rotate it around its central axis, translate it to the
edge of the larger ring and rotate it around the edge to a point
corresponding to its count (there are 10 positions around the edge of
the larger ring). For each of these new rings we recursively perform
the operations, each ring becoming 10 smaller rings. Notice how the
\axiomType{DHMATRIX} operations are used to build up the proper matrix
composing all these transformations.


\noindent
{\tt 1.\ \ \ drawRingsInner(s,\ n,\ dh)\ ==}\newline
```

```
{\tt 2.\ \ \ \ \ n\ =\ 0\ =>}\newline
{\tt 3.\ \ \ \ \ \ \ drawRing(s,\ dh)}\newline
{\tt 4.\ \ \ \ \ \ \ void()}\newline
{\tt 5.\ \ \ \ \ t\ :=\ 0.0@DFLOAT}\newline
{\tt 6.\ \ \ \ \ p\ :=\ 0.0@DFLOAT}\newline
{\tt 7.\ \ \ \ \ tr\ :=\ 1.0@DFLOAT}\newline
{\tt 8.\ \ \ \ \ inc\ :=\ 0.1@DFLOAT}\newline
{\tt 9.\ \ \ \ \ for\ i\ in\ 1..10\ repeat}\newline
{\tt 10.\ \ \ \ \ \ tr\ :=\ tr\ +\ inc}\newline
{\tt 11.\ \ \ \ \ \ inc\ :=\ -inc}\newline
{\tt 12.\ \ \ \ \ \
dh'\ :=\ dh*rotatez(t)*translate(tr,0.0@DFLOAT,0.0@DFLOAT)*}\newline
{\tt 13.\ \ \ \ \ \ \ \ \ \ \ \ \ \
rotatey(p)*scale(0.35@DFLOAT,\ 0.48@DFLOAT,\ 0.4@DFLOAT)}\newline
{\tt 14.\ \ \ \ \ \ drawRingsInner(s,\ n-1,\ dh')}\newline
{\tt 15.\ \ \ \ \ \ t\ :=\ t\ +\ 36.0@DFLOAT}\newline
{\tt 16.\ \ \ \ \ \ p\ :=\ p\ +\ 90.0@DFLOAT}\newline
{\tt 17.\ \ \ \ void()}\newline
{\tt 18.\ \ }\newline
{\tt 19.\ \ drawRing(s,\ dh)\ ==}\newline
{\tt 20.\ \ \ \ free\ torusRot}\newline
{\tt 21.\ \ \ \ torusRot\ :=\ dh}\newline
{\tt 22.\ \ \ \
makeObject(torus,\ 0..2*\%pi,\ 0..2*\%pi,\ var1Steps\ ==\ 6,}\newline
{\tt 23.\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
space\ ==\ s,\ var2Steps\ ==\ 15)}\newline
{\tt 24.\ \ }\newline
{\tt 25.\ \ torus(u\ ,v)\ ==}\newline
{\tt 26.\ \ \ \ cu\ :=\ cos(u)/6}\newline
{\tt 27.\ \ \ \
torusRot*point\ [(1+cu)*cos(v),(1+cu)*sin(v),(sin\ u)/6]}\newline

\noindent

%\input{gallery/scherk.htex}
\endscroll
\autobuttons
\end{page}
```

———-

scherk.input

— ug21.ht —

```
\begin{page}{ugFscherkPage}{G.14. scherk.input}
\beginscroll
```

```
%

Scherk's minimal surface, defined by:
\texht{$e^z \cos(x) = \cos(y)$}{\axiom{exp(z) * cos(x) = cos(y)}}.
See: {\it A Comprehensive Introduction to Differential Geometry,} Vol. 3,
by Michael Spivak, Publish Or Perish, Berkeley, 1979, pp. 249-252.


\noindent
{\tt 1.\ \ \ (xOffset,\ yOffset):DFLOAT}\newline
{\tt 2.\ \ \ \ }\newline
{\tt 3.\ \ \ }\newline
{\tt 4.\ \ \ drawScherk(m,n)\ ==}\newline
{\tt 5.\ \ \ \ \ \ free\ xOffset,\ yOffset}\newline
{\tt 6.\ \ \ \ \ \ space\ :=\ createThreeSpace()}\newline
{\tt 7.\ \ \ \ \ \ for\ i\ in\ 0..m-1\ repeat}\newline
{\tt 8.\ \ \ \ \ \ \ \ xOffset\ :=\ i*\%pi}\newline
{\tt 9.\ \ \ \ \ \ \ \ for\ j\ in\ 0\ ..\ n-1\ repeat}\newline
{\tt 10.\ \ \ \ \ \ \ \ \ rem(i+j,\ 2)\ =\ 0\ =>\ 'iter}\newline
{\tt 11.\ \ \ \ \ \ \ \ \ yOffset\ :=\ j*\%pi}\newline
{\tt 12.\ \ \ \ \ \ \ \ \ drawOneScherk(space)}\newline
{\tt 13.\ \ \ \
makeViewport3D(space,\ "Scherk's\ Minimal\ Surface")}\newline
{\tt 14.\ \ }\newline
{\tt 15.\ \ scherk1(u,v)\ ==}\newline
{\tt 16.\ \ \ \ x\ :=\ cos(u)/exp(v)}\newline
{\tt 17.\ \ \ \
point\ [xOffset\ +\ acos(x),\ yOffset\ +\ u,\ v,\ abs(v)]}\newline
{\tt 18.\ \ \ }\newline
{\tt 19.\ \ scherk2(u,v)\ ==}\newline
{\tt 20.\ \ \ \ x\ :=\ cos(u)/exp(v)}\newline
{\tt 21.\ \ \ \
point\ [xOffset\ -\ acos(x),\ yOffset\ +\ u,\ v,\ abs(v)]}\newline
{\tt 22.\ \ \ }\newline
{\tt 23.\ \ scherk3(u,v)\ ==\ }\newline
{\tt 24.\ \ \ \ x\ :=\ exp(v)\ *\ cos(u)}\newline
{\tt 25.\ \ \ \
point\ [xOffset\ +\ u,\ yOffset\ +\ acos(x),\ v,\ abs(v)]}\newline
{\tt 26.\ \ \ }\newline
{\tt 27.\ \ scherk4(u,v)\ ==\ }\newline
{\tt 28.\ \ \ \ x\ :=\ exp(v)\ *\ cos(u)}\newline
{\tt 29.\ \ \ \
point\ [xOffset\ +\ u,\ yOffset\ -\ acos(x),\ v,\ abs(v)]}\newline
{\tt 30.\ \ \ }\newline
{\tt 31.\ \ drawOneScherk(s)\ ==}\newline
{\tt 32.\ \ \ \
makeObject(scherk1,-\%pi/2..\%pi/2,0..\%pi/2,space==s,}\newline
{\tt 33.\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
var1Steps\ ==\ 28,\ var2Steps\ ==\ 28)}\newline
{\tt 34.\ \ \ \
```

```
makeObject(scherk2,-\%pi/2..\%pi/2,0..\%pi/2,space==s,}\newline
{\tt 35.\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
var1Steps\ ==\ 28,\ var2Steps\ ==\ 28)}\newline
{\tt 36.\ \ \ \
makeObject(scherk3,-\%pi/2..\%pi/2,-\%pi/2..0,space==s,}\newline
{\tt 37.\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
var1Steps\ ==\ 28,\ var2Steps\ ==\ 28)}\newline
{\tt 38.\ \ \ \
makeObject(scherk4,-\%pi/2..\%pi/2,-\%pi/2..0,space==s,}\newline
{\tt 39.\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
var1Steps\ ==\ 28,\ var2Steps\ ==\ 28)}\newline
{\tt 40.\ \ \ \ void()}\newline

\noindent
\endscroll
\autobuttons
\end{page}
```

Chapter 21

Hypertex Language Pages

Creating Hyperdoc Pages

⇒ "notitle" (ViewportPage) 3.50 on page 662
⇒ "notitle" (BitMaps) 3.9 on page 168
⇒ "notitle" (CPHelp) 3.18 on page 279

— **hyperdoc.ht** —

```
\begin{page}{Hyperdoc}{Creating Hyperdoc Pages}

\beginscroll
This document tells how to create \HyperName pages.
To start with, it is rather meager but it will grow with time.
\beginmenu
\menulink{Viewports}{ViewportPage} Including live graphics in documents.
\menulink{Gadjets}{BitMaps} Bitmaps for use in macros.
\menulink{Control Panel Bits}{CPHelp} Development page for help
facility for viewports. yuck.
%\menulink{Test Pages}{TestPage} Some test pages left by J.M.
%\menulink{Paste Pages}{PastePage} Examples of how to use paste in areas.
\endmenu

\endscroll
\autobuttons
\end{page}
```

21.1 htxadvpage1.ht

Input Areas

⇒ “notitle” (HTXAdvPage2) 21.2 on page 2615

— **htxadvpage1.ht** —

```
\begin{page}{HTXAdvPage1}{Input areas}
\centerline{\fbox{{\tt \thispage}}}\newline
\begin{scroll}

You have probably seen input areas in other Hyperdoc
pages. They provide {\it dynamic link} capabilities.
Instead of having a choice between certain actions,
they allow you to specify an action on--the--fly.
To use them, you need the following commands:
\beginImportant
\newline
{\tt \\inputstring\{{\it label}\}\{{\it length}\}\{{\it default value}\}}
\newline
{\tt \\stringvalue\{{\it label}\}}
\endImportant

The first command puts up an input area of the {\it length}
specified. The {\it default value} is placed in it.
The first argument, {\it label} gives a name to the
contents of the input area.
You can refer to those contents by using
the second command. Never place a {\tt \\stringvalue} command
in an "exposed" part of the page. It is only meant
to be used as an argument to an {\it action}.
Here are some examples.




\beginImportant
\begin{paste}{HTXAdvPage1xPaste1}{HTXAdvPage1xPatch1}
\pastebutton{HTXAdvPage1xPaste1}{Interpret}
\newline
{\tt Page name \\tab\{16\} }
{\tt \\inputstring\{pagetogo\}\{30\}\{RootPage\}}\newline
{\tt \\newline}\newline
{\tt \\downlink\{GO!\}\{\\stringvalue\{pagetogo\}\}}\newline
\end{paste}
\endImportant
```

```
\beginImportant
\begin{paste}{HTXAdvPage1xPaste2}{HTXAdvPage1xPatch2}
\pastebutton{HTXAdvPage1xPaste2}{Interpret}
\newline
{\tt File to edit \\tab\{16\}}\newline
{\tt \\inputstring\{filetoedit\}\{30\}\{/etc/passwd\}}\newline
{\tt \\newline}\newline
{\tt \\unixcommand\{Ready!\}\{xterm  -e vi \\stringvalue\{filetoedit\}\}}
\end{paste}
\endImportant


\end{scroll}
\beginmenu
\menulink{Next Page --- Radio boxes}{HTXAdvPage2}
\endmenu

\end{page}
```

———-

HTXAdvPage1xPatch1 patch

— htxadvpage1.ht —

```
\begin{patch}{HTXAdvPage1xPatch1}
\begin{paste}{HTXAdvPage1xPaste1A}{HTXAdvPage1xPatch1A}
\pastebutton{HTXAdvPage1xPaste1A}{Source}
\newline
Page name \tab{16}
\inputstring{pagetogo}{30}{RootPage}
\newline
\downlink{GO!}{\stringvalue{pagetogo}}
\end{paste}
\end{patch}
```

———-

HTXAdvPage1xPatch1A patch

— htxadvpage1.ht —

```
\begin{patch}{HTXAdvPage1xPatch1A}
```

```
\begin{paste}{HTXAdvPage1xPaste1B}{HTXAdvPage1xPatch1}
\pastebutton{HTXAdvPage1xPaste1B}{Interpret}
\newline
{\tt Page name \\tab\{16\} }
{\tt \\inputstring\{pagetogo\}\{30\}\{RootPage\}}\newline
{\tt \\newline}\newline
{\tt \\downlink\{GO!\}\{\\stringvalue\{pagetogo\}\}}\newline
\end{paste}
\end{patch}
```

———-

HTXAdvPage1xPatch2 patch

— htxadvpage1.ht —

```
\begin{patch}{HTXAdvPage1xPatch2}
\begin{paste}{HTXAdvPage1xPaste2A}{HTXAdvPage1xPatch2A}
\pastebutton{HTXAdvPage1xPaste2A}{Source}
\newline
File to edit \tab{16}
\inputstring{filetoedit}{30}{/etc/passwd}
\newline
\unixcommand{Ready!}{xterm  -e vi \stringvalue{filetoedit}}
\end{paste}
\end{patch}
```

———-

HTXAdvPage1xPatch2A patch

— htxadvpage1.ht —

```
\begin{patch}{HTXAdvPage1xPatch2A}
\begin{paste}{HTXAdvPage1xPaste2B}{HTXAdvPage1xPatch2}
\pastebutton{HTXAdvPage1xPaste2B}{Interpret}
\newline
{\tt File to edit \\tab\{16\}}\newline
{\tt \\inputstring\{filetoedit\}\{30\}\{/etc/passwd\}}\newline
{\tt \\newline}\newline
{\tt \\unixcommand\{Ready!\}\{xterm  -e vi \\stringvalue\{filetoedit\}\}}
\end{paste}
\end{patch}
```

———-

21.2 htxadvpage2.ht

Radio buttons

⇒ "notitle" (HTXAdvPage3) 21.3 on page 2618
— **htxadvpage2.ht** —

```
\begin{page}{HTXAdvPage2}{Radio buttons}
\centerline{\fbox{{\tt \thispage}}}\newline
\begin{scroll}


If you just want to make a multiple-choice
type selection, why not use the {\it radio buttons}.

You need to use bitmaps for the active areas (the buttons) but
Hyperdoc will keep track of the currently activated button.
You can use this boolean information somewhere else on your page.
The commands to use are:
\beginImportant
\newline
{\tt \\radioboxes\{{\it group name}\}\{{\it bitmap file1}\}
\{{\it bitmap file0}\}}
\newline
{\tt \\radiobox[{\it initial state}]\{{\it label}\}
\{{\it group name}\}}
\newline
{\tt \\boxvalue\{{\it label}\}}
\endImportant

The {\tt \\radioboxes} command sets up a group of {\tt \\radiobox}
buttons. The {\it group name} is a label for the group. The filenames
for the bitmaps are specified in {\it bitmap file1} and
{\it bitmap file0}.
The first one should denote an activated button and the
second a de-activated one.

To display each button in a group, use {\tt \\radiobox}.  The
{\it initial state} should be either {\tt 1} or {\tt 0} depending on
whether the button should first be displayed as activated or not.  The
second {\it label} argument defines the name by which the current
state of the button can be referred to.  The third argument specifies
which group this button belongs to.

The {\tt \\boxvalue} command can then be used in various
```

```
actions. The value of it will
be either {\tt t} or {\tt nil}.

In the example below, we use the {\tt \\htbmfile} macro
defined in {\bf util.ht} so that we do not have to write
the full bitmap file pathnames.

This is how we set up the group. The {\tt \\radioboxes} command
does not display anything.
Note that these commands cannot be included in a {\it patch}.
This is why we display this time the source and the result
at the same time.
\beginImportant
\newline
{\tt \\radioboxes\{group\}
\{\\htbmfile\{pick\}\}\{\\htbmfile\{unpick\}\}}\newline
{\tt \\newline \\table\{}\newline
{\tt \{\\radiobox[1]\{b1\}\{group\}\}}\newline
{\tt \{\\radiobox[0]\{b2\}\{group\}\}}\newline
{\tt \{\\radiobox[0]\{b3\}\{group\}\}\}}\newline
{\tt \\newline}\newline
{\tt \\lispcommand\{lisp\}\{(pprint (list}\newline
{\tt \\boxvalue\{b1\} \\boxvalue\{b2\} \\boxvalue\{b3\}))\}}\newline
{\tt \\newline}\newline
{\tt \\unixcommand\{unix\}\{echo '\\boxvalue\{b1\}}\newline
{\tt \\boxvalue\{b2\} \\boxvalue\{b3\}'\}}
\endImportant
\radioboxes{group}{\htbmfile{pick}}{\htbmfile{unpick}}
\table{
{\radiobox[1]{b1}{group}}
{\radiobox[0]{b2}{group}}
{\radiobox[0]{b3}{group}}}
\newline
\lispcommand{lisp}{(pprint (list
\boxvalue{b1} \boxvalue{b2} \boxvalue{b3}))}
\newline
\unixcommand{unix}{echo '\boxvalue{b1}
\boxvalue{b2} \boxvalue{b3}'}
\endImportant




You can only set one radio button at a time. If you want
a non--exclusive selection, try {\tt \\inputbox}.
The syntax for this command is
\beginImportant
\newline
{\tt \\inputbox[{\it initial state}]
\{{\it label}\}\{{\it bitmap file1}\}\{{\it bitmap file0}\}}
```

```
\endImportant

There is no group command for these.
\beginImportant
\newline
{\tt \\table\{}\newline
{\tt \{\\inputbox[1]\{c1\}
\{\\htbmfile\{pick\}\}\{\\htbmfile\{unpick\}\}\}}\newline
{\tt \{\\inputbox\{c2\}
\{\\htbmfile\{pick\}\}\{\\htbmfile\{unpick\}\}\}}\newline
{\tt \{\\inputbox[1]\{c3\}
\{\\htbmfile\{pick\}\}\{\\htbmfile\{unpick\}\}\}\}}\newline
{\tt \\newline}\newline
{\tt \\lispcommand\{lisp\}\{(pprint (list}\newline
{\tt \\boxvalue\{c1\} \\boxvalue\{c2\} \\boxvalue\{c3\}))\}}\newline
{\tt \\newline}\newline
{\tt \\unixcommand\{unix\}\{echo }\newline
{\tt '\\boxvalue\{c1\} \\boxvalue\{c2\} \\boxvalue\{c3\}'\}}\newline
\endImportant
\table{
{\inputbox[1]{c1}{\htbmfile{pick}}{\htbmfile{unpick}}}
{\inputbox{c2}{\htbmfile{pick}}{\htbmfile{unpick}}}
{\inputbox[1]{c3}{\htbmfile{pick}}{\htbmfile{unpick}}}}
\newline
\lispcommand{lisp}{(pprint (list
\boxvalue{c1} \boxvalue{c2} \boxvalue{c3}))}
\newline
\unixcommand{unix}{echo
'\boxvalue{c1} \boxvalue{c2} \boxvalue{c3}'}
\endImportant


Note that the {\it initial state} is an
optional argument. If omitted
the button will initially
be deactivated.

\end{scroll}
\beginmenu
\menulink{Next Page --- Macros}{HTXAdvPage3}
\endmenu

\end{page}
```

21.3 htxadvpage3.ht

Macros

⇒ “notitle” (HTXAdvPage4) 21.4 on page 2619

— htxadvpage3.ht —

```
\begin{page}{HTXAdvPage3}{Macros}
\centerline{\fbox{{\tt \thispage}}}\newline
\begin{scroll}


Sometimes you may find yourself having to
write
almost the same piece of Hyperdoc
text many times. Thankfully, there is a command to ease
the work.
It is the {\tt \\newcommand} command and provides
a macro facility for Hyperdoc.
In this way, you can give a short name to a sequence of Hyperdoc
text and use that name to include the sequence in your pages.
The way this works is the following
\beginImportant
\newline
\centerline{{\tt \\newcommand\{\\{\it name}\}
[{\it number of arguments}]\{{\it Hyperdoc text}\}}}
\endImportant
and here is an example from {\bf util.ht}
\beginImportant
\newline
{\tt \\newcommand\{\\axiomSig\}[2]
\{\\axiomType\{\#1\} \{\\tt ->\} \\axiomType\{\#2\}\}}
\newline
{\tt \\newcommand\{\\axiomType\}[1]
\{\\lispdownlink\{\#1\}\{(|spadType| '|\#1|)\}\}}
\endImportant

You see that a macro's definition can invoke another.
Don't create a circular definition though!
Notice how the arguments of the macro are used
in the definition. The {\tt \#{\it n}} construct
is the place--holder of the {\it n}'th argument.

To use the macro, just treat it as an ordinary command.
For instance
\beginImportant
\newline
{\tt \\axiomSig\{Integer\}\{List Integer\}}
```

```
\endImportant
displays and acts like this
\beginImportant
\newline
\axiomSig{Integer}{List Integer}
\endImportant

The best way to familiarise yourself to
macros is to study the macros defined in
\centerline{{\bf util.ht}}
It is highly probable that a good many of them
will prove useful to you.
Clever use of macros will allow you to
create Hyperdoc text that can be
formatted by other programs (such as TeX).
The Axiom User Guide was written
in such a way as to make translation in
Hyperdoc form and TeX form a mechanical process.

\end{scroll}
\beginmenu
\menulink{Next Page --- Patch and Paste}{HTXAdvPage4}
\endmenu

\end{page}
```

21.4 htxadvpage4.ht

Patch and Paste

⇒ “notitle” (HTXAdvPage5) 21.5 on page 2623

— **htxadvpage4.ht** —

```
\begin{page}{HTXAdvPage4}{Patch and Paste}
\centerline{\fbox{{\tt \thispage}}}\newline
\begin{scroll}


A powerful Hyperdoc feature is
the ability to {\it replace}
part of a displayed page with another part
when an active area is clicked. The group commands
{\it patch} and {\it paste} offer this facility.
A {\it paste} region can appear anywhere
```

```
within a page or a {\it patch}. A {\it patch} region must be defined
outside a page definition.

We need a few  objects to define the {\it paste}
region. These are a {\it name} with which to
refer to it, some way of specifying what it
is to be replaced by and a {\it trigger} for the
replacement. A {\it patch} is how we specify the
second of these objects.
The {\it patch} is generally a sequence of Hyperdoc
text.

If we want to have the option of returning to the original
(or ,indeed, proceeding to a {\it third} alternative)
we clearly must include a {\it paste} in the {\it patch}.

Let us start with a simple example. We wish to
have the word {\tt initial} somewhere on the page replaced by the
word {\tt final} at a click of a button.
Let us first define the {\it patch}. It will just contain
the word {\tt final}. Here is a definition of a
patch called {\tt patch1} (note that
the actual definition must be outside this page's definition).
\beginImportant
\newline
{\tt \\begin\{patch\}\{patch1\}} \newline
{\tt final}\newline
{\tt \\end\{patch\}}
\endImportant
We now define a {\it paste} region exactly where we
want the word {\tt initial} to appear.
\beginImportant
\newline
{\tt \\begin\{paste\}\{paste1\}\{patch1\}}\newline
{\tt initial}\newline
{\tt \\end\{paste\}}
\centerline{{\it results in}}
\begin{paste}{paste1}{patch1}
initial
\end{paste}
\endImportant
We have specified first the name of the {\it paste} region
which is {\tt paste1} and then the name of the
replacement {\it patch} which is {\tt patch1}.
Something is missing -- the trigger.
To include a trigger we write
\beginImportant
\newline
{\tt \\pastebutton\{paste1\}\{trigger\}
\centerline{{\it results in}}
```

```
\pastebutton{paste1}{trigger}
\endImportant
This new command {\tt \\pastebutton} displays the second argument
as an active area. The first argument specifies the {\it paste}
region it refers to. Clicking on {\tt trigger} above will
replace the word {\tt initial} with the word {\tt final}.

We can, if we like, include the {\tt \\pastebutton} in the {\it paste}
region.
Let us improve on the situation by providing a way
of going back to the original word {\tt initial} on the page.
The {\it patch} must itself include a {\it paste}.
What will the replacement {\it patch} for this new {\it paste}
be ? Well, we  have to define a second {\it patch}
that contains all the stuff in the original {\it paste}
region. Here is the updated {\it patch} for the first replacement.
The {\tt \\MenuDotBitmap} macro is defined in {\bf util.ht}.
It displays a button bitmap.
This time we put the {\tt \\pastebutton}
inside the {\it paste}.
\beginImportant
\newline
{\tt \\begin\{patch\}\{Patch1\}}\newline
{\tt \\begin\{paste\}\{Paste2\}\{Patch2\}}\newline
{\tt \\pastebutton\{Paste2\}\{\\MenuDotBitmap\}}\newline
{\tt final}\newline
{\tt \\end\{paste\}}\newline
{\tt \\end\{patch\}}\newline
\endImportant
and the new {\tt Patch2} {\it patch}
\beginImportant
\newline
{\tt \\begin\{patch\}\{Patch2\}}\newline
{\tt \\begin\{paste\}\{Paste3\}\{Patch1\}}\newline
{\tt \\pastebutton\{Paste3\}\{\\MenuDotBitmap\}}\newline
{\tt initial}\newline
{\tt \\end\{paste\}}\newline
{\tt \\end\{patch\}}\newline
\endImportant

Remember that these {\it patch} definitons must
occur outside a {\tt \\begin\{page\} - \\end\{page\}} group.
What is left now is to define the starting {\it paste}
region.
\beginImportant
\newline
{\tt \\begin\{paste\}\{Paste1\}\{Patch1\}}\newline
{\tt \\pastebutton\{Paste1\}\{\\MenuDotBitmap\}}\newline
{\tt initial}\newline
{\tt \\end\{paste\}}
```

```
\centerline{{\it results in}}
\begin{paste}{Paste1}{Patch1}
\pastebutton{Paste1}{\MenuDotBitmap}
initial
\end{paste}
\endImportant

Clicking on the button above next to {\tt initial}
will replace the {\tt Paste1} region with
{\tt Patch1}. That {\it patch}
also contains a {\it paste} region ({\tt Paste2}).
Clicking on {\it its} button will put up
{\tt Patch2} which has a {\it paste} region ({\tt Paste3}).
Clicking on {\it its} button will put up {\tt Patch1}
again. In that way, we close the chain of replacements.




\end{scroll}
\beginmenu
\menulink{Next Page --- Axiom paste-ins}{HTXAdvPage5}
\endmenu

\end{page}
```

———-

patch1 patch

— **htxadvpage4.ht** —

```
\begin{patch}{patch1}
final
\end{patch}
```

———-

Patch1 patch

— **htxadvpage4.ht** —

```
\begin{patch}{Patch1}
```

```
\begin{paste}{Paste2}{Patch2}
\pastebutton{Paste2}{\MenuDotBitmap}
final
\end{paste}
\end{patch}
```

———-

Patch2 patch

— **htxadvpage4.ht** —

```
\begin{patch}{Patch2}
\begin{paste}{Paste3}{Patch1}
\pastebutton{Paste3}{\MenuDotBitmap}
initial
\end{paste}
\end{patch}
```

———-

21.5 htxadvpage5.ht

Axiom paste-ins

⇒ "notitle" (HTXAdvPage6) 21.6 on page 2626

— **htxadvpage5.ht** —

```
\begin{page}{HTXAdvPage5}{Axiom paste-ins}
\centerline{\fbox{{\tt \thispage}}}\newline
\begin{scroll}

The {\it paste} and {\it patch} facility
(see \downlink{previous page}{HTXAdvPage4})
is used to display (or hide) the Axiom output
of an Axiom command ({\tt\\axiomcommand})
included in a
Hyperdoc page.

A mechanism has been set up to {\it automatically} generate
these paste-ins. It amounts to replacing an
{\tt \\axiomcommand} by a {\tt \\pastecommand} in the
```

```
Hyperdoc page.

In the case of a \axiomOp{draw} Axiom command
, where the result is to create an interactive viewport,
the appropriate command to use is {\tt \\pastegraph}.
The effect of this is to include (as the output)
the Axiom generated {\it image} of the graph
as an active area. Clicking on it will put up an
interactive viewport.
The {\tt \\pastegraph} command should be used only when
the result of the associated Axiom operation is
to {\it create} an interactive viewport. It is {\it not} necessarily
appropriate for all commands whose result
is a \axiomType{Two{}Dimensional{}Viewport} or
\axiomType{Three{}Dimensional{}Viewport}.
The {\tt \\pastecommand} and {\tt \\pastegraph}
are  macros defined in {\bf util.ht}.



There is no automatic paste-in generation facility
for Axiom piles (the {\tt \\begin\{axiomsrc\}} command).


The automatic paste-in generation mechanism works by invoking Axiom
with a particular option.  Hyperdoc is also started automatically. It
reads the {\tt \\pastecommand} and {\tt \\pastegraph} commands in all
pages in a specified {\bf somefile.ht} and passes them to Axiom for
evaluation. Hyperdoc captures the output and writes it out (to a file
called {\bf somefile.pht}) as the body of some {\it patch}
definitions. The commands encountered are written to a file
{\bf somefile.input} which you can {\tt )read} from an Axiom session.
It also creates directories for the graphics images encountered. Those
files and directories will be written under the {\it current}
directory.  The idea is that you then include the {\it patch}
definitions in {\bf somefile.pht} in your local database using {\bf
htadd}.


You can try this feature now. Edit a file called, say, {\bf trypaste.ht}
in a directory, say {\bf /tmp}. Put the following Hyperdoc text
in it.
\beginImportant
\newline
{\tt \\begin\{page\}\{TryPaste\}\{Trying out paste-in generation\}}\newline
{\tt \\begin\{scroll\}}\newline
{\tt \\pastecommand\{f z == z**2 \\bound\{f\}\}}\newline
{\tt \\pastegraph\{draw(f,-1..1) \\free\{f\}\}}\newline
{\tt \\pastecommand\{x:= f 3 \\free\{f\}\}}\newline
{\tt \\end\{scroll\}}\newline
```

```
{\tt \\end\{page\}}\newline
\endImportant

From the directory that contains the {\bf trypaste.ht},
issue
\centerline{
{\tt htadd -l ./trypaste.ht}
}
You will get the
{\bf ht.db} database file.
Set the environment variable {\tt HTPATH} so that
it points first to your directory and then the system directory.
In the {\bf /bin/csh}, you might use
\centerline{
{\tt setenv HTPATH /tmp:\$AXIOM/doc}
}
Make sure that no {\bf trypaste.input} or {\bf trypaste.pht}
files exist in the directory.
Then issue
\centerline{
{\tt axiom -paste trypaste.ht}
}
 and wait for
Axiom to finish.

There is a modification you will wish to make to
the {\bf trypaste.pht} file.
This is because the generated Hyperdoc text will assume that
the {\it viewport} data will be located in the
{\it system} directory.
\centerline{{\bf \env{AXIOM}/doc/viewports}}
You may want to place your images in a different directory,
say, {\bf /u/sugar/viewports}.
If so, then change all occurences of
\beginImportant
\newline
{\tt \\env\{AXIOM\}/doc/viewports/}
\centerline{{\it by}}
{\tt /u/sugar/viewports/}
\endImportant
in the file {\bf trypaste.pht}. The last step
is to include the {\it patch} definitions
in {\bf trypaste.pht} to your local database.
Issue
\centerline{
{\tt htadd -l ./trypaste.pht}
}
to update the database. If you have provided
a link to your pages from the {\tt RootPage}
via the {\tt \\localinfo} macro, you should now
```

```
be able to start Axiom or Hyperdoc
and see the computed Axiom output whenever you
click on the buttons to the left of each command.



\end{scroll}
\beginmenu
\menulink{Next Page --- Miscellaneous}{HTXAdvPage6}
\endmenu

\end{page}
```

———-

21.6 htxadvpage6.ht

Miscellaneous

⇒ “notitle” (HTXAdvTopPage) 21.7 on page 2630

— **htxadvpage6.ht** —

```
\begin{page}{HTXAdvPage6}{Miscellaneous}
\centerline{\fbox{{\tt \thispage}}}\newline
\begin{scroll}


We present here a few commands that may be of some use
to you.
You may want to know certain parameters so that you can pass
them to one of the action {\tt \\command}s.

The {\tt \\thispage} command shows the name of the
current page.

\beginImportant
\begin{paste}{HTXAdvPage6xPaste1}{HTXAdvPage6xPatch1}
\pastebutton{HTXAdvPage6xPaste1}{Interpret}
\newline
{\tt \\thispage}\newline
{\tt \\newline}\newline
{\tt \\lispcommand\{Lisp\}\{(pprint "\\thispage")\}}\newline
\end{paste}
\endImportant
```

```
The {\tt \\windowid} command shows the X Windows
{\it WindowID} of the current window.

\beginImportant
\begin{paste}{HTXAdvPage6xPaste2}{HTXAdvPage6xPatch2}
\pastebutton{HTXAdvPage6xPaste2}{Interpret}
\newline
{\tt \\windowid}\newline
{\tt \\newline}\newline
{\tt \\lispcommand\{Lisp\}\{(pprint \\windowid)\}}\newline
\end{paste}
\endImportant

% \examplenumber not documented

The {\tt \\env} command gets the value of an environment
variable. It is an error to use {\tt \\env} with an undefined
environment variable.

\beginImportant
\begin{paste}{HTXAdvPage6xPaste3}{HTXAdvPage6xPatch3}
\pastebutton{HTXAdvPage6xPaste3}{Interpret}
\newline
{\tt \\env\{AXIOM\}}
\end{paste}
\endImportant


The {\tt \\nolines} command, if included somewhere
in the page, eliminates the horizontal lines that
delimit the header and footer regions.


% \beep not documented

%\returnbutton{homebutton}{ReturnPage}

%\upbutton{upbutton}{UpPage}


\end{scroll}
\beginmenu
\menulink{Back to Menu}{HTXAdvTopPage}
\endmenu

\end{page}
```

———-

HTXAdvPage6xPatch1 patch

— **htxadvpage6.ht** —

```
\begin{patch}{HTXAdvPage6xPatch1}
\begin{paste}{HTXAdvPage6xPaste1A}{HTXAdvPage6xPatch1A}
\pastebutton{HTXAdvPage6xPaste1A}{Source}
\newline
\thispage
\newline
\lispcommand{Lisp}{(pprint "\thispage")}
\end{paste}
\end{patch}
```

———

HTXAdvPage6xPatch1A patch

— **htxadvpage6.ht** —

```
\begin{patch}{HTXAdvPage6xPatch1A}
\begin{paste}{HTXAdvPage6xPaste1B}{HTXAdvPage6xPatch1}
\pastebutton{HTXAdvPage6xPaste1B}{Interpret}
\newline
{\tt \\thispage}\newline
{\tt \\newline}\newline
{\tt \\lispcommand\{Lisp\}\{(pprint "\\thispage")\}}\newline
\end{paste}
\end{patch}
```

———

HTXAdvPage6xPatch2 patch

— **htxadvpage6.ht** —

```
\begin{patch}{HTXAdvPage6xPatch2}
\begin{paste}{HTXAdvPage6xPaste2A}{HTXAdvPage6xPatch2A}
\pastebutton{HTXAdvPage6xPaste2A}{Source}
\newline
\windowid
\newline
```

```
\lispcommand{Lisp}{(pprint \windowid)}
\end{paste}
\end{patch}
```

———-

HTXAdvPage6xPatch2A patch

— htxadvpage6.ht —

```
\begin{patch}{HTXAdvPage6xPatch2A}
\begin{paste}{HTXAdvPage6xPaste2B}{HTXAdvPage6xPatch2}
\pastebutton{HTXAdvPage6xPaste2B}{Interpret}
\newline
{\tt \\windowid}\newline
{\tt \\newline}\newline
{\tt \\lispcommand\{Lisp\}\{(pprint \\windowid)\}}\newline
\end{paste}
\end{patch}
```

———-

HTXAdvPage6xPatch3 patch

— htxadvpage6.ht —

```
\begin{patch}{HTXAdvPage6xPatch3}
\begin{paste}{HTXAdvPage6xPaste3A}{HTXAdvPage6xPatch3A}
\pastebutton{HTXAdvPage6xPaste3A}{Source}
\newline
\env{AXIOM}
\end{paste}
\end{patch}
```

———-

HTXAdvPage6xPatch3A patch

— htxadvpage6.ht —

```
\begin{patch}{HTXAdvPage6xPatch3A}
\begin{paste}{HTXAdvPage6xPaste3B}{HTXAdvPage6xPatch3}
\pastebutton{HTXAdvPage6xPaste3B}{Interpret}
\newline
{\tt \\env\{AXIOM\}}\newline
\end{paste}
\end{patch}
```

———-

21.7 htxadvtoppage.ht

Advanced features in Hyperdoc

⇒ "notitle" (HTXAdvPage1) 21.1 on page 2612
⇒ "notitle" (HTXAdvPage2) 21.2 on page 2615
⇒ "notitle" (HTXAdvPage3) 21.3 on page 2618
⇒ "notitle" (HTXAdvPage4) 21.4 on page 2619
⇒ "notitle" (HTXAdvPage5) 21.5 on page 2623
⇒ "notitle" (HTXAdvPage6) 21.6 on page 2626

— **htxadvtoppage.ht** —

```
\begin{page}{HTXAdvTopPage}{Advanced features in Hyperdoc}
\centerline{\fbox{{\tt \thispage}}}\newline
\beginscroll
\beginmenu
\menudownlink{Creating input areas}{HTXAdvPage1}
\menudownlink{Creating radio boxes}{HTXAdvPage2}
\menudownlink{Define new macros     }{HTXAdvPage3}
\menudownlink{Using patch and paste}{HTXAdvPage4}
\menudownlink{Generate paste-ins for Axiom commands}{HTXAdvPage5}
\menudownlink{Miscellaneous}{HTXAdvPage6}
\endmenu
\endscroll
\end{page}
```

———-

21.8 htxformatpage1.ht

Using the special characters

⇒ "notitle" (HTXFormatPage2) 21.9 on page 2633

— htxformatpage1.ht —

```
\begin{page}{HTXFormatPage1}{Using the special characters}
\centerline{\fbox{{\tt \thispage}}}\newline
\begin{scroll}

You can display the special characters ({\tt \$ \\ \{ \{ \[ \] \% \#})
by simply inserting the backslash {\tt \\} character just before any
one of them. So,
\beginImportant
\begin{paste}{HTXFormatPage1xPaste1}{HTXFormatPage1xPatch1}
\pastebutton{HTXFormatPage1xPaste1}{Interpret}
\newline
{\tt the characters \\\$, \\\\, \\\{, \\\}, \\\[, \\\], \\\%, \\\# }
\end{paste}
\endImportant


The {\tt \%} character is used in Hyperdoc as a comment marker.
If it is encountered on any line (without being preceded by the {\tt \\}
 character,
of course), Hyperdoc will ignore all remaining characters on that line.
\beginImportant
\begin{paste}{HTXFormatPage1xPaste2}{HTXFormatPage1xPatch2}
\pastebutton{HTXFormatPage1xPaste2}{Interpret}
\newline
{\tt  the latest figures indicate \% GET THE LATEST FIGURES}\newline
{\tt a steady rise}\indent{0}
\end{paste}
\endImportant

Earlier versions of Hyperdoc merged the words "indicate" and "a"
into one word in the example above. This no longer occurs.

\end{scroll}
\beginmenu
\menulink{Next -- Formatting without commands}{HTXFormatPage2}
\endmenu

\end{page}
```

———

HTXFormatPage1xPatch1 patch

— htxformatpage1.ht —

```
\begin{patch}{HTXFormatPage1xPatch1}
\begin{paste}{HTXFormatPage1xPaste1A}{HTXFormatPage1xPatch1A}
\pastebutton{HTXFormatPage1xPaste1A}{Source}
\newline
the characters \$, \\, \{, \}, \[, \],  \%, \#
\end{paste}
\end{patch}
\begin{patch}{HTXFormatPage1xPatch1A}
\begin{paste}{HTXFormatPage1xPaste1B}{HTXFormatPage1xPatch1}
\pastebutton{HTXFormatPage1xPaste1B}{Interpret}
\newline
{\tt the characters \\\$, \\\\, \\\{, \\\}, \\\[, \\\], \\\%, \\\# }
\end{paste}
\end{patch}
```

———

HTXFormatPage1xPatch2 patch

— htxformatpage1.ht —

```
\begin{patch}{HTXFormatPage1xPatch2}
\begin{paste}{HTXFormatPage1xPaste2A}{HTXFormatPage1xPatch2A}
\pastebutton{HTXFormatPage1xPaste2A}{Source}
\newline
the latest figures indicate % GET THE LATEST FIGURES
a steady rise
\end{paste}
\end{patch}
\begin{patch}{HTXFormatPage1xPatch2A}
\begin{paste}{HTXFormatPage1xPaste2B}{HTXFormatPage1xPatch2}
\pastebutton{HTXFormatPage1xPaste2B}{Interpret}
\newline
{\tt  the latest figures indicate \% GET THE LATEST FIGURES}\newline
{\tt a steady rise}\indent{0}
\end{paste}
\end{patch}
```

———

21.9 htxformatpage2.ht

Formatting without commands

— htxformatpage2.ht —

```
\begin{page}{HTXFormatPage2}{Formatting without commands}
\centerline{\fbox{{\tt \thispage}}}\newline
\begin{scroll}

Hyperdoc will interpret normal text in a {\em source file}
according to the following rules. \newline
\menuitemstyle{\indentrel{4}
Spaces mark the ends of words. The number of spaces between
words is {\em not} significant, that is, you cannot control word spacing
by inserting or removing extra space characters.
\indentrel{-4}}
\beginImportant
\begin{paste}{HTXFormatPage2xxPaste1}{HTXFormatPage2xPatch1}
\pastebutton{HTXFormatPage2xxPaste1}{Interpret}
\begin{verbatim}
word    spacing is       not  important
\end{verbatim}
\end{paste}
\endImportant
\menuitemstyle{\indentrel{4}
End-of-line  characters are not significant.
You can break up lines in the source file as you like. The end-of-line
will be interpreted as a space.
Take advantage of this feature to improve the readability of the
source file.
\indentrel{-4}}
\beginImportant
\begin{paste}{HTXFormatPage2xxPaste2}{HTXFormatPage2xPatch2}
\pastebutton{HTXFormatPage2xxPaste2}{Interpret}
\begin{verbatim}
This is
one
sentence.
\end{verbatim}
\end{paste}
\endImportant
\menuitemstyle{\indentrel{4}
A blank line marks the end of a paragraph.
Leaving more blank lines that necessary has no effect.
\indentrel{-4}}
\beginImportant
\begin{paste}{HTXFormatPage2xxPaste3}{HTXFormatPage2xPatch3}
```

```
\pastebutton{HTXFormatPage2xxPaste3}{Interpret}
\begin{verbatim}
some end.% A COMMENT


Start a paragraph

Start another paragraph.
\end{verbatim}
\end{paste}
\endImportant
\menuitemstyle{\indentrel{4}
The two-character combination {\tt \{\}} can be used to indicate
possible breaking of long words. It does not affect the formatting
in any other way.
\indentrel{-4}}
\beginImportant
\begin{paste}{HTXFormatPage2xxPaste4}{HTXFormatPage2xPatch4}
\pastebutton{HTXFormatPage2xxPaste4}{Interpret}
\begin{verbatim}
Generalized{}Multivariate{}Factorize
One{}Dimensional{}Array{}Aggregate
Elementary{}Function{}Definite{}Integration
Elementary{}Functions{}Univariate{}Puiseux{}Series
Finite{}Field{}Cyclic{}Group{}Extension{}ByPolynomial
\end{verbatim}
\end{paste}
\endImportant




\end{scroll}
\beginmenu
\menulink{Next -- Using different fonts}{HTXFormatPage3}
\endmenu

\end{page}
```

———-

HTXFormatPage2xPatch1 patch

— **htxformatpage2.ht** —

```
\begin{patch}{HTXFormatPage2xPatch1}
\begin{paste}{HTXFormatPage2xxPaste1A}{HTXFormatPage2xPatch1A}
```

```
\pastebutton{HTXFormatPage2xxPaste1A}{Source}
\newline
word    spacing is       not  important
\end{paste}
\end{patch}
\begin{patch}{HTXFormatPage2xPatch1A}
\begin{paste}{HTXFormatPage2xxPaste1B}{HTXFormatPage2xPatch1}
\pastebutton{HTXFormatPage2xxPaste1B}{Interpret}
\begin{verbatim}
word    spacing is       not  important
\end{verbatim}
\end{paste}
\end{patch}
```

———-

HTXFormatPage2xPatch2 patch

— htxformatpage2.ht —

```
\begin{patch}{HTXFormatPage2xPatch2}
\begin{paste}{HTXFormatPage2xxPaste2A}{HTXFormatPage2xPatch2A}
\pastebutton{HTXFormatPage2xxPaste2A}{Source}
\newline
This is
one
sentence.
\end{paste}
\end{patch}
```

———-

HTXFormatPage2xPatch2A patch

— htxformatpage2.ht —

```
\begin{patch}{HTXFormatPage2xPatch2A}
\begin{paste}{HTXFormatPage2xxPaste2B}{HTXFormatPage2xPatch2}
\pastebutton{HTXFormatPage2xxPaste2B}{Interpret}
\begin{verbatim}
This is
one
sentence.
```

```
\end{verbatim}
\end{paste}
\end{patch}
```

———–

HTXFormatPage2xPatch3 patch

— htxformatpage2.ht —

```
\begin{patch}{HTXFormatPage2xPatch3}
\begin{paste}{HTXFormatPage2xxPaste3A}{HTXFormatPage2xPatch3A}
\pastebutton{HTXFormatPage2xxPaste3A}{Source}
\newline
some end.% A COMMENT


Start a paragraph.

Start another paragraph.
\end{paste}
\end{patch}
```

———–

HTXFormatPage2xPatch3A patch

— htxformatpage2.ht —

```
\begin{patch}{HTXFormatPage2xPatch3A}
\begin{paste}{HTXFormatPage2xxPaste3B}{HTXFormatPage2xPatch3}
\pastebutton{HTXFormatPage2xxPaste3B}{Interpret}
\begin{verbatim}
some end.% A COMMENT


Start a paragraph

Start another paragraph.
\end{verbatim}
\end{paste}
\end{patch}
```

———–

HTXFormatPage2xPatch4 patch

— **htxformatpage2.ht** —

```
\begin{patch}{HTXFormatPage2xPatch4}
\begin{paste}{HTXFormatPage2xxPaste4A}{HTXFormatPage2xPatch4A}
\pastebutton{HTXFormatPage2xxPaste4A}{Source}
\newline
Generalized{}Multivariate{}Factorize
One{}Dimensional{}Array{}Aggregate
Elementary{}Function{}Definite{}Integration
Elementary{}Functions{}Univariate{}Puiseux{}Series
Finite{}Field{}Cyclic{}Group{}Extension{}ByPolynomial
\end{paste}
\end{patch}
```

———-

HTXFormatPage2xPatch4A patch

— **htxformatpage2.ht** —

```
\begin{patch}{HTXFormatPage2xPatch4A}
\begin{paste}{HTXFormatPage2xxPaste4B}{HTXFormatPage2xPatch4}
\pastebutton{HTXFormatPage2xxPaste4B}{Interpret}
\begin{verbatim}
Generalized{}Multivariate{}Factorize
One{}Dimensional{}Array{}Aggregate
Elementary{}Function{}Definite{}Integration
Elementary{}Functions{}Univariate{}Puiseux{}Series
Finite{}Field{}Cyclic{}Group{}Extension{}ByPolynomial
\end{verbatim}
\end{paste}
\end{patch}
```

———-

21.10 htxformatpage3.ht

Using different fonts

— **htxformatpage3.ht** —

```
\begin{page}{HTXFormatPage3}{Using different fonts}
\centerline{\fbox{{\tt \thispage}}}\newline
\begin{scroll}

You can use various fonts for the text. Hyperdoc makes
four {\em logical} fonts available to you: a {\em roman} font, an
{\em emphasised} font, a {\em bold} font and a
{\em typewriter} font. The actual font that corresponds to
each logical font is determined by the end user via a
defaults file. The colour for each of these fonts can also
be specified.

Normal text is displayed with the roman font.
If you want to emphasize some text, use the {\tt \\em}
command in the following way.
\beginImportant
\begin{paste}{HTXFormatPage3xPaste1}{HTXFormatPage3xPatch1}
\pastebutton{HTXFormatPage3xPaste1}{Interpret}
\newline
{\tt this is \{\\em emphasised\} text}
\end{paste}
\endImportant

Note the use of the braces to enclose command and "arguments".
All font commands are specified in the same way. The {\tt \\em} command
will in fact {\em switch} between roman and emphasised
font every time it is used.
\beginImportant
\begin{paste}{HTXFormatPage3xPaste2}{HTXFormatPage3xPatch2}
\pastebutton{HTXFormatPage3xPaste2}{Interpret}
\newline
{\tt \{\\em this is \{\\em emphasised\} text\}}
\end{paste}
\endImportant

If you want to be sure that the emphasized font will be used, specify
the {\tt \\it} command. Similarly, you can explicitly select the roman
font with the {\tt \\rm} command.
\beginImportant
\begin{paste}{HTXFormatPage3xPaste3}{HTXFormatPage3xPatch3}
\pastebutton{HTXFormatPage3xPaste3}{Interpret}
\newline
{\tt \{\\em this is \{\\it emphasised\} text and this is \{\\rm roman\}\}}
\end{paste}
\endImportant


The bold font is selected with the {\tt \\bf} command and the typewriter
font with the {\tt \\tt} command. All these commands can be applied to
individual characters, words, sentences etc.
```

```
\beginImportant
\begin{paste}{HTXFormatPage3xPaste4}{HTXFormatPage3xPatch4}
\pastebutton{HTXFormatPage3xPaste4}{Interpret}
\newline
{\tt \{\\bf U\}\{\\tt g\}\{\\it l\}\{\\rm y\}}
\end{paste}
\endImportant


Currently, Hyperdoc does not adjust its internal spacing rules
to each font individually. This means that, for consistent results,
users are encouraged to specify (in the defaults file)
"character-cell" fonts that are not
too small or too large for Hyperdoc. Here is the correspondence
between the above font commands and the defaults names:\newline
\menuitemstyle{RmFont \tab{26} {\tt \\rm} or {\tt \\em} }\newline
\menuitemstyle{BoldFont \tab{26} {\tt \\bf} }\newline
\menuitemstyle{EmphasizeFont \tab{26} {\tt \\it} or {\tt \\em} }\newline
\menuitemstyle{Ttfont \tab{26} {\tt \\tt} }\newline

Hyperdoc uses two more logical fonts that can be specified by
the end user : AxiomFont and ActiveFont. However, you cannot
explicitly use these fonts in your text. The ActiveFont is automatically
used for active area text and the AxiomFont is reserved for
active Axiom commands.




\end{scroll}
\beginmenu
\menulink{Next -- Indentation}{HTXFormatPage4}
\endmenu

\end{page}
```

———-

HTXFormatPage3xPatch1 patch

— **htxformatpage3.ht** —

```
\begin{patch}{HTXFormatPage3xPatch1}
\begin{paste}{HTXFormatPage3xPaste1A}{HTXFormatPage3xPatch1A}
\pastebutton{HTXFormatPage3xPaste1A}{Source}
\newline
this is {\em emphasised} text
```

```
\end{paste}
\end{patch}
\begin{patch}{HTXFormatPage3xPatch1A}
\begin{paste}{HTXFormatPage3xPaste1B}{HTXFormatPage3xPatch1}
\pastebutton{HTXFormatPage3xPaste1B}{Interpret}
\newline
{\tt this is \{\\em emphasised\} text}
\end{paste}
\end{patch}
```

———-

HTXFormatPage3xPatch2 patch

— htxformatpage3.ht —

```
\begin{patch}{HTXFormatPage3xPatch2}
\begin{paste}{HTXFormatPage3xPaste2A}{HTXFormatPage3xPatch2A}
\pastebutton{HTXFormatPage3xPaste2A}{Source}
\newline
{\em this is {\em emphasised} text}
\end{paste}
\end{patch}
\begin{patch}{HTXFormatPage3xPatch2A}
\begin{paste}{HTXFormatPage3xPaste2B}{HTXFormatPage3xPatch2}
\pastebutton{HTXFormatPage3xPaste2B}{Interpret}
\newline
{\tt \{\\em this is \{\\em emphasised\} text\}}
\end{paste}
\end{patch}
```

———-

HTXFormatPage3xPatch3 patch

— htxformatpage3.ht —

```
\begin{patch}{HTXFormatPage3xPatch3}
\begin{paste}{HTXFormatPage3xPaste3A}{HTXFormatPage3xPatch3A}
\pastebutton{HTXFormatPage3xPaste3A}{Source}
\newline
{\em this is {\it emphasised} text and this is {\rm roman}}
\end{paste}
```

```
\end{patch}
\begin{patch}{HTXFormatPage3xPatch3A}
\begin{paste}{HTXFormatPage3xPaste3B}{HTXFormatPage3xPatch3}
\pastebutton{HTXFormatPage3xPaste3B}{Interpret}
\newline
{\tt \{\\em this is \{\\it emphasised\} text
and this is \{\\rm roman\}\}}
\end{paste}
\end{patch}
```

———-

HTXFormatPage3xPatch4 patch

— htxformatpage3.ht —

```
\begin{patch}{HTXFormatPage3xPatch4}
\begin{paste}{HTXFormatPage3xPaste4A}{HTXFormatPage3xPatch4A}
\pastebutton{HTXFormatPage3xPaste4A}{Source}
\newline
{\bf U}{\tt g}{\it l}{\rm y}
\end{paste}
\end{patch}
\begin{patch}{HTXFormatPage3xPatch4A}
\begin{paste}{HTXFormatPage3xPaste4B}{HTXFormatPage3xPatch4}
\pastebutton{HTXFormatPage3xPaste4B}{Interpret}
\newline
{\tt \{\\bf U\}\{\\tt g\}\{\\it l\}\{\\rm y\}}
\end{paste}
\end{patch}
```

———-

21.11 htxformatpage4.ht

Indentation

— htxformatpage4.ht —

```
\begin{page}{HTXFormatPage4}{Indentation}
\centerline{\fbox{{\tt \thispage}}}\newline
\begin{scroll}
```

```
You can control the indentation of lines of text in Hyperdoc with
some useful commands.
Use the command {\tt \\par} to force a new paragraph if you don't want
to use the blank-line rule. The first line of a new paragraph
will normally be indented by a standard small amount. If you
just want to start on a new line, use the {\tt \\newline}
command.
\beginImportant
\begin{paste}{HTXFormatPage4xPaste1}{HTXFormatPage4xPatch1}
\pastebutton{HTXFormatPage4xPaste1}{Interpret}
\newline
{\tt let us start a new line \\newline here }
\end{paste}
\endImportant

The  command {\tt \\indent\{{\it value}\}} will
set the left-page-margin {\it value} characters
to the right of the standard left-page-margin.
The initial (standard) state of a page can be reset by the
{\tt \\indent\{0\}} command.
The first lines of paragraphs will be indented
by the {\it extra} standard  amount. The {\tt \\indent\{{\it value}\}}
command does {\em not} force a new line or paragraph.

You can also use the {\tt \\indentrel\{{\it value}\}} command.
Here, the {\it value} argument is a {\em relative} indentation
which can be positive or negative.
Otherwise, it behaves in the same way as the {\tt \\indent} command.
\beginImportant
\begin{paste}{HTXFormatPage4xPaste2}{HTXFormatPage4xPatch2}
\pastebutton{HTXFormatPage4xPaste2}{Interpret}
\newline
{\tt let us start a new line \\newline \\indent\{10\} here }\newline
{\tt \\newline \\indentrel\{-5\} there}\newline
{\tt \\newline \\indentrel\{-5\} back}
\end{paste}
\endImportant

The {\tt \\centerline\{{\it some text}\}} command will center its
argument between the current left and right margins. The argument of
the command should not be more than a paragraph of text and should not
contain any commands that change the left margin. The centered text
will start on a new line.
\beginImportant
\begin{paste}{HTXFormatPage4xPaste3}{HTXFormatPage4xPatch3}
\pastebutton{HTXFormatPage4xPaste3}{Interpret}
\newline
{\tt previous text. \\centerline\{This could}\newline
{\tt be some heading.\} Carry on}
```

```
\end{paste}
\endImportant

Placing text in vertically aligned columns is easily done with the
{\tt \\tab\{{\it value}\}} command. The {\tt \\tab} command has the
immediate effect of placing the next word {\it value} characters to
the right of the current left margin.
\beginImportant
\begin{paste}{HTXFormatPage4xPaste4}{HTXFormatPage4xPatch4}
\pastebutton{HTXFormatPage4xPaste4}{Interpret}
\newline
{\tt \\indent\{5\}\\newline}\newline
{\tt Team A \\tab\{17\}Score\\tab\{25\}Team B\\tab\{42\}Score\\newline}
\newline
{\tt 012345678901234567890123456789012345678901234567890\\newline}
\newline
{\tt Green-Red\\tab\{17\}4\\tab\{25\}Blue-Black\\tab\{42\}6\\newline}
\newline
{\tt \\indent\{0\}}
\end{paste}
\endImportant

If you wish to preserve the indentation of a piece of text
you can use the {\tt verbatim} group command. Simply place
a {\tt \\begin\{verbatim\}} and {\tt \\end\{verbatim\}} around
the text. Note that Hyperdoc commands will
not be interpreted within the {\tt verbatim} group.
\beginImportant
\begin{paste}{HTXFormatPage4xPaste5}{HTXFormatPage4xPatch5}
\pastebutton{HTXFormatPage4xPaste5}{Interpret}
\newline
{\tt \\begin\{verbatim\}}
\begin{verbatim}
This    spacing     will be      preserved
                    {\bf is}     preserved
\end{verbatim}
{\tt \\end\{verbatim\}}\newline
\end{paste}







\end{scroll}
\beginmenu
\menulink{Next -- Creating Lists and Tables}{HTXFormatPage5}
\endmenu
```

```
\end{page}
```

———-

HTXFormatPage4xPatch1 patch

— htxformatpage4.ht —

```
\begin{patch}{HTXFormatPage4xPatch1}
\begin{paste}{HTXFormatPage4xPaste1A}{HTXFormatPage4xPatch1A}
\pastebutton{HTXFormatPage4xPaste1A}{Source}
\newline
let us start a new line \newline here
\end{paste}
\end{patch}
```

———-

HTXFormatPage4xPatch1A patch

— htxformatpage4.ht —

```
\begin{patch}{HTXFormatPage4xPatch1A}
\begin{paste}{HTXFormatPage4xPaste1B}{HTXFormatPage4xPatch1}
\pastebutton{HTXFormatPage4xPaste1B}{Interpret}
\newline
{\tt let us start a new line \\newline here }
\end{paste}
\end{patch}
```

———-

HTXFormatPage4xPatch2 patch

— htxformatpage4.ht —

```
\begin{patch}{HTXFormatPage4xPatch2}
\begin{paste}{HTXFormatPage4xPaste2A}{HTXFormatPage4xPatch2A}
\pastebutton{HTXFormatPage4xPaste2A}{Source}
\newline
```

```
let us start a new line\newline\indent{10} here
\newline\indentrel{-5} there
\newline\indentrel{-5} back
\end{paste}
\end{patch}
```

———-

HTXFormatPage4xPatch2A patch

— htxformatpage4.ht —

```
\begin{patch}{HTXFormatPage4xPatch2A}
\begin{paste}{HTXFormatPage4xPaste2B}{HTXFormatPage4xPatch2}
\pastebutton{HTXFormatPage4xPaste2B}{Interpret}
\newline
{\tt let us start a new line \\newline \\indent\{10\} here }\newline
{\tt \\newline \\indentrel\{-5\} there}\newline
{\tt \\newline \\indentrel\{-5\} back}
\end{paste}
\end{patch}
```

———-

HTXFormatPage4xPatch3 patch

— htxformatpage4.ht —

```
\begin{patch}{HTXFormatPage4xPatch3}
\begin{paste}{HTXFormatPage4xPaste3A}{HTXFormatPage4xPatch3A}
\pastebutton{HTXFormatPage4xPaste3A}{Source}
\newline
previous text. \centerline{This could
be some heading.} Carry on
\end{paste}
\end{patch}
```

———-

HTXFormatPage4xPatch3A patch

— htxformatpage4.ht —

```
\begin{patch}{HTXFormatPage4xPatch3A}
\begin{paste}{HTXFormatPage4xPaste3B}{HTXFormatPage4xPatch3}
\pastebutton{HTXFormatPage4xPaste3B}{Interpret}
\newline
{\tt previous text. \\centerline\{This could}\newline
{\tt be some heading.\} Carry on}
\end{paste}
\end{patch}
```

———-

HTXFormatPage4xPatch4 patch

— htxformatpage4.ht —

```
\begin{patch}{HTXFormatPage4xPatch4}
\begin{paste}{HTXFormatPage4xPaste4A}{HTXFormatPage4xPatch4A}
\pastebutton{HTXFormatPage4xPaste4A}{Source}
\newline
\indent{5}\newline
Team A \tab{17}Score\tab{25}Team B\tab{42}Score\newline
01234567890123456789012345678901234567890123456789\newline
Green-Red\tab{17}4\tab{25}Blue-Black\tab{42}6\newline
\indent{0}
\end{paste}
\end{patch}
\begin{patch}{HTXFormatPage4xPatch4A}
\begin{paste}{HTXFormatPage4xPaste4B}{HTXFormatPage4xPatch4}
\pastebutton{HTXFormatPage4xPaste4B}{Interpret}
\newline
{\tt \\indent\{5\}\\newline}\newline
{\tt Team A \\tab\{17\}Score\\tab\{25\}Team B\\tab\{42\}Score\\newline}
\newline
{\tt 01234567890123456789012345678901234567890123456789\\newline}
\newline
{\tt Green-Red\\tab\{17\}4\\tab\{25\}Blue-Black\\tab\{42\}6\\newline}
\newline
{\tt \\indent\{0\}}
\end{paste}
\end{patch}
```

———-

HTXFormatPage4xPatch5 patch

— htxformatpage4.ht —

```
\begin{patch}{HTXFormatPage4xPatch5}
\begin{paste}{HTXFormatPage4xPaste5A}{HTXFormatPage4xPatch5A}
\pastebutton{HTXFormatPage4xPaste5A}{Source}
\begin{verbatim}
This    spacing     will be      preserved
                    {\bf is}     preserved
\end{verbatim}
\end{paste}
\end{patch}
```

———-

HTXFormatPage4xPatch5A patch

— htxformatpage4.ht —

```
\begin{patch}{HTXFormatPage4xPatch5A}
\begin{paste}{HTXFormatPage4xPaste5B}{HTXFormatPage4xPatch5}
\pastebutton{HTXFormatPage4xPaste5B}{Interpret}
\newline
{\tt \\begin\{verbatim\}}
\begin{verbatim}
This    spacing     will be      preserved
                    {\bf is}     preserved
\end{verbatim}
{\tt \\end\{verbatim\}}\newline
\end{paste}
\end{patch}
```

———-

21.12 htxformatpage5.ht

Creating Lists and Tables

⇒ "notitle" (HTXFormatPage6) 21.13 on page 2653

— htxformatpage5.ht —

```
\begin{page}{HTXFormatPage5}{Creating Lists and Tables}
\centerline{\fbox{{\tt \thispage}}}\newline
\begin{scroll}

The {\tt \\begin\{items\} {\rm -}  \\end\{items\}}
group command constructs itemized lists.
The {\tt \\item} command separates the items in the list.
The indentation rules for the list group are different from those
of a paragraph. The first line of an item will
normally extend further to the left than the rest of the lines.
Both commands accept {\em optional} arguments.
Optional arguments are enclosed in square brackets ({\tt \[ \]}) rather
than braces.

The indentation of subsequent lines in an item is determined by the
optional argument {\it some text} in the
{\tt \\begin\{items\}\[{\it some text}\]}
command. The optional argument is {\em not}
displayed. Its width is calculated and used to indent all subsequent
lines in the group except from the first line of each new item. This
indentation rule applies to all text {\em before} the first
{\tt \\item} command as well.

The {\tt \\item\[{\it some text}\]} command specifies the start of a
new item.  The {\it some text} optional argument will be displayed in
{\em bold} font at the current left-page-margin. Then, the text
following the command will be displayed in normal fashion with the
above indentation rule.
\beginImportant
\begin{paste}{HTXFormatPage5xPaste1}{HTXFormatPage5xPatch1}
\pastebutton{HTXFormatPage5xPaste1}{Interpret}
\newline
{\tt \\indent\{5\}}\newline
{\tt \\begin\{items\}\[how wide am I\]}\newline
{\tt Here we carry on but a \\newline} \newline
{\tt new line will be indented } \newline
{\tt \\item\[how wide am I\] fits nicely.
Here is a \\newline new line in an item.}\newline
{\tt \\item\[again\] to show another item}\newline
{\tt \\item\[\\\\tab\]\\tab\{0\} can be used \\tab\{15\} effectively}
\newline
```

```
{\tt \\end\{items\}}\newline
{\tt \\indent\{0\}}\newline
\end{paste}
\endImportant


Note that the {\tt \\begin\{items\}} command immediately sets the left-
page-margin to a new value. Subsequent
{\tt \\tab} or {\tt \\centerline} commands
refer to this new margin.
Any explicit margin setting commands included
in the group {\em will} have the normal effect.
The {\tt \\par} command does not produce
the standard paragraph indentation within a list group --- it behaves
instead like {\tt \\newline}.


You can nest list groups like the following example suggests.
\beginImportant
\begin{paste}{HTXFormatPage5xPaste2}{HTXFormatPage5xPatch2}
\pastebutton{HTXFormatPage5xPaste2}{Interpret}
\newline
{\tt \\begin\{items\}\[quitealot\]}\newline
{\tt A nested list:}\newline
{\tt \\item\[The first\] item of an itemized list is on this line.}
\newline
{\tt \\item\[The second\] item of the list starts here.
It contains another}\newline
{\tt list nested inside it.}\newline
{\tt \\begin\{items\}\[somuchmore\]}\newline
{\tt \\item \[First\]\\tab\{0\}This is the first item of an
enumerated}\newline
{\tt list that is nested within the itemized list.}\newline
{\tt \\item \[Second\]\\tab\{0\}This is the second item of the
inner list.}\newline
{\tt \\end\{items\}}\newline
{\tt This is the rest of the second item of the outer list.  It}\newline
{\tt is no more interesting than any other part of the item.}\newline
{\tt \\item\[The third\] item of the list.}\newline
{\tt \\end\{items\}}\newline
\end{paste}
\endImportant

Another facility for presenting lists is the {\tt \\table} command.
The correct syntax for it is : {\tt \\table\{\{{\it item a}\}
\{{\it item b}\} {\it ..}\}}.
The items in the braces will be placed in as many aligned columns
as is possible for the current window dimensions or page width.
If one item is particularly long there will probably be only one column
in the table. Here is a table of color names.
```

```
\beginImportant
\begin{paste}{HTXFormatPage5xPaste3}{HTXFormatPage5xPatch3}
\pastebutton{HTXFormatPage5xPaste3}{Interpret}
\newline
{\tt
\\table\{
\{Dark Orchid\} \{Dark Salmon\} \{Dark Sea Green\} \{Dark Slate Blue\}
\{Dark Slate Gray\} \{Dark Turquoise\} \{Dark Violet\} \{Deep Pink\}
\{Deep Sky Blue\} \{Dodger Blue\} \{Floral White\} \{Forest Green\}
\{Ghost White\} \{Hot Pink\} \{Indian Red\} \{Lavender Blush\}
\}
}
\end{paste}
\endImportant




\end{scroll}
\beginmenu
\menulink{Next -- Boxes and Lines}{HTXFormatPage6}
\endmenu

\end{page}
```

HTXFormatPage5xPatch1 patch

— htxformatpage5.ht —

```
\begin{patch}{HTXFormatPage5xPatch1}
\begin{paste}{HTXFormatPage5xPaste1A}{HTXFormatPage5xPatch1A}
\pastebutton{HTXFormatPage5xPaste1A}{Source}
\newline
\indent{5}
\begin{items}[how wide am I]
Here we carry on but a \newline
new line will be indented
\item[how wide am I] fits nicely. Here is a \newline new line in an item.
\item[again] to show another item
\item[\\tab]\tab{0} can be used \tab{15} effectively
\end{items}
\indent{0}
\end{paste}
\end{patch}
```

———-

HTXFormatPage5xPatch1A patch

— htxformatpage5.ht —

```
\begin{patch}{HTXFormatPage5xPatch1A}
\begin{paste}{HTXFormatPage5xPaste1B}{HTXFormatPage5xPatch1}
\pastebutton{HTXFormatPage5xPaste1B}{Interpret}
\newline
{\tt \\indent\{5\}}\newline
{\tt \\begin\{items\}\[how wide am I\]}\newline
{\tt Here we carry on but a \\newline} \newline
{\tt new line will be indented } \newline
{\tt \\item\[how wide am I\] fits nicely.
Here is a \\newline new line in an item.}\newline
{\tt \\item\[again\] to show another item}\newline
{\tt \\item\[\\\\tab\]\\tab\{0\} can be used \\tab\{15\} effectively}
\newline
{\tt \\end\{items\}}\newline
{\tt \\indent\{0\}}\newline
\end{paste}
\end{patch}
```

———-

HTXFormatPage5xPatch2 patch

— htxformatpage5.ht —

```
\begin{patch}{HTXFormatPage5xPatch2}
\begin{paste}{HTXFormatPage5xPaste2A}{HTXFormatPage5xPatch2A}
\pastebutton{HTXFormatPage5xPaste2A}{Source}
\newline
\begin{items}[quitealot]
A nested list:
\item[The first] item of an itemized list is on this line.
\item[The second] item of the list starts here.  It contains another
list nested inside it.
\begin{items}[somuchmore]
\item [First]\tab{0}This is the first item of the
list that is nested within the itemized list.
\item [Second]\tab{0}This is the second item of the inner list.
\end{items}
```

```
This is the rest of the second item of the outer list.  It
is no more interesting than any other part of the item.
\item [The third] item of the list.
\end{items}
\end{paste}
\end{patch}
```

———-

HTXFormatPage5xPatch2A patch

— htxformatpage5.ht —

```
\begin{patch}{HTXFormatPage5xPatch2A}
\begin{paste}{HTXFormatPage5xPaste2B}{HTXFormatPage5xPatch2}
\pastebutton{HTXFormatPage5xPaste2B}{Interpret}
\newline
{\tt \\begin\{items\}\[quitealot\]}\newline
{\tt A nested list:}\newline
{\tt \\item\[The first\] item of an itemized list is on this line.}
\newline
{\tt \\item\[The second\] item of the list starts here.
It contains another}\newline
{\tt list nested inside it.}\newline
{\tt \\begin\{items\}\[somuchmore\]}\newline
{\tt \\item \[First\]\\tab\{0\}This is the first item of an
enumerated}\newline
{\tt list that is nested within the itemized list.}\newline
{\tt \\item \[Second\]\\tab\{0\}This is the second item of
the inner list.}\newline
{\tt \\end\{items\}}\newline
{\tt This is the rest of the second item of the outer list.  It}
\newline
{\tt is no more interesting than any other part of the item.}\newline
{\tt \\item\[The third\] item of the list.}\newline
{\tt \\end\{items\}}\newline
\end{paste}
\end{patch}
```

———-

HTXFormatPage5xPatch3 patch

— htxformatpage5.ht —

```
\begin{patch}{HTXFormatPage5xPatch3}
\begin{paste}{HTXFormatPage5xPaste3A}{HTXFormatPage5xPatch3A}
\pastebutton{HTXFormatPage5xPaste3A}{Source}
\newline
\table{
{Dark Orchid} {Dark Salmon} {Dark Sea Green} {Dark Slate Blue}
{Dark Slate Gray}
{Dark Turquoise} {Dark Violet} {Deep Pink} {Deep Sky Blue} {Dodger Blue}
{Floral White} {Forest Green} {Ghost White} {Hot Pink} {Indian Red}
{Lavender Blush}
}
\end{paste}
\end{patch}
```

———-

HTXFormatPage5xPatch3A patch

— **htxformatpage5.ht** —

```
\begin{patch}{HTXFormatPage5xPatch3A}
\begin{paste}{HTXFormatPage5xPaste3B}{HTXFormatPage5xPatch3}
\pastebutton{HTXFormatPage5xPaste3B}{Interpret}
\newline
{\tt
\\table\{
\{Dark Orchid\} \{Dark Salmon\} \{Dark Sea Green\} \{Dark Slate Blue\}
\{Dark Slate Gray\} \{Dark Turquoise\} \{Dark Violet\} \{Deep Pink\}
\{Deep Sky Blue\} \{Dodger Blue\} \{Floral White\} \{Forest Green\}
\{Ghost White\} \{Hot Pink\} \{Indian Red\} \{Lavender Blush\}
\}
}
\end{paste}
\end{patch}
```

———-

21.13 htxformatpage6

Boxes and Lines

— **htxformatpage6.ht** —

```
\begin{page}{HTXFormatPage6}{Boxes and Lines}
\centerline{\fbox{{\tt \thispage}}}\newline
\begin{scroll}

The {\tt \\fbox} command can be used to place a box around one or more
words. The argument of the {\tt \\fbox} command is the text that will be
placed in the box. This command should only be used for text that can fit
in one line.

\beginImportant
\begin{paste}{HTXFormatPage6xPaste1}{HTXFormatPage6xPatch1}
\pastebutton{HTXFormatPage6xPaste1}{Interpret}
\newline
{\tt \\fbox\{Boxed!\}}\newline
\end{paste}
\endImportant


Use the {\tt \\horizontalline} command to draw a horizontal line
across the window. This might be useful for added emphasis.

\beginImportant
\begin{paste}{HTXFormatPage6xPaste2}{HTXFormatPage6xPatch2}
\pastebutton{HTXFormatPage6xPaste2}{Interpret}
\newline
{\tt \\horizontalline}\newline
\end{paste}
\endImportant

\end{scroll}
\beginmenu
\menulink{Next -- Micro-Spacing}{HTXFormatPage7}
\endmenu

\end{page}
```

HTXFormatPage6xPatch1 patch

— **htxformatpage6.ht** —

```
\begin{patch}{HTXFormatPage6xPatch1}
\begin{paste}{HTXFormatPage6xPaste1A}{HTXFormatPage6xPatch1A}
\pastebutton{HTXFormatPage6xPaste1A}{Source}
\newline
\fbox{Boxed!}
```

```
\end{paste}
\end{patch}
\begin{patch}{HTXFormatPage6xPatch1A}
\begin{paste}{HTXFormatPage6xPaste1B}{HTXFormatPage6xPatch1}
\pastebutton{HTXFormatPage6xPaste1B}{Interpret}
\newline
{\tt \\fbox\{Boxed!\}}\newline
\end{paste}
\end{patch}
```

———-

HTXFormatPage6xPatch2 patch

— **htxformatpage6.ht** —

```
\begin{patch}{HTXFormatPage6xPatch2}
\begin{paste}{HTXFormatPage6xPaste2A}{HTXFormatPage6xPatch2A}
\pastebutton{HTXFormatPage6xPaste2A}{Source}
\newline
\horizontalline
\end{paste}
\end{patch}
\begin{patch}{HTXFormatPage6xPatch2A}
\begin{paste}{HTXFormatPage6xPaste2B}{HTXFormatPage6xPatch2}
\pastebutton{HTXFormatPage6xPaste2B}{Interpret}
\newline
{\tt \\horizontalline}\newline
\end{paste}
\end{patch}
```

———-

21.14 htxformatpage7

Micro-Spacing

— **htxformatpage7.ht** —

```
\begin{page}{HTXFormatPage7}{Micro-Spacing}
\centerline{\fbox{{\tt \thispage}}}\newline
\begin{scroll}
```

```
There are three commands that one can use to exercise finer control
over the appearance of text on a page: {\tt \\space}, {\tt \\hspace}
and {\tt \\vspace}.

The {\tt \\space\{{\it value}\}} command accepts an integer argument
and simply changes the position of the next character to the right or
to the left.  A negative argument will move the next character to the
left and a positive one to the right. The unit of movement is {\it the
width of a character}. In this way one can overstrike characters to
produce various effects.

\beginImportant
\begin{paste}{HTXFormatPage7xPaste1}{HTXFormatPage7xPatch1}
\pastebutton{HTXFormatPage7xPaste1}{Interpret}
\newline
{\tt 0\\space\{-1\}\\}\newline
{\tt underlined\\space\{-10\}__________}\newline
\end{paste}
\endImportant


The {\tt \\hspace\{{\it value}\}} command
is similar to the {\tt \\space\{{\it value}\}} command.
It also accepts an integer argument and
changes the position of the next character to the right or to the left.
A negative argument will move the next character to the left and a
positive one to the right. The unit of movement is {\it a pixel}.
The {\it value} argument specifies an offset from the default placement
of the character.

\beginImportant
\begin{paste}{HTXFormatPage7xPaste2}{HTXFormatPage7xPatch2}
\pastebutton{HTXFormatPage7xPaste2}{Interpret}
\newline
{\tt x\\hspace\{-4\}x\\hspace\{-3\}x\\hspace\{-2\}x\\hspace\{-1\}x\%}\newline
{\tt x\\hspace\{1\}x\\hspace\{2\}x\\hspace\{3\}x\\hspace\{4\}x}
\end{paste}
\endImportant

The {\tt \\vspace\{{\it value}\}} command is similar to the
{\tt \\hspace\{{\it value}\}} command but (as the name suggests)
works in the vertical direction. The unit of movement is {\it a
pixel}. The {\it value} argument specifies an offset from {\it
the next line}. A negative argument moves the next character up
and a positive down. This command can be used for subscripts and
superscripts. One drawback in the use of {\tt \\vspace} is that
it can only work with a particular font at a time. This is
because the inter-line spacing depends on the font being used
and the value of it is needed to get "back" on the line.
```

```
In general, the command {\tt \\vspace\{{\it - ils}\}} will
have a null effect when {\it ils} = ( font ascent + font descent + 5 ).
The example below assumes that {\it ils} = 25 e.g. the Rom14 font
on the RISC System/6000.

\beginImportant
\begin{paste}{HTXFormatPage7xPaste3}{HTXFormatPage7xPatch3}
\pastebutton{HTXFormatPage7xPaste3}{Interpret}
\newline
{\tt CO\\vspace\{-18\}2\\vspace\{-32\} + CaO ->}\newline
{\tt CaCO\\vspace\{-18\}3\\vspace\{-32\}\\newline}\newline
{\tt R\\space\{-1\}~\\vspace\{-18\}v\\vspace\{-32\}}\newline
{\tt \\hspace\{4\}-\\hspace\{8\}---\\hspace\{-12\}}\newline
{\tt \\vspace\{-32\}1\\space\{-1\}\\vspace\{-7\}2}\newline
{\tt \\vspace\{-36\}\\hspace\{8\}g\\space\{-1\}~}\newline
{\tt \\vspace\{-18\}v\\vspace\{-32\}\\hspace\{2\}R}\newline
{\tt \\space\{-1\}~ = T\\vspace\{-18\}v\\vspace\{-32\}}\newline
{\tt \\vspace\{-25\}}
\end{paste}
\endImportant


\end{scroll}
\beginmenu
\menulink{Next -- Bitmaps and Images}{HTXFormatPage8}
\endmenu

\end{page}
```

HTXFormatPage7xPatch1 patch

— htxformatpage7.ht —

```
\begin{patch}{HTXFormatPage7xPatch1}
\begin{paste}{HTXFormatPage7xPaste1A}{HTXFormatPage7xPatch1A}
\pastebutton{HTXFormatPage7xPaste1A}{Source}
\newline
0\space{-1}\\
underlined\space{-10}__________
\end{paste}
\end{patch}
\begin{patch}{HTXFormatPage7xPatch1A}
\begin{paste}{HTXFormatPage7xPaste1B}{HTXFormatPage7xPatch1}
\pastebutton{HTXFormatPage7xPaste1B}{Interpret}
\newline
```

```
{\tt 0\\space\{-1\}\\}\newline
{\tt underlined\\space\{-10\}__________}\newline
\end{paste}
\end{patch}
```

———-

HTXFormatPage7xPatch2 patch

— htxformatpage7.ht —

```
\begin{patch}{HTXFormatPage7xPatch2}
\begin{paste}{HTXFormatPage7xPaste2A}{HTXFormatPage7xPatch2A}
\pastebutton{HTXFormatPage7xPaste2A}{Source}
\newline
x\hspace{-4}x\hspace{-3}x\hspace{-2}x\hspace{-1}x%
x\hspace{1}x\hspace{2}x\hspace{3}x\hspace{4}x
\end{paste}
\end{patch}
```

———-

HTXFormatPage7xPatch2A patch

— htxformatpage7.ht —

```
\begin{patch}{HTXFormatPage7xPatch2A}
\begin{paste}{HTXFormatPage7xPaste2B}{HTXFormatPage7xPatch2}
\pastebutton{HTXFormatPage7xPaste2B}{Interpret}
\newline
{\tt x\\hspace\{-4\}x\\hspace\{-3\}x\\hspace\{-2\}x\\hspace\{-1\}x\%}
\newline
{\tt x\\hspace\{1\}x\\hspace\{2\}x\\hspace\{3\}x\\hspace\{4\}x}
\end{paste}
\end{patch}
```

———-

HTXFormatPage7xPatch3 patch

— htxformatpage7.ht —

```
\begin{patch}{HTXFormatPage7xPatch3}
\begin{paste}{HTXFormatPage7xPaste3A}{HTXFormatPage7xPatch3A}
\pastebutton{HTXFormatPage7xPaste3A}{Source}
\newline
CO\vspace{-18}2\vspace{-32} + CaO ->
CaCO\vspace{-18}3\vspace{-32}\newline
R\space{-1}~\vspace{-18}v\vspace{-32}
\hspace{4}-\hspace{8}---\hspace{-12}
\vspace{-32}1\space{-1}\vspace{-7}2
\vspace{-36}\hspace{8}g\space{-1}~
\vspace{-18}v\vspace{-32}\hspace{2}R
\space{-1}~ = T\vspace{-18}v\vspace{-32}
\vspace{-25}
\end{paste}
\end{patch}
```

HTXFormatPage7xPatch3A patch

— htxformatpage7.ht —

```
\begin{patch}{HTXFormatPage7xPatch3A}
\begin{paste}{HTXFormatPage7xPaste3B}{HTXFormatPage7xPatch3}
\pastebutton{HTXFormatPage7xPaste3B}{Interpret}
\newline
{\tt CO\\vspace\{-18\}2\\vspace\{-32\} + CaO ->}\newline
{\tt CaCO\\vspace\{-18\}3\\vspace\{-32\}\\newline}\newline
{\tt R\\space\{-1\}~\\vspace\{-18\}v\\vspace\{-32\}}\newline
{\tt \\hspace\{4\}-\\hspace\{8\}---\\hspace\{-12\}}\newline
{\tt \\vspace\{-32\}1\\space\{-1\}\\vspace\{-7\}2}\newline
{\tt \\vspace\{-36\}\\hspace\{8\}g\\space\{-1\}~}\newline
{\tt \\vspace\{-18\}v\\vspace\{-32\}\\hspace\{2\}R}\newline
{\tt \\space\{-1\}~ = T\\vspace\{-18\}v\\vspace\{-32\}}\newline
{\tt \\vspace\{-25\}}
\end{paste}
\end{patch}
```

21.15 htxformatpage8

Bitmaps and Images

— htxformatpage8.ht —

```
\begin{page}{HTXFormatPage8}{Bitmaps and Images}
\centerline{\fbox{{\tt \thispage}}}\newline
\begin{scroll}

The commands {\tt \\inputbitmap\{{\it filename}\}}
and {\tt \\inputimage\{{\it filename}\}}
allow you to include an X11 bitmap or an
Axiom-generated viewport in a Hyperdoc
page.

In the case of the {\tt \\inputbitmap} command
the {\it filename} parameter must be the full pathname
of an X11 bitmap file.

\beginImportant
\begin{paste}{HTXFormatPage8xPaste1}{HTXFormatPage8xPatch1}
\pastebutton{HTXFormatPage8xPaste1}{Interpret}
\newline
{\tt \\inputbitmap\{\env{AXIOM}/doc/bitmaps/sup.bitmap\} }
\end{paste}
\endImportant

The {\it filename} parameter of the {\tt \\inputimage} command must be
the full pathname of a {\it compressed XPM image} file without the
name extensions.  Hyperdoc always adds ".xpm.Z" to whatever filename
you give and looks for the augmented filename.  Such files can be
generated by Axiom command \axiomOp{write} with the {\tt "image"} or
{\tt "pixmap"} options.

\beginImportant
\begin{paste}{HTXFormatPage8xPaste2}{HTXFormatPage8xPatch2}
\pastebutton{HTXFormatPage8xPaste2}{Interpret}
\newline
{\tt \\inputimage\{\env{AXIOM}/doc/viewports/ugProblemNumericPage30.view/image\}}
\end{paste}
\endImportant

Be careful not to break the pathname across lines.

The {\tt \\inputimage} command will automatically select
the {\it image.xpm} or the {\it image.bm} file for you
based on the capabilities of your X server.
```

```
For your convenience, there are two macros defined
in \centerline{ {\bf util.ht}.}
The {\tt \\viewport} macro eliminates the need to specify
the {\tt .view/image} part and the
{\tt \\axiomViewport} macro automatically selects viewport
files in the system directories. The above {\tt \\inputimage}
could have been written
\beginImportant
{\tt \\viewport\{\env{AXIOM}/doc/viewports/ugProblemNumericPage30\}}
\endImportant
or
\beginImportant
{\tt \\axiomViewport\{ugProblemNumericPage30\}}
\endImportant



\end{scroll}
\beginmenu
\menulink{Back to Formatting menu}{HTXFormatTopPage}
\endmenu

\end{page}
```

———-

HTXFormatPage8xPatch1 patch

— htxformatpage8.ht —

```
\begin{patch}{HTXFormatPage8xPatch1}
\begin{paste}{HTXFormatPage8xPaste1A}{HTXFormatPage8xPatch1A}
\pastebutton{HTXFormatPage8xPaste1A}{Source}
\newline
\inputbitmap{\env{AXIOM}/doc/bitmaps/sup.bitmap}
\end{paste}
\end{patch}
\begin{patch}{HTXFormatPage8xPatch1A}
\begin{paste}{HTXFormatPage8xPaste1B}{HTXFormatPage8xPatch1}
\pastebutton{HTXFormatPage8xPaste1B}{Interpret}
\newline
{\tt \\inputbitmap\{\env{AXIOM}/doc/bitmaps/sup.bitmap\} }
\end{paste}
\end{patch}
```

———-

HTXFormatPage8xPatch2 patch

— htxformatpage8.ht —

```
\begin{patch}{HTXFormatPage8xPatch2}
\begin{paste}{HTXFormatPage8xPaste2A}{HTXFormatPage8xPatch2A}
\pastebutton{HTXFormatPage8xPaste2A}{Source}
\newline
\inputimage{\env{AXIOM}/doc/viewports/ugProblemNumericPage30.view/image}
\end{paste}
\end{patch}
```

———-

HTXFormatPage8xPatch2A patch

— htxformatpage8.ht —

```
\begin{patch}{HTXFormatPage8xPatch2A}
\begin{paste}{HTXFormatPage8xPaste2B}{HTXFormatPage8xPatch2}
\pastebutton{HTXFormatPage8xPaste2B}{Interpret}
\newline
{\tt \\inputimage
\{\env{AXIOM}/doc/viewports/ugProblemNumericPage30.view/image\}}
\end{paste}
\end{patch}
```

———-

21.16 htxformattoppage.ht

Formatting in Hyperdoc

⇒ "notitle" (HTXFormatPage1) 21.8 on page 2631
⇒ "notitle" (HTXFormatPage2) 21.9 on page 2633
⇒ "notitle" (HTXFormatPage3) 21.10 on page 2637
⇒ "notitle" (HTXFormatPage4) 21.11 on page 2641
⇒ "notitle" (HTXFormatPage5) 21.12 on page 2648
⇒ "notitle" (HTXFormatPage6) 21.13 on page 2653

⇒ "notitle" (HTXFormatPage7) 21.14 on page 2655
⇒ "notitle" (HTXFormatPage8) 21.15 on page 2660

— **htxformattoppage.ht** —

```
\begin{page}{HTXFormatTopPage}{Formatting in Hyperdoc}
\centerline{\fbox{{\tt \thispage}}}\newline

Hyperdoc offers various facilities for formatting text and images.
You can learn about these facilities by clicking on the topics below.
\begin{scroll}
\beginmenu
\menudownlink{Special Characters}{HTXFormatPage1}
\menudownlink{Formatting without commands}{HTXFormatPage2}
\menudownlink{Using different fonts}{HTXFormatPage3}
\menudownlink{Indentation}{HTXFormatPage4}
\menudownlink{Creating Lists and Tables}{HTXFormatPage5}
\menudownlink{Boxes and Lines}{HTXFormatPage6}
\menudownlink{Micro-Spacing}{HTXFormatPage7}
\menudownlink{Bitmaps and Images}{HTXFormatPage8}
\endmenu
\end{scroll}
\end{page}
```

———-

21.17 htxintropage1.ht

What Hyperdoc does

⇒ "notitle" (HTXIntroPage2) 21.18 on page 2664
⇒ "notitle" (ugHyperPage) 8 on page 1707

— **htxintropage1.ht** —

```
\begin{page}{HTXIntroPage1}{What Hyperdoc does}
\centerline{\fbox{{\tt \thispage}}}\newline
\beginscroll

Take a close look at the objects in the Hyperdoc window you are now
reading.  Most of them are text. Resize the window using the window
manager facilities. The text is reformatted to fit the window
border. This action is performed by Hyperdoc. At the simplest level,
it provides a method for {\em formatting} text in a window.  In fact,
it can place other things on the window as well, such as bitmaps or
color images. The {\em buttons} you see at either side at the top of
```

```
the window are bitmaps.

Move the cursor so that it rests on one of those buttons. You notice
that the cursor has changed appearance. This indicates that there is
an action associated with the button. This action will be performed
when you click the mouse button over the {\em active area}. If you are
familiar with Hyperdoc, you know that the active area can be words,
bitmaps, images or {\em input areas}. In fact, anything that can be
displayed in a Hyperdoc window can be an active area.

So, what can the action associated with an active area be? Hyperdoc
allows quite a bit of freedom in defining that action. We will have a
close look at this issue \downlink{later on}{HTXLinkTopPage}. For now,
recall the various actions that you have encountered so far ---
executing Axiom commands, popping up new windows, providing parameters
for other active areas, and replacing or changing the contents of the
window. The most common action is to bring up some Hyperdoc text in
the same or a different window.  This lets us create {\em links}
between pieces of text and images. A system with such capability is
usually called a {\em hypertext} system.  Hyperdoc is in fact much
more.

\endscroll
\beginmenu
\menulink{Next -- How Hyperdoc does it}{HTXIntroPage2}
\menuwindowlink{Review some features of Hyperdoc}{ugHyperPage}
\endmenu

\helppage{ugHyperPage}
\end{page}
```

———-

21.18 htxintropage2.ht

How Hyperdoc does it

⇒ "notitle" (HTXIntroPage3) 21.19 on page 2666

— **htxintropage2.ht** —

```
\begin{page}{HTXIntroPage2}{How Hyperdoc does it}
{\centerline{\fbox{{\tt \thispage}}}\newline}
\beginscroll

Hyperdoc can read the {\em hypertext} information from standard text
files. This means that you can create or change this information with
```

```
any text editor. Once this information has been entered into the
files, a special program, called {\bf htadd}, scans these files and
produces a database (another file called {\bf ht.db}) of {\em objects}
encountered in the files. Hyperdoc consults this database when it
first starts and so knows where it might find the definitions of these
objects. You can maintain several such databases on different
directories. You indicate which database you want Hyperdoc to consult
by setting an {\em environment variable} called {\bf HTPATH}.

In general, hypertext must obviously use some kind of special (that is,
non-textual) marks for all the  extra functionality it provides. In
Hyperdoc, these marks are some special characters --- special
in the sense that they are not interpreted as ordinary displayable text.
These characters, however, are part of the standard ASCII set.
There is also a way to display these special characters as text .
The Hyperdoc special characters are :

\beginImportant
\noindent{\em Special Characters}:
{\tt \table{{\$}{\\}{\{}{\}}{\[}{\]}{\%}{\#}}}
\endImportant

Hyperdoc uses the special characters to distinguish between
{\em text} and {\em commands} (by {\em text}, we mean here anything
displayable). The commands are instructions to
Hyperdoc to treat some text in a particular way. Some commands
define special Hyperdoc objects. The most important objects
are {\em pages}, {\em patches}, and {\em macros}.
A {\em page} is a description of the contents of a
Hyperdoc window. A {\em patch} is a portion of a page.
A {\em macro} is a user-defined new Hyperdoc command.
Some commands allow special text {\em formatting} and others
associate some text with an action.

In order to display anything at all in \HyperName, you must define a
{\em page}. The next section explains how to define a {\em page} and put
some simple text into it.
\endscroll
\beginmenu
\menudownlink{Next -- Define a simple text page}{HTXIntroPage3}
\endmenu

\end{page}
```

21.19 htxintropage3.ht

A simple text page

⇒ “notitle” (HTXLinkPage6) 21.26 on page 2693
⇒ “notitle” (HTXTryPage) 21.29 on page 2700
⇒ “notitle” (HTXFormatTopPage) 21.16 on page 2662

— **htxintropage3.ht** —

```
\begin{page}{HTXIntroPage3}{A simple text page}
{\centerline{\fbox{{\tt \thispage}}}\newline}
\begin{scroll}


A page is defined by a {\em group} command. Group commands are used to
delimit a group, that is, to declare where a group starts and where it
ends. The proper syntax for a page definition is as follows:
\beginImportant
{\tt \\begin\{page\}\{{\it name}\}\{{\it a title}\}}
\newline
.
\newline
.
\newline
.
\newline
{\tt \\end\{page\}}
\beginImportant

Note the use of the special characters {\tt \\}, {\tt \{} and {\tt
\}}.  The {\tt \\} (backslash) character introduces a command, in this
case, {\tt begin}.  The {\tt \{ \}} (braces) delimit the {\em
parameters} to the command.  The first parameter (the word {\tt page})
specifies this as a page definition command.

The second parameter can be any single unbroken word consisting of
alphanumeric characters only, and specifies the name of the page by
which it can be referred to by other commands. You should choose this
internal name with care so as to avoid potential conflict with page
names that are defined by the Axiom system. This caveat only applies
in the case where you have started Hyperdoc with the Axiom database
--- see \downlink{later on}{HTXLinkPage6}. It is suggested that the
page names you define start with the letters {\tt UX} (standing for
{\tt U}ser e{\tt X}tensions). You can have a look at the Axiom system
database file {\centerline{\bf \env{AXIOM}/doc/ht.db} }
which contains the names of all pages, macros and patches used by
Axiom.
```

```
The third parameter specifies a title for the page.
The title of a page is the area at the very top
of the window, between the buttons. Virtually anything
that can be put in the main page can also be put in the
title. As an example, {\em this} page's
declaration is like this:\newline
{\tt \\begin\{page\}\{\thispage\}\{A simple text page\}}

Everything you type between the {\tt \\begin\{page\}} command and the
next {\tt \\end\{page\}} command will become the body of the page. It
is an error to insert another {\tt \\begin\{page\}} between the two,
that is, this group command cannot be nested.

There is another useful group command that should be mentioned here
--- the {\em scroll} command. It controls the portion of the page that
will be scrollable. Hyperdoc will split a page in three sections:
a {\em header}, a {\em scroll region} and a {\em footer}. Hyperdoc
will always try to keep the header and footer regions visible on the
page; the header at the top and the footer at the bottom. The middle
scroll region will be truncated and a scroll bar will be automatically
provided if the window becomes too small for the whole contents of the
page. Only one scroll region can be defined in a page and the correct
syntax is as follows:
\beginImportant
{\tt \\begin\{scroll\}}
\newline
.
\newline
.
\newline
.
\newline
{\tt \\end\{scroll\}}
\beginImportant

This group should be placed inside the relevant page group. The text
between the {\tt \\begin\{page\}} and {\tt \\begin\{scroll\}} commands
defines the header region, the text inside the scroll group defines
the scroll region and the text between the {\tt \\end\{scroll\}} and
{\tt \\end\{page\}} commands defines the footer region. It is
important to keep the header and footer areas small.  Use them to
display information that might be needed at any time by the user.  If
you don't define a scroll region in your page, you may find that a
portion of the page is truncated.

You are now ready to experiment with a page of your own. If you just
want to display some text on a page, you don't need any other
Hyperdoc commands. Just make sure that the text you type for the
title, header, scroll and footer regions does not contain (for the
moment) any of the Hyperdoc special characters.
```

```
\end{scroll}
\beginmenu
\menuwindowlink{Try out what you learned}{HTXTryPage}
\menudownlink{Next -- Learn how to format text}{HTXFormatTopPage}
\endmenu
\end{page}
```

———-

21.20 htxintrotoppage.ht

First Steps

⇒ “notitle” (HTXIntroPage1) 21.17 on page 2663
⇒ “notitle” (HTXIntroPage2) 21.18 on page 2664
⇒ “notitle” (HTXIntroPage3) 21.19 on page 2666

— **htxintrotoppage.ht** —

```
\begin{page}{HTXIntroTopPage}{First Steps}
\centerline{\fbox{{\tt \thispage}}}\newline
\beginscroll

Hyperdoc is both a way of presenting information and
a customisable front-end. Axiom uses
it for its own purpose as a front-end and documentation system.
Hyperdoc has special facilities that allow it to interact
very closely with Axiom.  The \Browse{} facility, the Basic
Commands section and the ability to execute Axiom commands
by clicking on Hyperdoc text are witness to this.

These pages will show you the features of Hyperdoc that might
make it appropriate for your own use in, for example, providing
documentation for Axiom code that you write or some other purpose.

It is recommended that you get familiar with the {\em use} of
Hyperdoc before proceeding.

\endscroll
\beginmenu
\menudownlink{What Hyperdoc does}{HTXIntroPage1}
\menudownlink{How Hyperdoc does it}{HTXIntroPage2}
\menudownlink{Define a simple text page}{HTXIntroPage3}
\endmenu
```

```
\end{page}
```

———-

21.21 htxlinkpage1.ht

Linking to a named page

⇒ "notitle" (HTXLinkTopPage) 21.27 on page 2698
⇒ "notitle" (TestHelpPage) 21.21 on page 2672
⇒ "notitle" (HTXLinkPage2) 21.22 on page 2672

— **htxlinkpage1.ht** —

```
\begin{page}{HTXLinkPage1}{Linking to a named page}
\centerline{\fbox{{\tt \thispage}}}\newline
\begin{scroll}

In Hyperdoc, hypertext links are specified by different flavors of the
{\tt \\link} command. These commands take two arguments. One argument
specifies the active area, that is, the {\it trigger} of the link. The
second argument specifies the {\it target} of the link, that is, a
page. The trigger can be quite arbitrary Hyperdoc text and can include
images or whole paragraphs. The trigger text will be formatted in the
normal fashion but its default font will be the font specified by the
ActiveFont resource.

The simplest kind of Hyperdoc link is a link to a named page.
Clicking on the trigger will cause the named page to appear in a
Hyperdoc window.
There are three flavors for such a link.
\begin{items}[123456]
\item\menuitemstyle{{\tt \\windowlink\{{\it trigger}\}
\{{\it page name}\}}}
\newline
This link command, when activated, will create a new
window for the named page.
\newline
There will be no \centerline{\UpBitmap{}  or \ReturnBitmap{}}
buttons on the new page.
The new page will have a \centerline{\ExitBitmap{}} button, however.
The original page containing the {\tt \\windowlink}
command will be unaffected.
\item\menuitemstyle{{\tt \\downlink\{{\it trigger}\}\{{\it page name}\}} }
\newline This link command, when activated, will cause the
```

```
current page to be replaced by the target page
in the same Hyperdoc window.
A \centerline{\UpBitmap{}}  button will automatically be placed
on the new page allowing you to get back to the page
containing the {\tt \\downlink} command.
If the current page has a \centerline{\ReturnBitmap{}} button then
the target page will also carry it. The associated
target page of that button will be the same as it is
in the current page.
\item\menuitemstyle{{\tt \\memolink\{{\it trigger}\}\{{\it page name}\}}}
\newline This link command is similar to the {\tt \\downlink} command.
In addition, it will cause a \centerline{\ReturnBitmap{}}
button to be included in
the target page and all pages {\tt \\downlink}ed from it.
This button will act as a
direct link to the page containing the {\tt \\memolink} command allowing
a short-cut to be taken.
\end{items}

\beginImportant
\begin{paste}{HTXLinkPage1xPaste1}{HTXLinkPage1xPatch1}
\pastebutton{HTXLinkPage1xPaste1}{Interpret}
\newline
{\tt \\windowlink
\{windowlink to Actions menu\}\{HTXLinkTopPage\}\\newline}
\newline
{\tt \\downlink\{downlink to Actions menu\}\{HTXLinkTopPage\}\\newline}
\newline
{\tt \\memolink\{memolink to Actions menu\}\{HTXLinkTopPage\}}
\end{paste}
\endImportant


There is a fourth button that can appear at the top of the page
next to the \centerline{\ExitBitmap{}} button.
Its purpose is to provide access to a particular {\it help page}
associated with the current page.
That is the \centerline{\HelpBitmap{}} button. The command to use
is
\centerline{{\tt \\helppage\{{\it help page name}\}}}
The {\tt \\helppage} command {\it must } be placed
just before the {\tt \\end\{page\}} command.
For instance, to get a help button on this page
the following command is used.
\centerline{{\tt {\\helppage\{TestHelpPage\}}}}
Clicking on the help button at the top
will display the {\tt TestHelpPage} page in a new window.

\end{scroll}
\beginmenu
```

```
\menulink{Next -- Standard Pages}{HTXLinkPage2}
\endmenu

\helppage{TestHelpPage}
\end{page}
```

———-

HTXLinkPage1xPatch1 patch

— **htxlinkpage1.ht** —

```
\begin{patch}{HTXLinkPage1xPatch1}
\begin{paste}{HTXLinkPage1xPaste1A}{HTXLinkPage1xPatch1A}
\pastebutton{HTXLinkPage1xPaste1A}{Source}
\newline
\windowlink{windowlink to Actions
menu}{HTXLinkTopPage}\newline
\downlink{downlink to Actions menu}{HTXLinkTopPage}\newline
\memolink{memolink to Actions menu}{HTXLinkTopPage}
\end{paste}
\end{patch}
```

———-

HTXLinkPage1xPatch1A patch

— **htxlinkpage1.ht** —

```
\begin{patch}{HTXLinkPage1xPatch1A}
\begin{paste}{HTXLinkPage1xPaste1B}{HTXLinkPage1xPatch1}
\pastebutton{HTXLinkPage1xPaste1B}{Interpret}
\newline
{\tt \\windowlink
\{windowlink to Actions menu\}\{HTXLinkTopPage\}\\newline}
\newline
{\tt \\downlink\{downlink to Actions menu\}\{HTXLinkTopPage\}\\newline}
\newline
{\tt \\memolink\{memolink to Actions menu\}\{HTXLinkTopPage\}}
\end{paste}
\end{patch}
```

———-

Test Help Page

— **htxlinkpage1.ht** —

```
\begin{page}{TestHelpPage}{Test Help Page}
\begin{scroll}

\vspace{100}
\centerline{Is this any help?}
\end{scroll}
\end{page}
```

21.22 htxlinkpage2.ht

Standard Pages

⇒ "notitle" (HTXLinkPage6) 21.26 on page 2693
⇒ "notitle" (SpadNotConnectedPage) 2.1 on page 114
⇒ "notitle" (UnknownPage) 2.1 on page 114
⇒ "notitle" (ErrorPage) 2.1 on page 115
⇒ "notitle" (ProtectedQuitPage) 2.1 on page 114
⇒ "notitle" (HTXLinkPage3) 21.23 on page 2675

— **htxlinkpage2.ht** —

```
\begin{page}{HTXLinkPage2}{Standard Pages}
\centerline{\fbox{{\tt \thispage}}}\newline
\begin{scroll}

You have reached this page after performing
a series of mouse clicks on Hyperdoc
active areas. Each time, a {\tt \\link}
command was activated. Well, how does it all
start?
The answer is that Hyperdoc always puts up
a particular page called {\tt RootPage} when
it starts up. If this page is not found in the database,
Hyperdoc will immediately exit.
It is, of course, desirable that the {\tt RootPage}
contains links to other pages!
It is possible to override
Axiom's choice of {\tt RootPage} and provide your own
to Hyperdoc. This is done in the same way as
```

```
you would override any Axiom-defined page and is
discussed in \downlink{How to use your pages with Hyperdoc}{HTXLinkPage6}.



You may have noticed that Hyperdoc
uses some pages when certain events occur.
There is a page that is put up, for instance,
whenever Hyperdoc cannot connect to Axiom.
Another page is put up whenever there is a formatting
error and yet another when a request for an unknown page
is made. Finally, there is a page that prompts
for confirmation when you press the
exit button on the initial page.


These pages have standard names and must be provided
in the Hyperdoc page database.
They are already defined in the Axiom system
Hyperdoc page database so that you do not have to
define them yourself.

Here are the pages required by Hyperdoc. You can click on any of these
to see their contents. Click on their exit buttons when you are finished.

\beginImportant
\begin{paste}{HTXLinkPage2xPaste1}{HTXLinkPage2xPatch1}
\pastebutton{HTXLinkPage2xPaste1}{Interpret}
\newline
{\tt \\table\{}\newline
{\tt \{\\windowlink\{SpadNotConnectedPage\}\{SpadNotConnectedPage\}\}}
\newline
{\tt \{\\windowlink\{UnknownPage\}\{UnknownPage\}\}}\newline
{\tt \{\\windowlink\{ErrorPage\}\{ErrorPage\}\}}\newline
{\tt \{\\windowlink\{ProtectedQuitPage\}\{ProtectedQuitPage\}\}}\newline
{\tt \}}\newline
\end{paste}
\endImportant


In addition, Hyperdoc uses certain bitmaps for its buttons.
They are also provided in the Axiom system
bitmap directory and Hyperdoc knows where to find them.

The bitmap files required by Hyperdoc are the following.
\newline
\tab{7}{\it exit.bitmap}\tab{22} = \tab{25}{\ExitBitmap{}} \newline
\tab{7}{\it help2.bitmap}\tab{22} = \tab{25}{\HelpBitmap{}} \newline
\tab{7}{\it up3.bitmap}\tab{22} = \tab{25}{\UpBitmap{}}\newline
\tab{7}{\it return3.bitmap}\tab{22} = \tab{25}{\ReturnBitmap{}}\newline
```

```
\tab{7}{\it noop.bitmap}\tab{22} = \tab{25}{\NoopBitmap{}}

These files must exist in your current directory if
the {\tt AXIOM} environment variable is not set.
If it is, then Hyperdoc will assume that it points
to the Axiom system directory and will look for
these files in
{\bf \$AXIOM/doc/bitmaps}.

\end{scroll}
\beginmenu
\menulink{Next -- Active Axiom commands}{HTXLinkPage3}
\endmenu

\end{page}
```

———-

HTXLinkPage2xPatch1 patch

— htxlinkpage2.ht —

```
\begin{patch}{HTXLinkPage2xPatch1}
\begin{paste}{HTXLinkPage2xPaste1A}{HTXLinkPage2xPatch1A}
\pastebutton{HTXLinkPage2xPaste1A}{Source}
\newline
\table{
{\windowlink{SpadNotConnectedPage}{SpadNotConnectedPage}}
{\windowlink{UnknownPage}{UnknownPage}}
{\windowlink{ErrorPage}{ErrorPage}}
{\windowlink{ProtectedQuitPage}{ProtectedQuitPage}}
}
\end{paste}
\end{patch}
```

———-

HTXLinkPage2xPatch1A patch

— htxlinkpage2.ht —

```
\begin{patch}{HTXLinkPage2xPatch1A}
\begin{paste}{HTXLinkPage2xPaste1B}{HTXLinkPage2xPatch1}
```

```
\pastebutton{HTXLinkPage2xPaste1B}{Interpret}
\newline
{\tt \\table\{}\newline
{\tt \{\\windowlink\{SpadNotConnectedPage\}\{SpadNotConnectedPage\}\}}
\newline
{\tt \{\\windowlink\{UnknownPage\}\{UnknownPage\}\}}\newline
{\tt \{\\windowlink\{ErrorPage\}\{ErrorPage\}\}}\newline
{\tt \{\\windowlink\{ProtectedQuitPage\}\{ProtectedQuitPage\}\}}\newline
{\tt \}}\newline
\end{paste}
\end{patch}
```

———-

21.23 htxlinkpage3.ht

Active Axiom commands

⇒ "notitle" (HTXLinkPage4) 21.24 on page 2681

— **htxlinkpage3.ht.ht** —

```
\begin{page}{HTXLinkPage3}{Active Axiom commands}
\centerline{\fbox{{\tt \thispage}}}\newline
\begin{scroll}

This section explains how to include Axiom
commands in your page. The commands we will
introduce are actually {\it macros} that are defined
in
\centerline{{\bf \env{AXIOM}/doc/util.ht}}
This means that you can use them only if you include
this file in your Hyperdoc database.

The first command to learn is
\horizontalline
{\tt \\axiomcommand\{ {\it command }
{\tt \ \\free\{}{\it var1 var2 ...}
{\tt \}\ \\bound\{}{\it var}{\tt \}\ \}} }
\horizontalline


The {\tt \\free\{\}} and {\tt \\bound\{\}} directives are optional.
We will come to them in a minute. The {\it command} above is the
text of the Axiom command. Only single line commands are allowed
here.
This text will be displayed in the reserved AxiomFont logical
```

```
font.  The area of the text will be active and clicking on it
will attempt to send the command to Axiom for evaluation.
A new Axiom interpreter window (and Axiom frame)
will be created if this was the first Axiom command
activated in the current page. If not, the command will be sent
to the already opened Axiom interpreter window for the current page.
Note that it {\it is} necessary to escape special
Hyperdoc characters with the {\tt '\\'} backslash character.
The exceptions are the characters {\tt \[\]}; they do not
need to be escaped in this context.

\beginImportant
\begin{paste}{HTXLinkPage3xPaste1}{HTXLinkPage3xPatch1}
\pastebutton{HTXLinkPage3xPaste1}{Interpret}
\newline
{\tt \\axiomcommand\{ l:=brace[1,2,3] ; length:=\\\# l ; m:=[1,2]\}}
\end{paste}
\endImportant


The optional {\tt \\free\{\}} and {\tt \\bound\{\}} directives
provide dependency control. The reader of a Hyperdoc
page is not forced to click on the commands in the
order in which in they appear on the page.  If the
correct {\tt \\free\{\}} and {\tt \\bound\{\}}
specifications are made, clicking on a command
will result in execution of all those other
commands that should be executed before it.
This will {\it only} happen the first time the command is
clicked.

So, how are the dependencies specified?  The arguments of the {\tt
\\free\{\}} directive must be space-separated words (labels). The
argument of {\tt \\bound\{\}} must be a single (unique for the page)
label. Each label in the {\tt \\free\{\}} list must exist as an
argument to one (and only one) {\tt \\bound\{\}} directive somewhere
in the current page.  When the command is activated, Hyperdoc will
look in the {\tt \\free\{\}} list and check each label.  For each
label, it will find the command that specifies that label in its {\tt
\\bound\{\}} directive and execute it if it has not been already
executed.  The order of labels in the {\tt \\free\{\}} directive list
is respected. Hyperdoc will follow all dependency links recursively.

Here is  an example.
Clicking on the third command will automatically
execute all of them in the correct sequence.
Note that in this case the order of labels in the last
line is immaterial since {\tt v2} explicitly depends on {\tt v1}.

\beginImportant
```

```
\begin{paste}{HTXLinkPage3xPaste2}{HTXLinkPage3xPatch2}
\pastebutton{HTXLinkPage3xPaste2}{Interpret}
\newline
{\tt \\axiomcommand\{a:=1;d:=4 \\bound\{v1\}\}}\newline
{\tt \\newline}\newline
{\tt \\axiomcommand\{b:=a+3 \\free\{v1\} \\bound\{v2\}\}}\newline
{\tt \\newline}\newline
{\tt \\axiomcommand\{c:=b+d \\free\{v1 v2\} \\bound\{v3\}\}}\newline
\end{paste}
\endImportant

The second command deals with multi-line Axiom
code. This is the command to use for execution of
an Axiom {\it pile}. It is a {\it group}
command. The proper syntax for it is as follows:
\horizontalline
{\tt \\begin\{spadsrc\}
\ [\\free\{{\it var1 var2} ...\}\ \\bound\{{\it var}\}]}
\newline
.
\newline
.
\newline
{\tt \\end\{spadsrc\}}
\horizontalline

Again, the {\tt \\free} and {\tt \\bound} directives are
optional. If they are specified (in exactly the same way
as {\tt \\axiomcommand}), they must be enclosed in
square brackets {\tt []}.
The lines between the {\tt \\begin} and {\tt \\end}
contain the Axiom statements. Indentation
will be respected. Hyperdoc will
actually save this part in a temporary file
and instruct Axiom to read the file
with the {\tt )read} system command.

Here is an example. The execution of the following
fragment is dependent on the {\tt v3} label.
Make sure that previous commands are active (and
hence the label {\tt v3} is "visible") before
trying to execute it. If the label {\tt v3}
is not seen in the page, Hyperdoc will
print an error message on standard output
and ignore the dependency.


\beginImportant
\begin{paste}{HTXLinkPage3xPaste3}{HTXLinkPage3xPatch3}
\pastebutton{HTXLinkPage3xPaste3}{Interpret}
```

```
\newline
{\tt \\begin\{spadsrc\}\ [\\free\{v3\}\ \\bound\{v4\}]}\newline
{\tt f\ x\ ==}\newline
{\tt \ \ \ x+c}\newline
{\tt f\ 3}\newline
{\tt \\end\{spadsrc\}}
\end{paste}
\endImportant

There is, in fact, more that one can do
with Axiom commands. In pages elsewhere
in the system, Axiom commands appear next
to button like this \ \MenuDotBitmap{}.
Clicking on this button, one can see the output
for that command. The output has been
pre-computed and is also stored in
Hyperdoc files. This is done using
{\it patch} and {\it paste}.
It is the same mechanism that
is used to alternatively display
Hyperdoc source and interpreted
result in this and other pages.
It is explained \downlink{later on}{HTXAdvPage5}.


\end{scroll}
\beginmenu
\menulink{Next -- Linking to Lisp}{HTXLinkPage4}
\endmenu

\end{page}
```

HTXLinkPage3xPatch1 patch

— htxlinkpage3.ht —

```
\begin{patch}{HTXLinkPage3xPatch1}
\begin{paste}{HTXLinkPage3xPaste1A}{HTXLinkPage3xPatch1A}
\pastebutton{HTXLinkPage3xPaste1A}{Source}
\newline
\axiomcommand{ l:=brace[1,2,3] ; length:=\# l ; m:=[1,2]}
\end{paste}
\end{patch}
```

———-

HTXLinkPage3xPatch1A patch

— htxlinkpage3.ht —

```
\begin{patch}{HTXLinkPage3xPatch1A}
\begin{paste}{HTXLinkPage3xPaste1B}{HTXLinkPage3xPatch1}
\pastebutton{HTXLinkPage3xPaste1B}{Interpret}
\newline
{\tt \\axiomcommand\{ l:=brace[1,2,3] ; length:=\\\# l ; m:=[1,2]\}}
\end{paste}
\end{patch}
```

———-

HTXLinkPage3xPatch2 patch

— htxlinkpage3.ht —

```
\begin{patch}{HTXLinkPage3xPatch2}
\begin{paste}{HTXLinkPage3xPaste2A}{HTXLinkPage3xPatch2A}
\pastebutton{HTXLinkPage3xPaste2A}{Source}
\newline
\axiomcommand{a:=1;d:=4 \bound{v1}}
\newline
\axiomcommand{b:=a+3 \free{v1} \bound{v2}}
\newline
\axiomcommand{c:=b+d \free{v1 v2} \bound{v3}}
\end{paste}
\end{patch}
```

———-

HTXLinkPage3xPatch2A patch

— htxlinkpage3.ht —

```
\begin{patch}{HTXLinkPage3xPatch2A}
\begin{paste}{HTXLinkPage3xPaste2B}{HTXLinkPage3xPatch2}
\pastebutton{HTXLinkPage3xPaste2B}{Interpret}
```

```
\newline
{\tt \\axiomcommand\{a:=1;d:=4 \\bound\{v1\}\}}\newline
{\tt \\newline}\newline
{\tt \\axiomcommand\{b:=a+3 \\free\{v1\} \\bound\{v2\}\}}\newline
{\tt \\newline}\newline
{\tt \\axiomcommand\{c:=b+d \\free\{v1 v2\} \\bound\{v3\}\}}\newline
\end{paste}
\end{patch}
```

———-

HTXLinkPage3xPatch3 patch

— **htxlinkpage3.ht** —

```
\begin{patch}{HTXLinkPage3xPatch3}
\begin{paste}{HTXLinkPage3xPaste3A}{HTXLinkPage3xPatch3A}
\pastebutton{HTXLinkPage3xPaste3A}{Source}
\newline
\begin{spadsrc} [\free{v3} \bound{v4}]
f x ==
   x+c
f 3
\end{spadsrc}
\end{paste}
\end{patch}
```

———-

HTXLinkPage3xPatch3A patch

— **htxlinkpage3.ht** —

```
\begin{patch}{HTXLinkPage3xPatch3A}
\begin{paste}{HTXLinkPage3xPaste3B}{HTXLinkPage3xPatch3}
\pastebutton{HTXLinkPage3xPaste3B}{Interpret}
\newline
{\tt \\begin\{spadsrc\}\ [\\free\{v3\}\ \\bound\{v4\}]}\newline
{\tt f\ x\ ==}\newline
{\tt \ \ \ x+c}\newline
{\tt f\ 3}\newline
{\tt \\end\{spadsrc\}}
\end{paste}
```

```
\end{patch}
```

———-

21.24 htxlinkpage4.ht

Linking to Lisp

⇒ "notitle" (HTXLinkPage5) 21.25 on page 2690

— **htxlinkpage4.ht** —

```
\begin{page}{HTXLinkPage4}{Linking to Lisp}
\centerline{\fbox{{\tt \thispage}}}\newline
\begin{scroll}

Another feature of the Axiom\hspace{2}--Hyperdoc
link is the ability to execute {\it Lisp}
code at a click of a button.
There are two things one can do.

The first is to cause the evaluation
of a {\it Lisp} form and ignore (as far as Hyperdoc
is concerned) its value. The evaluation of the function
might have an effect however on your Axiom session.

The command for this is
\horizontalline
\centerline{ {\tt \\lispcommand\{{\it text}\}\{{\it Lisp form}\}}}
\horizontalline

Here is an example. We will first define a {\it Lisp} function
and then execute it.  Notice that the Hyperdoc
special characters must be escaped (this is on top
of {\it Lisp} escaping conventions).


\beginImportant
\begin{paste}{HTXLinkPage4xPaste1}{HTXLinkPage4xPatch1}
\pastebutton{HTXLinkPage4xPaste1}{Interpret}
\newline
{\tt \\lispcommand\{Definition\}\{(defun HTXTESTFUNCTION ()}\newline
{\tt  (print "Hello from HyperDoc \\\\\\\\ \\\% \\\{ \\\}"))\}} \newline
{\tt \\newline}\newline
{\tt \\lispcommand\{Execution\}\{(HTXTESTFUNCTION)\}} \newline
\end{paste}
\endImportant
```

```
Your command will be executed as soon as
Axiom completes any computation it might be
carrying out.


%\axiomcommand{)lisp (defun f () (pprint "hello"))}
%\lispcommand{f}{(|f|)}


The second thing you can do is quite powerful. It allows you
to delegate to a {\it Lisp} function
the {\it dynamic} creation of a page. This is used
in \Browse{} to present
the Axiom Library in a hypertext form.

The command to use is a lot like the {\tt link} commands
you encountered \downlink{earlier}{HTXLinkPage1} and comes in three
flavors.
\centerline{{\tt \\lispwindowlink\{{\it trigger}\}\{{\it Lisp form}\}}}
\centerline{{\tt \\lispdownlink\{{\it trigger}\}\{{\it Lisp form}\}}}
\centerline{{\tt \\lispmemolink\{{\it trigger}\}\{{\it Lisp form}\}}}

The difference between the three versions is the same as before.
When such a link is activated, Hyperdoc issues the
{\it Lisp form} to Axiom and waits for a full
page definition. An important point to note is that
Hyperdoc does {\it not} use
the value of the {\it Lisp form} but, instead, it
depends on its {\it side-effects}.
What {\it must} happen during evaluation
of the form is enough evaluations of a special {\it Lisp}
function called {\bf issueHT} to define a page.
The argument of {\bf issueHT} is a string
containing Hyperdoc text. Perhaps an example will clarify
matters.

First we will define a {\it Lisp} function that accepts
a string argument and calls {\bf issueHT} a few times.
The strings that are passed to {\bf issueHT} construct
a Hyperdoc page that would just contain our
original argument centered roughly on the page.
Then we write the {\tt \\lisplink} with a call to
the function. Finally, we execute a {\it Lisp}
command that just pretty--prints the function's definition.



\beginImportant
\begin{paste}{HTXLinkPage4xPaste2}{HTXLinkPage4xPatch2}
```

```
\pastebutton{HTXLinkPage4xPaste2}{Interpret}
\newline
{\tt \\lispcommand\{Definition\}\{(defun HTXTESTPAGE (x) (|issueHT|}\newline
{\tt  "\\\\\\\\begin\\\{page\\\}\\\{LispTestPage\\\}\\\{Lisp Test Page\\\}}\newline
{\tt  \\\\\\\\vspace\\\{150\\\} \\\\\\\\centerline\\\{") (|issueHT| x)
(|issueHT|}\newline
{\tt  "\\\} \\\\\\\\end\\\{page\\\}" ) ) \}}\newline
{\tt \\newline}\newline
{\tt \\lispwindowlink\{Link to it\}\{(HTXTESTPAGE "Hi there")\}}\newline
{\tt \\newline}\newline
{\tt \\lispcommand\{Show Lisp definition\}
\{(pprint (symbol-function 'HTXTESTPAGE))\}}\newline
\end{paste}
\endImportant

The {\tt '\\\{'} and {\tt '\\\}'} is required to escape
Hyperdoc's special characters {\tt '\{'} and {\tt  '\}'}.
The {\tt '\\\\\\\\'} has the following rationale.
We need to send to Hyperdoc (from {\it Lisp}) the sequence
{\tt \\begin}. But {\tt '\\'} is a special {\it Lisp}
character. Therefore the {\it Lisp} string must be
{\tt '\\\\begin'}. But to specify this
in Hyperdoc we need to escape the two {\tt '\\'}.
Therefore, we write {\tt '\\\\\\\\begin'}.


The definition of {\tt HTXTESTPAGE} would have been written in {\it Lisp}
as follows.
\begin{verbatim}
(defun HTXTESTPAGE (X)
   (|issueHT|
      "\\begin{page}{LispTestPage}{Lisp Test Page}
\\vspace{200} \\centerline{")
   (|issueHT| X)
   (|issueHT| "} \\end{page}"))
\end{verbatim}



You should not execute {\tt HTXTESTPAGE} in the
{\it Lisp} environment manually. It is meant to
be executed {\it only} in response to a
Hyperdoc request.

Can you pop-up a named page from {\it Lisp} regardless of
user action? Yes --- use {\it Lisp} function {\bf linkToHTPage}
with the page name as a string argument. Click on the
{\tt \\axiomcommand} below. Then, in your Axiom
session, you can repeat it if you like.
```

```
\beginImportant
\begin{paste}{HTXLinkPage4xPaste3}{HTXLinkPage4xPatch3}
\pastebutton{HTXLinkPage4xPaste3}{Interpret}
\newline
{\tt \\axiomcommand\{)lisp (|linkToHTPage| "RootPage")\}}
\end{paste}
\endImportant

You can also pop-up a {\it dynamic} page regardless of user action.
To do this, make sure you evaluate the {\it Lisp form}
{\bf (|startHTPage| 50)} before using {\bf issueHT}.
The example below requires the {\tt HTXTESTPAGE} function
to be defined in {\it Lisp} so you should make sure
you have executed the command above that defines it.

\beginImportant
\begin{paste}{HTXLinkPage4xPaste4}{HTXLinkPage4xPatch4}
\pastebutton{HTXLinkPage4xPaste4}{Interpret}
\newline
{\tt \\axiomcommand\{
)lisp (progn (|startHTPage| 50)(HTXTESTPAGE "Immediately"))\}}
\end{paste}
\endImportant

Now, the most important use of this facility
so far has been in the \Browse{} and Basic Commands components of
Hyperdoc. Instead of giving you details of the various
\Browse{} {\it Lisp} functions, a few macros are defined in
\centerline{{\bf \$AXIOM/doc/util.ht}}

The most important defined macros are
\beginImportant
\table{
{ {\tt \\axiomType\{{\it constructor}\}} }
{ {\tt \\axiomOp\{{\it operation}\}} }
{ {\tt \\axiomOpFrom\{{\it operation }\}\{{\it constructor}\}}}
}
\endImportant

Here are some examples of their use.
\beginImportant
\begin{paste}{HTXLinkPage4xPaste5}{HTXLinkPage4xPatch5}
\pastebutton{HTXLinkPage4xPaste5}{Interpret}
\newline
{\tt \\axiomType\{Expression Integer\}}\newline
{\tt \newline}\newline
{\tt \\axiomType\{Expression\}}\newline
{\tt \newline}\newline
{\tt \\axiomType\{EXPR\}}\newline
{\tt \newline}\newline
```

```
{\tt \\axiomOp\{reduce\}}\newline
{\tt \newline}\newline
{\tt \\axiomOp\{as*\}}\newline
{\tt \newline}\newline
{\tt \\axiomOpFrom\{reduce\}\{Expression\}}\newline
\end{paste}
\endImportant

The macro {\tt \\axiomType} brings up the \Browse{}
constructor page for the constructor specified.
You can specify a full name, or an abbreviation
or just the top level name.
The macro {\tt \\axiomOp} brings up a list of operations
matching the argument.
The macro {\tt \\axiomOpFrom} shows documentation
about the specified operation whose origin is
constructor. No wildcard in the operation name
or type abbreviation is
allowed here.  You should also specify just the top level type.

\end{scroll}
\beginmenu
\menulink{Next -- Linking to Unix}{HTXLinkPage5}
\endmenu

\end{page}
```

———-

HTXLinkPage4xPatch1 patch

— htxlinkpage4.ht —

```
\begin{patch}{HTXLinkPage4xPatch1}
\begin{paste}{HTXLinkPage4xPaste1A}{HTXLinkPage4xPatch1A}
\pastebutton{HTXLinkPage4xPaste1A}{Source}
\newline
\lispcommand{Definition}{(defun HTXTESTFUNCTION ()
(print "Hello from HyperDoc \\\\ \% \{ \}"))}
\newline
\lispcommand{Execution}{(HTXTESTFUNCTION)}
\end{paste}
\end{patch}
```

———-

HTXLinkPage4xPatch1A patch

— **htxlinkpage4.ht** —

```
\begin{patch}{HTXLinkPage4xPatch1A}
\begin{paste}{HTXLinkPage4xPaste1B}{HTXLinkPage4xPatch1}
\pastebutton{HTXLinkPage4xPaste1B}{Interpret}
\newline
{\tt \\lispcommand\{Definition\}\{
(defun HTXTESTFUNCTION ()
 (print "Hello from HyperDoc \\\\\\\\ \\\% \\\{ \\\}"))\}} \newline
{\tt \\newline}\newline
{\tt \\lispcommand\{Execution\}\{(HTXTESTFUNCTION)\}} \newline
\end{paste}
\end{patch}
```

————

HTXLinkPage4xPatch2 patch

⇒ "Definition" (LispFunctions) 3.71 on page 952

— **htxlinkpage4.ht** —

```
\begin{patch}{HTXLinkPage4xPatch2}
\begin{paste}{HTXLinkPage4xPaste2A}{HTXLinkPage4xPatch2A}
\pastebutton{HTXLinkPage4xPaste2A}{Source}
\newline
\lispcommand{Definition}{(defun HTXTESTPAGE (x) (|issueHT|
"\\\\begin\{page\}\{LispTestPage\}\{Lisp Test Page\}
\\\\vspace\{150\} \\\\centerline\{") (|issueHT| x) (|issueHT|
"\} \\\\end\{page\}" ) ) }
\newline
\lispwindowlink{Link to it}{(HTXTESTPAGE "Hi there")}
\newline
\lispcommand{Show Lisp definition}{(pprint (symbol-function 'HTXTESTPAGE))}
\end{paste}
\end{patch}
```

————

HTXLinkPage4xPatch2A patch

— **htxlinkpage4.ht** —

```
\begin{patch}{HTXLinkPage4xPatch2A}
\begin{paste}{HTXLinkPage4xPaste2B}{HTXLinkPage4xPatch2}
\pastebutton{HTXLinkPage4xPaste2B}{Interpret}
\newline
{\tt \\lispcommand\{Definition\}\{(defun HTXTESTPAGE (x)
(|issueHT|}\newline
{\tt  "\\\\\\\\begin\\\{page\\\}\\\{LispTestPage\\\}
\\\{Lisp Test Page\\\}}\newline
{\tt  \\\\\\\\vspace\\\{150\\\} \\\\\\\\centerline\\\{")
(|issueHT| x) (|issueHT|}\newline
{\tt  "\\\} \\\\\\\\end\\\{page\\\}" ) ) \}}\newline
{\tt \\newline}\newline
{\tt \\lispwindowlink\{Link to it\}\{(HTXTESTPAGE "Hi there")\}}
\newline
{\tt \\newline}\newline
{\tt \\lispcommand\{Show Lisp definition\}
\{(pprint (symbol-function 'HTXTESTPAGE))\}}\newline
\end{paste}
\end{patch}
```

———-

HTXLinkPage4xPatch3 patch

— htxlinkpage4.ht —

```
\begin{patch}{HTXLinkPage4xPatch3}
\begin{paste}{HTXLinkPage4xPaste3A}{HTXLinkPage4xPatch3A}
\pastebutton{HTXLinkPage4xPaste3A}{Source}
\newline
\axiomcommand{)lisp (|linkToHTPage| "RootPage")}
\end{paste}
\end{patch}
```

———-

HTXLinkPage4xPatch3A patch

— htxlinkpage4.ht —

```
\begin{patch}{HTXLinkPage4xPatch3A}
\begin{paste}{HTXLinkPage4xPaste3B}{HTXLinkPage4xPatch3}
\pastebutton{HTXLinkPage4xPaste3B}{Interpret}
```

```
\newline
{\tt \\axiomcommand\{)lisp (|linkToHTPage| "RootPage")\}}
\end{paste}
\end{patch}
```

———-

HTXLinkPage4xPatch4 patch

— htxlinkpage4.ht —

```
\begin{patch}{HTXLinkPage4xPatch4}
\begin{paste}{HTXLinkPage4xPaste4A}{HTXLinkPage4xPatch4A}
\pastebutton{HTXLinkPage4xPaste4A}{Source}
\newline
\axiomcommand{)lisp (progn (|startHTPage| 50)(HTXTESTPAGE "Immediately"))}
\end{paste}
\end{patch}
```

———-

HTXLinkPage4xPatch4A patch

— htxlinkpage4.ht —

```
\begin{patch}{HTXLinkPage4xPatch4A}
\begin{paste}{HTXLinkPage4xPaste4B}{HTXLinkPage4xPatch4}
\pastebutton{HTXLinkPage4xPaste4B}{Interpret}
\newline
{\tt \\axiomcommand\{
)lisp (progn (|startHTPage| 50)(HTXTESTPAGE "Immediately"))\}}
\end{paste}
\end{patch}
```

———-

HTXLinkPage4xPatch5 patch

— htxlinkpage4.ht —

```
\begin{patch}{HTXLinkPage4xPatch5}
\begin{paste}{HTXLinkPage4xPaste5A}{HTXLinkPage4xPatch5A}
\pastebutton{HTXLinkPage4xPaste5A}{Source}
\newline
\axiomType{Expression Integer}
\newline
\axiomType{Expression}
\newline
\axiomType{EXPR}
\newline
\axiomOp{reduce}
\newline
\axiomOp{as*}
\newline
\axiomOpFrom{reduce}{Expression}
\end{paste}
\end{patch}
```

———-

HTXLinkPage4xPatch5A patch

— htxlinkpage4.ht —

```
\begin{patch}{HTXLinkPage4xPatch5A}
\begin{paste}{HTXLinkPage4xPaste5B}{HTXLinkPage4xPatch5}
\pastebutton{HTXLinkPage4xPaste5B}{Interpret}
\newline
{\tt \\axiomType\{Expression Integer\}}\newline
{\tt \newline}\newline
{\tt \\axiomType\{Expression\}}\newline
{\tt \newline}\newline
{\tt \\axiomType\{EXPR\}}\newline
{\tt \newline}\newline
{\tt \\axiomOp\{reduce\}}\newline
{\tt \newline}\newline
{\tt \\axiomOp\{as*\}}\newline
{\tt \newline}\newline
{\tt \\axiomOpFrom\{reduce\}\{Expression\}}\newline
\end{paste}
\end{patch}
```

———-

21.25 htxlinkpage5.ht

Linking to Unix

— htxlinkpage5.ht —

```
\begin{page}{HTXLinkPage5}{Linking to Unix}
\centerline{\fbox{{\tt \thispage}}}\newline
\begin{scroll}

Let us conclude the tour of Hyperdoc
actions that can be triggered with a click of a button
with two more facilities. These are
\beginImportant
\table{
{ {\tt \\unixcommand\{{\it trigger text}\}\{{\it unix command}\}}}
{ {\tt \\unixlink\{{\it trigger text}\}\{{\it unix command}\}}}
}
\endImportant


The first one, {\tt \\unixcommand}, is very much like
{\tt \\axiomcommand} and {\tt \\lispcommand}.
The trigger text becomes an active area. Clicking on it
will force Hyperdoc to pass the second argument
to the system as a shell command to be executed.
The shell used is {\bf /bin/sh}.
Hyperdoc ignores the output of the command.


\beginImportant
\begin{paste}{HTXLinkPage5xPaste1}{HTXLinkPage5xPatch1}
\pastebutton{HTXLinkPage5xPaste1}{Interpret}
\newline
{\tt \\unixcommand\{List \\\$HOME directory\}\{ls \\\$HOME\}}\newline
\end{paste}
\endImportant

The {\tt \\unixlink} command delegates to a another
program the creation of a dynamic page. When the trigger
text is activated, Hyperdoc will invoke the command
specified in the second argument. It will then start reading
the {\it standard output} of the command until
a complete page has been received. It is important that
a single page and nothing more is written by the command.
This command is essentially a {\tt \\downlink}, i.e.
the new page replaces the current page in the window.
There aren't any other flavours of {\tt \\unixlink}.
```

```
A trivial example is to use {\bf cat} on a Hyperdoc
file known to contain just one page.

\beginImportant
\begin{paste}{HTXLinkPage5xPaste2}{HTXLinkPage5xPatch2}
\pastebutton{HTXLinkPage5xPaste2}{Interpret}
\newline
{\tt \\unixlink\{Some file\}} \newline
{\tt \{cat\\ \\env\{AXIOM\}/doc/HTXplay.ht\}}
\end{paste}
\endImportant


Two things to notice in the second argument of
{\tt \\unixlink}: You must use a {\it hard space}
{\tt '\\\ '} to preserve the spacing in the command.
Also, the {\tt \\env} command allows you to use
an environment variable in Hyperdoc text.

With a little ingenuity (and maybe some shell and {\bf awk} scripts !)
, one can use these
facilities to create, say, a point-and-click
directory viewer which allows you to edit
a file by clicking on its name.

\end{scroll}
\beginmenu
\menulink{Next -- How to use your pages with Hyperdoc}{HTXLinkPage6}
\endmenu

\end{page}
```

HTXLinkPage5xPatch1 patch

— htxlinkpage5.ht —

```
\begin{patch}{HTXLinkPage5xPatch1}
\begin{paste}{HTXLinkPage5xPaste1A}{HTXLinkPage5xPatch1A}
\pastebutton{HTXLinkPage5xPaste1A}{Source}
\newline
\unixcommand{List \$HOME directory}{ls \$HOME}
\end{paste}
\end{patch}
```

———-

HTXLinkPage5xPatch1A patch

— htxlinkpage5.ht —

```
\begin{patch}{HTXLinkPage5xPatch1A}
\begin{paste}{HTXLinkPage5xPaste1B}{HTXLinkPage5xPatch1}
\pastebutton{HTXLinkPage5xPaste1B}{Interpret}
\newline
{\tt \\unixcommand\{List \\\$HOME directory\}\{ls \\\$HOME\}}\newline
\end{paste}
\end{patch}
```

———-

HTXLinkPage5xPatch2 patch

— htxlinkpage5.ht —

```
\begin{patch}{HTXLinkPage5xPatch2}
\begin{paste}{HTXLinkPage5xPaste2A}{HTXLinkPage5xPatch2A}
\pastebutton{HTXLinkPage5xPaste2A}{Source}
\newline
\unixlink{Some file}
{cat\ \env{AXIOM}/doc/HTXplay.ht}
\end{paste}
\end{patch}
```

———-

HTXLinkPage5xPatch2A patch

— htxlinkpage5.ht —

```
\begin{patch}{HTXLinkPage5xPatch2A}
\begin{paste}{HTXLinkPage5xPaste2B}{HTXLinkPage5xPatch2}
\pastebutton{HTXLinkPage5xPaste2B}{Interpret}
\newline
{\tt \\unixlink\{Some file\}} \newline
{\tt \{cat\\ \\env\{AXIOM\}/doc/HTXplay.ht\}}
```

```
\end{paste}
\end{patch}
```

———-

21.26 htxlinkpage6.ht

How to use your pages with Hyperdoc

— **htxlinkpage6.ht** —

```
\begin{page}{HTXLinkPage6}{How to use your pages with Hyperdoc}
\centerline{\fbox{{\tt \thispage}}}\newline
\begin{scroll}

Let us say that you have written a few Hyperdoc
pages and you would like to incorporate them in the system.
Here is what you should do.

Put all your files in some directory and make sure that
they all have the {\bf .ht} extension.

You will need a way of "hooking" into a system--defined
Hyperdoc page. The proper way to do this is to use
the {\tt \\localinfo} macro. The Axiom system
Hyperdoc page database includes, as it should,
a {\tt RootPage}. This is the page that first comes up
when you start Hyperdoc. This page contains
a line like this.
\beginImportant
\newline
{\tt \\localinfo}
\endImportant

This macro is defined in
\centerline{ {\bf \env{AXIOM}/doc/util.ht}}
to be (see \downlink{Macros}{HTXAdvPage3} to learn how to define macros):
\beginImportant
\newline
{\tt \\newcommand\{\\localinfo\}\{\}}
\endImportant
which is an empty definition (the second argument of {\tt \\newcommand}).
The idea then is that you {\it override} this definition of the macro
with your own.
To do that, include a definition like the following in one (possibly the
one that contains your top--level page) of your files. You can
```

```
put this definition in its own file if you like.
\beginImportant
\newline
{\tt \\newcommand\{\\localinfo\}
\{\\menuwindowlink\{{\it active text}\}} \newline
{\tt \{{\it page name}\} \\tab\{16\}{\it short description}\}}
\endImportant

If you have a look at the initial Hyperdoc page, you will probably be
able to decipher what this does. The macro {\tt \\menuwindowlink} is
defined (again in {\bf util.ht}) and is responsible for putting the
little square to the left of the active area.  Specify a word or two
for {\it active text}. That will become the trigger of the {\tt
\\link}. Specify the page name of your top--level page in {\it page
name}. Finally, you can give a comment about the topic under {\it
short description}. That will appear to the right of the {\it active
text}.

The next thing you need to do is to create a {\it local database}
for your files. You will use the {\bf \env{AXIOM}/bin/htadd} program.
This program will create an {\bf ht.db} file that summarises your
definitions and acts as an index. Let us present an example
of its use. Suppose you have two files {\bf user1.ht} and {\bf user2.ht}
in directory {\bf /u/sugar/Hyperdoc}. You should create the {\bf ht.db}
in that same directory. To create the {\bf ht.db} file you issue to
the unix shell:
\beginImportant
\newline
{\tt htadd -f /u/sugar/Hyperdoc
/u/sugar/Hyperdoc/user1.ht /u/sugar/Hyperdoc/user2.ht}
\centerline{or ,if you are already in /u/sugar/Hyperdoc}
{\tt htadd -l ./user1.ht ./user2.ht}
\endImportant


The options and conventions for {\bf htadd} will be explained below.
To start Hyperdoc with your own pages, you now need to tell
it where to search for {\bf ht.db} files and Hyperdoc {\bf .ht}
files. To do this, define the shell environment variable
{\bf HTPATH}. The value should be a colon {\tt ':'} separated
list of directory full pathnames.
The order of the directories is respected with earlier entries overriding
later ones. Since we want all the Axiom pages but need to override the
{\tt \\localinfo} macro, we should use the value
\centerline{{\bf /u/sugar/Hyperdoc:\env{AXIOM}/doc}}
The way that you define environment variables depends on the shell
you are using. In the {\bf /bin/csh}, it would be
\newline
{\bf setenv HTPATH /u/sugar/Hyperdoc:\env{AXIOM}{}/doc{}/hypertex{}/pages}
```

```
\beginImportant
\begin{paste}{HTXLinkPage6xPaste1}{HTXLinkPage6xPatch1}
\pastebutton{HTXLinkPage6xPaste1}{Options for {\bf htadd}}
\newline
\end{paste}
\endImportant


\beginImportant
\begin{paste}{HTXLinkPage6xPaste2}{HTXLinkPage6xPatch2}
\pastebutton{HTXLinkPage6xPaste2}{Where does Hyperdoc look for files}
\newline
\end{paste}
\endImportant



\end{scroll}
\beginmenu
\menulink{Back to Actions menu}{HTXLinkTopPage}
\endmenu

\end{page}
```

———-

HTXLinkPage6xPatch1 patch

— htxlinkpage6.ht —

```
\begin{patch}{HTXLinkPage6xPatch1}
\begin{paste}{HTXLinkPage6xPaste1A}{HTXLinkPage6xPatch1A}
\pastebutton{HTXLinkPage6xPaste1A}{Hide}
\newline
Name:

{\tt htadd - create or modify a Hyperdoc database}
\vspace{}
\newline
Syntax:

{\tt htadd [ -l | -s | -f\ }{\it path}{\tt ]
[ -d | -n ]\ }{\it filename ...}
\vspace{}
\newline
```

```
Options:\indentrel{4}\newline
{\tt -l}\tab{8}\indentrel{8}
build {\bf ht.db} database in current working directory.
This is the default behaviour if no {\tt -l}, {\tt -s} or {\tt -f}
is specified.

\indentrel{-8}\newline
{\tt -s}\tab{8}\indentrel{8}
build {\bf ht.db} database in {\it system} directory. The
system directory is built as follows. If the {\tt AXIOM}
variable is defined, the {\bf \$AXIOM/doc} directory
is used. If {\tt AXIOM} is not defined, the
{\bf /usr/local/axiom/doc} directory is used.


\indentrel{-8}\newline
{\tt -f\ }{\it path}\newline\tab{8}\indentrel{8}
build {\bf ht.db} database in specified {\it path}.

\indentrel{-8}\newline
{\tt -d}\tab{8}\indentrel{8}
delete the entries in the specified files from {\bf ht.db}.

\indentrel{-8}\newline
{\tt -n}\tab{8}\indentrel{8}
delete {\bf ht.db} and create a new one using only the files
specified.

If none of {\tt -n} and {\tt -d} is specified, the {\bf ht.db}
is updated with the entries in the file specified.


\indentrel{-8}
\indentrel{-4}
\vspace{}\newline
Filename interpretation :
\indentrel{12}\newline
A full pathname (i.e. anything that has a {\tt '/'} in it)
will be taken do be a completely specified file.
Otherwise, the following interpretation will occur:
If the {\tt HTPATH} variable is defined, the directories
specified in it will be tried in order. If {\tt HTPATH}
is not defined, then, if {\tt AXIOM} is defined, the
{\bf \$AXIOM/doc} will be tried, else
the file will be deemed missing and {\bf htadd} will fail.
\indentrel{-12}
\end{paste}
\end{patch}
```

———-

HTXLinkPage6xPatch1A patch

— htxlinkpage6.ht —

```
\begin{patch}{HTXLinkPage6xPatch1A}
\begin{paste}{HTXLinkPage6xPaste1B}{HTXLinkPage6xPatch1}
\pastebutton{HTXLinkPage6xPaste1B}{Options for {\bf htadd}}
\newline
\end{paste}
\end{patch}
```

———-

HTXLinkPage6xPatch2 patch

— htxlinkpage6.ht —

```
\begin{patch}{HTXLinkPage6xPatch2}
\begin{paste}{HTXLinkPage6xPaste2A}{HTXLinkPage6xPatch2A}
\pastebutton{HTXLinkPage6xPaste2A}{Hide}
\indentrel{12}\newline

The Hyperdoc program is \centerline{{\bf \env{AXIOM}/lib/hypertex}} If
{\tt AXIOM} is defined and {\tt HTPATH} is not (this is the case when
Axiom starts Hyperdoc) Hyperdoc will look in
\centerline{{\bf \env{AXIOM}/doc}}
for the {\bf ht.db} file and all
Hyperdoc pages.  If {\tt HTPATH} is defined, it is assumed that it
alone points to the directories to be searched (the above default will
NOT be searched unless explicitly specified in {\tt HTPATH}).  For
each directory in {\tt HTPATH}, the {\bf ht.db} file, if there, will
be read.  Each file listed in {\bf ht.db} will then be searched for in
the complete sequence of directories in {\tt HTPATH}. Note that the
{\bf ht.db} does not keep full pathnames of files.  If a {\it page},
{\it macro} or {\it patch} (specified in some {\bf ht.db}) happens to
be (in a file) in more than one of the directories specified in
{\tt HTPATH}, Hyperdoc will print a warning and explain which version in
which file is ignored. Generally, earlier directories in {\tt HTPATH}
are preferred over later ones.

\indentrel{-12}\newline
\end{paste}
```

```
\end{patch}
```

———-

HTXLinkPage6xPatch2A patch

— **htxlinkpage6.ht** —

```
\begin{patch}{HTXLinkPage6xPatch2A}
\begin{paste}{HTXLinkPage6xPaste2B}{HTXLinkPage6xPatch2}
\pastebutton{HTXLinkPage6xPaste2B}{Where does Hyperdoc look for files}}
\newline
\end{paste}
\end{patch}
```

———-

21.27 htxlinktoppage.ht

Actions in Hyperdoc

⇒ "notitle" (HTXLinkPage1) 21.21 on page 2669
⇒ "notitle" (HTXLinkPage2) 21.22 on page 2672
⇒ "notitle" (HTXLinkPage3) 21.23 on page 2675
⇒ "notitle" (HTXLinkPage4) 21.24 on page 2681
⇒ "notitle" (HTXLinkPage5) 21.25 on page 2690
⇒ "notitle" (HTXLinkPage6) 21.26 on page 2693

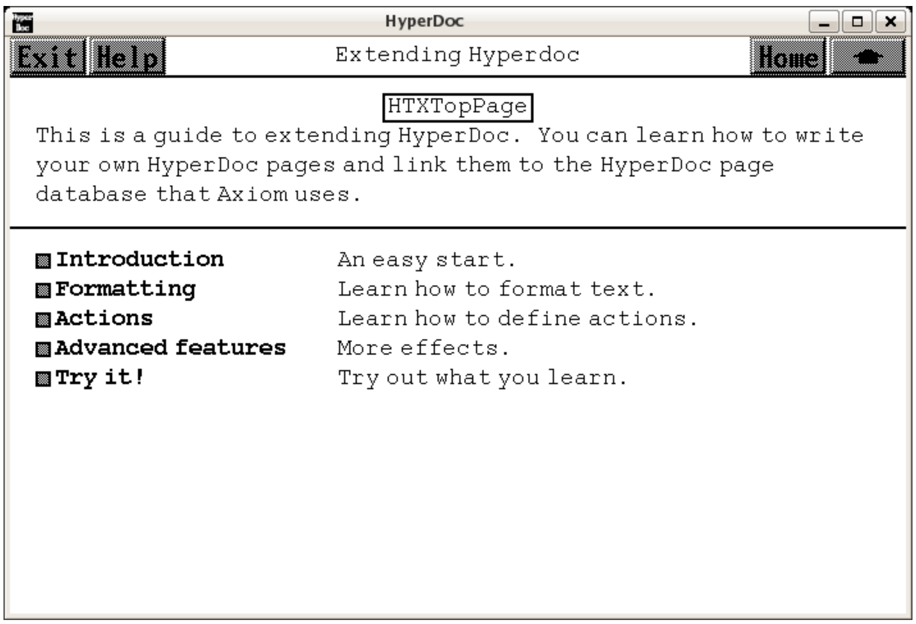
— **htxlinktoppage.ht** —

```
\begin{page}{HTXLinkTopPage}{Actions in Hyperdoc}
\centerline{\fbox{{\tt \thispage}}}\newline
Hyperdoc offers various types of hypertext links.
You can learn about these facilities by clicking on the topics below.
\begin{scroll}
\beginmenu
\menudownlink{Linking to a named page}{HTXLinkPage1}
\menudownlink{Standard pages}{HTXLinkPage2}
\menudownlink{Active Axiom commands}{HTXLinkPage3}
\menudownlink{Linking to Lisp}{HTXLinkPage4}
\menudownlink{Linking to Unix}{HTXLinkPage5}
\menudownlink{How to use your pages with Hyperdoc}{HTXLinkPage6}
\endmenu
```

```
\end{scroll}
\end{page}
```

21.28 htxtoppage.ht

Extending Hyperdoc



⇐ "Reference" (TopReferencePage) 3.1 on page 123
⇒ "Introduction" (HTXIntroTopPage) 21.20 on page 2668
⇒ "Formatting" (HTXFormatTopPage) 21.16 on page 2662
⇒ "Actions" (HTXLinkTopPage) 21.27 on page 2698
⇒ "Advanced features" (HTXAdvTopPage) 21.7 on page 2630
⇒ "Try it!" (HTXTryPage) 21.29 on page 2700

— **htxtoppage.ht** —

```
\begin{page}{HTXTopPage}{Extending Hyperdoc}
\centerline{\fbox{{\tt \thispage}}}\newline
This is a guide to extending Hyperdoc. You can learn
how to write your own Hyperdoc pages and link them to the
Hyperdoc page database that Axiom uses.
\begin{scroll}
\beginmenu
```

```
\menumemolink{Introduction}{HTXIntroTopPage}
\tab{20} An easy start.
\menumemolink{Formatting}{HTXFormatTopPage}
\tab{20} Learn how to format text.
\menumemolink{Actions}{HTXLinkTopPage}
\tab{20} Learn how to define actions.
\menumemolink{Advanced features}{HTXAdvTopPage}
\tab{20} More effects.
\menuwindowlink{Try it!}{HTXTryPage}
\tab{20} Try out what you learn.
\endmenu
\end{scroll}
\end{page}
```

———

21.29 htxtrypage.ht

Try out Hyperdoc

— htxtrypage.ht —

```
\begin{page}{HTXTryPage}{Try out Hyperdoc}
\centerline{\fbox{{\tt \thispage}}}\newline

This page allows you to quickly experiment with Hyperdoc.
It is a good idea to keep it handy as you learn about various commands.

\beginscroll

We are going to use here the Hyperdoc facilities that allow
us to communicate with external programs and files. For more
information see \downlink{later on}{HTXLinkPage5}.
\beginmenu
\item\menuitemstyle{
In order to use the buttons at the bottom of this page, you must
first specify a name for the file you are going to use to hold
Hyperdoc commands. Edit the input area below to change the
name of the file.}
\item\menuitemstyle{
If the file you specified does not yet exist, click on the
{\bf Initialize} button below. This action will fill the file
with the minimum of Hyperdoc commands necessary to define a page.}
\item\menuitemstyle{
If you want to edit the file, just click on the {\bf Edit} button.
This action will pop up a window, and invoke the {\it vi}
```

```
editor on the file. Alternatively, use an editor of your choice.}
\item\menuitemstyle{
Once you have finished making the changes to the file, update it and
click on the {\bf Link} button. Hyperdoc will then read
the file, interpret it as a new page, and display the page on
this window. If you change the file and want to display it again,
just get back to this page and click on {\bf Link} again. }
\endmenu
\endscroll
\beginmenu
{\it Filename: }{\inputstring{filename}{40}{\env{HOME}/HTXplay.ht}}
\menuunixcommand{Initialize}
{cp\space{1}\env{AXIOM}/doc/HTXplay.ht
\stringvalue{filename}} \tab{20} Get a fresh copy from the system.
\menuunixcommand{Edit}{xterm -T "\stringvalue{filename}" -e vi
\stringvalue{filename}} \tab{20} Edit the file.
\menuunixwindow{Link}{cat \space{1}\stringvalue{filename}}
\tab{20} Link to the page defined in the file.
\endmenu
{\it Important : The file must contain
one and only one page definition and must not contain any
macro or patch definitions.}
\end{page}
```

———-

Chapter 22

NAG Library Routines

22.1 nagaux.ht

NAG On-line Documentation

— nagaux.ht —

```
\begin{page}{manpageXXonline}{NAG On-line Documentation}
\beginscroll
\begin{verbatim}


     DOC INTRO(3NAG)   Foundation Library (12/10/92)   DOC INTRO(3NAG)



                    Introduction to NAG On-Line Documentation


          The on-line documentation for the NAG Foundation Library has been
          generated automatically from the same base material used to
          create the printed Reference Manual. To make the documentation
          readable on the widest range of machines, only the basic set of
          ascii characters has been used.

          Certain mathematical symbols have been constructed using plain
          ascii characters:

          integral signs              /
                                      |
```

```
                              /

summation signs               --
                              >
                              --

square root signs                ----
                                /
                              \/
```

Large brackets are constructed using vertical stacks of the equivalent ascii character:

```
         (  )  [  ]  {  }  |
         (  )  [  ]  {  }  |
         (  )  [  ]  {  }  |
```

Fractions are represented as:

```
          a
         ---
         x+1
```

Greek letters are represented by their names enclosed in round brackets:

```
        (alpha)  (beta)  (gamma) .....

        (Alpha)  (Beta)  (Gamma) .....
```

Some characters are accented using:

```
        ^     ~
        X     X     X
```

Other mathematical symbols are represented as follows:

```
       *           times

       <=>         left-right arrow

       <-          left arrow

       ~           similar to
```

```
                 ~=          similar or equal to

                 ==          equivalent to

                 >=          greater than or equal to

                 <=          less than or equal to

                 >>          much greater than

                 <<          much less than

                 >~          greater than or similar to

                 /=          not equal to

                 dd          partial derivative

                 +-          plus or minus

                 (nabla)     Nabla

\end{verbatim}
\endscroll
\end{page}
```

———-

NAG Documentation: summary

— nagaux.ht —

```
\begin{page}{manpageXXsummary}{NAG Documentation: summary}
\beginscroll
\begin{verbatim}



     SUMMARY(3NAG)     Foundation Library (12/10/92)     SUMMARY(3NAG)



          Introduction                                    List of Routines
          List of Routines

          The NAG Foundation Library contains three categories of routines
          which can be called by users. They are listed separately in the
```

```
three sections below.

Fully Documented Routines
      254 routines, for each of which an individual routine
      document is provided. These are regarded as the primary
      contents of the Foundation Library.

Fundamental Support Routines
      83 comparatively simple routines which are documented in
      compact form in the relevant Chapter Introductions (F06,
      X01, X02).

Routines from the NAG Fortran Library
      An additional 167 routines from the NAG Fortran Library,
      which are used as auxiliaries in the Foundation Library.
      They are not documented in this publication, but can be
      called if you are already familiar with their use in the
      Fortran Library. Only their names are given here.


Note: all the routines in the above categories have names ending
in 'F'. Occasionally this publication may refer to routines whose
names end in some other letter (e.g. 'Z', 'Y', 'X'). These are
auxiliary routines whose names may be passed as parameters to a
Foundation Library routine; you only need to know their names,
not how to call them directly.

Fully Documented Routines

The Foundation Library contains 254 user-callable routines, for
each of which an individual routine document is provided, in the
following chapters:

C02 -- Zeros of Polynomials

C02AFF  All zeros of complex polynomial, modified Laguerre method

C02AGF  All zeros of real polynomial, modified Laguerre method

C05 -- Roots of One or More Transcendental Equations

C05ADF  Zero of continuous function in given interval, Bus and
        Dekker algorithm

C05NBF  Solution of system of nonlinear equations using function
        values only

C05PBF  Solution of system of nonlinear equations using 1st
        derivatives
```

```
C05ZAF  Check user's routine for calculating 1st derivatives

C06 -- Summation of Series

C06EAF  Single 1-D real discrete Fourier transform, no extra
        workspace

C06EBF  Single 1-D Hermitian discrete Fourier transform, no extra
        workspace

C06ECF  Single 1-D complex discrete Fourier transform, no extra
        workspace

C06EKF  Circular convolution or correlation of two real vectors,
        no extra workspace

C06FPF  Multiple 1-D real discrete Fourier transforms

C06FQF  Multiple 1-D Hermitian discrete Fourier transforms

C06FRF  Multiple 1-D complex discrete Fourier transforms

C06FUF  2-D complex discrete Fourier transform

C06GBF  Complex conjugate of Hermitian sequence

C06GCF  Complex conjugate of complex sequence

C06GQF  Complex conjugate of multiple Hermitian sequences

C06GSF  Convert Hermitian sequences to general complex sequences

D01 -- Quadrature

D01AJF  1-D quadrature, adaptive, finite interval, strategy due
        to Piessens and de Doncker, allowing for badly-behaved
        integrands

D01AKF  1-D quadrature, adaptive, finite interval, method
        suitable for oscillating functions

D01ALF  1-D quadrature, adaptive, finite interval, allowing for
        singularities at user-specified break-points

D01AMF  1-D quadrature, adaptive, infinite or semi-infinite
        interval

D01ANF  1-D quadrature, adaptive, finite interval, weight
        function cos((omega)x) or sin((omega)x)
```

```
D01APF  1-D quadrature, adaptive, finite interval, weight
        function with end-point singularities of algebraico-
        logarithmic type

D01AQF  1-D quadrature, adaptive, finite interval, weight
        function 1/(x-c), Cauchy principal value (Hilbert
        transform)

D01ASF  1-D quadrature, adaptive, semi-infinite interval, weight
        function cos((omega)x) or sin((omega)x)

D01BBF  Weights and abscissae for Gaussian quadrature rules

D01FCF  Multi-dimensional adaptive quadrature over hyper-
        rectangle

D01GAF  1-D quadrature, integration of function defined by data
        values, Gill-Miller method

D01GBF  Multi-dimensional quadrature over hyper-rectangle, Monte
        Carlo method

D02 -- Ordinary Differential Equations

D02BBF  ODEs, IVP, Runge-Kutta-Merson method, over a range,
        intermediate output

D02BHF  ODEs, IVP, Runge-Kutta-Merson method, until function of
        solution is zero

D02CJF  ODEs, IVP, Adams method, until function of solution is
        zero, intermediate output

D02EJF  ODEs, stiff IVP, BDF method, until function of solution
        is zero, intermediate output

D02GAF  ODEs, boundary value problem, finite difference technique
        with deferred correction, simple nonlinear problem

D02GBF  ODEs, boundary value problem, finite difference technique
        with deferred correction, general linear problem

D02KEF  2nd order Sturm-Liouville problem, regular/singular
        system, finite/infinite range, eigenvalue and
        eigenfunction, user-specified break-points

D02RAF  ODEs, general nonlinear boundary value problem, finite
        difference technique with deferred correction,
        continuation facility
```

```
D03 -- Partial Differential Equations

D03EDF  Elliptic PDE, solution of finite difference equations by
        a multigrid technique

D03EEF  Discretize a 2nd order elliptic PDE on a rectangle

D03FAF  Elliptic PDE, Helmholtz equation, 3-D Cartesian co-
        ordinates

E01 -- Interpolation

E01BAF  Interpolating functions, cubic spline interpolant, one
        variable

E01BEF  Interpolating functions, monotonicity-preserving,
        piecewise cubic Hermite, one variable

E01BFF  Interpolated values, interpolant computed by E01BEF,
        function only, one variable

E01BGF  Interpolated values, interpolant computed by E01BEF,
        function and 1st derivative, one variable

E01BHF  Interpolated values, interpolant computed by E01BEF,
        definite integral, one variable

E01DAF  Interpolating functions, fitting bicubic spline, data on
        rectangular grid

E01SAF  Interpolating functions, method of Renka and Cline, two
        variables

E01SBF  Interpolated values, evaluate interpolant computed by
        E01SAF, two variables

E01SEF  Interpolating functions, modified Shepard's method, two
        variables

E01SFF  Interpolated values, evaluate interpolant computed by
        E01SEF, two variables

E02 -- Curve and Surface Fitting

E02ADF  Least-squares curve fit, by polynomials, arbitrary data
        points

E02AEF  Evaluation of fitted polynomial in one variable from
        Chebyshev series form (simplified parameter list)
```

```
E02AGF  Least-squares polynomial fit, values and derivatives may
        be constrained, arbitrary data points,

E02AHF  Derivative of fitted polynomial in Chebyshev series form

E02AJF  Integral of fitted polynomial in Chebyshev series form

E02AKF  Evaluation of fitted polynomial in one variable, from
        Chebyshev series form

E02BAF  Least-squares curve cubic spline fit (including
        interpolation)

E02BBF  Evaluation of fitted cubic spline, function only

E02BCF  Evaluation of fitted cubic spline, function and
        derivatives

E02BDF  Evaluation of fitted cubic spline, definite integral

E02BEF  Least-squares cubic spline curve fit, automatic knot
        placement

E02DAF  Least-squares surface fit, bicubic splines

E02DCF  Least-squares surface fit by bicubic splines with
        automatic knot placement, data on rectangular grid

E02DDF  Least-squares surface fit by bicubic splines with
        automatic knot placement, scattered data

E02DEF  Evaluation of a fitted bicubic spline at a vector of
        points

E02DFF  Evaluation of a fitted bicubic spline at a mesh of points

E02GAF  L -approximation by general linear function
         1

E02ZAF  Sort 2-D data into panels for fitting bicubic splines

E04 -- Minimizing or Maximizing a Function

E04DGF  Unconstrained minimum, pre-conditioned conjugate gradient
        algorithm, function of several variables using 1st
        derivatives

E04DJF  Read optional parameter values for E04DGF from external
        file
```

E04DKF Supply optional parameter values to E04DGF

E04FDF Unconstrained minimum of a sum of squares, combined Gauss-Newton and modified Newton algorithm using function values only

E04GCF Unconstrained minimum of a sum of squares, combined Gauss-Newton and quasi-Newton algorithm, using 1st derivatives

E04JAF Minimum, function of several variables, quasi-Newton algorithm, simple bounds, using function values only

E04MBF Linear programming problem

E04NAF Quadratic programming problem

E04UCF Minimum, function of several variables, sequential QP method, nonlinear constraints, using function values and optionally 1st derivatives

E04UDF Read optional parameter values for E04UCF from external file

E04UEF Supply optional parameter values to E04UCF

E04YCF Covariance matrix for nonlinear least-squares problem

F01 -- Matrix Factorizations

F01BRF LU factorization of real sparse matrix

F01BSF LU factorization of real sparse matrix with known sparsity pattern

F01MAF LL^T factorization of real sparse symmetric positive-definite matrix

F01MCF LDL^T factorization of real symmetric positive-definite variable-bandwidth matrix

F01QCF QR factorization of real m by n matrix (m>=n)

F01QDF Operations with orthogonal matrices, compute QB or $Q^T B$ after factorization by F01QCF

F01QEF Operations with orthogonal matrices, form columns of Q after factorization by F01QCF

F01RCF QR factorization of complex m by n matrix (m>=n)

F01RDF Operations with unitary matrices, compute QB or $Q^H B$ after factorization by F01RCF

F01REF Operations with unitary matrices, form columns of Q after factorization by F01RCF

F02 -- Eigenvalues and Eigenvectors

F02AAF All eigenvalues of real symmetric matrix

F02ABF All eigenvalues and eigenvectors of real symmetric matrix

F02ADF All eigenvalues of generalized real symmetric-definite eigenproblem

F02AEF All eigenvalues and eigenvectors of generalized real symmetric-definite eigenproblem

F02AFF All eigenvalues of real matrix

F02AGF All eigenvalues and eigenvectors of real matrix

F02AJF All eigenvalues of complex matrix

F02AKF All eigenvalues and eigenvectors of complex matrix

F02AWF All eigenvalues of complex Hermitian matrix

F02AXF All eigenvalues and eigenvectors of complex Hermitian matrix

F02BBF Selected eigenvalues and eigenvectors of real symmetric matrix

F02BJF All eigenvalues and optionally eigenvectors of generalized eigenproblem by QZ algorithm, real matrices

F02FJF Selected eigenvalues and eigenvectors of sparse symmetric eigenproblem

F02WEF SVD of real matrix

F02XEF SVD of complex matrix

F04 -- Simultaneous Linear Equations

```
F04ADF  Approximate solution of complex simultaneous linear
        equations with multiple right-hand sides

F04ARF  Approximate solution of real simultaneous linear
        equations, one right-hand side

F04ASF  Accurate solution of real symmetric positive-definite
        simultaneous linear equations, one right-hand side

F04ATF  Accurate solution of real simultaneous linear equations,
        one right-hand side

F04AXF  Approximate solution of real sparse simultaneous linear
        equations (coefficient matrix already factorized by
        F01BRF or F01BSF)

F04FAF  Approximate solution of real symmetric positive-definite
        tridiagonal simultaneous linear equations, one right-hand
        side

F04JGF  Least-squares (if rank = n) or minimal least-squares (if
        rank <n) solution of m real equations in n unknowns, rank
        <=n, m>=n

F04MAF  Real sparse symmetric positive-definite simultaneous
        linear equations (coefficient matrix already factorized)

F04MBF  Real sparse symmetric simultaneous linear equations

F04MCF  Approximate solution of real symmetric positive-definite
        variable-bandwidth simultaneous linear equations
        (coefficient matrix already factorized)

F04QAF  Sparse linear least-squares problem, m real equations in
        n unknowns

F07 -- Linear Equations (LAPACK)

F07ADF  (DGETRF) LU factorization of real m by n matrix

F07AEF  (DGETRS) Solution of real system of linear equations,
        multiple right-hand sides, matrix already factorized by
        F07ADF

F07FDF  (DPOTRF) Cholesky factorization of real symmetric
        positive-definite matrix

F07FEF  (DPOTRS) Solution of real symmetric positive-definite
        system of linear equations, multiple right-hand sides,
```

```
       matrix already factorized by F07FDF

G01 -- Simple Calculations on Statistical Data

G01AAF  Mean, variance, skewness, kurtosis etc, one variable,
        from raw data

G01ADF  Mean, variance, skewness, kurtosis etc, one variable,
        from frequency table

G01AEF  Frequency table from raw data


G01AFF  Two-way contingency table analysis, with (chi) /Fisher's
        exact test

G01ALF  Computes a five-point summary (median, hinges and
        extremes)

G01ARF  Constructs a stem and leaf plot

G01EAF  Computes probabilities for the standard Normal
        distribution

G01EBF  Computes probabilities for Student's t-distribution

                                        2
G01ECF  Computes probabilities for (chi)  distribution

G01EDF  Computes probabilities for F-distribution

G01EEF  Computes upper and lower tail probabilities and
        probability density function for the beta distribution

G01EFF  Computes probabilities for the gamma distribution

G01FAF  Computes deviates for the standard Normal distribution

G01FBF  Computes deviates for Student's t-distribution

                                       2
G01FCF  Computes deviates for the (chi)  distribution

G01FDF  Computes deviates for the F-distribution

G01FEF  Computes deviates for the beta distribution

G01FFF  Computes deviates for the gamma distribution

G01HAF  Computes probabilities for the bivariate Normal
```

```
        distribution

G02 -- Correlation and Regression Analysis

G02BNF  Kendall/Spearman non-parametric rank correlation
        coefficients, no missing values, overwriting input data

G02BQF  Kendall/Spearman non-parametric rank correlation
        coefficients, no missing values, preserving input data

G02BXF  Computes (optionally weighted) correlation and covariance
        matrices

G02CAF  Simple linear regression with constant term, no missing
        values

G02DAF  Fits a general (multiple) linear regression model

G02DGF  Fits a general linear regression model for new dependent
        variable

G02DNF  Computes estimable function of a general linear
        regression model and its standard error

G02FAF  Calculates standardized residuals and influence
        statistics

G02GBF  Fits a generalized linear model with binomial errors

G02GCF  Fits a generalized linear model with Poisson errors

G03 -- Multivariate Methods

G03AAF  Performs principal component analysis

G03ADF  Performs canonical correlation analysis

G03BAF  Computes orthogonal rotations for loading matrix,
        generalized orthomax criterion

G05 -- Random Number Generators

G05CAF  Pseudo-random double precision numbers, uniform
        distribution over (0,1)

G05CBF  Initialise random number generating routines to give
        repeatable sequence

G05CCF  Initialise random number generating routines to give non-
        repeatable sequence
```

G05CFF Save state of random number generating routines

G05CGF Restore state of random number generating routines

G05DDF Pseudo-random double precision numbers, Normal distribution

G05DFF Pseudo-random double precision numbers, Cauchy distribution

G05DPF Pseudo-random double precision numbers, Weibull distribution

G05DYF Pseudo-random integer from uniform distribution

G05DZF Pseudo-random logical (boolean) value

G05EAF Set up reference vector for multivariate Normal distribution

G05ECF Set up reference vector for generating pseudo-random integers, Poisson distribution

G05EDF Set up reference vector for generating pseudo-random integers, binomial distribution

G05EHF Pseudo-random permutation of an integer vector

G05EJF Pseudo-random sample from an integer vector

G05EXF Set up reference vector from supplied cumulative distribution function or probability distribution function

G05EYF Pseudo-random integer from reference vector

G05EZF Pseudo-random multivariate Normal vector from reference vector

G05FAF Generates a vector of pseudo-random numbers from a uniform distribution

G05FBF Generates a vector of pseudo-random numbers from a (negative) exponential distribution

G05FDF Generates a vector of pseudo-random numbers from a Normal distribution

G05FEF Generates a vector of pseudo-random numbers from a beta distribution

G05FFF Generates a vector of pseudo-random numbers from a gamma distribution

G05HDF Generates a realisation of a multivariate time series from a VARMA model

G08 -- Nonparameteric Statistics

G08AAF Sign test on two paired samples

G08ACF Median test on two samples of unequal size

G08AEF Friedman two-way analysis of variance on k matched samples

G08AFF Kruskal-Wallis one-way analysis of variance on k samples of unequal size

G08AGF Performs the Wilcoxon one sample (matched pairs) signed rank test

G08AHF Performs the Mann-Whitney U test on two independent samples

G08AJF Computes the exact probabilities for the Mann-Whitney U statistic, no ties in pooled sample

G08AKF Computes the exact probabilities for the Mann-Whitney U statistic, ties in pooled sample

G08CGF Performs the (chi)^2 goodness of fit test, for standard continuous distributions

G13 -- Time Series Analysis

G13AAF Univariate time series, seasonal and non-seasonal differencing

G13ABF Univariate time series, sample autocorrelation function

G13ACF Univariate time series, partial autocorrelations from autocorrelations

G13ADF Univariate time series, preliminary estimation, seasonal ARIMA model

G13AFF Univariate time series, estimation, seasonal ARIMA model

G13AGF Univariate time series, update state set for forecasting

G13AHF Univariate time series, forecasting from state set

G13AJF Univariate time series, state set and forecasts, from fully specified seasonal ARIMA model

G13ASF Univariate time series, diagnostic checking of residuals, following G13AFF

G13BAF Multivariate time series, filtering (pre-whitening) by an ARIMA model

G13BCF Multivariate time series, cross correlations

G13BDF Multivariate time series, preliminary estimation of transfer function model

G13BEF Multivariate time series, estimation of multi-input model

G13BJF Multivariate time series, state set and forecasts from fully specified multi-input model

G13CBF Univariate time series, smoothed sample spectrum using spectral smoothing by the trapezium frequency (Daniell) window

G13CDF Multivariate time series, smoothed sample cross spectrum using spectral smoothing by the trapezium frequency (Daniell) window

M01 -- Sorting

M01CAF Sort a vector, double precision numbers

M01DAF Rank a vector, double precision numbers

M01DEF Rank rows of a matrix, double precision numbers

M01DJF Rank columns of a matrix, double precision numbers

M01EAF Rearrange a vector according to given ranks, double precision numbers

M01ZAF Invert a permutation

S -- Approximations of Special Functions

z

S01EAF Complex exponential, e

S13AAF Exponential integral $E_1(x)$

S13ACF Cosine integral Ci(x)

S13ADF Sine integral Si(x)

S14AAF Gamma function

S14ABF Log Gamma function

S14BAF Incomplete gamma functions P(a,x) and Q(a,x)

S15ADF Complement of error function erfc x

S15AEF Error function erf x

S17ACF Bessel function $Y_0(x)$

S17ADF Bessel function $Y_1(x)$

S17AEF Bessel function $J_0(x)$

S17AFF Bessel function $J_1(x)$

S17AGF Airy function Ai(x)

S17AHF Airy function Bi(x)

S17AJF Airy function Ai'(x)

S17AKF Airy function Bi'(x)

S17DCF Bessel functions $Y_{(nu)+a}(z)$, real a>=0, complex z, (nu)=0,1,2,...

S17DEF Bessel functions $J_{(nu)+a}(z)$, real a>=0, complex z, (nu)=0,1,2,...

S17DGF Airy functions Ai(z) and Ai'(z), complex z

S17DHF Airy functions Bi(z) and Bi'(z), complex z

S17DLF Hankel functions $H^{(j)}_{(nu)+a}(z)$, j=1,2, real a>=0, complex z, (nu)=0,1,2,...

S18ACF Modified Bessel function $K_0(x)$

S18ADF Modified Bessel function $K_1(x)$

S18AEF Modified Bessel function $I_0(x)$

S18AFF Modified Bessel function $I_1(x)$

S18DCF Modified Bessel functions $K_{(nu)+a}(z)$, real a>=0, complex z, (nu)=0,1,2,...

S18DEF Modified Bessel functions $I_{(nu)+a}(z)$, real a>=0, complex z, (nu)=0,1,2,...

S19AAF Kelvin function ber x

S19ABF Kelvin function bei x

S19ACF Kelvin function ker x

S19ADF Kelvin function kei x

S20ACF Fresnel integral S(x)

S20ADF Fresnel integral C(x)

S21BAF Degenerate symmetrised elliptic integral of 1st kind $R_C(x,y)$

S21BBF Symmetrised elliptic integral of 1st kind $R_F(x,y,z)$

S21BCF Symmetrised elliptic integral of 2nd kind $R_D(x,y,z)$

S21BDF Symmetrised elliptic integral of 3rd kind $R_J(x,y,z,r)$

```
X04 -- Input/Output Utilities

X04AAF  Return or set unit number for error messages

X04ABF  Return or set unit number for advisory messages

X04CAF  Print a real general matrix

X04DAF  Print a complex general matrix

X05 -- Date and Time Utilities

X05AAF  Return date and time as an array of integers

X05ABF  Convert array of integers representing date and time to
        character string

X05ACF  Compare two character strings representing date and time

X05BAF  Return the CPU time


Fundamental Support Routines

The following fundamental support routines are provided and are
documented in compact form in the relevant chapter introductory
material:

F06 -- Linear Algebra Support Routines

F06AAF  (DROTG) Generate real plane rotation

F06EAF  (DDOT) Dot product of two real vectors

F06ECF  (DAXPY) Add scalar times real vector to real vector

F06EDF  (DSCAL) Multiply real vector by scalar

F06EFF  (DCOPY) Copy real vector

F06EGF  (DSWAP) Swap two real vectors

F06EJF  (DNRM2) Compute Euclidean norm of real vector

F06EKF  (DASUM) Sum the absolute values of real vector elements

F06EPF  (DROT) Apply real plane rotation
```

F06GAF (ZDOTU) Dot product of two complex vectors, unconjugated

F06GBF (ZDOTC) Dot product of two complex vectors, conjugated

F06GCF (ZAXPY) Add scalar times complex vector to complex vector

F06GDF (ZSCAL) Multiply complex vector by complex scalar

F06GFF (ZCOPY) Copy complex vector

F06GGF (ZSWAP) Swap two complex vectors

F06JDF (ZDSCAL) Multiply complex vector by real scalar

F06JJF (DZNRM2) Compute Euclidean norm of complex vector

F06JKF (DZASUM) Sum the absolute values of complex vector elements

F06JLF (IDAMAX) Index, real vector element with largest absolute value

F06JMF (IZAMAX) Index, complex vector element with largest absolute value

F06PAF (DGEMV) Matrix-vector product, real rectangular matrix

F06PBF (DGBMV) Matrix-vector product, real rectangular band matrix

F06PCF (DSYMV) Matrix-vector product, real symmetric matrix

F06PDF (DSBMV) Matrix-vector product, real symmetric band matrix

F06PEF (DSPMV) Matrix-vector product, real symmetric packed matrix

F06PFF (DTRMV) Matrix-vector product, real triangular matrix

F06PGF (DTBMV) Matrix-vector product, real triangular band matrix

F06PHF (DTPMV) Matrix-vector product, real triangular packed matrix

F06PJF (DTRSV) System of equations, real triangular matrix

F06PKF (DTBSV) System of equations, real triangular band matrix

F06PLF (DTPSV) System of equations, real triangular packed matrix

F06PMF (DGER) Rank-1 update, real rectangular matrix

F06PPF (DSYR) Rank-1 update, real symmetric matrix

F06PQF (DSPR) Rank-1 update, real symmetric packed matrix

F06PRF (DSYR2) Rank-2 update, real symmetric matrix

F06PSF (DSPR2) Rank-2 update, real symmetric packed matrix

F06SAF (ZGEMV) Matrix-vector product, complex rectangular matrix

F06SBF (ZGBMV) Matrix-vector product, complex rectangular band matrix

F06SCF (ZHEMV) Matrix-vector product, complex Hermitian matrix

F06SDF (ZHBMV) Matrix-vector product, complex Hermitian band matrix

F06SEF (ZHPMV) Matrix-vector product, complex Hermitian packed matrix

F06SFF (ZTRMV) Matrix-vector product, complex triangular matrix

F06SGF (ZTBMV) Matrix-vector product, complex triangular band matrix

F06SHF (ZTPMV) Matrix-vector product, complex triangular packed matrix

F06SJF (ZTRSV) System of equations, complex triangular matrix

F06SKF (ZTBSV) System of equations, complex triangular band matrix

F06SLF (ZTPSV) System of equations, complex triangular packed matrix

F06SMF (ZGERU) Rank-1 update, complex rectangular matrix, unconjugated vector

F06SNF (ZGERC) Rank-1 update, complex rectangular matrix, conjugated vector

F06SPF (ZHER) Rank-1 update, complex Hermitian matrix

F06SQF (ZHPR) Rank-1 update, complex Hermitian packed matrix

```
F06SRF  (ZHER2) Rank-2 update, complex Hermitian matrix

F06SSF  (ZHPR2) Rank-2 update, complex Hermitian packed matrix

F06YAF  (DGEMM) Matrix-matrix product, two real rectangular
        matrices

F06YCF  (DSYMM) Matrix-matrix product, one real symmetric matrix,
        one real rectangular matrix

F06YFF  (DTRMM) Matrix-matrix product, one real triangular
        matrix, one real rectangular matrix

F06YJF  (DTRSM) Solves a system of equations with multiple right-
        hand sides, real triangular coefficient matrix

F06YPF  (DSYRK) Rank-k update of a real symmetric matrix

F06YRF  (DSYR2K) Rank-2k update of a real symmetric matrix

F06ZAF  (ZGEMM) Matrix-matrix product, two complex rectangular
        matrices

F06ZCF  (ZHEMM) Matrix-matrix product, one complex Hermitian
        matrix, one complex rectangular matrix

F06ZFF  (ZTRMM) Matrix-matrix product, one complex triangular
        matrix, one complex rectangular matrix

F06ZJF  (ZTRSM) Solves system of equations with multiple right-
        hand sides, complex triangular coefficient matrix

F06ZPF  (ZHERK) Rank-k update of a complex Hermitian matrix

F06ZRF  (ZHER2K) Rank-2k update of a complex Hermitian matrix

F06ZTF  (ZSYMM) Matrix-matrix product, one complex symmetric
        matrix, one complex rectangular matrix

F06ZUF  (ZSYRK) Rank-k update of a complex symmetric matrix

F06ZWF  (ZSYR2K) Rank-2k update of a complex symmetric matrix

X01 -- Mathematical Constants

X01AAF  (pi)

X01ABF  Euler's constant, (gamma)
```

```
X02 -- Machine Constants

X02AHF  Largest permissible argument for SIN and COS

X02AJF  Machine precision

X02AKF  Smallest positive model number

X02ALF  Largest positive model number

X02AMF  Safe range of floating-point arithmetic

X02ANF  Safe range of complex floating-point arithmetic

X02BBF  Largest representable integer

X02BEF  Maximum number of decimal digits that can be represented

X02BHF  Parameter of floating-point arithmetic model, b

X02BJF  Parameter of floating-point arithmetic model, p

X02BKF  Parameter of floating-point arithmetic model, e
                                                       min

X02BLF  Parameter of floating-point arithmetic model, e
                                                       max

X02DJF  Parameter of floating-point arithmetic model, ROUNDS


Routines from the NAG Fortran Library

A number of routines from the NAG Fortran Library are used in the
Foundation Library as auxiliaries and are not documented here:

A00AAF

A02AAF  A02ABF  A02ACF

C02AJF

C05AZF  C05NCF  C05PCF

C06FAF  C06FBF  C06FCF  C06FFF  C06FJF  C06FKF

C06HAF  C06HBF  C06HCF  C06HDF

D02CBF  D02CHF  D02NMF  D02NSF  D02NVF  D02PAF
```

D02XAF D02XKF D02YAF D02ZAF

E02AFF

E04GBF E04GEF E04YAF

F01ADF F01AEF F01AFF F01AGF F01AHF F01AJF

F01AKF F01AMF F01APF F01ATF F01AUF F01AVF

F01AWF F01AXF F01BCF F01BTF F01CRF F01LBF

F01LZF F01QAF F01QFF F01QGF F01QJF F01QKF

F01RFF F01RGF F01RJF F01RKF

F02AMF F02ANF F02APF F02AQF F02AVF F02AYF

F02BEF F02SWF F02SXF F02SYF F02SZF F02UWF

F02UXF F02UYF F02WDF F02WUF F02XUF

F03AAF F03ABF F03AEF F03AFF

F04AAF F04AEF F04AFF F04AGF F04AHF F04AJF

F04AMF F04ANF F04AYF F04LDF F04YAF F04YCF

F06BAF F06BCF F06BLF F06BMF F06BNF F06CAF

F06CCF F06CLF F06DBF F06DFF F06FBF F06FCF

F06FDF F06FGF F06FJF F06FLF F06FPF F06FQF

F06FRF F06FSF F06HBF F06HGF F06HQF F06HRF

F06KFF F06KJF F06KLF F06QFF F06QHF F06QKF

F06QRF F06QSF F06QTF F06QVF F06QWF F06QXF

F06RAF F06RJF F06TFF F06THF F06TTF F06TXF

F06VJF F06VKF F06VXF

F07AGF F07AHF F07AJF F07FGF F07FHF F07FJF

F07TJF

G01CEF

```
        G02BAF  G02BUF  G02BWF  G02DDF

        G13AEF

        M01CBF  M01CCF  M01DBF  M01DCF  M01DFF  M01ZBF

        P01ABF  P01ACF

        S01BAF  S07AAF  S15ABF

        X03AAF

        X04BAF  X04BBF  X04CBF  X04DBF

\end{verbatim}
\endscroll
\end{page}
```

NAG Documentation: introduction

— nagaux.ht —

```
\begin{page}{manpageXXintro}{NAG Documentation: introduction}
\beginscroll
\begin{verbatim}



     INTRO(3NAG)       Foundation Library (12/10/92)       INTRO(3NAG)



          Introduction                               Essential Introduction
          Essential Introduction to the NAG Foundation Library

          This document is essential reading for any prospective user of
          the Library.

          This document appears in both the Handbook and the Reference
          Manual for the NAG Foundation Library, but with a different
          Section 3 to describe the different forms of routine
          documentation in the two publications.

          1. The Library and its Documentation
```

```
1.1. Structure of the Library

1.2. Structure of the Documentation

1.3. On-line Documentation

1.4. Implementations of the Library

1.5. Library Identification

1.6. Fortran Language Standards

2. Using the Library

2.1. General Advice

2.2. Programming Advice

2.3. Error handling and the Parameter IFAIL

2.4. Input/output in the Library

2.5. Auxiliary Routines

3. Using the Reference Manual

3.1. General Guidance

3.2. Structure of Routine Documents

3.3. Specifications of Parameters

3.3.1. Classification of Parameters

3.3.2. Constraints and Suggested Values

3.3.3. Array Parameters

3.4. Implementation-dependent Information

3.5. Example Programs and Results

3.6. Summary for New Users

4. Relationship between the Foundation Library
   and other NAG Libraries

4.1. NAG Fortran Library
```

1. The Library and its Documentation

1.1. Structure of the Library

The NAG Foundation Library is a comprehensive collection of Fortran 77 routines for the solution of numerical and statistical problems. The word 'routine' is used to denote 'subroutine' or ' function'.

The Library is divided into chapters, each devoted to a branch of numerical analysis or statistics. Each chapter has a three-character name and a title, e.g.

    D01 -- Quadrature

Exceptionally one chapter (S) has a one-character name. (The chapters and their names are based on the ACM modified SHARE classification index [1].)

All documented routines in the Library have six-character names, beginning with the characters of the chapter name, e.g.

    D01AJF

Note that the second and third characters are digits, not letters; e.g. 0 is the digit zero, not the letter O. The last letter of each routine name is always 'F'.

1.2. Structure of the Documentation

There are two types of manual for the NAG Foundation Library: a Handbook and a Reference Manual.

The Handbook has the same chapter structure as the Library: each chapter of routines in the Library has a corresponding chapter (of the same name) in the Handbook. The chapters occur in alphanumeric order. General introductory documents and indexes are placed at the beginning of the Handbook.

Each chapter in the Handbook contains a Chapter Introduction, followed by concise summaries of the functionality and parameter specifications of each routine in the chapter. Exceptionally, in some chapters (F06, X01, X02) which contain simple support routines, there are no concise summaries: all the routines are described together in the Chapter Introduction.

The Reference Manual provides complete reference documentation for the NAG Foundation Library. In the Reference Manual, each chapter consists of the following documents:

Chapter Introduction, e.g. Introduction -- D01;

Chapter Contents, e.g. Contents -- D01;

routine documents, one for each documented routine in the chapter.

A routine document has the same name as the routine which it describes. Within each chapter, routine documents occur in alphanumeric order. As in the Handbook, chapters F06, X01 and X02 do not contain separate documentation for individual routines.

The general introductory documents, indexes and chapter introductions are the same in the Reference Manual as in the Handbook. The only exception is that the Essential Introduction contains a different Section 3 in the two publications, to describe the different forms of routine documentation.

1.3. On-line Documentation

Extensive on-line documentation is included as an integral part of the Foundation Library product. This consists of a number of components:

-- general introductory material, including the Essential Introduction

-- a summary list of all documented routines

-- a KWIC Index

-- Chapter Introductions

-- routine documents

-- example programs, data and results.

The material has been derived in a number of forms to cater for different user requirements, e.g. UNIX man pages, plain text, RICH TEXT format etc, and the appropriate version is included on the distribution media. For each implementation of the Foundation Library the specific documentation (Installers' Note, Users' Note etc) gives details of what is provided.

1.4. Implementations of the Library

The NAG Foundation Library is available on many different computer systems. For each distinct system, an implementation of the Library is prepared by NAG, e.g. the IBM RISC System/6000 implementation. The implementation is distributed as a tested compiled library.

An implementation is usually specific to a range of machines; it may also be specific to a particular operating system or compilation system.

Essentially the same facilities are provided in all implementations of the Library, but, because of differences in arithmetic behaviour and in the compilation system, routines cannot be expected to give identical results on different systems, especially for sensitive numerical problems.

The documentation supports all implementations of the Library, with the help of a few simple conventions, and a small amount of implementation-dependent information, which is published in a separate Users' Note for each implementation (see Section 3.4).

1.5. Library Identification

You must know which implementation of the Library you are using or intend to use. To find out which implementation of the Library is available on your machine, you can run a program which calls the NAG Foundation Library routine A00AAF. This routine has no parameters; it simply outputs text to the advisory message unit (see Section 2.4). An example of the output is:

```
*** Start of NAG Foundation Library implementation details ***
Implementation title: IBM RISC System/6000
Precision: FORTRAN double precision
Product Code: FFIB601D
Release: 1
*** End of NAG Foundation Library implementation details ***
```

(The product code can be ignored, except possibly when communicating with NAG; see Section 4.)

1.6. Fortran Language Standards

All routines in the Library conform to ANSI Standard Fortran 90 [8].

Most of the routines in the Library were originally written to conform to the earlier Fortran 66 [6] and Fortran 77 [7] standards, and their calling sequences contain some parameters which are not strictly necessary in Fortran 90.

2. Using the Library

2.1. General Advice

A NAG Foundation Library routine cannot be guaranteed to return meaningful results, irrespective of the data supplied to it. Care and thought must be exercised in:

(a) formulating the problem;

(b) programming the use of library routines;

(c) assessing the significance of the results.

2.2. Programming Advice

The NAG Foundation Library and its documentation are designed on the assumption that users know how to write a calling program in Fortran.

When programming a call to a routine, read the routine document carefully, especially the description of the Parameters. This states clearly which parameters must have values assigned to them on entry to the routine, and which return useful values on exit. See Section 3.3 for further guidance.

If a call to a Library routine results in an unexpected error message from the system (or possibly from within the Library), check the following:

Has the NAG routine been called with the correct number of parameters?

Do the parameters all have the correct type?

Have all array parameters been dimensioned correctly?

Remember that all floating-point parameters must be declared to be double precision, either with an explicit DOUBLE PRECISION declaration (or COMPLEX(KIND(1.0D0)) if they are complex), or by using a suitable IMPLICIT statement.

Avoid the use of NAG-type names for your own program units or COMMON blocks: in general, do not use names which contain a three-character NAG chapter name embedded in them; they may clash with the names of an auxiliary routine or COMMON block used by the NAG Library.

2.3. Error handling and the Parameter IFAIL

NAG Foundation Library routines may detect various kinds of error, failure or warning conditions. Such conditions are handled in a systematic way by the Library. They fall roughly into three classes:

(i) an invalid value of a parameter on entry to a routine;

(ii) a numerical failure during computation (e.g. approximate singularity of a matrix, failure of an iteration to converge);

(iii) a warning that although the computation has been completed, the results cannot be guaranteed to be completely reliable.

All three classes are handled in the same way by the Library, and are all referred to here simply as 'errors'.

The error-handling mechanism uses the parameter IFAIL, which is the last parameter in the calling sequence of most NAG Foundation Library routines. IFAIL serves two purposes:

(i) it allows users to specify what action a Library routine should take if it detects an error;

(ii) it reports the outcome of a call to a Library routine, either success (IFAIL = 0) or failure (IFAIL /= 0, with different values indicating different reasons for the failure, as explained in Section 6 of the routine document) .

For the first purpose IFAIL must be assigned a value before calling the routine; since IFAIL is reset by the routine, it must be passed as a variable, not as an integer constant. Allowed values on entry are:

IFAIL=0: an error message is output, and execution is terminated ('hard failure');

IFAIL=+1: execution continues without any error message;

```
    IFAIL=-1: an error message is output, and execution
    continues.

The settings IFAIL =+-1 are referred to as 'soft failure'.
The safest choice is to set IFAIL to 0, but this is not always
convenient: some routines return useful results even though a
failure (in some cases merely a warning) is indicated. However,
if IFAIL is set to +- 1 on entry, it is essential for the program
to test its value on exit from the routine, and to take
appropriate action.

The specification of IFAIL in Section 5 of a routine document
suggests a suitable setting of IFAIL for that routine.

2.4. Input/output in the Library

Most NAG Foundation Library routines perform no output to an
external file, except possibly to output an error message. All
error messages are written to a logical error message unit. This
unit number (which is set by default to 6 in most
implementations) can be changed by calling the Library routine
X04AAF.

Some NAG Foundation Library routines may optionally output their
final results, or intermediate results to monitor the course of
computation. All output other than error messages is written to a
logical advisory message unit. This unit number (which is also
set by default to 6 in most implementations) can be changed by
calling the Library routine X04ABF. Although it is logically
distinct from the error message unit, in practice the two unit
numbers may be the same.

All output from the Library is formatted.

The only Library routines which perform input from an external
file are a few 'option-setting' routines in Chapter E04: the unit
number is a parameter to the routine, and all input is formatted.

You must ensure that the relevant Fortran unit numbers are
associated with the desired external files, either by an OPEN
statement in your calling program, or by operating system
commands.

2.5. Auxiliary Routines

In addition to those Library routines which are documented and
are intended to be called by users, the Library also contains
many auxiliary routines.

In general, you need not be concerned with them at all, although
```

```
you may be made aware of their existence if, for example, you
examine a memory map of an executable program which calls NAG
routines. The only exception is that when calling some NAG
Foundation Library routines, you may be required or allowed to
supply the name of an auxiliary routine from the Library as an
external procedure parameter. The routine documents give the
necessary details. In such cases, you only need to supply the
name of the routine; you never need to know details of its
parameter-list.

NAG auxiliary routines have names which are similar to the name
of the documented routine(s) to which they are related, but with
last letter 'Z', 'Y', and so on, e.g. D01AJZ is an auxiliary
routine called by D01AJF.

3. Using the Reference Manual

3.1. General Guidance

The Reference Manual is designed to serve the following
functions:

    -- to give background information about different areas of
    numerical and statistical computation;

    -- to advise on the choice of the most suitable NAG
    Foundation Library routine or routines to solve a particular
    problem;

    -- to give all the information needed to call a NAG
    Foundation Library routine correctly from a Fortran program,
    and to assess the results.

At the beginning of the Manual are some general introductory
documents. The following may help you to find the chapter, and
possibly the routine, which you need to solve your problem:

    Contents     -- a list of routines in the Library, by
    Summary      chapter;

    KWIC Index   -- a keyword index to chapters and routines.

Having found a likely chapter or routine, you should read the
corresponding Chapter Introduction, which gives background
information about that area of numerical computation, and
recommendations on the choice of a routine, including indexes,
tables or decision trees.

When you have chosen a routine, you must consult the routine
document. Each routine document is essentially self-contained (it
```

may contain references to related documents). It includes a description of the method, detailed specifications of each parameter, explanations of each error exit, and remarks on accuracy.

Example programs which illustrate the use of each routine are distributed with the Library in machine-readable form.

3.2. Structure of Routine Documents

All routine documents have the same structure, consisting of nine numbered sections:

1. Purpose

2. Specification

3. Description

4. References

5. Parameters (see Section 3.3 below)

6. Error Indicators

7. Accuracy

8. Further Comments

9. Example (see Section 3.5 below)

In a few documents, Section 5 also includes a description of printed output which may optionally be produced by the routine.

3.3. Specifications of Parameters

Section 5 of each routine document contains the specification of the parameters, in the order of their appearance in the parameter list.

3.3.1. Classification of Parameters

Parameters are classified as follows:

Input : you must assign values to these parameters on or before entry to the routine, and these values are unchanged on exit from the routine.

Output : you need not assign values to these parameters on or before entry to the routine; the routine may assign values to them.

Input/Output : you must assign values to these parameters on or before entry to the routine, and the routine may then change these values.

Workspace: array parameters which are used as workspace by the routine. You must supply arrays of the correct type and dimension, but you need not be concerned with their contents.

External Procedure: a subroutine or function which must be supplied (e.g. to evaluate an integrand or to print intermediate output). Usually it must be supplied as part of your calling program, in which case its specification includes full details of its parameter-list and specifications of its parameters (all enclosed in a box). Its parameters are classified in the same way as those of the Library routine, but because you must write the procedure rather than call it, the significance of the classification is different:

Input : values may be supplied on entry, which your procedure must not change.

Output : you may or must assign values to these parameters before exit from your procedure.

Input/Output : values may be supplied on entry, and you may or must assign values to them before exit from your procedure.

Occasionally, as mentioned in Section 2.5, the procedure can be supplied from the NAG Library, and then you only need to know its name.

User Workspace: array parameters which are passed by the Library routine to an external procedure parameter. They are not used by the routine, but you may use them to pass information between your calling program and the external procedure.

3.3.2. Constraints and Suggested Values

The word 'Constraint:' or 'Constraints:' in the specification of an Input parameter introduces a statement of the range of valid values for that parameter, e.g.

Constraint: N > 0.

If the routine is called with an invalid value for the parameter (e.g. N = 0), the routine will usually take an error exit, returning a non-zero value of IFAIL (see Section 2.3).

In newer documents constraints on parameters of type CHARACTER only list uppercase alphabetic characters, e.g.

    Constraint: STRING = 'A' or 'B'.

In practice all routines with CHARACTER parameters will permit the use of lower case characters.

The phrase 'Suggested Value:' introduces a suggestion for a reasonable initial setting for an Input parameter (e.g. accuracy or maximum number of iterations) in case you are unsure what value to use; you should be prepared to use a different setting if the suggested value turns out to be unsuitable for your problem.

3.3.3. Array Parameters

Most array parameters have dimensions which depend on the size of the problem. In Fortran terminology they have 'adjustable dimensions': the dimensions occurring in their declarations are integer variables which are also parameters of the Library routine.

For example, a Library routine might have the specification:

```
     SUBROUTINE <name> (M, N, A, B, LDB)
     INTEGER       M, N, A(N), B(LDB,N), LDB
```

For a one-dimensional array parameter, such as A in this example, the specification would begin:

```
 3:  A(N) -- DOUBLE PRECISION array                          Input
```

You must ensure that the dimension of the array, as declared in your calling (sub)program, is at least as large as the value you supply for N. It may be larger; but the routine uses only the first N elements.

For a two-dimensional array parameter, such as B in the example, the specification might be:

```
 4:  B(LDB,N) -- DOUBLE PRECISION array               Input/Output
     On entry: the m by n matrix B.
```

and the parameter LDB might be described as follows:

```
 5:  LDB -- INTEGER                                          Input
     On entry: the first dimension of the array B as declared in
```

     the (sub)program from which <name> is called. Constraint:
     LDB >= M.

You must supply the first dimension of the array B, as declared in your calling (sub)program, through the parameter LDB, even though the number of rows actually used by the routine is determined by the parameter M. You must ensure that the first dimension of the array is at least as large as the value you supply for M. The extra parameter LDB is needed because Fortran does not allow information about the dimensions of array parameters to be passed automatically to a routine.

You must also ensure that the second dimension of the array, as declared in your calling (sub)program, is at least as large as the value you supply for N. It may be larger, but the routine only uses the first N columns.

A program to call the hypothetical routine used as an example in this section might include the statements:

```
     INTEGER AA(100), BB(100,50)
     LDB = 100
     .
     .
     .
     M = 80
     N = 20
     CALL <name>(M,N,AA,BB,LDB)
```

Fortran requires that the dimensions which occur in array declarations, must be greater than zero. Many NAG routines are designed so that they can be called with a parameter like N in the above example set to 0 (in which case they would usually exit immediately without doing anything). If so, the declarations in the Library routine would use the 'assumed size' array dimension, and would be given as:

```
     INTEGER        M, N, A(*), B(LDB,*), LDB
```

However, the original declaration of an array in your calling program must always have constant dimensions, greater than or equal to 1.

Consult an expert or a textbook on Fortran, if you have difficulty in calling NAG routines with array parameters.

3.4. Implementation-dependent Information

In order to support all implementations of the Foundation Library, the Manual has adopted a convention of using bold italics to distinguish terms which have different interpretations in different implementations.

For example, machine precision denotes the relative precision to which double precision floating-point numbers are stored in the computer, e.g. in an implementation with approximately 16 decimal digits of precision, machine precision has a value of approximately 10^{-16}.

The precise value of machine precision is given by the function X02AJF. Other functions in Chapter X02 return the values of other implementation-dependent constants, such as the overflow threshold, or the largest representable integer. Refer to the X02

Chapter Introduction for more details.

For each implementation of the Library, a separate Users' Note is provided. This is a short document, revised at each Mark. At most installations it is available in machine-readable form. It gives any necessary additional information which applies specifically to that implementation, in particular:

-- the interpretation of bold italicised terms;

-- the values returned by X02 routines;

-- the default unit numbers for output (see Section 2.4).

3.5. Example Programs and Results

The last section of each routine document describes an example problem which can be solved by simple use of the routine. The example programs themselves, together with data and results, are not printed in the routine document, but are distributed in machine-readable form with the Library. The programs are designed so that they can fairly easily be modified, and so serve as the basis for a simple program to solve a user's own problem.

The results distributed with each implementation were obtained using that implementation of the Library; they may not be identical to the results obtained with other implementations.

3.6. Summary for New Users

If you are unfamiliar with the NAG Foundation Library and are thinking of using a routine from it, please follow these instructions:

(a) read the whole of the Essential Introduction;

(b) consult the Contents Summary or KWIC Index to choose an appropriate chapter or routine;

(c) read the relevant Chapter Introduction;

(d) choose a routine, and read the routine document. If the routine does not after all meet your needs, return to steps (b) or (c);

(e) read the Users' Note for your implementation;

(f) consult local documentation, which should be provided by your local support staff, about access to the NAG Library on your computing system.

You should now be in a position to include a call to the routine in a program, and to attempt to run it. You may of course need to refer back to the relevant documentation in the case of difficulties, for advice on assessment of results, and so on.

As you become familiar with the Library, some of steps (a) to (f) can be omitted, but it is always essential to:

-- be familiar with the Chapter Introduction;

-- read the routine document;

-- be aware of the Users' Note for your implementation.

4. Relationship between the Foundation Library and other NAG Libraries

4.1. NAG Fortran Library

The Foundation Library is a strict subset of the full NAG Fortran Library (Mark 15 or later). Routines in both libraries have identical source code (apart from any modifications necessary for implementation on a specific system) and hence can be called in exactly the same way, though you should consult the relevant implementation-specific documentation for details such as values of machine constants.

By its very nature, the Foundation Library cannot contain the same extensive range of routines as the full Fortran Library. If your application requires a routine which is not in the Foundation Library, then please consult NAG for information on relevant material available in the Fortran Library.

Some routines which occur as user-callable routines in the full Fortran Library are included as auxiliary routines in the Foundation Library but they are not documented in this publication and direct calls to them should only be made if you are already familiar with their use in the Fortran Library. A list of all such auxiliary routines is given at the end of the Foundation Library Contents Summary.

Whereas the full Fortran Library may be provided in either a single precision or a double precision version, the Foundation Library is always provided in double precision.

4.2. NAG Workstation Library

The Foundation Library is a successor product to an earlier, smaller subset of the full NAG Fortran Library which was called the NAG Workstation Library. The Foundation Library has greater functionality than the Workstation Library but is not strictly upwards compatible, i.e., a number of routines in the earlier product have been replaced by new material to reflect recent algorithmic developments.
If you have used the Workstation Library and wish to convert your programs to call routines from the Foundation Library, please consult the document 'Converting from the Workstation Library' in this Manual.

4.3. NAG C Library

NAG has also developed a library of numerical and statistical software for use by C programmers. This now contains over 200 user-callable functions and provides similar (but not identical) coverage to that of the Foundation Library. Please contact NAG for further details if you have a requirement for similar quality library code in C.

5. Contact between Users and NAG

If you are using the NAG Foundation Library in a multi-user environment and require further advice please consult your local support staff who will be receiving regular information from NAG. This covers such matters as:

-- obtaining a copy of the Users' Note for your implementation;

-- obtaining information about local access to the Library;

-- seeking advice about using the Library;

-- reporting suspected errors in routines or documents;

-- making suggestions for new routines or features;

-- purchasing NAG documentation.

If you are unable to make contact with a local source of support or are in a single-user environment then please contact NAG directly at any one of the addresses given at the beginning of this publication.

6. General Information about NAG

NAG produces and distributes numerical, symbolic, statistical and graphical software for the solution of problems in a wide range of applications in such areas as science, engineering, financial analysis and research.

For users who write programs and build packages NAG produces sub-program libraries in a range of computer languages (Ada, C, Fortran, Pascal, Turbo Pascal). NAG also provides a number of Fortran programming support products in the NAGWare range -- Fortran 77 programming tools, Fortran 90 compilers for a number of machine platforms (including PC-DOS) and VecPar 77 for restructuring and tuning programs for execution on vector or parallel computers.

For users who do not wish to program in the traditional sense but want the same reliability and qualities offered by our libraries, NAG provides several powerful mathematical and statistical packages for interactive use. A major addition to this range of packages is Axiom -- the powerful symbolic solver which includes a Hypertext system and graphical capabilities.

For further details of any of these products, please contact NAG at one of the addresses given at the beginning of this publication.

References [2], [3], [4], and [5] discuss various aspects of the design and development of the NAG Library, and NAG's technical policies and organisation.

7. References

[1] (1960--1976) Collected Algorithms from ACM Index by subject to algorithms.

[2] Ford B (1982) Transportable Numerical Software. Lecture Notes in Computer Science. 142 128--140.

```
     [3]   Ford B, Bentley J, Du Croz J J and Hague S J (1979) The NAG
           Library 'machine'. Software Practice and Experience. 9(1)
           65--72.

     [4]   Ford B and Pool J C T (1984) The Evolving NAG Library
           Service. Sources and Development of Mathematical Software.
           (ed W Cowell) Prentice-Hall. 375--397.

     [5]   Hague S J, Nugent S M and Ford B (1982) Computer-based
           Documentation for the NAG Library. Lecture Notes in Computer
           Science. 142 91--127.

     [6]   (1966) USA Standard Fortran. Publication X3.9. American
           National Standards Institute.

     [7]   (1978) American National Standard Fortran. Publication X3.9.
           American National Standards Institute.

     [8]   (1991) American National Standard Programming Language
           Fortran 90. Publication X3.198.
           American National Standards Institute.
```


```
\end{verbatim}
\endscroll
\end{page}
```

———-

NAG Documentation: keyword in context

— nagaux.ht —

```
\begin{page}{manpageXXkwic}{NAG Documentation: keyword in context}
\beginscroll
\begin{verbatim}



     KWIC(3NAG)       Foundation Library (12/10/92)       KWIC(3NAG)



          Introduction                               Keywords in Context
          Keywords in Context

          Pre-computed weights and                              D01BBF
          abscissae
```

```
for Gaussian quadrature rules, restricted choice of ...

Sum the                                                 F06EKF
absolute
values of real vector elements (DASUM)

Sum the                                                 F06JKF
absolute
values of complex vector elements (DZASUM)

Index, real vector element with largest                 F06JLF
absolute
value (IDAMAX)

Index, complex vector element with largest              F06JMF
absolute
value (IZAMAX)

ODEs, IVP,                                              D02CJF
Adams
method, until function of solution is zero, ...

1-D quadrature,                                         D01AJF
adaptive
, finite interval, strategy due to Piessens and de ...

1-D quadrature,                                         D01AKF
adaptive
, finite interval, method suitable for oscillating ...

1-D quadrature,                                         D01ALF
adaptive
, finite interval, allowing for singularities at ...

1-D quadrature,                                         D01AMF
adaptive
, infinite or semi-infinite interval

1-D quadrature,                                         D01ANF
adaptive
, finite interval, weight function cos((omega)x) ...

1-D quadrature,                                         D01APF
adaptive
, finite interval, weight function with end-point ...

1-D quadrature,                                         D01AQF
adaptive
, finite interval, weight function 1/(x-c), ...
```

```
type

Two-way contingency table                          G01AFF
analysis
            2
, with (chi) /Fisher's exact test

Performs principal component                       G03AAF
analysis

Performs canonical correlation                     G03ADF
analysis

Friedman two-way                                   G08AEF
analysis
of variance on k matched samples

Kruskal-Wallis one-way                             G08AFF
analysis
of variance on k samples of unequal size

L -                                                E02GAF
 1
approximation
by general linear function

Approximation                                      E02

Approximation                                      S
of special functions

ARIMA
model

Univariate time series, estimation, seasonal       G13AFF
ARIMA
model

ARIMA
model

ARIMA
model

Safe range of floating-point                       X02AMF
arithmetic

Safe range of complex floating-point               X02ANF
arithmetic
```

Parameter of floating-point arithmetic model, b	X02BHF
Parameter of floating-point arithmetic model, p	X02BJF
Parameter of floating-point arithmetic model, e_{min}	X02BKF
Parameter of floating-point arithmetic model, e_{max}	X02BLF
Parameter of floating-point arithmetic model, ROUNDS	X02DJF
Univariate time series, sample autocorrelation function	G13ABF
Univariate time series, partial autocorrelations from autocorrelations	G13ACF
Univariate time series, partial autocorrelations from autocorrelations	G13ACF
Least-squares cubic spline curve fit, automatic knot placement	E02BEF
Least-squares surface fit by bicubic splines with automatic knot placement, data on rectangular grid	E02DCF
Least-squares surface fit by bicubic splines with automatic knot placement, scattered data	E02DDF
B-splines	E02
Matrix-vector product, real rectangular band matrix (DGBMV)	F06PBF

```
x

Bessel                                                    S17ACF
function Y (x)
          0

Bessel                                                    S17ADF
function Y (x)
          1

Bessel                                                    S17AEF
function J (x)
          0

Bessel                                                    S17AFF
function J (x)
          1

Bessel                                                    S17DCF
functions Y      (z), complex z, real (nu)>=0, ...
           (nu)+n

Bessel                                                    S17DEF
functions J      (z), complex z, real (nu)>=0, ...
           (nu)+n

Modified                                                  S18ACF
Bessel
function K (x)
          0

Modified                                                  S18ADF
Bessel
function K (x)
          1

Modified                                                  S18AEF
Bessel
function I (x)
          0

Modified                                                  S18AFF
Bessel
function I (x)
          1

Modified                                                  S18DCF
Bessel
functions K      (z), complex z, real (nu)>=0, ...
           (nu)+n
```

Dekker
algorithm

Computes upper and lower tail and probability density
function probabilities for the beta distribution — G01EEF

Solution of system of nonlinear equations using 1st derivatives — C05PBF

Check user's routine for calculating 1st derivatives — C05ZAF

derivative
, one variable

Least-squares polynomial fit, values and derivatives
may be constrained, arbitrary data points — E02AGF

Derivative
of fitted polynomial in Chebyshev series form — E02AHF

Evaluation of fitted cubic spline, function and derivatives — E02BCF

derivatives

derivatives

derivatives

Computes deviates
for the standard Normal distribution — G01FAF

Computes deviates
for Student's t-distribution — G01FBF

Computes deviates
for the (chi)^2 distribution — G01FCF

Computes deviates
for the F-distribution — G01FDF

```
Computes                                                G01FEF
deviates
for the beta distribution

Computes                                                G01FFF
deviates
for the gamma distribution

Univariate time series,                                 G13ASF
diagnostic
checking of residuals, following G13AFF

ODEs, boundary value problem, finite                    D02GAF
difference
technique with deferred correction, simple ...

ODEs, boundary value problem, finite                    D02GBF
difference
technique with deferred correction, general linear ...

difference
technique with deferred correction, continuation ...

Elliptic PDE, solution of finite                        D03EDF
difference
equations by a multigrid technique

Univariate time series, seasonal and non-seasonal       G13AAF
differencing

Single 1-D real                                         C06EAF
discrete
Fourier transform, no extra workspace

Single 1-D Hermitian                                    C06EBF
discrete
Fourier transform, no extra workspace

Single 1-D complex                                      C06ECF
discrete
Fourier transform, no extra workspace

Multiple 1-D real                                       C06FPF
discrete
Fourier transforms

Multiple 1-D Hermitian                                  C06FQF
discrete
Fourier transforms
```

Multiple 1-D complex discrete Fourier transforms	C06FRF
2-D complex discrete Fourier transform	C06FUF
Discretize a 2nd order elliptic PDE on a rectangle	D03EEF
Computes probabilities for the standard Normal distribution	G01EAF
Computes probabilities for Student's t- distribution	G01EBF
Computes probabilities for (chi)^2 distribution	G01ECF
Computes probabilities for F- distribution	G01EDF
distribution	
Computes probabilities for the gamma distribution	G01EFF
Computes deviates for the standard Normal distribution	G01FAF
Computes deviates for Student's t- distribution	G01FBF
Computes deviates for the (chi)^2 distribution	G01FCF
Computes deviates for the F- distribution	G01FDF
Computes deviates for the beta distribution	G01FEF
Computes deviates for the gamma distribution	G01FFF
Computes probability for the bivariate Normal distribution	G01HAF

```
Pseudo-random real numbers, uniform                     G05CAF
distribution
over (0,1)

Pseudo-random real numbers, Normal                      G05DDF
distribution

Pseudo-random real numbers, Cauchy                      G05DFF
distribution

Pseudo-random real numbers, Weibull                     G05DPF
distribution

Pseudo-random integer from uniform                      G05DYF
distribution

Set up reference vector for multivariate Normal         G05EAF
distribution

distribution

distribution

Set up reference vector from supplied cumulative        G05EXF
distribution
function or probability distribution function

distribution
function

Generates a vector of random numbers from a uniform     G05FAF
distribution

distribution

Generates a vector of random numbers from a Normal      G05FDF
distribution

distribution

distribution

     2
(chi)  goodness of fit test, for standard continuous    G08CGF
distributions

Inverse                                                 G01F
distributions
```

```
of real matrix

All                                                       F02AGF
eigenvalues
and eigenvectors of real matrix

All                                                       F02AJF
eigenvalues
of complex matrix

All                                                       F02AKF
eigenvalues
and eigenvectors of complex matrix

All                                                       F02AWF
eigenvalues
of complex Hermitian matrix

All                                                       F02AXF
eigenvalues
and eigenvectors of complex Hermitian matrix

Selected                                                  F02BBF
eigenvalues
and eigenvectors of real symmetric matrix

All                                                       F02BJF
eigenvalues
and optionally eigenvectors of generalized ...

Selected                                                  F02FJF
eigenvalues
and eigenvectors of sparse symmetric eigenproblem

All eigenvalues and                                       F02ABF
eigenvectors
of real symmetric matrix

All eigenvalues and                                       F02AEF
eigenvectors
of generalized real eigenproblem of the form ...

All eigenvalues and                                       F02AGF
eigenvectors
of real matrix

All eigenvalues and                                       F02AKF
eigenvectors
of complex matrix
```

```
-distribution

LU                                                      F01BRF
factorization
of real sparse matrix

LU                                                      F01BSF
factorization
of real sparse matrix with known sparsity pattern

  T
LL                                                      F01MAF
factorization
of real sparse symmetric positive-definite matrix

   T
LDL                                                     F01MCF
factorization
of real symmetric positive-definite ...

QR                                                      F01QCF
factorization
of real m by n matrix (m>=n)

                             T
factorization
by F01QCF

factorization
by F01QCF

QR                                                      F01RCF
factorization
of complex m by n matrix (m>=n)

                                     H
factorization
by F01RCF

factorization
by F01RCF

LU                                                      F07ADF
factorization
of real m by n matrix (DGETRF)

Cholesky                                                F07FDF
factorization
of real symmetric positive-definite matrix ...
```

```
Multivariate time series,                               G13BAF
filtering
(pre-whitening) by an ARIMA model

1-D quadrature, adaptive,                               D01AJF
finite
interval, strategy due to Piessens and de Doncker, ...

1-D quadrature, adaptive,                               D01AKF
finite
interval, method suitable for oscillating functions

1-D quadrature, adaptive,                               D01ALF
finite
interval, allowing for singularities at user-specified

1-D quadrature, adaptive,                               D01ANF
finite
interval, weight function cos((omega)x) or sin...

1-D quadrature, adaptive,                               D01APF
finite
interval, weight function with end-point ...

1-D quadrature, adaptive,                               D01AQF
finite
interval, weight function 1/(x-c), Cauchy ...

ODEs, boundary value problem,                           D02GAF
finite
difference technique with deferred correction, simple .

ODEs, boundary value problem,                           D02GBF
finite
difference technique with deferred correction, general

finite/infinite
range, eigenvalue and eigenfunction, ...

ODEs, general nonlinear boundary value problem,         D02RAF
finite
difference technique with deferred correction, ...

Elliptic PDE, solution of                               D03EDF
finite
difference equations by a multigrid technique

                                             2
Fisher's
exact test
```

```
Interpolating functions,                                E01DAF
fitting
bicubic spline, data on rectangular grid

Least-squares curve                                     E02ADF
fit
, by polynomials, arbitrary data points

Evaluation of                                           E02AEF
fitted
polynomial in one variable from Chebyshev series form .

Least-squares polynomial                                E02AGF
fit
, values and derivatives may be constrained, arbitrary

Derivative of                                           E02AHF
fitted
polynomial in Chebyshev series form

Integral of                                             E02AJF
fitted
polynomial in Chebyshev series form

Evaluation of                                           E02AKF
fitted
polynomial in one variable, from Chebyshev series form

Least-squares curve cubic spline                        E02BAF
fit
(including interpolation)

Evaluation of                                           E02BBF
fitted
cubic spline, function only

Evaluation of                                           E02BCF
fitted
cubic spline, function and derivatives

Evaluation of                                           E02BDF
fitted
cubic spline, definite integral

Least-squares cubic spline curve                        E02BEF
fit
, automatic knot placement

Least-squares surface                                   E02DAF
```

```
floating-point
arithmetic

Safe range of complex                                     X02ANF
floating-point
arithmetic

Parameter of                                              X02BHF
floating-point
arithmetic model, b

Parameter of                                              X02BJF
floating-point
arithmetic model, p

Parameter of                                              X02BKF
floating-point
arithmetic model, e
                   min

Parameter of                                              X02BLF
floating-point
arithmetic model, e
                   max

Parameter of                                              X02DJF
floating-point
arithmetic model, ROUNDS

Univariate time series, update state set for              G13AGF
forecasting

Univariate time series,                                   G13AHF
forecasting
from state set

Univariate time series, state set and                     G13AJF
forecasts
, from fully specified seasonal ARIMA model

Multivariate time series, state set and                   G13BJF
forecasts
from fully specified multi-input model

Single 1-D real discrete                                  C06EAF
Fourier
transform, no extra workspace

Single 1-D Hermitian discrete                             C06EBF
Fourier
```

```
orthomax criterion

Generate                                                  F06AAF
real plane rotation (DROTG)

Initialise random number                                  G05CBF
generating
routines to give repeatable sequence

Initialise random number                                  G05CCF
generating
routines to give non-repeatable sequence

Save state of random number                               G05CFF
generating
routines

Restore state of random number                            G05CGF
generating
routines

Set up reference vector for                               G05ECF
generating
pseudo-random integers, Poisson distribution

Set up reference vector for                               G05EDF
generating
pseudo-random integers, binomial distribution

Generates                                                 G05FAF
a vector of random numbers from a uniform ...

Generates                                                 G05FBF
a vector of random numbers from an (negative) ...

Generates                                                 G05FDF
a vector of random numbers from a Normal distribution

Generates                                                 G05FEF
a vector of pseudo-random numbers from a beta ...

Generates                                                 G05FFF
a vector of pseudo-random numbers from a gamma ...

Generates                                                 G05HDF
a realisation of a multivariate time series from a ...

Gill-Miller
method
```

```
                    2
Performs the (chi)                                      G08CGF
goodness
of fit test, for standard continuous distributions

Goodness                                                G08
of fit tests

Unconstrained minimum, pre-conditioned conjugate        E04DGF
gradient
algorithm, function of several variables using 1st ...

Hankel                                                  S17DLF
           (j)
functions H      (z), j=1,2, ...
           (nu)+n

Elliptic PDE,                                           D03FAF
Helmholtz
equation, 3-D Cartesian co-ordinates

Hermite
, one variable

Single 1-D                                              C06EBF
Hermitian
discrete Fourier transform, no extra workspace

Multiple 1-D                                            C06FQF
Hermitian
discrete Fourier transforms

Complex conjugate of                                    C06GBF
Hermitian
sequence

Complex conjugate of multiple                           C06GQF
Hermitian
sequences

Convert                                                 C06GSF
Hermitian
sequences to general complex sequences

All eigenvalues of complex                              F02AWF
Hermitian
matrix

All eigenvalues and eigenvectors of complex             F02AXF
Hermitian
```

```
Exponential                                           S13AAF
integral
E (x)
 1

Cosine                                                S13ACF
integral
Ci(x)

Sine                                                  S13ADF
integral
Si(x)

Fresnel                                               S20ACF
integral
S(x)

Fresnel                                               S20ADF
integral
C(x)

Degenerate symmetrised elliptic                       S21BAF
integral
of 1st kind R (x,y)
             C

Symmetrised elliptic                                  S21BBF
integral
of 1st kind R (x,y,z)
             F

Symmetrised elliptic                                  S21BCF
integral
of 2nd kind R (x,y,z)
             D

Symmetrised elliptic                                  S21BDF
integral
of 3rd kind R (x,y,z,r)
             J

1-D quadrature,                                       D01GAF
integration
of function defined by data values, Gill-Miller ...

Numerical                                             D01
integration

Interpolating                                         E01BAF
functions, cubic spline interpolant, one variable
```

```
iterative
refinement

ODEs,                                                 D02BBF
IVP
, Runge-Kutta-Merson method, over a range, intermediate

ODEs,                                                 D02BHF
IVP
, Runge-Kutta-Merson method, until function of solution
is ...

ODEs,                                                 D02CJF
IVP
, Adams method, until function of solution is zero, ...

ODEs, stiff                                           D02EJF
IVP
, BDF method, until function of solution is zero, ...

Kelvin function                                       S19ADF
kei
x

Kelvin                                                S19AAF
function ber x

Kelvin                                                S19ABF
function bei x

Kelvin                                                S19ACF
function ker x

Kelvin                                                S19ADF
function kei x

Kendall/Spearman                                      G02BNF
non-parametric rank correlation ...

Kendall/Spearman                                      G02BQF
non-parametric rank correlation ...

Kelvin function                                       S19ACF
ker
x

Least-squares cubic spline curve fit, automatic       E02BEF
knot
placement
```

```
Least-squares                                          E02AGF
polynomial fit, values and derivatives may be ...

Least-squares                                          E02BAF
curve cubic spline fit (including interpolation)

Least-squares                                          E02BEF
cubic spline curve fit, automatic knot placement

Least-squares                                          E02DAF
surface fit, bicubic splines

Least-squares                                          E02DCF
surface fit by bicubic splines with automatic ...

Least-squares                                          E02DDF
surface fit by bicubic splines with automatic ...

Covariance matrix for nonlinear                        E04YCF
least-squares
problem

Least-squares                                          F04JGF
(if rank=n) or minimal least-squares (if ...

Least-squares (if rank=n) or minimal                   F04JGF
least-squares
(if rank<n) solution of m real equations ...

Sparse linear                                          F04QAF
least-squares
problem, m real equations in n unknowns

linear
problem

L -approximation by general                            E02GAF
 1
linear
function

Linear                                                 E04MBF
programming problem

Solution of complex simultaneous                       F04ADF
linear
equations with multiple right-hand sides

Solution of real simultaneous                          F04ARF
linear
```

```
equations, one right-hand side

linear
equations, one right-hand side using iterative ...

Solution of real simultaneous                          F04ATF
linear
equations, one right-hand side using iterative ...

Solution of real sparse simultaneous                   F04AXF
linear
equations (coefficient matrix already factorized)

linear
equations, one right-hand side

Real sparse symmetric positive-definite simultaneous   F04MAF
linear
equations (coefficient matrix already factorized by ...

Real sparse symmetric simultaneous                     F04MBF
linear
equations

linear
equations (coefficient matrix already factorized by ...

Sparse                                                 F04QAF
linear
least-squares problem, m real equations in n ...

Solution of real system of                             F07AEF
linear
equations, multiple right-hand sides, matrix already ..

linear
equations, multiple right-hand sides, matrix already ..

Simple                                                 G02CAF
linear
regression with constant term, no missing values

Fits a general (multiple)                              G02DAF
linear
regression model

Fits a general                                         G02DGF
linear
regression model for new dependent variable
```

```
Computes estimable function of a general              G02DNF
linear
regression model and its standard error

Fits a generalized                                    G02GBF
linear
model with binomial errors

Fits a generalized                                    G02GCF
linear
model with Poisson errors

Basic                                                 F06
Linear
Algebra Subprograms

2nd order Sturm-                                      D02KEF
Liouville
problem, regular/singular system, finite/infinite ...

Computes orthogonal rotations for                     G03BAF
loading
matrix, generalized orthomax criterion

Location                                              G08
tests

Log                                                   S14ABF
Gamma function

algebraico-
logarithmic
type

Computes upper and                                    G01EEF
lower
tail and probability density function probabilities for

LU                                                    F01BRF
factorization of real sparse matrix

LU                                                    F01BSF
factorization of real sparse matrix with known sparsity

LU                                                    F07ADF
factorization of real m by n matrix (DGETRF)

Machine                                               X02AJF
precision
```

Machine Constants	X02
Performs the Mann-Whitney U test on two independent samples	G08AHF
Computes the exact probabilities for the Mann-Whitney U statistic, no ties in pooled sample	G08AJF
Computes the exact probabilities for the Mann-Whitney U statistic, ties in pooled sample	G08AKF
Friedman two-way analysis of variance on k matched samples	G08AEF
Performs the Wilcoxon one-sample (matched pairs) signed rank test	G08AGF
Mathematical Constants	X01
Maximization	E04
Maximum number of decimal digits that can be represented	X02BEF
Mean , variance, skewness, kurtosis etc, one variable, from	G01AAF
Mean , variance, skewness, kurtosis etc, one variable, from	G01ADF
Computes a five-point summary (median , hinges and extremes)	G01ALF
Median test on two samples of unequal size	G08ACF
ODEs, IVP, Runge-Kutta- Merson method, over a range, intermediate output	D02BBF
ODEs, IVP, Runge-Kutta- Merson	D02BHF

method, until function of solution is zero (simple ...

Evaluation of a fitted bicubic spline at a E02DFF
mesh
of points

Miller
method

Least-squares (if rank=n) or F04JGF
minimal
least-squares (if rank<n) solution of m real ...

Minimization E04

Unconstrained E04DGF
minimum
, pre-conditioned conjugate gradient algorithm, ...

Unconstrained E04FDF
minimum
of a sum of squares, combined Gauss-Newton and ...

Unconstrained E04GCF
minimum
of a sum of squares, combined Gauss-Newton and ...

Minimum E04JAF
, function of several variables, quasi-Newton ...

Minimum E04UCF
, function of several variables, sequential QP method,

missing
values, overwriting input data

missing
values, preserving input data

Simple linear regression with constant term, no G02CAF
missing
values

Fits a general (multiple) linear regression G02DAF
model

Fits a general linear regression G02DGF
model
for new dependent variable

model
and its standard error

Fits a generalized linear G02GBF
model
with binomial errors

Fits a generalized linear G02GCF
model
with Poisson errors

model

model

Univariate time series, estimation, seasonal ARIMA G13AFF
model

model

model

model

Multivariate time series, estimation of multi-input G13BEF
model

model

Smallest positive X02AKF
model
number

Largest positive X02ALF
model
number

Parameter of floating-point arithmetic X02BHF
model
, b

Parameter of floating-point arithmetic X02BJF
model
, p

Parameter of floating-point arithmetic X02BKF
model
, e
min

and modified Newton algorithm using function values ...

Newton
algorithm using function values only
Newton
and quasi-Newton algorithm, using 1st derivatives

Newton
algorithm, using 1st derivatives

Minimum, function of several variables, quasi- E04JAF
Newton
algorithm, simple bounds, using function values only

Kendall/Spearman G02BNF
non-parametric
rank correlation coefficients, no missing ...

Kendall/Spearman G02BQF
non-parametric
rank correlation coefficients, no missing ...

Initialise random number generating routines to give G05CCF
non-repeatable
sequence

Univariate time series, seasonal and G13AAF
non-seasonal
differencing

Non-parametric G08
tests

Solution of system of C05NBF
nonlinear
equations using function values only

Solution of system of C05PBF
nonlinear
equations using 1st derivatives

nonlinear
problem

ODEs, general D02RAF
nonlinear
boundary value problem, finite difference technique ...

nonlinear
constraints, using function values and optionally ...

Covariance matrix for nonlinear least-squares problem	E04YCF
Nonlinear optimization	E04
Nonlinear regression	E04
Compute Euclidean norm of real vector (DNRM2)	F06EJF
Compute Euclidean norm of complex vector (DZNRM2)	F06JJF
Computes probabilities for the standard Normal distribution	G01EAF
Computes deviates for the standard Normal distribution	G01FAF
Computes probability for the bivariate Normal distribution	G01HAF
Pseudo-random real numbers, Normal distribution	G05DDF
Set up reference vector for multivariate Normal distribution	G05EAF
Pseudo-random multivariate Normal vector from reference vector	G05EZF
Generates a vector of random numbers from a Normal distribution	G05FDF
Numerical integration	D01

```
matrices, compute QB or Q B after ...

Operations with                                        F01QEF
orthogonal
matrices, form columns of Q after factorization ...

Computes                                               G03BAF
orthogonal
rotations for loading matrix, generalized orthomax ...

orthomax
criterion

oscillating
functions

Incomplete gamma functions                             S14BAF
P(a,x)
and Q(a,x)

Matrix-vector product, real symmetric                  F06PEF
packed
matrix (DSPMV)

Matrix-vector product, real triangular                 F06PHF
packed
matrix (DTPMV)

System of equations, real triangular                   F06PLF
packed
matrix (DTPSV)

Rank-1 update, real symmetric                          F06PQF
packed
matrix (DSPR)

Rank-2 update, real symmetric                          F06PSF
packed
matrix (DSPR2)

Matrix-vector product, complex Hermitian               F06SEF
packed
matrix (ZHPMV)

Matrix-vector product, complex triangular              F06SHF
packed
matrix (ZTPMV)

System of equations, complex triangular                F06SLF
packed
```

```
Invert a                                                M01ZAF
permutation

Interpolating functions, monotonicity-preserving,       E01BEF
piecewise
cubic Hermite, one variable

Piessens
and de Doncker, allowing for badly-behaved integrands

Generate real                                           F06AAF
plane
rotation (DROTG)
Apply real                                              F06EPF
plane
rotation (DROT)

Constructs a stem and leaf                              G01ARF
plot

Fits a generalized linear model with                    G02GCF
Poisson
errors

Poisson
distribution

All zeros of complex                                    C02AFF
polynomial
, modified Laguerre method

All zeros of real                                       C02AGF
polynomial
, modified Laguerre method

Least-squares curve fit, by                             E02ADF
polynomials
, arbitrary data points

Evaluation of fitted                                    E02AEF
polynomial
in one variable from Chebyshev series form ...

Least-squares                                           E02AGF
polynomial
fit, values and derivatives may be constrained, ...

Derivative of fitted                                    E02AHF
polynomial
```

```
Computes                                                G01EAF
probabilities
for the standard Normal distribution

Computes                                                G01EBF
probabilities
for Student's t-distribution

Computes                                                G01ECF
probabilities
         2
for (chi)  distribution

Computes                                                G01EDF
probabilities
for F-distribution

Computes upper and lower tail and                       G01EEF
probability
density function probabilities for the beta ...

probabilities
for the beta distribution

Computes                                                G01EFF
probabilities
for the gamma distribution

Computes                                                G01HAF
probability
for the bivariate Normal distribution

probability
distribution function

Computes the exact                                      G08AJF
probabilities
for the Mann-Whitney U statistic, no ties in ...

Computes the exact                                      G08AKF
probabilities
for the Mann-Whitney U statistic, ties in ...

Dot                                                     F06EAF
product
of two real vectors (DDOT)

Dot                                                     F06GAF
product
```

of two complex vectors, unconjugated (ZDOTU)

Dot F06GBF
product
of two complex vectors, conjugated (ZDOTC)

Matrix-vector F06PAF
product
, real rectangular matrix (DGEMV)

Matrix-vector F06PBF
product
, real rectangular band matrix (DGBMV)

Matrix-vector F06PCF
product
, real symmetric matrix (DSYMV)

Matrix-vector F06PDF
product
, real symmetric band matrix (DSBMV)

Matrix-vector F06PEF
product
, real symmetric packed matrix (DSPMV)

Matrix-vector F06PFF
product
, real triangular matrix (DTRMV)

Matrix-vector F06PGF
product
, real triangular band matrix (DTBMV)

Matrix-vector F06PHF
product
, real triangular packed matrix (DTPMV)

Matrix-vector F06SAF
product
, complex rectangular matrix (ZGEMV)

Matrix-vector F06SBF
product
, complex rectangular band matrix (ZGBMV)

Matrix-vector F06SCF
product
, complex Hermitian matrix (ZHEMV)

Matrix-vector F06SDF
product
, complex Hermitian band matrix (ZHBMV)

Matrix-vector F06SEF
product
, complex Hermitian packed matrix (ZHPMV)

Matrix-vector F06SFF
product
, complex triangular matrix (ZTRMV)

Matrix-vector F06SGF
product
, complex triangular band matrix (ZTBMV)

Matrix-vector F06SHF
product
, complex triangular packed matrix (ZTPMV)

Matrix-matrix F06YAF
product
, two real rectangular matrices (DGEMM)

Matrix-matrix F06YCF
product
, one real symmetric matrix, one real rectangular ...

Matrix-matrix F06YFF
product
, one real triangular matrix, one real rectangular ...

Matrix-matrix F06ZAF
product
, two complex rectangular matrices (ZGEMM)

Matrix-matrix F06ZCF
product
, one complex Hermitian matrix, one complex ...

Matrix-matrix F06ZFF
product
, one complex triangular matrix, one complex ...

Matrix-matrix F06ZTF
product
, one complex symmetric matrix, one complex ...

Linear E04MBF
programming

```
problem

Pseudo-random                                            G05CAF
real numbers, uniform distribution over (0,1)

Pseudo-random                                            G05DDF
real numbers, Normal distribution

Pseudo-random                                            G05DFF
real numbers, Cauchy distribution

Pseudo-random                                            G05DPF
real numbers, Weibull distribution

Pseudo-random                                            G05DYF
integer from uniform distribution

Pseudo-random                                            G05DZF
logical (boolean) value

Set up reference vector for generating                   G05ECF
pseudo-random
integers, Poisson distribution

Set up reference vector for generating                   G05EDF
pseudo-random
integers, binomial distribution

Pseudo-random                                            G05EHF
permutation of an integer vector

Pseudo-random                                            G05EJF
sample from an integer vector

Pseudo-random                                            G05EYF
integer from reference vector

Pseudo-random                                            G05EZF
multivariate Normal vector from reference vector

Generates a vector of                                    G05FEF
pseudo-random
numbers from a beta distribution

Generates a vector of                                    G05FFF
pseudo-random
numbers from a gamma distribution

Incomplete gamma functions P(a,x) and                    S14BAF
Q(a,x)
```

QP E04NAF
problem

Minimum, function of several variables, sequential E04UCF
QP
method, nonlinear constraints, using function values
and ...

QR F01QCF
factorization of real m by n matrix (m<=n)

QR F01RCF
factorization of complex m by n matrix (m<=n)

1-D D01AJF
quadrature
, adaptive, finite interval, strategy due to ...

1-D D01AKF
quadrature
, adaptive, finite interval, method suitable for ...

1-D D01ALF
quadrature
, adaptive, finite interval, allowing for ...
1-D D01AMF
quadrature
, adaptive, infinite or semi-infinite interval

1-D D01ANF
quadrature
, adaptive, finite interval, weight function
cos((omega)x) ...

1-D D01APF
quadrature
, adaptive, finite interval, weight function with ...

1-D D01AQF
quadrature
, adaptive, finite interval, weight function ...

1-D D01ASF
quadrature
, adaptive, semi-infinite interval, weight function ...

Pre-computed weights and abscissae for Gaussian D01BBF
quadrature
rules, restricted choice of rule

```
Multi-dimensional adaptive                                D01FCF
quadrature
over hyper-rectangle

1-D                                                       D01GAF
quadrature
, integration of function defined by data values, ...

Multi-dimensional                                         D01GBF
quadrature
over hyper-rectangle, Monte Carlo method

quasi-Newton
algorithm, using 1st derivatives

Minimum, function of several variables,                   E04JAF
quasi-Newton
algorithm, simple bounds, using function values ...

QZ
algorithm, real matrices

Pseudo-                                                   G05CAF
random
real numbers, uniform distribution over (0,1)

Initialise                                                G05CBF
random
number generating routines to give repeatable sequence

Initialise                                                G05CCF
random
number generating routines to give non-repeatable ...

Save state of                                             G05CFF
random
number generating routines

Restore state of                                          G05CGF
random
number generating routines

Pseudo-                                                   G05DDF
random
real numbers, Normal distribution

Pseudo-                                                   G05DFF
random
real numbers, Cauchy distribution
```

Pseudo-
random
real numbers, Weibull distribution G05DPF

Pseudo-
random
integer from uniform distribution G05DYF

Pseudo-
random
logical (boolean) value G05DZF

Set up reference vector for generating pseudo-
random
integers, Poisson distribution G05ECF

Set up reference vector for generating pseudo-
random
integers, binomial distribution G05EDF

Pseudo-
random
permutation of an integer vector G05EHF

Pseudo-
random
sample from an integer vector G05EJF

Pseudo-
random
integer from reference vector G05EYF

Pseudo-
random
multivariate Normal vector from reference vector G05EZF

Generates a vector of
random
numbers from a uniform distribution G05FAF

Generates a vector of
random
numbers from an (negative) exponential distribution G05FBF

Generates a vector of
random
numbers from a Normal distribution G05FDF

Generates a vector of pseudo- G05FEF

random
numbers from a beta distribution

Generates a vector of pseudo- G05FFF
random
numbers from a gamma distribution

ODEs, IVP, Runge-Kutta-Merson method, over a D02BBF
range
, intermediate output

range
, eigenvalue and eigenfunction, user-specified ...

Safe X02AMF
range
of floating-point arithmetic

Safe X02ANF
range
of complex floating-point arithmetic

Least-squares (if F04JGF
rank=n)
or minimal least-squares (if rank<n) ...

rank
<n) solution of m real equations in n unknowns, ...

rank
<=n,m>=n

Rank-1 F06PMF
update, real rectangular matrix (DGER)

Rank-1 F06PPF
update, real symmetric matrix (DSYR)

Rank-1 F06PQF
update, real symmetric packed matrix (DSPR)

Rank-2 F06PRF
update, real symmetric matrix (DSYR2)

Rank-2 F06PSF
update, real symmetric packed matrix (DSPR2)

Rank-1 F06SMF
update, complex rectangular matrix, unconjugated ...

Rank-1 F06SNF
update, complex rectangular matrix, conjugated vector .

Rank-1 F06SPF
update, complex Hermitian matrix (ZHER)

Rank-1 F06SQF
update, complex Hermitian packed matrix (ZHPR)

Rank-2 F06SRF
update, complex Hermitian matrix (ZHER2)

Rank-2 F06SSF
update, complex Hermitian packed matrix (ZHPR2)

Rank-k F06YPF
update of a real symmetric matrix (DSYRK)

Rank-2k F06YRF
update of a real symmetric matrix (DSYR2K)

Rank-k F06ZPF
update of a complex Hermitian matrix (ZHERK)

Rank-2k F06ZRF
update of a complex Hermitian matrix (ZHER2K)

Rank-k F06ZUF
update of a complex symmetric matrix (ZSYRK)

Rank-2k F06ZWF
update of a complex symmetric matrix (ZHER2K)

Kendall/Spearman non-parametric G02BNF
rank
correlation coefficients, no missing values,
overwriting ...

Kendall/Spearman non-parametric G02BQF
rank
correlation coefficients, no missing values, preserving

rank
test

Rank M01DAF
a vector, real numbers

Rank M01DEF
rows of a matrix, real numbers

```
Rank-1 update, complex                                  F06SMF
rectangular
matrix, unconjugated vector (ZGERU)

Rank-1 update, complex                                  F06SNF
rectangular
matrix, conjugated vector (ZGERC)

Matrix-matrix product, two real                         F06YAF
rectangular
matrices (DGEMM)

rectangular
matrix (DSYMM)

rectangular
matrix (DTRMM)

Matrix-matrix product, two complex                      F06ZAF
rectangular
matrices (ZGEMM)

rectangular
matrix (ZHEMM)

rectangular
matrix (ZTRMM)

rectangular
matrix (ZSYMM)

Set up                                                  G05EAF
reference
vector for multivariate Normal distribution

Set up                                                  G05ECF
reference
vector for generating pseudo-random integers, ...

Set up                                                  G05EDF
reference
vector for generating pseudo-random integers, ...

Set up                                                  G05EXF
reference
vector from supplied cumulative distribution ...

Pseudo-random integer from                              G05EYF
reference
```

```
right-hand
side using iterative refinement

Solution of real simultaneous linear equations, one      F04ATF
right-hand
side using iterative refinement

right-hand
side

Solves a system of equations with multiple               F06YJF
right-hand
sides, real triangular coefficient matrix (DTRSM)

Solves system of equations with multiple                 F06ZJF
right-hand
sides, complex triangular coefficient matrix (ZTRSM)

Solution of real system of linear equations, multiple    F07AEF
right-hand
sides, matrix already factorized by F07ADF (DGETRS)

right-hand
sides, matrix already factorized by F07FDF (DPOTRS)

Generate real plane                                      F06AAF
rotation
(DROTG)

Apply real plane                                         F06EPF
rotation
(DROT)

Computes orthogonal                                      G03BAF
rotations
for loading matrix, generalized orthomax criterion

rules
, restricted choice of rule

rule
ODEs, IVP,                                               D02BBF
Runge-Kutta-Merson
method, over a range, intermediate output

ODEs, IVP,                                               D02BHF
Runge-Kutta-Merson
method, until function of solution is zero ...
```

```
Safe                                                       X02AMF
range of floating-point arithmetic

Safe                                                       X02ANF
range of complex floating-point arithmetic

Pseudo-random                                              G05EJF
sample
from an integer vector

Sign test on two paired                                    G08AAF
samples

Median test on two                                         G08ACF
samples
of unequal size

Friedman two-way analysis of variance on k matched         G08AEF
samples

Kruskal-Wallis one-way analysis of variance on k           G08AFF
samples
of unequal size

Performs the Wilcoxon one-                                 G08AGF
sample
(matched pairs) signed rank test

Performs the Mann-Whitney U test on two independent        G08AHF
samples

sample

sample

Univariate time series,                                    G13ABF
sample
autocorrelation function

Univariate time series, smoothed                           G13CBF
sample
spectrum using spectral smoothing by the trapezium ...

Multivariate time series, smoothed                         G13CDF
sample
cross spectrum using spectral smoothing by the ...

Add                                                        F06ECF
scalar
times real vector to real vector (DAXPY)
```

```
Multiply real vector by                              F06EDF
scalar
(DSCAL)

Add                                                  F06GCF
scalar
times complex vector to complex vector (ZAXPY)

Multiply complex vector by complex                   F06GDF
scalar
(ZSCAL)

Multiply complex vector by real                      F06JDF
scalar
(ZDSCAL)

scattered
data

Univariate time series,                              G13AAF
seasonal
and non-seasonal differencing

Univariate time series, seasonal and non-            G13AAF
seasonal
differencing

Univariate time series, preliminary estimation,      G13ADF
seasonal
ARIMA model

Univariate time series, estimation,                  G13AFF
seasonal
ARIMA model

seasonal
ARIMA model

1-D quadrature, adaptive, infinite or                D01AMF
semi-infinite
interval

1-D quadrature, adaptive,                            D01ASF
semi-infinite
interval, weight function cos((omega)x) ...
Complex conjugate of Hermitian                       C06GBF
sequence

Complex conjugate of complex                         C06GCF
```

sequence

Complex conjugate of multiple Hermitian sequences — C06GQF

Convert Hermitian
sequences
to general complex sequences — C06GSF

Convert Hermitian sequences to general complex sequences — C06GSF

sequence

sequence

Minimum, function of several variables,
sequential
QP method, nonlinear constraints, using function ... — E04UCF

Interpolating functions, modified
Shepard's
method, two variables — E01SEF

Sign
test on two paired samples — G08AAF

Performs the Wilcoxon one-sample (matched pairs)
signed
rank test — G08AGF

Solution of complex
simultaneous
linear equations with multiple right-hand sides — F04ADF

Solution of real
simultaneous
linear equations, one right-hand side — F04ARF

Solution of real symmetric positive-definite
simultaneous
linear equations, one right-hand side using ... — F04ASF

Solution of real
simultaneous
linear equations, one right-hand side using ... — F04ATF

Solution of real sparse
simultaneous
linear equations (coefficient matrix already ... — F04AXF

by the trapezium frequency (Daniell) window

Multivariate time series, G13CDF
smoothed
sample cross spectrum using spectral smoothing by ...

smoothing
by the trapezium frequency (Daniell) window

Sort E02ZAF
2-D data into panels for fitting bicubic splines

Sort M01CAF
a vector, real numbers

LU factorization of real F01BRF
sparse
matrix

LU factorization of real F01BSF
sparse
matrix with known sparsity pattern

LU factorization of real sparse matrix with known F01BSF
sparsity
pattern

LL^T factorization of real F01MAF
sparse
symmetric positive-definite matrix

Selected eigenvalues and eigenvectors of F02FJF
sparse
symmetric eigenproblem

Solution of real F04AXF
sparse
simultaneous linear equations (coefficient matrix ...

Real F04MAF
sparse
symmetric positive-definite simultaneous linear ...

Real F04MBF
sparse
symmetric simultaneous linear equations

Sparse F04QAF
linear least-squares problem, m real equations in ...

```
Kendall/                                                G02BNF
Spearman
non-parametric rank correlation coefficients, no ...

Kendall/                                                G02BQF
Spearman
non-parametric rank correlation coefficients, no ...

Approximation of                                        S
special
functions

Univariate time series, smoothed sample                 G13CBF
spectrum
using spectral smoothing by the trapezium frequency ...

spectral
smoothing by the trapezium frequency (Daniell) window

Multivariate time series, smoothed sample cross         G13CDF
spectrum
using spectral smoothing by the trapezium frequency ...

spectral
smoothing by the trapezium frequency (Daniell) window

Interpolating functions, cubic                          E01BAF
spline
interpolant, one variable

Interpolating functions, fitting bicubic                E01DAF
spline
, data on rectangular grid

Least-squares curve cubic                               E02BAF
spline
fit (including interpolation)

Evaluation of fitted cubic                              E02BBF
spline
, function only

Evaluation of fitted cubic                              E02BCF
spline
, function and derivatives

Evaluation of fitted cubic                              E02BDF
spline
, definite integral
```

Least-squares cubic spline curve fit, automatic knot placement	E02BEF
Least-squares surface fit, bicubic splines	E02DAF
Least-squares surface fit by bicubic splines with automatic knot placement, data on rectangular grid	E02DCF
Least-squares surface fit by bicubic splines with automatic knot placement, scattered data	E02DDF
Evaluation of a fitted bicubic spline at a vector of points	E02DEF
Evaluation of a fitted bicubic spline at a mesh of points	E02DFF
Sort 2-D data into panels for fitting bicubic splines	E02ZAF
B-splines	E02
Least-squares curve fit, by polynomials, arbitrary data points	E02ADF
Least-squares polynomial fit, values and derivatives may be ...	E02AGF
Least-squares curve cubic spline fit (including interpolation)	E02BAF
Least-squares cubic spline curve fit, automatic knot placement	E02BEF
Least-squares surface fit, bicubic splines	E02DAF
Least-	E02DCF

```
squares
surface fit by bicubic splines with automatic knot ...

Least-                                                E02DDF
squares
surface fit by bicubic splines with automatic knot ...
Unconstrained minimum of a sum of                     E04FDF
squares
, combined Gauss-Newton and modified Newton algorithm .

Unconstrained minimum of a sum of                     E04GCF
squares
, combined Gauss-Newton and quasi-Newton algorithm, ...

Covariance matrix for nonlinear least-                E04YCF
squares
problem

Least-                                                F04JGF
squares
(if rank=n) or minimal least-squares (if ...

Least-squares (if rank=n) or minimal least-           F04JGF
squares
(if rank<n) solution of m real equations in n ...

Sparse linear least-                                  F04QAF
squares
problem, m real equations in n unknowns

Computes probabilities for the                        G01EAF
standard
Normal distribution

Computes deviates for the                             G01FAF
standard
Normal distribution

standard
error

                  2
Performs the (chi)  goodness of fit test, for         G08CGF
standard
continuous distributions

Calculates                                            G02FAF
standardized
residuals and influence statistics
```

```
Calculates standardized residuals and influence         G02FAF
statistics

statistic
, no ties in pooled sample

statistic
, ties in pooled sample
Constructs a                                             G01ARF
stem
and leaf plot

ODEs,                                                    D02EJF
stiff
IVP, BDF method, until function of solution is zero, ..

Computes probabilities for                               G01EBF
Student's
t-distribution

Computes deviates for                                    G01FBF
Student's
t-distribution

2nd order                                                D02KEF
Sturm-Liouville
problem, regular/singular system, ...

Basic Linear Algebra                                     F06
Subprograms

Unconstrained minimum of a                               E04FDF
sum
of squares, combined Gauss-Newton and modified Newton .

Unconstrained minimum of a                               E04GCF
sum
of squares, combined Gauss-Newton and quasi-Newton ...

Sum                                                      F06EKF
the absolute values of real vector elements (DASUM)

Sum                                                      F06JKF
the absolute values of complex vector elements (DZASUM)

Computes a five-point                                    G01ALF
summary
(median, hinges and extremes)

Least-squares                                            E02DAF
```

```
surface
fit, bicubic splines

Least-squares                                          E02DCF
surface
fit by bicubic splines with automatic knot placement, .

Least-squares                                          E02DDF
surface
fit by bicubic splines with automatic knot placement, .
SVD                                                    F02WEF
of real matrix

SVD                                                    F02XEF
of complex matrix

Swap                                                   F06EGF
two real vectors (DSWAP)

Swap                                                   F06GGF
two complex vectors (ZSWAP)

Fresnel integral                                       S20ACF
S(x)

  T
LL  factorization of real sparse                       F01MAF
symmetric
positive-definite matrix

   T
LDL  factorization of real                             F01MCF
symmetric
positive-definite variable-bandwidth matrix

All eigenvalues of real                                F02AAF
symmetric
matrix

All eigenvalues and eigenvectors of real               F02ABF
symmetric
matrix

symmetric
and B is positive-definite

symmetric
and B is positive-definite

Selected eigenvalues and eigenvectors of real          F02BBF
```

```
symmetric
matrix

Selected eigenvalues and eigenvectors of sparse         F02FJF
symmetric
eigenproblem

Solution of real                                        F04ASF
symmetric
positive-definite simultaneous linear equations, ...

Solution of real                                        F04FAF
symmetric
positive-definite tridiagonal simultaneous linear ...

Real sparse                                             F04MAF
symmetric
positive-definite simultaneous linear equations ...

Real sparse                                             F04MBF
symmetric
simultaneous linear equations

Solution of real                                        F04MCF
symmetric
positive-definite variable-bandwidth simultaneous ...

Matrix-vector product, real                             F06PCF
symmetric
matrix (DSYMV)

Matrix-vector product, real                             F06PDF
symmetric
band matrix (DSBMV)

Matrix-vector product, real                             F06PEF
symmetric
packed matrix (DSPMV)

Rank-1 update, real                                     F06PPF
symmetric
matrix (DSYR)

Rank-1 update, real                                     F06PQF
symmetric
packed matrix (DSPR)

Rank-2 update, real                                     F06PRF
symmetric
matrix (DSYR2)
```

```
Rank-2 update, real                                     F06PSF
symmetric
packed matrix (DSPR2)

Matrix-matrix product, one real                         F06YCF
symmetric
matrix, one real rectangular matrix (DSYMM)

Rank-k update of a real                                 F06YPF
symmetric
matrix (DSYRK)

Rank-2k update of a real                                F06YRF
symmetric
matrix (DSYR2K)

Matrix-matrix product, one complex                      F06ZTF
symmetric
matrix, one complex rectangular matrix (ZSYMM)

Rank-k update of a complex                              F06ZUF
symmetric
matrix (ZSYRK)

Rank-2k update of a complex                             F06ZWF
symmetric
matrix (ZHER2K)

Cholesky factorization of real                          F07FDF
symmetric
positive-definite matrix (DPOTRF)

Solution of real                                        F07FEF
symmetric
positive-definite system of linear equations, ...

Degenerate                                              S21BAF
symmetrised
elliptic integral of 1st kind R (x,y)
                               C

Symmetrised                                             S21BBF
elliptic integral of 1st kind R (x,y,z)
                               F

Symmetrised                                             S21BCF
elliptic integral of 2nd kind R (x,y,z)
                               D
```

Computes probabilities for Student's t-distribution G01EBF

Computes deviates for Student's t-distribution G01FBF

table

Frequency table from raw data G01AEF

Two-way contingency table analysis, with (chi)^{2}/Fisher's exact test G01AFF

Computes upper and lower tail and probability density function probabilities for the G01EEF

2

test

Sign test on two paired samples G08AAF

Median test on two samples of unequal size G08ACF

test

Performs the Mann-Whitney U test on two independent samples G08AHF

Performs the (chi)^{2} goodness of fit test, for standard continuous distributions G08CGF

Goodness of fit tests G08

Location tests G08

Non-parametric G08

tests

ties
in pooled sample

ties
in pooled sample

Generates a realisation of a multivariate G05HDF
time
series from a VARMA model

Univariate G13AAF
time
series, seasonal and non-seasonal differencing

Univariate G13ABF
time
series, sample autocorrelation function

Univariate G13ACF
time
series, partial autocorrelations from autocorrelations

Univariate G13ADF
time
series, preliminary estimation, seasonal ARIMA model

Univariate G13AFF
time
series, estimation, seasonal ARIMA model

Univariate G13AGF
time
series, update state set for forecasting

Univariate G13AHF
time
series, forecasting from state set

Univariate G13AJF
time
series, state set and forecasts, from fully specified .

Univariate G13ASF
time
series, diagnostic checking of residuals, following
G13AFF

Multivariate G13BAF

Single 1-D Hermitian discrete Fourier transform , no extra workspace	C06EBF
Single 1-D complex discrete Fourier transform , no extra workspace	C06ECF
Multiple 1-D real discrete Fourier transforms	C06FPF
Multiple 1-D Hermitian discrete Fourier transforms	C06FQF
Multiple 1-D complex discrete Fourier transforms	C06FRF
2-D complex discrete Fourier transform	C06FUF
transform)	
trapezium frequency (Daniell) window trapezium frequency (Daniell) window	
Matrix-vector product, real triangular matrix (DTRMV)	F06PFF
Matrix-vector product, real triangular band matrix (DTBMV)	F06PGF
Matrix-vector product, real triangular packed matrix (DTPMV)	F06PHF
System of equations, real triangular matrix (DTRSV)	F06PJF
System of equations, real triangular band matrix (DTBSV)	F06PKF
System of equations, real triangular packed matrix (DTPSV)	F06PLF

```
Matrix-vector product, complex                         F06SFF
triangular
matrix (ZTRMV)

Matrix-vector product, complex                         F06SGF
triangular
band matrix (ZTBMV)

Matrix-vector product, complex                         F06SHF
triangular
packed matrix (ZTPMV)

System of equations, complex                           F06SJF
triangular
matrix (ZTRSV)

System of equations, complex                           F06SKF
triangular
band matrix (ZTBSV)

System of equations, complex                           F06SLF
triangular
packed matrix (ZTPSV)

Matrix-matrix product, one real                        F06YFF
triangular
matrix, one real rectangular matrix (DTRMM)

triangular
coefficient matrix (DTRSM)

Matrix-matrix product, one complex                     F06ZFF
triangular
matrix, one complex rectangular matrix (ZTRMM)

triangular
coefficient matrix (ZTRSM)

Solution of real symmetric positive-definite           F04FAF
tridiagonal
simultaneous linear equations, one right-hand side

Two-way                                                G01AFF
                                      2
contingency table analysis, with (chi)  ...

Sign test on                                           G08AAF
two
paired samples
```

Rank-2 F06PRF
update
, real symmetric matrix (DSYR2)

Rank-2 F06PSF
update
, real symmetric packed matrix (DSPR2)

Rank-1 F06SMF
update
, complex rectangular matrix, unconjugated vector
(ZGERU)

Rank-1 F06SNF
update
, complex rectangular matrix, conjugated vector (ZGERC)

Rank-1 F06SPF
update
, complex Hermitian matrix (ZHER)

Rank-1 F06SQF
update
, complex Hermitian packed matrix (ZHPR)

Rank-2 F06SRF
update
, complex Hermitian matrix (ZHER2)

Rank-2 F06SSF
update
, complex Hermitian packed matrix (ZHPR2)
Rank-k F06YPF
update
of a real symmetric matrix (DSYRK)

Rank-2k F06YRF
update
of a real symmetric matrix (DSYR2K)

Rank-k F06ZPF
update
of a complex Hermitian matrix (ZHERK)

Rank-2k F06ZRF
update
of a complex Hermitian matrix (ZHER2K)

Rank-k F06ZUF
update

of a complex symmetric matrix (ZSYRK)

Rank-2k update of a complex symmetric matrix (ZHER2K) — F06ZWF

Univariate time series, update state set for forecasting — G13AGF

Computes upper and lower tail and probability density function ... — G01EEF

Input/output utilities — X04

Mean, variance, skewness, kurtosis etc, one variable, from raw data — G01AAF

Mean, variance, skewness, kurtosis etc, one variable, from ... — G01ADF

Friedman two-way analysis of variance on k matched samples — G08AEF

Kruskal-Wallis one-way analysis of variance on k samples of unequal size — G08AFF

VARMA model

Circular convolution or correlation of two real vectors, no extra workspace — C06EKF

Evaluation of a fitted bicubic spline at a vector of points — E02DEF

Dot product of two real vectors (DDOT) — F06EAF

Add scalar times real vector — F06ECF

to real vector (DAXPY)

Add scalar times real vector to real vector (DAXPY) F06ECF

Multiply real vector by scalar (DSCAL) F06EDF

Copy real vector (DCOPY) F06EFF

Swap two real vectors (DSWAP) F06EGF

Compute Euclidean norm of real vector (DNRM2) F06EJF

Sum the absolute values of real vector elements (DASUM) F06EKF

Dot product of two complex vectors , unconjugated (ZDOTU) F06GAF

Dot product of two complex vectors , conjugated (ZDOTC) F06GBF

Add scalar times complex vector to complex vector (ZAXPY) F06GCF

Add scalar times complex vector to complex vector (ZAXPY) F06GCF

Multiply complex vector by complex scalar (ZSCAL) F06GDF

Copy complex vector (ZCOPY) F06GFF

```
Swap two complex                                         F06GGF
vectors
(ZSWAP)

Multiply complex                                         F06JDF
vector
by real scalar (ZDSCAL)

Compute Euclidean norm of complex                        F06JJF
vector
(DZNRM2)

Sum the absolute values of complex                       F06JKF
vector
elements (DZASUM)

Index, real                                              F06JLF
vector
element with largest absolute value (IDAMAX)

Index, complex                                           F06JMF
vector
element with largest absolute value (IZAMAX)

Matrix-                                                  F06PAF
vector
product, real rectangular matrix (DGEMV)

Matrix-                                                  F06PBF
vector
product, real rectangular band matrix (DGBMV)

Matrix-                                                  F06PCF
vector
product, real symmetric matrix (DSYMV)

Matrix-                                                  F06PDF
vector
product, real symmetric band matrix (DSBMV)

Matrix-                                                  F06PEF
vector
product, real symmetric packed matrix (DSPMV)

Matrix-                                                  F06PFF
vector
product, real triangular matrix (DTRMV)

Matrix-                                                  F06PGF
vector
```

product, real triangular band matrix (DTBMV)

Matrix-
vector
product, real triangular packed matrix (DTPMV) F06PHF

Matrix-
vector
product, complex rectangular matrix (ZGEMV) F06SAF

Matrix-
vector
product, complex rectangular band matrix (ZGBMV) F06SBF

Matrix-
vector
product, complex Hermitian matrix (ZHEMV) F06SCF

Matrix-
vector
product, complex Hermitian band matrix (ZHBMV) F06SDF

Matrix-
vector
product, complex Hermitian packed matrix (ZHPMV) F06SEF

Matrix-
vector
product, complex triangular matrix (ZTRMV) F06SFF

Matrix-
vector
product, complex triangular band matrix (ZTBMV) F06SGF

Matrix-
vector
product, complex triangular packed matrix (ZTPMV) F06SHF

vector
(ZGERU)

Rank-1 update, complex rectangular matrix, conjugated
vector
(ZGERC) F06SNF

Set up reference
vector
for multivariate Normal distribution G05EAF

Set up reference
vector G05ECF

```
for generating pseudo-random integers, Poisson ...

Set up reference                                         G05EDF
vector
for generating pseudo-random integers, binomial ...

Pseudo-random permutation of an integer                  G05EHF
vector

Pseudo-random sample from an integer                     G05EJF
vector

Set up reference                                         G05EXF
vector
from supplied cumulative distribution function or ...

Pseudo-random integer from reference                     G05EYF
vector

Pseudo-random multivariate Normal                        G05EZF
vector
from reference vector

vector

Generates a                                              G05FAF
vector
of random numbers from a uniform distribution

Generates a                                              G05FBF
vector
of random numbers from an (negative) exponential ...

Generates a                                              G05FDF
vector
of random numbers from a Normal distribution

Generates a                                              G05FEF
vector
of pseudo-random numbers from a beta distribution

Generates a                                              G05FFF
vector
of pseudo-random numbers from a gamma distribution

Sort a                                                   M01CAF
vector
, real numbers

Rank a                                                   M01DAF
```

```
vector
, real numbers

Rearrange a                                                M01EAF
vector
according to given ranks, real numbers

Kruskal-                                                   G08AFF
Wallis
one-way analysis of variance on k samples of unequal ..

Pseudo-random real numbers,                                G05DPF
Weibull
distribution

1-D quadrature, adaptive, finite interval,                 D01ANF
weight
function cos((omega)x) or sin((omega)x)

1-D quadrature, adaptive, finite interval,                 D01APF
weight
function with end-point singularities of ...

1-D quadrature, adaptive, finite interval,                 D01AQF
weight
function 1/(x-c), Cauchy principal value ...

1-D quadrature, adaptive, semi-infinite interval,          D01ASF
weight
function cos((omega)x) or sin((omega)x)

Pre-computed                                               D01BBF
weights
and abscissae for Gaussian quadrature rules, ...

Computes (optionally                                       G02BXF
weighted)
correlation and covariance matrices

Multivariate time series, filtering (pre-                  G13BAF
whitening)
by an ARIMA model

Performs the Mann-                                         G08AHF
Whitney
U test on two independent samples

Computes the exact probabilities for the Mann-             G08AJF
Whitney
U statistic, no ties in pooled sample
```

```
Computes the exact probabilities for the Mann-          G08AKF
Whitney
U statistic, ties in pooled sample

Performs the                                            G08AGF
Wilcoxon
one-sample (matched pairs) signed rank test

window

window

Two-way contingency table analysis, with                G01AFF
     2
(chi)
/Fisher's exact test

Computes probabilities for                              G01ECF
     2
(chi)
distribution

Computes deviates for the                               G01FCF
     2
(chi)
distribution

Performs the                                            G08CGF
     2
(chi)
goodness of fit test, for standard continuous ...

All                                                     C02AFF
zeros
of complex polynomial, modified Laguerre method

All                                                     C02AGF
zeros
of real polynomial, modified Laguerre method

Zero                                                    C05ADF
of continuous function in given interval, Bus and
Dekker ...

zero
(simple driver)

zero
, intermediate output (simple driver)
```

```
          zero
          , intermediate output (simple driver)

\end{verbatim}
\endscroll
\end{page}
```

———-

NAG Documentation: conversion

— nagaux.ht —

```
\begin{page}{manpageXXconvert}{NAG Documentation: conversion}
\beginscroll
\begin{verbatim}


     CONVERSION(3NAG)  Foundation Library (12/10/92)  CONVERSION(3NAG)



          Introduction              Converting from the Workstation Library
          Converting from the Workstation Library

          The NAG Foundation Library is a successor product to an earlier,
          smaller subset of the full NAG Fortran Library which was called
          the NAG Workstation Library. The Foundation Library has been
          designed to be upwards compatible, in terms of functionality,
          with the Workstation Library. However some routines that were
          present in the Workstation Library have been replaced by more up-
          to-date routines from the NAG Fortran Library, which provide
          improved algorithms or software design.

          The list below gives the names of those routines which were
          available in the Workstation Library, but are not included in the
          Foundation Library. For each such routine, it also gives the name
          of the routine in the Foundation Library which best covers the
          same functionality.

              Workstation    Foundation
              Library        Library

              C02AEF         C02AGF
```

D02CBF	D02CJF
D02CHF	D02CJF
D02EBF	D02EJF
D02EHF	D02EJF
D02HAF	D02GAF
D02HBF	D02RAF
D02SAF	D02RAF
E02DBF	E02DEF
E04VDF	E04UCF
E04ZCF	E04UCF (see Note 1)
F01BTF	F07ADF
F01BXF	F07FDF
F02WAF	F02WEF
F04AYF	F07AEF
F04AZF	F07FEF
F04YAF	G02DAF
G01ABF	G02BXF (with M = 2)
G01BAF	G01EBF
G01BBF	G01EDF
G01BCF	G01ECF
G01BDF	G01EEF
G01CAF	G01FBF
G01CBF	G01FDF
G01CCF	G01FCF
G01CDF	G01FEF

```
    G01CEF          G01FAF

    G02BAF          G02BXF

    G02BGF          G02BXF

    G02CEF          G02DAF (see Note 2)

    G02CGF          G02DAF

    G02CJF          G02DAF

    G05DBF          G05FBF

    G05DCF          G05CAF (see Note 3)

    G05DEF          G05FFF

    G05DHF          G05FFF (see Note 4)

    G05EGF          G05HDF

    G05EWF          G05HDF

    G08ABF          G08AGF

    G08ADF          G08AHF

    M01AKF          M01DAF

    M01APF          M01CAF

    S15ABF          G01EAF

    S15ACF          G01EAF

    X02AAF          X02AJF

    X02ACF          X02ALF

Notes:

1.    E04ZCF checks user-supplied routines for evaluating the
      first derivatives of the objective function and constraint
      functions supplied to E04VDF. This functionality is now
      provided by E04UCF, using the optional parameters Verify
      Objective Gradients and Verify Constraint Gradients.

2.    G02CEF selects variables to be included in a linear
      regression performed by G02CGF. This functionality is now
```

```
                provided by the parameter ISX of G02DAF.

          3.    A call to G05DCF can be replaced by a simple transformation
                of the result of a call to G05CAF. The statement
                     X = G05DCF(A,B)
                can be replaced by the statements

                     X = G05CAF(X)
                     X = A + B*LOG(X/(1.0D0-X))

                                                        2
          4.    G05DHF generates random numbers from a (chi)  distribution
                with N degrees of freedom. This can be achieved by calling
                G05FFF with the values DBLE(N)/2.0D0 and 2.0D0 for the
                parameters A and B respectively.

\end{verbatim}
\endscroll
\end{page}
```

22.2 nagc.ht

Zeros of Polynomials

— nagc.ht —

```
\begin{page}{manpageXXc02}{NAG Documentation: c02}
\beginscroll
\begin{verbatim}



     C02(3NAG)        Foundation Library (12/10/92)        C02(3NAG)



          C02 -- Zeros of Polynomials                  Introduction -- C02
                                    Chapter C02
                               Zeros of Polynomials

          1. Scope of the Chapter

          This chapter is concerned with computing the zeros of a
          polynomial with real or complex coefficients.
```

2. Background to the Problems

Let $f(z)$ be a polynomial of degree n with complex coefficients a_i:

$$f(z)==a_0 z^n +a_1 z^{n-1} +a_2 z^{n-2} +...+a_{n-1} z+a_n , \quad a_0 /=0.$$

A complex number z_1 is called a zero of $f(z)$ (or equivalently a root of the equation $f(z)=0$), if:

$$f(z_1)=0.$$

If z_1 is a zero, then $f(z)$ can be divided by a factor $(z-z_1)$:

$$f(z)=(z-z_1)f_1(z) \qquad (1)$$

where $f_1(z)$ is a polynomial of degree $n-1$. By the Fundamental Theorem of Algebra, a polynomial $f(z)$ always has a zero, and so the process of dividing out factors $(z-z_i)$ can be continued until we have a complete factorization of $f(z)$

$$f(z)==a_0 (z-z_1)(z-z_2)...(z-z_n).$$

Here the complex numbers $z_1,z_2,...,z_n$ are the zeros of $f(z)$; they may not all be distinct, so it is sometimes more convenient to write:

$$f(z)==a_0 (z-z_1)^{m_1} (z-z_2)^{m_2} ...(z-z_k)^{m_k} , \quad k<=n,$$

with distinct zeros $z_1,z_2,...,z_k$ and multiplicities $m_i>=1$. If $m_i=1$, z_i is called a single zero, if $m_i>1$, z_i is called a multiple or repeated zero; a multiple zero is also a zero of the derivative of $f(z)$.

If the coefficients of $f(z)$ are all real, then the zeros of $f(z)$ are either real or else occur as pairs of conjugate complex numbers $x+iy$ and $x-iy$. A pair of complex conjugate zeros are the zeros of a quadratic factor of $f(z)$, (z^2+rz+s), with real coefficients r and s.

Mathematicians are accustomed to thinking of polynomials as pleasantly simple functions to work with. However the problem of numerically computing the zeros of an arbitrary polynomial is far from simple. A great variety of algorithms have been proposed, of which a number have been widely used in practice; for a fairly comprehensive survey, see Householder [1]. All general algorithms are iterative. Most converge to one zero at a time; the corresponding factor can then be divided out as in equation (1) above - this process is called deflation or, loosely, dividing out the zero - and the algorithm can be applied again to the polynomial $f_1(z)$. A pair of complex conjugate zeros can be divided out together - this corresponds to dividing $f(z)$ by a quadratic factor.

Whatever the theoretical basis of the algorithm, a number of practical problems arise: for a thorough discussion of some of them see Peters and Wilkinson [2] and Wilkinson [3]. The most elementary point is that, even if z_1 is mathematically an exact zero of $f(z)$, because of the fundamental limitations of computer arithmetic the computed value of $f(z_1)$ will not necessarily be exactly 0.0. In practice there is usually a small region of values of z about the exact zero at which the computed value of $f(z)$ becomes swamped by rounding errors. Moreover in many algorithms this inaccuracy in the computed value of $f(z)$ results in a similar inaccuracy in the computed step from one iterate to the next. This limits the precision with which any zero can be computed. Deflation is another potential cause of trouble, since, in the notation of equation (1), the computed coefficients of $f_1(z)$ will not be completely accurate, especially if z_1 is not an exact zero of $f(z)$; so the zeros of the computed $f_1(z)$ will deviate from the zeros of $f(z)$.

A zero is called ill-conditioned if it is sensitive to small changes in the coefficients of the polynomial. An ill-conditioned zero is likewise sensitive to the computational inaccuracies just mentioned. Conversely a zero is called well-conditioned if it is comparatively insensitive to such perturbations. Roughly speaking a zero which is well separated from other zeros is well-conditioned, while zeros which are close together are ill-conditioned, but in talking about 'closeness' the decisive factor is not the absolute distance between neighbouring zeros but their ratio: if the ratio is close to 1 the zeros are ill-conditioned. In particular, multiple zeros are ill-conditioned. A multiple zero is usually split into a cluster of zeros by perturbations in the polynomial or computational inaccuracies.

2.1. References

[1] Householder A S (1970) The Numerical Treatment of a Single Nonlinear Equation. McGraw-Hill.

[2] Peters G and Wilkinson J H (1971) Practical Problems Arising in the Solution of Polynomial Equations. J. Inst. Maths Applics. 8 16--35.

[3] Wilkinson J H (1963) Rounding Errors in Algebraic Processes, Chapter 2. HMSO.

3. Recommendations on Choice and Use of Routines

3.1. Discussion

Two routines are available: C02AFF for polynomials with complex coefficients and C02AGF for polynomials with real coefficients.

C02AFF and C02AGF both use a variant of Laguerre's Method due to BT Smith to calculate each zero until the degree of the deflated polynomial is less than 3, whereupon the remaining zeros are obtained using the 'standard' closed formulae for a quadratic or linear equation.

The accuracy of the roots will depend on how ill-conditioned they are. Peters and Wilkinson [2] describe techniques for estimating the errors in the zeros after they have been computed.

3.2. Index

C02 -- Zeros of Polynomials Contents -- C02
Chapter C02

Zeros of Polynomials

```
        C02AGF  All zeros of real polynomial, modified Laguerre method

\end{verbatim}
\endscroll
\end{page}
```

———-

Roots of a complex polynomial equation

— **nagc.ht** —

```
\begin{page}{manpageXXc02aff}{NAG Documentation: c02aff}
\beginscroll
\begin{verbatim}



     C02AFF(3NAG)      Foundation Library (12/10/92)      C02AFF(3NAG)



          C02 -- Zeros of Polynomials                               C02AFF
                  C02AFF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          C02AFF finds all the roots of a complex polynomial equation,
          using a variant of Laguerre's Method.

          2. Specification

                 SUBROUTINE C02AFF (A, N, SCALE, Z, W, IFAIL)
                 INTEGER          N, IFAIL
                 DOUBLE PRECISION A(2,N+1), Z(2,N), W(4*(N+1))
                 LOGICAL          SCALE

          3. Description

          The routine attempts to find all the roots of the nth degree
          complex polynomial equation
```

$$P(z)=a_0 z^n +a_1 z^{n-1} +a_2 z^{n-2} +...+a_{n-1} z+a_n =0.$$

The roots are located using a modified form of Laguerre's Method, originally proposed by Smith [2].

The method of Laguerre [3] can be described by the iterative scheme

$$L(z_k)=z_{k+1} -z_k = \frac{-n*P(z_k)}{P'(z_k)+-\sqrt{H(z_k)}},$$

where $H(z_k)=(n-1)*[(n-1)*(P'(z_k))^2 -n*P(z_k)P''(z_k)]$, and z_0 is specified.

The sign in the denominator is chosen so that the modulus of the Laguerre step at z_k, viz. $|L(z_k)|$, is as small as possible. The method can be shown to be cubically convergent for isolated roots (real or complex) and linearly convergent for multiple roots. The routine generates a sequence of iterates z_1, z_2, z_3,..., such that $|P(z_{k+1})|<|P(z_k)|$ and ensures that $z_{k+1}+L(z_{k+1})$ 'roughly' lies inside a circular region of radius $|F|$ about z_k known to contain a zero of $P(z)$; that is, $|L(z_{k+1})|<=|F|$, where F denotes the Fejer bound (see Marden [1]) at the point z_k. Following Smith [2], F is taken to be min(B,1.445*n*R), where B is an upper bound for the magnitude of the smallest zero given by

$$B=1.0001*\min(\sqrt{n}*L(z_k),|r_1|,|a_n/a_0|^{1/n}),$$

r_1 is the zero X of smaller magnitude of the quadratic equation

$$2(P''(z_k)/(2*n*(n-1)))X^2 +2(P'(z_k)/n)X+P(z_k)=0$$

k k k

and the Cauchy lower bound R for the smallest zero is computed (using Newton's Method) as the positive root of the polynomial equation

$$|a_0|z^n+|a_1|z^{n-1}+|a_2|z^{n-2}+...+|a_{n-1}|z-|a_n|=0.$$

Starting from the origin, successive iterates are generated according to the rule $z_{k+1}=z_k+L(z_k)$ for k = 1,2,3,... and $L(z_k)$ is 'adjusted' so that $|P(z_{k+1})|<|P(z_k)|$ and $|L(z_{k+1})|<=|F|$. The iterative procedure terminates if $P(z_{k+1})$ is smaller in absolute value than the bound on the rounding error in $P(z_{k+1})$ and the current iterate $z_p=z_{k+1}$ is taken to be a zero of P(z). The deflated polynomial $\tilde{P}(z)=P(z)/(z-z_p)$ of degree n-1 is then formed, and the above procedure is repeated on the deflated polynomial until n<3, whereupon the remaining roots are obtained via the 'standard' closed formulae for a linear (n = 1) or quadratic (n = 2) equation.

To obtain the roots of a quadratic polynomial, C02AHF(*) can be used.

4. References

[1] Marden M (1966) Geometry of Polynomials. Mathematical Surveys. 3 Am. Math. Soc., Providence, RI.

[2] Smith B T (1967) ZERPOL: A Zero Finding Algorithm for Polynomials Using Laguerre's Method. Technical Report. Department of Computer Science, University of Toronto, Canada.

[3] Wilkinson J H (1965) The Algebraic Eigenvalue Problem. Clarendon Press.

5. Parameters

1: A(2,N+1) -- DOUBLE PRECISION array Input
On entry: if A is declared with bounds (2,0:N), then A(1,i) and A(2,i) must contain the real and imaginary parts of a

```
                                                         i
                                n-i
        (i.e., the coefficient of z   ), for i=0,1,...,n.
        Constraint: A(1,0) /= 0.0 or A(2,0) /= 0.0.

    2:  N -- INTEGER                                        Input
        On entry: the degree of the polynomial, n. Constraint: N >=
        1.

    3:  SCALE -- LOGICAL                                    Input
        On entry: indicates whether or not the polynomial is to be
        scaled. See Section 8 for advice on when it may be
        preferable to set SCALE = .FALSE. and for a description of
        the scaling strategy. Suggested value: SCALE = .TRUE..

    4:  Z(2,N) -- DOUBLE PRECISION array                   Output
        On exit: the real and imaginary parts of the roots are
        stored in Z(1,i) and Z(2,i) respectively, for i=1,2,...,n.

    5:  W(4*(N+1)) -- DOUBLE PRECISION array            Workspace

    6:  IFAIL -- INTEGER                             Input/Output
        On entry: IFAIL must be set to 0, -1 or 1. For users not
        familiar with this parameter (described in the Essential
        Introduction) the recommended value is 0.

        On exit: IFAIL = 0 unless the routine detects an error (see
        Section 6).

   6. Error Indicators and Warnings

   Errors detected by the routine:

   If on entry IFAIL = 0 or -1, explanatory error messages are
   output on the current error message unit (as defined by X04AAF).

   IFAIL= 1
        On entry A(1,0) = 0.0 and A(2,0) = 0.0,

        or       N < 1.

   IFAIL= 2
        The iterative procedure has failed to converge. This error
        is very unlikely to occur. If it does, please contact NAG
        immediately, as some basic assumption for the arithmetic has
        been violated. See also Section 8.

   IFAIL= 3
        Either overflow or underflow prevents the evaluation of P(z)
        near some of its zeros. This error is very unlikely to
```

occur. If it does, please contact NAG immediately. See also Section 8.

7. Accuracy

All roots are evaluated as accurately as possible, but because of the inherent nature of the problem complete accuracy cannot be guaranteed.

8. Further Comments

If SCALE = .TRUE., then a scaling factor for the coefficients is chosen as a power of the base B of the machine so that the largest coefficient in magnitude approaches THRESH = B^{EMAX-P}. Users should note that no scaling is performed if the largest coefficient in magnitude exceeds THRESH, even if SCALE = .TRUE.. (For definition of B, EMAX and P see the Chapter Introduction X02.)

However, with SCALE = .TRUE., overflow may be encountered when the input coefficients $a_0,a_1,a_2,...,a_n$ vary widely in magnitude, particularly on those machines for which $B^{(4*P)}$ overflows. In such cases, SCALE should be set to .FALSE. and the coefficients scaled so that the largest coefficient in magnitude does not exceed $B^{(EMAX-2*P)}$.

Even so, the scaling strategy used in C02AFF is sometimes insufficient to avoid overflow and/or underflow conditions. In such cases, the user is recommended to scale the independent variable (z) so that the disparity between the largest and smallest coefficient in magnitude is reduced. That is, use the routine to locate the zeros of the polynomial d*P(cz) for some suitable values of c and d. For example, if the original polynomial was $P(z)=2^{-100}i+2^{100}z^{20}$, then choosing $c=2^{-10}$ and $d=2^{100}$, for instance, would yield the scaled polynomial $i+z^{20}$, which is well-behaved relative to overflow and underflow and has zeros which are 2^{10} times those of P(z).

If the routine fails with IFAIL = 2 or 3, then the real and imaginary parts of any roots obtained before the failure occurred are stored in Z in the reverse order in which they were found. Let n_R denote the number of roots found before the failure

```
          occurred. Then Z(1,n) and Z(2,n) contain the real and imaginary
          parts of the 1st root found, Z(1,n-1) and Z(2,n-1) contain the
          real and imaginary parts of the 2nd root found, ..., Z(1,n ) and
                                                                    R
          Z(2,n ) contain the real and imaginary parts of the n th root
               R                                               R
          found. After the failure has occurred, the remaining 2*(n-n )
                                                                     R
          elements of Z contain a large negative number (equal to


          -1/(X02AMF().\/2)).

          9. Example

                                                 5    4    3    2
          To find the roots of the polynomial a z +a z +a z +a z +a z+a =0,
                                               0    1    2    3    4   5
          where a =(5.0+6.0i), a =(30.0+20.0i), a =-(0.2+6.0i),
                 0              1                2
          a =(50.0+100000.0i), a =-(2.0-40.0i) and a =(10.0+1.0i).
           3                    4                   5

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Roots of a real polynomial equation

— nagc.ht —

```
\begin{page}{manpageXXc02agf}{NAG Documentation: c02agf}
\beginscroll
\begin{verbatim}


     C02AGF(3NAG)      Foundation Library (12/10/92)      C02AGF(3NAG)
```

C02 -- Zeros of Polynomials C02AGF

C02AGF -- NAG Foundation Library Routine Document

Note: Before using this routine, please read the Users' Note for your implementation to check implementation-dependent details. The symbol (*) after a NAG routine name denotes a routine that is not included in the Foundation Library.

1. Purpose

C02AGF finds all the roots of a real polynomial equation, using a variant of Laguerre's Method.

2. Specification

```
SUBROUTINE C02AGF (A, N, SCALE, Z, W, IFAIL)
INTEGER          N, IFAIL
DOUBLE PRECISION A(N+1), Z(2,N), W(2*(N+1))
LOGICAL          SCALE
```

3. Description

The routine attempts to find all the roots of the nth degree real polynomial equation

$$P(z)=a_0 z^n +a_1 z^{n-1} +a_2 z^{n-2} +...+a_{n-1} z+a_n =0.$$

The roots are located using a modified form of Laguerre's Method, originally proposed by Smith [2].

The method of Laguerre [3] can be described by the iterative scheme

$$L(z_k)=z_{k+1} -z_k = \frac{-n*P(z_k)}{P'(z_k)+-\sqrt{H(z_k)}},$$

where $H(z_k)=(n-1)*[(n-1)*(P'(z_k))^2 -n*P(z_k)P''(z_k)]$, and z_0 is specified.

The sign in the denominator is chosen so that the modulus of the Laguerre step at z_k, viz. $|L(z_k)|$, is as small as possible. The method can be shown to be cubically convergent for isolated roots (real or complex) and linearly convergent for multiple roots. The routine generates a sequence of iterates $z_1, z_2, z_3, ...$, such that $|P(z_{k+1})|<|P(z_k)|$ and ensures that $z_{k+1}+L(z_{k+1})$ 'roughly' lies inside a circular region of radius $|F|$ about z_k known to contain a zero of $P(z)$; that is, $|L(z_{k+1})|<=|F|$, where F denotes the Fejer bound (see Marden [1]) at the point z_k. Following Smith [2], F is taken to be min(B,1.445*n*R), where B is an upper bound for the magnitude of the smallest zero given by

$$B=1.0001*\min(\sqrt{n}*L(z_k),|r_1|,|a_n/a_0|^{1/n}),$$

r_1 is the zero X of smaller magnitude of the quadratic equation

$$2(P''(z_k)/(2*n*(n-1)))X^2+2(P'(z_k)/n)X+P(z_k)=0$$

and the Cauchy lower bound R for the smallest zero is computed (using Newton's Method) as the positive root of the polynomial equation

$$|a_0|z^n+|a_1|z^{n-1}+|a_2|z^{n-2}+...+|a_{n-1}|z-|a_n|=0.$$

Starting from the origin, successive iterates are generated according to the rule $z_{k+1}=z_k+L(z_k)$ for k=1,2,3,... and $L(z_k)$ is

k+1 k k+1

iterative procedure terminates if $P(z_{k+1})$ is smaller in absolute value than the bound on the rounding error in $P(z_{k+1})$ and the current iterate $z_p=z_{k-1}$ is taken to be a zero of $P(z)$ (as is its

conjugate z_p if z_p is complex). The deflated polynomial

$\tilde{P}(z)=P(z)/(z-z_p)$ of degree n-1 if z_p is real p p

$(\tilde{P}(z)=P(z)/((z-z_p)(z-z_p))$ of degree n-2 if z_p is complex) is then formed, and the above procedure is repeated on the deflated polynomial until n<3, whereupon the remaining roots are obtained via the 'standard' closed formulae for a linear (n = 1) or quadratic (n = 2) equation.

To obtain the roots of a quadratic polynomial, C02AJF(*) can be used.

4. References

[1] Marden M (1966) Geometry of Polynomials. Mathematical Surveys. 3 Am. Math. Soc., Providence, RI.

[2] Smith B T (1967) ZERPOL: A Zero Finding Algorithm for Polynomials Using Laguerre's Method. Technical Report. Department of Computer Science, University of Toronto, Canada.

[3] Wilkinson J H (1965) The Algebraic Eigenvalue Problem. Clarendon Press.

5. Parameters

1: A(N+1) -- DOUBLE PRECISION array Input
On entry: if A is declared with bounds (0:N), then A(i) must contain a_i (i.e., the coefficient of z^{n-i}), for i=0,1,...,n. Constraint: A(0) /= 0.0.

2: N -- INTEGER Input
On entry: the degree of the polynomial, n. Constraint: N >= 1.

3: SCALE -- LOGICAL Input
On entry: indicates whether or not the polynomial is to be scaled. See Section 8 for advice on when it may be preferable to set SCALE = .FALSE. and for a description of the scaling strategy. Suggested value: SCALE = .TRUE..

4: Z(2,N) -- DOUBLE PRECISION array Output
On exit: the real and imaginary parts of the roots are stored in Z(1,i) and Z(2,i) respectively, for i=1,2,...,n. Complex conjugate pairs of roots are stored in consecutive pairs of elements of Z; that is, Z(1,i+1) = Z(1,i) and Z(2,i+1)=-Z(2,i).

```
 5:  W(2*(N+1)) -- DOUBLE PRECISION array              Workspace

 6:  IFAIL -- INTEGER                                Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.
     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

IFAIL= 1
     On entry A(0) = 0.0,

     or       N < 1.

IFAIL= 2
     The iterative procedure has failed to converge. This error
     is very unlikely to occur. If it does, please contact NAG
     immediately, as some basic assumption for the arithmetic has
     been violated. See also Section 8.

IFAIL= 3
     Either overflow or underflow prevents the evaluation of P(z)
     near some of its zeros. This error is very unlikely to
     occur. If it does, please contact NAG immediately. See also
     Section 8.

7. Accuracy

All roots are evaluated as accurately as possible, but because of
the inherent nature of the problem complete accuracy cannot be
guaranteed.

8. Further Comments

If SCALE = .TRUE., then a scaling factor for the coefficients is
chosen as a power of the base B of the machine so that the
                                                       EMAX-P
largest coefficient in magnitude approaches THRESH = B      .
Users should note that no scaling is performed if the largest
coefficient in magnitude exceeds THRESH, even if SCALE = .TRUE..
(For definition of B, EMAX and P see the Chapter Introduction
X02.)
```

However, with SCALE = .TRUE., overflow may be encountered when the input coefficients $a_0,a_1,a_2,...,a_n$ vary widely in magnitude, particularly on those machines for which $B^{(4*P)}$ overflows. In such cases, SCALE should be set to .FALSE. and the coefficients scaled so that the largest coefficient in magnitude does not exceed $B^{(EMAX-2*P)}$.

Even so, the scaling strategy used in C02AGF is sometimes insufficient to avoid overflow and/or underflow conditions. In such cases, the user is recommended to scale the independent variable (z) so that the disparity between the largest and smallest coefficient in magnitude is reduced. That is, use the routine to locate the zeros of the polynomial d*P(cz) for some suitable values of c and d. For example, if the original polynomial was $P(z)=2^{-100}+2^{100}z^{20}$, then choosing $c=2^{-10}$ and $d=2^{100}$, for instance, would yield the scaled polynomial $1+z^{20}$, which is well-behaved relative to overflow and underflow and has zeros which are 2^{10} times those of P(z).

If the routine fails with IFAIL = 2 or 3, then the real and imaginary parts of any roots obtained before the failure occurred are stored in Z in the reverse order in which they were found. Let n_R denote the number of roots found before the failure occurred. Then Z(1,n) and Z(2,n) contain the real and imaginary parts of the 1st root found, Z(1,n-1) and Z(2,n-1) contain the real and imaginary parts of the 2nd root found, ..., $Z(1,n_R)$ and $Z(2,n_R)$ contain the real and imaginary parts of the n_Rth root found. After the failure has occurred, the remaining $2*(n-n_R)$ elements of Z contain a large negative number (equal to

-1/(X02AMF().\/2)).

9. Example

To find the roots of the 5th degree polynomial $z^5+2z^4+3z^3+4z^2+5z+6=0$.

```
        The example program is not reproduced here. The source code for
        all example programs is distributed with the NAG Foundation
        Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Roots of One or More Transcendental Equations

— nagc.ht —

```
\begin{page}{manpageXXc05}{NAG Documentation: c05}
\beginscroll
\begin{verbatim}



     C05(3NAG)         Foundation Library (12/10/92)         C05(3NAG)



          C05 -- Roots of One or More Transcendental Equations
                                                 Introduction -- C05
                                    Chapter C05
                   Roots of One or More Transcendental Equations

          1. Scope of the Chapter

          This chapter is concerned with the calculation of real zeros of
          continuous real functions of one or more variables. (Complex
          equations must be expressed in terms of the equivalent larger
          system of real equations.)

          2. Background to the Problems

          The chapter divides naturally into two parts.

          2.1. A Single Equation

          The first deals with the real zeros of a real function of a
          single variable f(x).

          At present, there is only one routine with a simple calling
          sequence. This routine assumes that the user can determine an
```

initial interval [a,b] within which the desired zero lies, that is $f(a)*f(b)<0$, and outside which all other zeros lie. The routine then systematically subdivides the interval to produce a final interval containing the zero. This final interval has a length bounded by the user's specified error requirements; the end of the interval where the function has smallest magnitude is returned as the zero. This routine is guaranteed to converge to a simple zero of the function. (Here we define a simple zero as a zero corresponding to a sign-change of the function.) The algorithm used is due to Bus and Dekker.

2.2. Systems of Equations

The routines in the second part of this chapter are designed to solve a set of nonlinear equations in n unknowns

$$f_i(x)=0, \quad i=1,2,...,n, \quad x=(x_1,x_2,...,x_n)^T \qquad (1)$$

where T stands for transpose.

It is assumed that the functions are continuous and differentiable so that the matrix of first partial derivatives of the functions, the Jacobian matrix $J_{ij}(x)=ddf_i/ddx_j$ evaluated at the point x, exists, though it may not be possible to calculate it directly.

The functions f_i must be independent, otherwise there will be an infinity of solutions and the methods will fail. However, even when the functions are independent the solutions may not be unique. Since the methods are iterative, an initial guess at the solution has to be supplied, and the solution located will usually be the one closest to this initial guess.

2.3. References

[1] Gill P E and Murray W (1976) Algorithms for the Solution of the Nonlinear Least-squares Problem. NAC 71 National Physical Laboratory.

[2] More J J, Garbow B S and Hillstrom K E (1974) User Guide for Minpack-1. ANL-80-74 Argonne National Laboratory.

[3] Ortega J M and Rheinboldt W C (1970) Iterative Solution of Nonlinear Equations in Several Variables. Academic Press.

[4] Rabinowitz P (1970) Numerical Methods for Nonlinear Algebraic Equations. Gordon and Breach.

3. Recommendations on Choice and Use of Routines

3.1. Zeros of Functions of One Variable

There is only one routine (C05ADF) for solving a single nonlinear equation. This routine is designed for solving problems where the function f(x) whose zero is to be calculated, can be coded as a user-supplied routine.

C05ADF may only be used when the user can supply an interval [a,b] containing the zero, that is f(a)*f(b)<0.

3.2. Solution of Sets of Nonlinear Equations

The solution of a set of nonlinear equations

$$f_i(x_1,x_2,\ldots,x_n)=0, \quad i=1,2,\ldots,n \tag{2}$$

can be regarded as a special case of the problem of finding a minimum of a sum of squares

$$s(x)=\sum_{i=1}^{m}[f_i(x_1,x_2,\ldots,x_n)]^2 \quad (m>=n). \tag{3}$$

So the routines in Chapter E04 of the Library are relevant as well as the special nonlinear equations routines.

There are two routines (C05NBF and C05PBF) for solving a set of nonlinear equations. These routines require the f_i (and possibly their derivatives) to be calculated in user-supplied routines. These should be set up carefully so the Library routines can work as efficiently as possible.

The main decision which has to be made by the user is whether to supply the derivatives $\frac{ddf_i}{ddx_j}$. It is advisable to do so if possible, since the results obtained by algorithms which use derivatives are generally more reliable than those obtained by algorithms which do not use derivatives.

C05PBF requires the user to provide the derivatives, whilst C05NBF does not. C05NBF and C05PBF are easy-to-use routines. A routine, C05ZAF, is provided for use in conjunction with C05PBF to check the user-provided derivatives for consistency with the functions themselves. The user is strongly advised to make use of this routine whenever C05PBF is used.

Firstly, the calculation of the functions and their derivatives should be ordered so that cancellation errors are avoided. This is particularly important in a routine that uses these quantities to build up estimates of higher derivatives.

Secondly, scaling of the variables has a considerable effect on the efficiency of a routine. The problem should be designed so that the elements of x are of similar magnitude. The same comment applies to the functions, all the f_i should be of comparable size.

The accuracy is usually determined by the accuracy parameters of the routines, but the following points may be useful:

(i) Greater accuracy in the solution may be requested by choosing smaller input values for the accuracy parameters. However, if unreasonable accuracy is demanded, rounding errors may become important and cause a failure.

(ii) Some idea of the accuracies of the x_i may be obtained by monitoring the progress of the routine to see how many figures remain unchanged during the last few iterations.

(iii) An approximation to the error in the solution x, given by e where e is the solution to the set of linear equations

$$J(x)e=-f(x)$$

where $f(x)=(f_1(x),f_2(x),...,f_n(x))^T$ (see Chapter F04).

(iv) If the functions $f_i(x)$ are changed by small amounts $(epsilon)_i$, for i=1,2,...,n, then the corresponding change in the solution x is given approximately by (sigma), where (sigma) is the solution of the set of linear equations

$$J(x)(sigma)=-(epsilon),$$ (see Chapter F04).

```
      Thus one can estimate the sensitivity of x to any
      uncertainties in the specification of f (x), for
                                              i
      i=1,2,...,n.

 3.3. Index

 Zeros of functions of one variable:
      Bus and Dekker algorithm                           C05ADF
 Zeros of functions of several variables:
      easy-to-use                                        C05NBF
      easy-to-use, derivatives required                  C05PBF
 Checking Routine:
      Checks user-supplied Jacobian                      C05ZAF


 C05 -- Roots of One or More Transcendental Equations
                                                   Contents -- C05
 Chapter C05

 Roots of One or More Transcendental Equations

 C05ADF  Zero of continuous function in given interval, Bus and
         Dekker algorithm

 C05NBF  Solution of system of nonlinear equations using function
         values only

 C05PBF  Solution of system of nonlinear equations using 1st
         derivatives

 C05ZAF  Check user's routine for calculating 1st derivatives

\end{verbatim}
\endscroll
\end{page}
```

Zero of a continuous function in a given interval

— nagc.ht —

```
\begin{page}{manpageXXc05adf}{NAG Documentation: c05adf}
\beginscroll
\begin{verbatim}
```

```
C05ADF(3NAG)      Foundation Library (12/10/92)      C05ADF(3NAG)



     C05 -- Roots of One or More Transcendental Equations       C05ADF
             C05ADF -- NAG Foundation Library Routine Document

     Note: Before using this routine, please read the Users' Note for
     your implementation to check implementation-dependent details.
     The symbol (*) after a NAG routine name denotes a routine that is
     not included in the Foundation Library.

     1. Purpose

     C05ADF locates a zero of a continuous function in a given
     interval by a combination of the methods of linear interpolation,
     extrapolation and bisection.

     2. Specification

            SUBROUTINE C05ADF (A, B, EPS, ETA, F, X, IFAIL)
            INTEGER          IFAIL
            DOUBLE PRECISION A, B, EPS, ETA, F, X
            EXTERNAL         F

     3. Description

     The routine attempts to obtain an approximation to a simple zero
     of the function f(x) given an initial interval [a,b] such that
     f(a)*f(b)<=0. The zero is found by calls to C05AZF(*) whose
     specification should be consulted for details of the method used.

     The approximation x to the zero (alpha) is determined so that one
     or both of the following criteria are satisfied:

          (i) |x-(alpha)|<EPS,

          (ii) |f(x)|<ETA.

     4. References

     None.

     5. Parameters

      1:  A -- DOUBLE PRECISION                                  Input
          On entry: the lower bound of the interval, a.
```

```
 2:  B -- DOUBLE PRECISION                                  Input
     On entry: the upper bound of the interval, b. Constraint: B
     /= A.

 3:  EPS -- DOUBLE PRECISION                                Input
     On entry: the absolute tolerance to which the zero is
     required (see Section 3). Constraint: EPS > 0.0.

 4:  ETA -- DOUBLE PRECISION                                Input
     On entry: a value such that if |f(x)|<ETA, x is accepted as
     the zero. ETA may be specified as 0.0 (see Section 7).

 5:  F -- DOUBLE PRECISION FUNCTION, supplied by the user.
                                                  External Procedure
     F must evaluate the function f whose zero is to be
     determined.

     Its specification is:

            DOUBLE PRECISION FUNCTION F (XX)
            DOUBLE PRECISION XX

      1:  XX -- DOUBLE PRECISION                            Input
          On entry: the point at which the function must be
          evaluated.
     F must be declared as EXTERNAL in the (sub)program from
     which C05ADF is called. Parameters denoted as Input
     must not be changed by this procedure.

 6:  X -- DOUBLE PRECISION                                 Output
     On exit: the approximation to the zero.

 7:  IFAIL -- INTEGER                                Input/Output
     Before entry, IFAIL must be assigned a value. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     Unless the routine detects an error (see Section 6), IFAIL
     contains 0 on exit.

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1
     On entry EPS <= 0.0,

     or       A = B,
```

or F(A)*F(B)>0.0.

IFAIL= 2
Too much accuracy has been requested in the computation, that is, EPS is too small for the computer being used. The final value of X is an accurate approximation to the zero.

IFAIL= 3
A change in sign of f(x) has been determined as occurring near the point defined by the final value of X. However, there is some evidence that this sign-change corresponds to a pole of f(x).

IFAIL= 4
Indicates that a serious error has occurred in C05AZF(*). Check all routine calls. Seek expert help.

7. Accuracy

This depends on the value of EPS and ETA. If full machine accuracy is required, they may be set very small, resulting in an error exit with IFAIL = 2, although this may involve more iterations than a lesser accuracy. The user is recommended to set ETA = 0.0 and to use EPS to control the accuracy, unless he has considerable knowledge of the size of f(x) for values of x near the zero.

8. Further Comments

The time taken by the routine depends primarily on the time spent evaluating F (see Section 5).

If it is important to determine an interval of length less than EPS containing the zero, or if the function F is expensive to evaluate and the number of calls to F is to be restricted, then use of C05AZF(*) is recommended. Use of C05AZF(*) is also recommended when the structure of the problem to be solved does not permit a simple function F to be written: the reverse communication facilities of C05AZF(*) are more flexible than the direct communication of F required by C05ADF.

9. Example

The example program below calculates the zero of $e^{-x}-x$ within the interval [0,1] to approximately 5 decimal places.

The example program is not reproduced here. The source code for all example programs is distributed with the NAG Foundation Library software and should be available on-line.

```
\end{verbatim}
\endscroll
\end{page}
```

———-

Solution of a system of nonlinear equations

— nagc.ht —

```
\begin{page}{manpageXXc05nbf}{NAG Documentation: c05nbf}
\beginscroll
\begin{verbatim}



     C05NBF(3NAG)      Foundation Library (12/10/92)      C05NBF(3NAG)



          C05 -- Roots of One or More Transcendental Equations       C05NBF
                  C05NBF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          C05NBF is an easy-to-use routine to find a solution of a system
          of nonlinear equations by a modification of the Powell hybrid
          method.

          2. Specification

                 SUBROUTINE C05NBF (FCN, N, X, FVEC, XTOL, WA, LWA, IFAIL)
                 INTEGER          N, LWA, IFAIL
                 DOUBLE PRECISION X(N), FVEC(N), XTOL, WA(LWA)
                 EXTERNAL         FCN

          3. Description

          The system of equations is defined as:

                        f (x ,x ,...,x )=0, for i=1,2,...,n.
```

i 1 2 n

C05NBF is based upon the MINPACK routine HYBRD1 (More et al [1]). It chooses the correction at each step as a convex combination of the Newton and scaled gradient directions. Under reasonable conditions this guarantees global convergence for starting points far from the solution and a fast rate of convergence. The Jacobian is updated by the rank-1 method of Broyden. At the starting point the Jacobian is approximated by forward differences, but these are not used again until the rank-1 method fails to produce satisfactory progress. For more details see Powell [2].

4. References

[1] More J J, Garbow B S and Hillstrom K E User Guide for MINPACK-1. Technical Report ANL-80-74. Argonne National Laboratory.

[2] Powell M J D (1970) A Hybrid Method for Nonlinear Algebraic Equations. Numerical Methods for Nonlinear Algebraic Equations. (ed P Rabinowitz) Gordon and Breach.

5. Parameters

1: FCN -- SUBROUTINE, supplied by the user.
External Procedure

FCN must return the values of the functions f_i at a point x.

Its specification is:

```
SUBROUTINE FCN (N, X, FVEC, IFLAG)
INTEGER          N, IFLAG
DOUBLE PRECISION X(N), FVEC(N)
```

1: N -- INTEGER Input

On entry: the number of equations, n.

2: X(N) -- DOUBLE PRECISION array Input

On entry: the components of the point x at which the functions must be evaluated.

3: FVEC(N) -- DOUBLE PRECISION array Output

On exit: the function values $f_i(x)$ (unless IFLAG is set to a negative value by FCN).

4: IFLAG -- INTEGER Input/Output

On entry: IFLAG > 0. On exit: in general, IFLAG should

```
          not be reset by FCN. If, however, the user wishes to
          terminate execution (perhaps because some illegal point
          X has been reached), then IFLAG should be set to a
          negative integer. This value will be returned through
          IFAIL.
     FCN must be declared as EXTERNAL in the (sub)program
     from which C05NBF is called. Parameters denoted as
     Input must not be changed by this procedure.

 2:  N -- INTEGER                                          Input
     On entry: the number of equations, n. Constraint: N > 0.

 3:  X(N) -- DOUBLE PRECISION array                 Input/Output
     On entry: an initial guess at the solution vector. On
     exit: the final estimate of the solution vector.

 4:  FVEC(N) -- DOUBLE PRECISION array                    Output
     On exit: the function values at the final point, X.

 5:  XTOL -- DOUBLE PRECISION                              Input
     On entry: the accuracy in X to which the solution is
     required. Suggested value: the square root of the machine
     precision. Constraint: XTOL >= 0.0.

 6:  WA(LWA) -- DOUBLE PRECISION array                 Workspace

 7:  LWA -- INTEGER                                        Input
     On entry: the dimension of the array WA. Constraint:
     LWA>=N*(3*N+13)/2.

 8:  IFAIL -- INTEGER                               Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

IFAIL< 0
     The user has set IFLAG negative in FCN. The value of IFAIL
     will be the same as the user's setting of IFLAG.

IFAIL= 1
```

On entry N <= 0,

or XTOL < 0.0,

or LWA<N*(3*N+13)/2.

IFAIL= 2
There have been at least 200*(N+1) evaluations of FCN. Consider restarting the calculation from the final point held in X.

IFAIL= 3
No further improvement in the approximate solution X is possible; XTOL is too small.

IFAIL= 4
The iteration is not making good progress. This failure exit may indicate that the system does not have a zero, or that the solution is very close to the origin (see Section 7). Otherwise, rerunning C05NBF from a different starting point may avoid the region of difficulty.

7. Accuracy

If $\hat{x}$ is the true solution, C05NBF tries to ensure that

$$||x-\hat{x}||<=XTOL*||\hat{x}||.$$

If this condition is satisfied with $XTOL=10^{-k}$, then the larger components of x have k significant decimal digits. There is a danger that the smaller components of x may have large relative errors, but the fast rate of convergence of C05NBF usually avoids this possibility.

If XTOL is less than machine precision, and the above test is satisfied with the machine precision in place of XTOL, then the routine exits with IFAIL = 3.

Note: this convergence test is based purely on relative error, and may not indicate convergence if the solution is very close to the origin.

The test assumes that the functions are reasonably well behaved. If this condition is not satisfied, then C05NBF may incorrectly indicate convergence. The validity of the answer can be checked, for example, by rerunning C05NBF with a tighter tolerance.

```
          8. Further Comments

          The time required by C05NBF to solve a given problem depends on n
          , the behaviour of the functions, the accuracy requested and the
          starting point. The number of arithmetic operations executed by
                                                           2
          C05NBF to process each call of FCN is about 11.5*n . Unless FCN
          can be evaluated quickly, the timing of C05NBF will be strongly
          influenced by the time spent in FCN.

          Ideally the problem should be scaled so that at the solution the
          function values are of comparable magnitude.

          9. Example

          To determine the values x ,..., x  which satisfy the tridiagonal
                                   1       9
          equations:

                                  (3-2x )x -2x =-1
                                       1  1   2

                       -x -1+(3-2x )x -2x   =-1,  i=2,3,...,8
                         i        i  i   i+1

                                  -x +(3-2x )x =-1.
                                    8      9  9

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Solution of a system of nonlinear equations

— nagc.ht —

```
\begin{page}{manpageXXc05pbf}{NAG Documentation: c05pbf}
\beginscroll
\begin{verbatim}
```

```
C05PBF(3NAG)      Foundation Library (12/10/92)      C05PBF(3NAG)
```

```
C05 -- Roots of One or More Transcendental Equations       C05PBF
        C05PBF -- NAG Foundation Library Routine Document
```

Note: Before using this routine, please read the Users' Note for your implementation to check implementation-dependent details. The symbol (*) after a NAG routine name denotes a routine that is not included in the Foundation Library.

1. Purpose

C05PBF is an easy-to-use routine to find a solution of a system of nonlinear equations by a modification of the Powell hybrid method. The user must provide the Jacobian.

2. Specification

```
       SUBROUTINE C05PBF (FCN, N, X, FVEC, FJAC, LDFJAC, XTOL,
      1                   WA, LWA, IFAIL)
       INTEGER          N, LDFJAC, LWA, IFAIL
       DOUBLE PRECISION X(N), FVEC(N), FJAC(LDFJAC,N), XTOL, WA
      1                 (LWA)
       EXTERNAL         FCN
```

3. Description

The system of equations is defined as:

$$f_i(x_1,x_2,...,x_n)=0, i=1,2,...,n.$$

C05PBF is based upon the MINPACK routine HYBRJ1 (More et al [1]). It chooses the correction at each step as a convex combination of the Newton and scaled gradient directions. Under reasonable conditions this guarantees global convergence for starting points far from the solution and a fast rate of convergence. The Jacobian is updated by the rank-1 method of Broyden. At the starting point the Jacobian is calculated, but it is not recalculated until the rank-1 method fails to produce satisfactory progress. For more details see Powell [2].

4. References

[1]   More J J, Garbow B S and Hillstrom K E User Guide for MINPACK-1. Technical Report ANL-80-74. Argonne National Laboratory.

[2] Powell M J D (1970) A Hybrid Method for Nonlinear Algebraic Equations. Numerical Methods for Nonlinear Algebraic Equations. (ed P Rabinowitz) Gordon and Breach.

5. Parameters

1: FCN -- SUBROUTINE, supplied by the user.
External Procedure

Depending upon the value of IFLAG, FCN must either return the values of the functions f_i at a point x or return the Jacobian at x.

Its specification is:

```
SUBROUTINE FCN (N, X, FVEC, FJAC, LDFJAC, IFLAG)
INTEGER          N, LDFJAC, IFLAG
DOUBLE PRECISION X(N), FVEC(N), FJAC(LDFJAC,N)
```

1: N -- INTEGER Input

On entry: the number of equations, n.

2: X(N) -- DOUBLE PRECISION array Input

On entry: the components of the point x at which the functions or the Jacobian must be evaluated.

3: FVEC(N) -- DOUBLE PRECISION array Output

On exit: if IFLAG = 1 on entry, FVEC must contain the function values $f_i(x)$ (unless IFLAG is set to a negative value by FCN). If IFLAG = 2 on entry, FVEC must not be changed.

4: FJAC(LDFJAC,N) -- DOUBLE PRECISION array Output

On exit: if IFLAG = 2 on entry, FJAC(i,j) must contain the value of $\frac{ddf_i}{ddx_j}$ at the point x, for i,j=1,2,...,n (unless IFLAG is set to a negative value by FCN).

If IFLAG = 1 on entry, FJAC must not be changed.

5: LDFJAC -- INTEGER Input

On entry: the first dimension of FJAC.

6: IFLAG -- INTEGER Input/Output

On entry: IFLAG = 1 or 2:

```
          if IFLAG = 1, FVEC is to be updated;

          if IFLAG = 2, FJAC is to be updated.
     On exit: in general, IFLAG should not be reset by FCN.
     If, however, the user wishes to terminate execution
     (perhaps because some illegal point x has been reached)
     then IFLAG should be set to a negative integer. This
     value will be returned through IFAIL.
FCN must be declared as EXTERNAL in the (sub)program
from which C05PBF is called. Parameters denoted as
Input must not be changed by this procedure.

 2:  N -- INTEGER                                      Input
     On entry: the number of equations, n. Constraint: N > 0.

 3:  X(N) -- DOUBLE PRECISION array             Input/Output
     On entry: an initial guess at the solution vector. On
     exit: the final estimate of the solution vector.

 4:  FVEC(N) -- DOUBLE PRECISION array                Output
     On exit: the function values at the final point, X.

 5:  FJAC(LDFJAC,N) -- DOUBLE PRECISION array         Output
     On exit: the orthogonal matrix Q produced by the QR
     factorization of the final approximate Jacobian.

 6:  LDFJAC -- INTEGER                                 Input
     On entry:
     the first dimension of the array FJAC as declared in the
     (sub)program from which C05PBF is called.
     Constraint: LDFJAC >= N.

 7:  XTOL -- DOUBLE PRECISION                          Input
     On entry: the accuracy in X to which the solution is
     required. Suggested value: the square root of the machine
     precision. Constraint: XTOL >= 0.0.

 8:  WA(LWA) -- DOUBLE PRECISION array             Workspace

 9:  LWA -- INTEGER                                    Input
     On entry: the dimension of the array WA. Constraint:
     LWA>=N*(N+13)/2.

10:  IFAIL -- INTEGER                           Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).
```

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL< 0
A negative value of IFAIL indicates an exit from C05PBF because the user has set IFLAG negative in FCN. The value of IFAIL will be the same as the user's setting of IFLAG.

IFAIL= 1
On entry N <= 0,

or LDFJAC < N,

or XTOL < 0.0,

or LWA<N*(N+13)/2.

IFAIL= 2
There have been 100*(N+1) evaluations of the functions. Consider restarting the calculation from the final point held in X.

IFAIL= 3
No further improvement in the approximate solution X is possible; XTOL is too small.

IFAIL= 4
The iteration is not making good progress. This failure exit may indicate that the system does not have a zero or that the solution is very close to the origin (see Section 7). Otherwise, rerunning C05PBF from a different starting point may avoid the region of difficulty.

7. Accuracy

If $\hat{x}$ is the true solution, C05PBF tries to ensure that

$$||x-\hat{x}||_2 \le XTOL*||\hat{x}||_2 .$$

If this condition is satisfied with $XTOL=10^{-k}$, then the larger components of x have k significant decimal digits. There is a danger that the smaller components of x may have large relative errors, but the fast rate of convergence of C05PBF usually avoids the possibility.

If XTOL is less than machine precision and the above test is satisfied with the machine precision in place of XTOL, then the routine exits with IFAIL = 3.

Note: this convergence test is based purely on relative error, and may not indicate convergence if the solution is very close to the origin.

The test assumes that the functions and Jacobian are coded consistently and that the functions are reasonably well behaved. If these conditions are not satisfied then C05PBF may incorrectly indicate convergence. The coding of the Jacobian can be checked using C05ZAF. If the Jacobian is coded correctly, then the validity of the answer can be checked by rerunning C05PBF with a tighter tolerance.

8. Further Comments

The time required by C05PBF to solve a given problem depends on n, the behaviour of the functions, the accuracy requested and the starting point. The number of arithmetic operations executed by C05PBF is about $11.5*n^2$ to process each evaluation of the functions and about $1.3*n^3$ to process each evaluation of the Jacobian. Unless FCN can be evaluated quickly, the timing of C05PBF will be strongly influenced by the time spent in FCN.

Ideally the problem should be scaled so that, at the solution, the function values are of comparable magnitude.

9. Example

To determine the values $x_1,\ldots, x_9$ which satisfy the tridiagonal equations:

$$(3-2x_1)x_1-2x_2=-1$$

$$-x_{i-1}+(3-2x_i)x_i-2x_{i+1}=-1, \quad i=2,3,\ldots,8.$$

$$-x_8+(3-2x_9)x_9=-1.$$

```
        The example program is not reproduced here. The source code for
        all example programs is distributed with the NAG Foundation
        Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

———

Checks the gradients of a set of non-linear functions

— nagc.ht —

```
\begin{page}{manpageXXc05zaf}{NAG Documentation: c05zaf}
\beginscroll
\begin{verbatim}



     C05ZAF(3NAG)      Foundation Library (12/10/92)      C05ZAF(3NAG)



          C05 -- Roots of One or More Transcendental Equations       C05ZAF
                  C05ZAF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          C05ZAF checks the user-provided gradients of a set of non-linear
          functions in several variables, for consistency with the
          functions themselves. The routine must be called twice.

          2. Specification

                 SUBROUTINE C05ZAF (M, N, X, FVEC, FJAC, LDFJAC, XP, FVECP,
                1                   MODE, ERR)
                 INTEGER          M, N, LDFJAC, MODE
                 DOUBLE PRECISION X(N), FVEC(M), FJAC(LDFJAC,N), XP(N),
                1                 FVECP(M), ERR(M)

          3. Description
```

C05ZAF is based upon the MINPACK routine CHKDER (More et al [1]). It checks the ith gradient for consistency with the ith function by computing a forward-difference approximation along a suitably chosen direction and comparing this approximation with the user-supplied gradient along the same direction. The principal characteristic of C05ZAF is its invariance under changes in scale of the variables or functions.

4. References

[1] More J J, Garbow B S and Hillstrom K E User Guide for MINPACK-1. Technical Report ANL-80-74. Argonne National Laboratory.

5. Parameters

1: M -- INTEGER Input
On entry: the number of functions.

2: N -- INTEGER Input
On entry: the number of variables. For use with C05PBF and C05PCF(*), M = N.

3: X(N) -- DOUBLE PRECISION array Input
On entry: the components of a point x, at which the consistency check is to be made. (See Section 8.)

4: FVEC(M) -- DOUBLE PRECISION array Input
On entry: when MODE = 2, FVEC must contain the functions evaluated at x.

5: FJAC(LDFJAC,N) -- DOUBLE PRECISION array Input
On entry: when MODE = 2, FJAC must contain the user-supplied gradients. (The ith row of FJAC must contain the gradient of the ith function evaluated at the point x.)

6: LDFJAC -- INTEGER Input
On entry:
the first dimension of the array FJAC as declared in the (sub)program from which C05ZAF is called.
Constraint: LDFJAC >= M.

7: XP(N) -- DOUBLE PRECISION array Output
On exit: when MODE = 1, XP is set to a neighbouring point to X.

8: FVECP(M) -- DOUBLE PRECISION array Input
On entry: when MODE = 2, FVECP must contain the functions evaluated at XP.

```
 9:  MODE -- INTEGER                                      Input
     On entry: the value 1 on the first call and the value 2 on
     the second call of C05ZAF.

10:  ERR(M) -- DOUBLE PRECISION array                    Output
     On exit: when MODE = 2, ERR contains measures of
     correctness of the respective gradients. If there is no loss
     of significance (see Section 8), then if ERR(i) is 1.0 the i
     th user-supplied gradient is correct, whilst if ERR(i) is 0.
     0 the ith gradient is incorrect. For values of ERR(i)
     between 0.0 and 1.0 the categorisation is less certain. In
     general, a value of ERR(i)>0.5 indicates that the ith
     gradient is probably correct.

6. Error Indicators and Warnings

None.

7. Accuracy

See below.

8. Further Comments

The time required by C05ZAF increases with M and N.

C05ZAF does not perform reliably if cancellation or rounding
errors cause a severe loss of significance in the evaluation of a
function. Therefore, none of the components of x should be
unusually small (in particular, zero) or any other value which
may cause loss of significance. The relative differences between
corresponding elements of FVECP and FVEC should be at least two
orders of magnitude greater than the machine precision.

9. Example

This example checks the Jacobian matrix for a problem with 15
functions of 3 variables. The results indicate that the first 7
gradients are probably incorrect (this is caused by a deliberate
error in the code to calculate the Jacobian).

The example program is not reproduced here. The source code for
all example programs is distributed with the NAG Foundation
Library software and should be available on-line.
```

```
\end{verbatim}
\endscroll
\end{page}
```

Discrete Fourier transform of real or complex data values

— **nagc.ht** —

```
\begin{page}{manpageXXc06}{NAG Documentation: c06}
\beginscroll
\begin{verbatim}



     C06(3NAG)         Foundation Library (12/10/92)         C06(3NAG)



          C06 -- Summation of Series                   Introduction -- C06
                                    Chapter C06
                                Summation of Series

          1. Scope of the Chapter

          This chapter is concerned with calculating the discrete Fourier
          transform of a sequence of real or complex data values, and
          applying it to calculate convolutions and correlations.

          2. Background to the Problems

          2.1. Discrete Fourier Transforms

          2.1.1.  Complex transforms

          Most of the routines in this chapter calculate the finite
          discrete Fourier transform  (DFT) of a sequence of n complex
          numbers z , for j=0,1,...,n-1. The transform is defined by:
                   j

                                  n-1
                          ^    1  --      (   2(pi)jk)
                          z = --- >  z exp(-i -------)                 (1)
                           k      --  j   (      n   )
                              \/n j=0

          for k=0,1,...,n-1. Note that equation (1) makes sense for all
                                             ^
          integral k and with this extension z  is periodic with period n,
                                              k
               ^  ^                            ^   ^
```

i.e. $z_k = z_{k\pm n}$, and in particular $z_{-k} = z_{n-k}$.

If we write $z_j = x_j + iy_j$ and $\hat{z}_k = a_k + ib_k$, then the definition of $\hat{z}_k$ may be written in terms of sines and cosines as:

$$a_k = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} \left(x_j \cos\left(\frac{2\pi jk}{n}\right) + y_j \sin\left(\frac{2\pi jk}{n}\right) \right)$$

$$b_k = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} \left(y_j \cos\left(\frac{2\pi jk}{n}\right) - x_j \sin\left(\frac{2\pi jk}{n}\right) \right).$$

The original data values z_j may conversely be recovered from the transform $\hat{z}_k$ by an inverse discrete Fourier transform:

$$z_j = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} \hat{z}_k \exp\left(+i \frac{2\pi jk}{n}\right) \qquad (2)$$

for $j=0,1,...,n-1$. If we take the complex conjugate of (2), we find that the sequence z_j is the DFT of the sequence $\hat{z}_k$. Hence the inverse DFT of the sequence $\hat{z}_k$ may be obtained by: taking the complex conjugates of the $\hat{z}_k$; performing a DFT; and taking the complex conjugates of the result.

Notes: definitions of the discrete Fourier transform vary. Sometimes (2) is used as the definition of the DFT, and (1) as the definition of the inverse. Also the scale-factor of $1/\sqrt{n}$ may be omitted in the definition of the DFT, and replaced by $1/n$ in the definition of the inverse.

2.1.2. Real transforms

If the original sequence is purely real valued, i.e. $z_j=x_j$, then

$$\hat{z}_k=a_k+ib_k=\frac{1}{\sqrt{n}}\sum_{j=0}^{n-1}x_j\exp\left(-i\frac{2\pi jk}{n}\right)$$

and $\hat{z}_{n-k}$ is the complex conjugate of $\hat{z}_k$. Thus the DFT of a real sequence is a particular type of complex sequence, called a Hermitian sequence, or half-complex or conjugate symmetric with the properties:

$a_{n-k}=a_k$ $b_{n-k}=-b_k$ $b_0=0$ and, if n is even, $b_{n/2}=0$.

Thus a Hermitian sequence of n complex data values can be represented by only n, rather than 2n, independent real values. This can obviously lead to economies in storage, the following scheme being used in this chapter: the real parts a_k for $0\le k\le n/2$ are stored in normal order in the first $n/2+1$ locations of an array X of length n; the corresponding non-zero imaginary parts are stored in reverse order in the remaining locations of X. In other words, if X is declared with bounds (0:n-1) in the user's (sub)program, the real and imaginary parts of $\hat{z}_k$ are stored as follows:

	if n=2s	if n=2s-1
X(0)	a_0	a_0
X(1)	a_1	a_1
X(2)	a_2	a_2

.	.	.
.	.	.
.	.	.
X(s-1)	a_{s-1}	a_{s-1}
X(s)	a_s	b_{s-1}
X(s+1)	b_{s-1}	b_{s-2}
.	.	.
.	.	.
.	.	.
X(n-2)	b_2	b_2
X(n-1)	b_1	b_1

Hence $$x_j = \frac{1}{\sqrt{n}}\left(a_0 + 2\sum_{k=0}^{n/2-1}\left(a_k \cos\left(\frac{2\pi jk}{n}\right) - b_k \sin\left(\frac{2\pi jk}{n}\right)\right) + a_{n/2}\right)$$

where $a_{n/2} = 0$ if n is odd.

2.1.3. Fourier integral transforms

The usual application of the discrete Fourier transform is that of obtaining an approximation of the Fourier integral transform

$$F(s) = \int_{-\infty}^{+\infty} f(t)\exp(-i2\pi st)\,dt$$

when f(t) is negligible outside some region (0,c). Dividing the region into n equal intervals we have

$$F(s)\simeq \frac{c}{n}\sum_{j=0}^{n-1} f_j \exp(-i2(pi)sjc/n)$$

and so

$$F_k\simeq \frac{c}{n}\sum_{j=0}^{n-1} f_j \exp(-i2(pi)jk/n)$$

for $k=0,1,...,n-1$, where $f_j=f(jc/n)$ and $F_k=F(k/c)$.

Hence the discrete Fourier transform gives an approximation to the Fourier integral transform in the region s=0 to s=n/c.

If the function f(t) is defined over some more general interval (a,b), then the integral transform can still be approximated by the discrete transform provided a shift is applied to move the point a to the origin.

2.1.4. Convolutions and correlations

One of the most important applications of the discrete Fourier transform is to the computation of the discrete convolution or correlation of two vectors x and y defined (as in Brigham [1]) by:

$$\text{convolution: } z_k = \sum_{j=0}^{n-1} x_j y_{k-j}$$

$$\text{correlation: } w_k = \sum_{j=0}^{n-1} x_j y_{k+j}$$

(Here x and y are assumed to be periodic with period n.)

Under certain circumstances (see Brigham [1]) these can be used as approximations to the convolution or correlation integrals defined by:

```
                   +infty
                   /
             z(s)= |     x(t)y(s-t)dt
                   /
                   -infty
```

and

```
          +infty
          /
    w(s)= |     x(t)y(s+t)dt,   -infty<s<+infty.
          /
          -infty
```

For more general advice on the use of Fourier transforms, see Hamming [2]; more detailed information on the fast Fourier transform algorithm can be found in Van Loan [3] and Brigham [1].

2.2. References

[1] Brigham E O (1973) The Fast Fourier Transform. Prentice-Hall.

[2] Hamming R W (1962) Numerical Methods for Scientists and Engineers. McGraw-Hill.

[3] Van Loan C (1992) Computational Frameworks for the Fast Fourier Transform. SIAM Philadelphia.

3. Recommendations on Choice and Use of Routines

3.1. One-dimensional Fourier Transforms

The choice of routine is determined first of all by whether the data values constitute a real, Hermitian or general complex sequence. It is wasteful of time and storage to use an inappropriate routine.

Two groups, each of three routines, are provided

	Group 1	Group 2
Real sequences	C06EAF	C06FPF
Hermitian sequences	C06EBF	C06FQF
General complex sequences	C06ECF	C06FRF

Group 1 routines each compute a single transform of length n, without requiring any extra working storage. The Group 1 routines impose some restrictions on the value of n, namely that no prime factor of n may exceed 19 and the total number of prime factors (including repetitions) may not exceed 20 (though the latter restriction only becomes relevant when $n>10^6$).

Group 2 routines are designed to perform several transforms in a single call, all with the same value of n. They do however require more working storage. Even on scalar processors, they may be somewhat faster than repeated calls to Group 1 routines because of reduced overheads and because they pre-compute and store the required values of trigonometric functions. Group 2 routines impose no practical restrictions on the value of n; however the fast Fourier transform algorithm ceases to be 'fast' if applied to values of n which cannot be expressed as a product of small prime factors. All the above routines are particularly efficient if the only prime factors of n are 2, 3 or 5.

If extensive use is to be made of these routines, users who are concerned about efficiency are advised to conduct their own timing tests.

To compute inverse discrete Fourier transforms the above routines should be used in conjunction with the utility routines C06GBF, C06GCF and C06GQF which form the complex conjugate of a Hermitian or general sequence of complex data values.

3.2. Multi-dimensional Fourier Transforms

C06FUF computes a 2-dimensional discrete Fourier transform of a 2-dimensional sequence of complex data values. This is defined by

$$\hat{z}_{k_1 k_2} = \frac{1}{\sqrt{n_1 n_2}} \sum_{j_1=0}^{n_1-1} \sum_{j_2=0}^{n_2-1} z_{j_1 j_2} \exp\left(-i \frac{2(pi)j_1 k_1}{n_1}\right) \exp\left(-i \frac{2(pi)j_2 k_2}{n_2}\right).$$

3.3. Convolution and Correlation

C06EKF computes either the discrete convolution or the discrete correlation of two real vectors.

3.4. Index

```
C06 -- Summation of Series                        Contents -- C06
Chapter C06

Summation of Series
```

```
         C06GQF  Complex conjugate of multiple Hermitian sequences

         C06GSF  Convert Hermitian sequences to general complex sequences

\end{verbatim}
\endscroll
\end{page}
```

Discrete Fourier transform of n real data values

— nagc.ht —

```
\begin{page}{manpageXXc06eaf}{NAG Documentation: c06eaf}
\beginscroll
\begin{verbatim}



     C06EAF(3NAG)      Foundation Library (12/10/92)      C06EAF(3NAG)



          C06 -- Summation of Series                                C06EAF
                  C06EAF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          C06EAF calculates the discrete Fourier transform of a sequence of
          n real data values. (No extra workspace required.)

          2. Specification

                 SUBROUTINE C06EAF (X, N, IFAIL)
                 INTEGER          N, IFAIL
                 DOUBLE PRECISION X(N)

          3. Description

          Given a sequence of n real data values x , for j=0,1,...,n-1,
                                                  j
```

this routine calculates their discrete Fourier transform defined by:

$$\hat{z}_k = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} x_j * \exp\left(-i \frac{2(pi)jk}{n}\right), k=0,1,...,n-1.$$

(Note the scale factor of $\frac{1}{\sqrt{n}}$ in this definition.) The transformed values $\hat{z}_k$ are complex, but they form a Hermitian sequence (i.e., $\hat{z}_{n-k}$ is the complex conjugate of $\hat{z}_k$), so they are completely determined by n real numbers (see also the Chapter Introduction).

To compute the inverse discrete Fourier transform defined by:

$$\hat{w}_k = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} x_j * \exp\left(+i \frac{2(pi)jk}{n}\right),$$

this routine should be followed by a call of C06GBF to form the complex conjugates of the $\hat{z}_k$.

The routine uses the fast Fourier transform (FFT) algorithm (Brigham [1]). There are some restrictions on the value of n (see Section 5).

4. References

[1] Brigham E O (1973) The Fast Fourier Transform. Prentice-Hall.

5. Parameters

1: X(N) -- DOUBLE PRECISION array Input/Output
On entry: if X is declared with bounds (0:N-1) in the (sub) program from which C06EAF is called, then X(j) must contain

x_j, for j=0,1,...,n-1. On exit: the discrete Fourier transform stored in Hermitian form. If the components of the transform $\hat{z}_k$ are written as a_k+ib_k, and if X is declared with bounds (0:N-1) in the (sub)program from which C06EAF is called, then for 0<=k<=n/2, a_k is contained in X(k), and for 1<=k<=(n-1)/2, b_k is contained in X(n-k). (See also Section 2.1.2 of the Chapter Introduction, and the Example Program.)

2: N -- INTEGER Input
On entry: the number of data values, n. The largest prime factor of N must not exceed 19, and the total number of prime factors of N, counting repetitions, must not exceed 20. Constraint: N > 1.

3: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1
At least one of the prime factors of N is greater than 19.

IFAIL= 2
N has more than 20 prime factors.

IFAIL= 3
N <= 1.

7. Accuracy

Some indication of accuracy can be obtained by performing a subsequent inverse transform and comparing the results with the original sequence (in exact arithmetic they would be identical).

8. Further Comments

The time taken by the routine is approximately proportional to n*logn, but also depends on the factorization of n. The routine

```
          is somewhat faster than average if the only prime factors of n
          are 2, 3 or 5; and fastest of all if n is a power of 2.

          On the other hand, the routine is particularly slow if n has
          several unpaired prime factors, i.e., if the 'square-free' part
          of n has several factors. For such values of n, routine C06FAF(*)
          (which requires an additional n elements of workspace) is
          considerably faster.

          9. Example

          This program reads in a sequence of real data values, and prints
          their discrete Fourier transform (as computed by C06EAF), after
          expanding it from Hermitian form into a full complex sequence.

          It then performs an inverse transform using C06GBF and C06EBF,
          and prints the sequence so obtained alongside the original data
          values.

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Discrete Fourier transform of a Hermitian sequence

— **nagc.ht** —

```
\begin{page}{manpageXXc06ebf}{NAG Documentation: c06ebf}
\beginscroll
\begin{verbatim}



     C06EBF(3NAG)      Foundation Library (12/10/92)      C06EBF(3NAG)



          C06 -- Summation of Series                                 C06EBF
                  C06EBF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
```

your implementation to check implementation-dependent details. The symbol (*) after a NAG routine name denotes a routine that is not included in the Foundation Library.

1. Purpose

C06EBF calculates the discrete Fourier transform of a Hermitian sequence of n complex data values. (No extra workspace required.)

2. Specification

```
SUBROUTINE C06EBF (X, N, IFAIL)
INTEGER          N, IFAIL
DOUBLE PRECISION X(N)
```

3. Description

Given a Hermitian sequence of n complex data values z_j (i.e., a sequence such that z_0 is real and z_{n-j} is the complex conjugate of z_j, for j=1,2,...,n-1) this routine calculates their discrete Fourier transform defined by:

$$\hat{x}_k = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} z_j * \exp\left(-i \frac{2(pi)jk}{n}\right), \quad k=0,1,...,n-1.$$

(Note the scale factor of $\frac{1}{\sqrt{n}}$ in this definition.) The transformed values $\hat{x}_k$ are purely real (see also the the Chapter Introduction).

To compute the inverse discrete Fourier transform defined by:

$$\hat{y}_k = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} z_j * \exp\left(+i \frac{2(pi)jk}{n}\right),$$

this routine should be preceded by a call of C06GBF to form the complex conjugates of the z_j.

The routine uses the fast Fourier transform (FFT) algorithm (Brigham [1]). There are some restrictions on the value of n (see Section 5).

4. References

[1] Brigham E O (1973) The Fast Fourier Transform. Prentice-Hall.

5. Parameters

1: X(N) -- DOUBLE PRECISION array Input/Output
On entry: the sequence to be transformed stored in Hermitian form. If the data values z_j are written as x_j+iy_j, and if X is declared with bounds (0:N-1) in the subroutine from which C06EBF is called, then for 0<=j<=n/2, x_j is contained in X(j), and for 1<=j<=(n-1)/2, y_j is contained in X(n-j). (See also Section 2.1.2 of the Chapter Introduction and the Example Program.) On exit: the components of the discrete Fourier transform $\hat{x}_k$. If X is declared with bounds (0:N-1) in the (sub)program from which C06EBF is called, then $\hat{x}_k$ is stored in X(k), for k=0,1,...,n-1.

2: N -- INTEGER Input
On entry: the number of data values, n. The largest prime factor of N must not exceed 19, and the total number of prime factors of N, counting repetitions, must not exceed 20. Constraint: N > 1.

3: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

```
IFAIL= 1
     At least one of the prime factors of N is greater than 19.

IFAIL= 2
     N has more than 20 prime factors.

IFAIL= 3
     N <= 1.

7. Accuracy

Some indication of accuracy can be obtained by performing a
subsequent inverse transform and comparing the results with the
original sequence (in exact arithmetic they would be identical).

8. Further Comments

The time taken by the routine is approximately proportional to
n*logn, but also depends on the factorization of n. The routine
is somewhat faster than average if the only prime factors of n
are 2, 3 or 5; and fastest of all if n is a power of 2.

On the other hand, the routine is particularly slow if n has
several unpaired prime factors, i.e., if the 'square-free' part
of n has several factors. For such values of n, routine C06FBF(*)
(which requires an additional n elements of workspace) is
considerably faster.

9. Example

This program reads in a sequence of real data values which is
assumed to be a Hermitian sequence of complex data values stored
in Hermitian form. The input sequence is expanded into a full
complex sequence and printed alongside the original sequence. The
discrete Fourier transform (as computed by C06EBF) is printed
out.

The program then performs an inverse transform using C06EAF and
C06GBF, and prints the sequence so obtained alongside the
original data values.

The example program is not reproduced here. The source code for
all example programs is distributed with the NAG Foundation
Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

———-

Discrete Fourier transform of n complex data values

— **nagc.ht** —

```
\begin{page}{manpageXXc06ecf}{NAG Documentation: c06ecf}
\beginscroll
\begin{verbatim}



     C06ECF(3NAG)      Foundation Library (12/10/92)      C06ECF(3NAG)



          C06 -- Summation of Series                                  C06ECF
                  C06ECF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          C06ECF calculates the discrete Fourier transform of a sequence of
          n complex data values. (No extra workspace required.)

          2. Specification

                 SUBROUTINE C06ECF (X, Y, N, IFAIL)
                 INTEGER          N, IFAIL
                 DOUBLE PRECISION X(N), Y(N)

          3. Description

          Given a sequence of n complex data values z , for j=0,1,...,n-1,
                                                     j
          this routine calculates their discrete Fourier transform defined
          by:

                           n-1
                    ^    1  --       (   2(pi)jk)
                    z = --- >  z *exp(-i -------), k=0,1,...,n-1.
                     k      --  j    (      n   )
                        \/n j=0
```

(Note the scale factor of $\frac{1}{\sqrt{n}}$ in this definition.)

To compute the inverse discrete Fourier transform defined by:

$$\hat{w}_k = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} z_j \exp\left(+i \frac{2(pi)jk}{n}\right),$$

this routine should be preceded and followed by calls of C06GCF to form the complex conjugates of the z_j and the $\hat{z}_k$.

The routine uses the fast Fourier transform (FFT) algorithm (Brigham [1]). There are some restrictions on the value of n (see Section 5).

4. References

[1] Brigham E O (1973) The Fast Fourier Transform. Prentice-Hall.

5. Parameters

1: X(N) -- DOUBLE PRECISION array Input/Output
On entry: if X is declared with bounds (0:N-1) in the (sub) program from which C06ECF is called, then X(j) must contain x_j, the real part of z_j, for j=0,1,...,n-1. On exit: the real parts a_k of the components of the discrete Fourier transform. If X is declared with bounds (0:N-1) in the (sub) program from which C06ECF is called, then a_k is contained in X(k), for k=0,1,...,n-1.

2: Y(N) -- DOUBLE PRECISION array Input/Output
On entry: if Y is declared with bounds (0:N-1) in the (sub) program from which C06ECF is called, then Y(j) must contain y_j, the imaginary part of z_j, for j=0,1,...,n-1. On exit: the imaginary parts b_k of the components of the discrete Fourier transform. If Y is declared with bounds (0:N-1) in

the (sub)program from which C06ECF is called, then b_k is contained in Y(k), for k=0,1,...,n-1.

3: N -- INTEGER Input
On entry: the number of data values, n. The largest prime factor of N must not exceed 19, and the total number of prime factors of N, counting repetitions, must not exceed 20. Constraint: N > 1.

4: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1
At least one of the prime factors of N is greater than 19.

IFAIL= 2
N has more than 20 prime factors.

IFAIL= 3
N <= 1.

7. Accuracy

Some indication of accuracy can be obtained by performing a subsequent inverse transform and comparing the results with the original sequence (in exact arithmetic they would be identical).

8. Further Comments

The time taken by the routine is approximately proportional to n*logn, but also depends on the factorization of n. The routine is somewhat faster than average if the only prime factors of n are 2, 3 or 5; and fastest of all if n is a power of 2.

On the other hand, the routine is particularly slow if n has several unpaired prime factors, i.e., if the 'square-free' part of n has several factors. For such values of n, routine C06FCF(*) (which requires an additional n real elements of workspace) is considerably faster.

```
          9. Example

          This program reads in a sequence of complex data values and
          prints their discrete Fourier transform.

          It then performs an inverse transform using C06GCF and C06ECF,
          and prints the sequence so obtained alongside the original data
          values.

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Circular convolution or correlation of two real vectors

— nagc.ht —

```
\begin{page}{manpageXXc06ekf}{NAG Documentation: c06ekf}
\beginscroll
\begin{verbatim}



     C06EKF(3NAG)      Foundation Library (12/10/92)      C06EKF(3NAG)



          C06 -- Summation of Series                                 C06EKF
                  C06EKF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          C06EKF calculates the circular convolution or correlation of two
          real vectors of period n. No extra workspace is required.

          2. Specification
```

```
      SUBROUTINE C06EKF (JOB, X, Y, N, IFAIL)
      INTEGER          JOB, N, IFAIL
      DOUBLE PRECISION X(N), Y(N)
```

3. Description

This routine computes:

if JOB =1, the discrete convolution of x and y, defined by:

$$z_k = \sum_{j=0}^{n-1} x_j y_{k-j} = \sum_{j=0}^{n-1} x_{k-j} y_j ;$$

if JOB =2, the discrete correlation of x and y defined by:

$$w_k = \sum_{j=0}^{n-1} x_j y_{k+j} .$$

Here x and y are real vectors, assumed to be periodic, with period n, i.e., $x_j = x_{j\pm n} = x_{j\pm 2n} = \ldots$; z and w are then also periodic with period n.

Note: this usage of the terms 'convolution' and 'correlation' is taken from Brigham [1]. The term 'convolution' is sometimes used to denote both these computations.

If $\hat{x}$, $\hat{y}$, $\hat{z}$ and $\hat{w}$ are the discrete Fourier transforms of these sequences,

$$\text{i.e., } \hat{x}_k = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} x_j \exp\left(-i \frac{2(pi)jk}{n}\right), \text{ etc,}$$

then $\hat{z}_k = \sqrt{n}.\hat{x}_k \hat{y}_k$

and $\hat{w}_k = \sqrt{n}.\bar{\hat{x}}_k \hat{y}_k$

(the bar denoting complex conjugate).

This routine calls the same auxiliary routines as C06EAF and C06EBF to compute discrete Fourier transforms, and there are some restrictions on the value of n.

4. References

[1] Brigham E O (1973) The Fast Fourier Transform. Prentice-Hall.

5. Parameters

1: JOB -- INTEGER Input

On entry: the computation to be performed:

if JOB = 1, $z_k = \sum_{j=0}^{n-1} x_j y_{k-j}$ (convolution);

if JOB = 2, $w_k = \sum_{j=0}^{n-1} x_j y_{k+j}$ (correlation).

Constraint: JOB = 1 or 2.

2: X(N) -- DOUBLE PRECISION array Input/Output

On entry: the elements of one period of the vector x. If X is declared with bounds (0:N-1) in the (sub)program from which C06EKF is called, then X(j) must contain x_j, for j=0,1,...,n-1. On exit: the corresponding elements of the discrete convolution or correlation.

3: Y(N) -- DOUBLE PRECISION array Input/Output

On entry: the elements of one period of the vector y. If Y is declared with bounds (0:N-1) in the (sub)program from which C06EKF is called, then Y(j) must contain y_j, for j=0,1,...,n-1. On exit: the discrete Fourier transform of the convolution or correlation returned in the array X; the transform is stored in Hermitian form, exactly as described in the document C06EAF.

```
 4:  N -- INTEGER                                          Input
     On entry: the number of values, n, in one period of the
     vectors X and Y. The largest prime factor of N must not
     exceed 19, and the total number of prime factors of N,
     counting repetitions, must not exceed 20. Constraint: N > 1.

 5:  IFAIL -- INTEGER                               Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1
     At least one of the prime factors of N is greater than 19.

IFAIL= 2
     N has more than 20 prime factors.

IFAIL= 3
     N <= 1.

IFAIL= 4
     JOB /= 1 or 2.

7. Accuracy

The results should be accurate to within a small multiple of the
machine precision.

8. Further Comments

The time taken by the routine is approximately proportional to
n*logn, but also depends on the factorization of n. The routine
is faster than average if the only prime factors are 2, 3 or 5;
and fastest of all if n is a power of 2.

The routine is particularly slow if n has several unpaired prime
factors, i.e., if the 'square free' part of n has several
factors. For such values of n, routine C06FKF(*) is considerably
faster (but requires an additional workspace of n elements).

9. Example

This program reads in the elements of one period of two real
```

```
          vectors x and y and prints their discrete convolution and
          correlation (as computed by C06EKF). In realistic computations
          the number of data values would be much larger.

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Discrete Fourier transforms of m sequences

— nagc.ht —

```
\begin{page}{manpageXXc06fpf}{NAG Documentation: c06fpf}
\beginscroll
\begin{verbatim}



     C06FPF(3NAG)      Foundation Library (12/10/92)      C06FPF(3NAG)



          C06 -- Summation of Series                                C06FPF
                  C06FPF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          C06FPF computes the discrete Fourier transforms of m sequences,
          each containing n real data values. This routine is designed to
          be particularly efficient on vector processors.

          2. Specification

                 SUBROUTINE C06FPF (M, N, X, INIT, TRIG, WORK, IFAIL)
                 INTEGER          M, N, IFAIL
                 DOUBLE PRECISION X(M*N), TRIG(2*N), WORK(M*N)
```

```
      CHARACTER*1      INIT
```

3. Description

Given m sequences of n real data values x_j^p, for j=0,1,...,n-1; p=1,2,...,m, this routine simultaneously calculates the Fourier transforms of all the sequences defined by:

$$\hat{z}_k^p = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} x_j^p \exp\left(-i \frac{2(pi)jk}{n}\right), \; k=0,1,\ldots,n-1; \; p=1,2,\ldots,m.$$

(Note the scale factor $\frac{1}{\sqrt{n}}$ in this definition.)

The transformed values $\hat{z}_k^p$ are complex, but for each value of p the $\hat{z}_k^p$ form a Hermitian sequence (i.e., $\hat{z}_{n-k}^p$ is the complex conjugate of $\hat{z}_k^p$), so they are completely determined by mn real numbers (see also the Chapter Introduction).

The discrete Fourier transform is sometimes defined using a positive sign in the exponential term:

$$\hat{z}_k^p = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} x_j^p \exp\left(+i \frac{2(pi)jk}{n}\right).$$

To compute this form, this routine should be followed by a call to C06GQF to form the complex conjugates of the $\hat{z}_k^p$.

The routine uses a variant of the fast Fourier transform (FFT) algorithm (Brigham [1]) known as the Stockham self-sorting algorithm, which is described in Temperton [2]. Special coding is provided for the factors 2, 3, 4, 5 and 6. This routine is designed to be particularly efficient on vector processors, and it becomes especially fast as M, the number of transforms to be computed in parallel, increases.

4. References

[1] Brigham E O (1973) The Fast Fourier Transform. Prentice-Hall.

[2] Temperton C (1983) Fast Mixed-Radix Real Fourier Transforms. J. Comput. Phys. 52 340--350.

5. Parameters

1: M -- INTEGER Input
On entry: the number of sequences to be transformed, m.
Constraint: M >= 1.

2: N -- INTEGER Input
On entry: the number of real values in each sequence, n.
Constraint: N >= 1.

3: X(M,N) -- DOUBLE PRECISION array Input/Output
On entry: the data must be stored in X as if in a two-dimensional array of dimension (1:M,0:N-1); each of the m sequences is stored in a row of the array. In other words, if the data values of the pth sequence to be transformed are denoted by x_j^p, for j=0,1,...,n-1, then the mn elements of the array X must contain the values

$$x_0^1,x_0^2,...,x_0^m, x_1^1,x_1^2,..., x_1^m,...,x_{n-1}^1,x_{n-1}^2,...,x_{n-1}^m.$$

On exit: the m discrete Fourier transforms stored as if in a two-dimensional array of dimension (1:M,0:N-1). Each of the m transforms is stored in a row of the array in Hermitian form, overwriting the corresponding original sequence. If the n components of the discrete Fourier transform $\hat{z}_k^p$ are written as $a_k^p+ib_k^p$, then for 0<=k<=n/2, a_k^p is contained in X(p,k), and for 1<=k<=(n-1)/2, b_k^p is contained in X(p,n-k). (See also Section 2.1.2 of the Chapter Introduction.)

4: INIT -- CHARACTER*1 Input

```
     On entry: if the trigonometric coefficients required to
     compute the transforms are to be calculated by the routine
     and stored in the array TRIG, then INIT must be set equal to
     'I' (Initial call).

     If INIT contains 'S' (Subsequent call), then the routine
     assumes that trigonometric coefficients for the specified
     value of n are supplied in the array TRIG, having been
     calculated in a previous call to one of C06FPF, C06FQF or
     C06FRF.

     If INIT contains 'R' (Restart then the routine assumes that
     trigonometric coefficients for the particular value of n are
     supplied in the array TRIG, but does not check that C06FPF,
     C06FQF or C06FRF have previously been called. This option
     allows the TRIG array to be stored in an external file, read
     in and re-used without the need for a call with INIT equal
     to 'I'. The routine carries out a simple test to check that
     the current value of n is consistent with the array TRIG.
     Constraint: INIT = 'I', 'S' or 'R'.

 5:  TRIG(2*N) -- DOUBLE PRECISION array             Input/Output
     On entry: if INIT = 'S' or 'R', TRIG must contain the
     required coefficients calculated in a previous call of the
     routine. Otherwise TRIG need not be set. On exit: TRIG
     contains the required coefficients (computed by the routine
     if INIT = 'I').

 6:  WORK(M*N) -- DOUBLE PRECISION array                Workspace

 7:  IFAIL -- INTEGER                                Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).


6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

IFAIL= 1
     On entry M < 1.

IFAIL= 2
```

```
     N < 1.

IFAIL= 3
     INIT is not one of 'I', 'S' or 'R'.

IFAIL= 4
     INIT = 'S', but none of C06FPF, C06FQF or C06FRF has
     previously been called.

IFAIL= 5
     INIT = 'S' or 'R', but the array TRIG and the current value
     of N are inconsistent.

7. Accuracy

Some indication of accuracy can be obtained by performing a
subsequent inverse transform and comparing the results with the
original sequence (in exact arithmetic they would be identical).

8. Further Comments

The time taken by the routine is approximately proportional to
nm*logn, but also depends on the factors of n. The routine is
fastest if the only prime factors of n are 2, 3 and 5, and is
particularly slow if n is a large prime, or has large prime
factors.

9. Example

This program reads in sequences of real data values and prints
their discrete Fourier transforms (as computed by C06FPF). The
Fourier transforms are expanded into full complex form using
C06GSF and printed. Inverse transforms are then calculated by
calling C06GQF followed by C06FQF showing that the original
sequences are restored.

The example program is not reproduced here. The source code for
all example programs is distributed with the NAG Foundation
Library software and should be available on-line.
```

```
\end{verbatim}
\endscroll
\end{page}
```

Discrete Fourier transforms of m Hermitian sequences

— nagc.ht —

```
\begin{page}{manpageXXc06fqf}{NAG Documentation: c06fqf}
\beginscroll
\begin{verbatim}



     C06FQF(3NAG)      Foundation Library (12/10/92)      C06FQF(3NAG)



          C06 -- Summation of Series                                 C06FQF
                  C06FQF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          C06FQF computes the discrete Fourier transforms of m Hermitian
          sequences, each containing n complex data values. This routine is
          designed to be particularly efficient on vector processors.

          2. Specification

                 SUBROUTINE C06FQF (M, N, X, INIT, TRIG, WORK, IFAIL)
                 INTEGER          M, N, IFAIL
                 DOUBLE PRECISION X(M*N), TRIG(2*N), WORK(M*N)
                 CHARACTER*1      INIT

          3. Description

                                                                 p
          Given m Hermitian sequences of n complex data values z , for
                                                                 j
          j=0,1,...,n-1; p=1,2,...,m, this routine simultaneously
          calculates the Fourier transforms of all the sequences defined
          by:

                      n-1
             ^p   1   --  p    (   2(pi)jk)
             x = --- >  z *exp(-i -------), k=0,1,...,n-1; p=1,2,...,m.
              k       --  j    (      n   )
                 \/n j=0
```

(Note the scale factor $\frac{1}{\sqrt{n}}$ in this definition.)

The transformed values are purely real (see also the Chapter Introduction).

The discrete Fourier transform is sometimes defined using a positive sign in the exponential term

$$\hat{x}^p_k = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} z^p_j \exp\left(+i \frac{2(\pi)jk}{n}\right).$$

To compute this form, this routine should be preceded by a call to C06GQF to form the complex conjugates of the $\hat{z}^p_j$.

The routine uses a variant of the fast Fourier transform (FFT) algorithm (Brigham [1]) known as the Stockham self-sorting algorithm, which is described in Temperton [2]. Special code is included for the factors 2, 3, 4, 5 and 6. This routine is designed to be particularly efficient on vector processors, and it becomes especially fast as m, the number of transforms to be computed in parallel, increases.

4. References

[1] Brigham E O (1973) The Fast Fourier Transform. Prentice-Hall.

[2] Temperton C (1983) Fast Mixed-Radix Real Fourier Transforms. J. Comput. Phys. 52 340--350.

5. Parameters

1: M -- INTEGER Input
On entry: the number of sequences to be transformed, m.
Constraint: M >= 1.

2: N -- INTEGER Input
On entry: the number of data values in each sequence, n.
Constraint: N >= 1.

3: X(M,N) -- DOUBLE PRECISION array Input/Output

On entry: the data must be stored in X as if in a two-dimensional array of dimension (1:M,0:N-1); each of the m sequences is stored in a row of the array in Hermitian form. If the n data values z_j^p are written as $x_j^p+iy_j^p$, then for $0<=j<=n/2$, x_j^p is contained in X(p,j), and for $1<=j<=(n-1)/2$, y_j^p is contained in X(p,n-j). (See also Section 2.1.2 of the Chapter Introduction.) On exit: the components of the m discrete Fourier transforms, stored as if in a two-dimensional array of dimension (1:M,0:N-1). Each of the m transforms is stored as a row of the array, overwriting the corresponding original sequence. If the n components of the discrete Fourier transform are denoted by $\hat{x}_k^p$, for k=0,1,...,n-1, then the mn elements of the array X contain the values

$$\hat{x}_0^1,\hat{x}_0^2,...,\hat{x}_0^m, \hat{x}_1^1,\hat{x}_1^2,..., \hat{x}_1^m,...,\hat{x}_{n-1}^1,\hat{x}_{n-1}^2,...,\hat{x}_{n-1}^m.$$

4: INIT -- CHARACTER*1 Input

On entry: if the trigonometric coefficients required to compute the transforms are to be calculated by the routine and stored in the array TRIG, then INIT must be set equal to 'I' (Initial call).

If INIT contains 'S' (Subsequent call), then the routine assumes that trigonometric coefficients for the specified value of n are supplied in the array TRIG, having been calculated in a previous call to one of C06FPF, C06FQF or C06FRF.

If INIT contains 'R' (Restart), then the routine assumes that trigonometric coefficients for the particular value of N are supplied in the array TRIG, but does not check that C06FPF, C06FQF or C06FRF have previously been called. This option allows the TRIG array to be stored in an external file, read in and re-used without the need for a call with INIT equal to 'I'. The routine carries out a simple test to check that the current value of n is compatible with the array TRIG. Constraint: INIT = 'I', 'S' or 'R'.

5: TRIG(2*N) -- DOUBLE PRECISION array Input/Output

```
     On entry: if INIT = 'S' or 'R', TRIG must contain the
     required coefficients calculated in a previous call of the
     routine. Otherwise TRIG need not be set. On exit: TRIG
     contains the required coefficients (computed by the routine
     if INIT = 'I').

 6:  WORK(M*N) -- DOUBLE PRECISION array                Workspace

 7:  IFAIL -- INTEGER                                Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

IFAIL= 1
     On entry M < 1.

IFAIL= 2
     On entry N < 1.

IFAIL= 3
     On entry INIT is not one of 'I', 'S' or 'R'.

IFAIL= 4
     On entry INIT = 'S', but none of C06FPF, C06FQF and C06FRF
     has previously been called.

IFAIL= 5
     On entry INIT = 'S' or 'R', but the array TRIG and the
     current value of n are inconsistent.

7. Accuracy

Some indication of accuracy can be obtained by performing a
subsequent inverse transform and comparing the results with the
original sequence (in exact arithmetic they would be identical).

8. Further Comments

The time taken by the routine is approximately proportional to
nm*logn, but also depends on the factors of n. The routine is
```

```
          fastest if the only prime factors of n are 2, 3 and 5, and is
          particularly slow if n is a large prime, or has large prime
          factors.

          9. Example

          This program reads in sequences of real data values which are
          assumed to be Hermitian sequences of complex data stored in
          Hermitian form. The sequences are expanded into full complex form
          using C06GSF and printed. The discrete Fourier transforms are
          then computed (using C06FQF) and printed out. Inverse transforms
          are then calculated by calling C06FPF followed by C06GQF showing
          that the original sequences are restored.

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Discrete Fourier transforms of m complex sequences

— nagc.ht —

```
\begin{page}{manpageXXc06frf}{NAG Documentation: c06frf}
\beginscroll
\begin{verbatim}



     C06FRF(3NAG)      Foundation Library (12/10/92)      C06FRF(3NAG)



          C06 -- Summation of Series                                C06FRF
                  C06FRF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose
```

C06FRF computes the discrete Fourier transforms of m sequences, each containing n complex data values. This routine is designed to be particularly efficient on vector processors.

2. Specification

```
SUBROUTINE C06FRF (M, N, X, Y, INIT, TRIG, WORK, IFAIL)
INTEGER          M, N, IFAIL
DOUBLE PRECISION X(M*N), Y(M*N), TRIG(2*N), WORK(2*M*N)
CHARACTER*1      INIT
```

3. Description

Given m sequences of n complex data values z_j^p, for $j=0,1,...,n-1$; $p=1,2,...,m$, this routine simultaneously calculates the Fourier transforms of all the sequences defined by:

$$\hat{z}_k^p = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} z_j^p \exp\left(-i\frac{2(pi)jk}{n}\right), \quad k=0,1,...,n-1;\ p=1,2,...,m.$$

(Note the scale factor $\frac{1}{\sqrt{n}}$ in this definition.)

The discrete Fourier transform is sometimes defined using a positive sign in the exponential term

$$\hat{z}_k^p = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} z_j^p \exp\left(+i\frac{2(pi)jk}{n}\right).$$

To compute this form, this routine should be preceded and followed by a call of C06GCF to form the complex conjugates of the z_j^p and the $\hat{z}_k^p$.

The routine uses a variant of the fast Fourier transform (FFT) algorithm (Brigham [1]) known as the Stockham self-sorting algorithm, which is described in Temperton [2]. Special code is provided for the factors 2, 3, 4, 5 and 6. This routine is designed to be particularly efficient on vector processors, and it becomes especially fast as m, the number of transforms to be computed in parallel, increases.

4. References

[1] Brigham E O (1973) The Fast Fourier Transform. Prentice-Hall.

[2] Temperton C (1983) Self-sorting Mixed-radix Fast Fourier Transforms. J. Comput. Phys. 52 1--23.

5. Parameters

1: M -- INTEGER Input

On entry: the number of sequences to be transformed, m.
Constraint: M >= 1.

2: N -- INTEGER Input

On entry: the number of complex values in each sequence, n.
Constraint: N >= 1.

3: X(M,N) -- DOUBLE PRECISION array Input/Output

4: Y(M,N) -- DOUBLE PRECISION array Input/Output

On entry: the real and imaginary parts of the complex data must be stored in X and Y respectively as if in a two-dimensional array of dimension (1:M,0:N-1); each of the m sequences is stored in a row of each array. In other words, if the real parts of the pth sequence to be transformed are denoted by x_j^p, for j=0,1,...,n-1, then the mn elements of the array X must contain the values

$$x_0^1,x_0^2,\ldots,x_0^m, x_1^1,x_1^2,\ldots,x_1^m,\ldots, x_{n-1}^1,x_{n-1}^2,\ldots,x_{n-1}^m .$$

On exit: X and Y are overwritten by the real and imaginary parts of the complex transforms.

5: INIT -- CHARACTER*1 Input

On entry: if the trigonometric coefficients required to compute the transforms are to be calculated by the routine and stored in the array TRIG, then INIT must be set equal to 'I' (Initial call).

If INIT contains 'S' (Subsequent call), then the routine assumes that trigonometric coefficients for the specified value of n are supplied in the array TRIG, having been

```
     calculated in a previous call to one of C06FPF, C06FQF or
     C06FRF.

     If INIT contains 'R' (Restart) then the routine assumes that
     trigonometric coefficients for the particular value of n are
     supplied in the array TRIG, but does not check that C06FPF,
     C06FQF or C06FRF have previously been called. This option
     allows the TRIG array to be stored in an external file, read
     in and re-used without the need for a call with INIT equal
     to 'I'. The routine carries out a simple test to check that
     the current value of n is compatible with the array TRIG.
     Constraint: INIT = 'I', 'S' or 'R'.

 6:  TRIG(2*N) -- DOUBLE PRECISION array             Input/Output
     On entry: if INIT = 'S' or 'R', TRIG must contain the
     required coefficients calculated in a previous call of the
     routine. Otherwise TRIG need not be set. On exit: TRIG
     contains the required coefficients (computed by the routine
     if INIT = 'I').

 7:  WORK(2*M*N) -- DOUBLE PRECISION array              Workspace

 8:  IFAIL -- INTEGER                                Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

IFAIL= 1
     On entry M < 1.

IFAIL= 2
     On entry N < 1.

IFAIL= 3
     On entry INIT is not one of 'I', 'S' or 'R'.

IFAIL= 4
     On entry INIT = 'S', but none of C06FPF, C06FQF and C06FRF
     has previously been called.
```

```
          IFAIL= 5
               On entry INIT = 'S' or 'R', but the array TRIG and the
               current value of n are inconsistent.

          7. Accuracy

          Some indication of accuracy can be obtained by performing a
          subsequent inverse transform and comparing the results with the
          original sequence (in exact arithmetic they would be identical).

          8. Further Comments

          The time taken by the routine is approximately proportional to
          nm*logn, but also depends on the factors of n. The routine is
          fastest if the only prime factors of n are 2, 3 and 5, and is
          particularly slow if n is a large prime, or has large prime
          factors.

          9. Example

          This program reads in sequences of complex data values and prints
          their discrete Fourier transforms (as computed by C06FRF).
          Inverse transforms are then calculated using C06GCF and C06FRF
          and printed out, showing that the original sequences are
          restored.

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

———

Discrete Fourier transform of bivariate complex data

— nagc.ht —

```
\begin{page}{manpageXXc06fuf}{NAG Documentation: c06fuf}
\beginscroll
\begin{verbatim}


     C06FUF(3NAG)      Foundation Library (12/10/92)      C06FUF(3NAG)
```

C06 -- Summation of Series C06FUF

C06FUF -- NAG Foundation Library Routine Document

Note: Before using this routine, please read the Users' Note for your implementation to check implementation-dependent details. The symbol (*) after a NAG routine name denotes a routine that is not included in the Foundation Library.

1. Purpose

C06FUF computes the two-dimensional discrete Fourier transform of a bivariate sequence of complex data values. This routine is designed to be particularly efficient on vector processors.

2. Specification

```
       SUBROUTINE C06FUF (M, N, X, Y, INIT, TRIGM, TRIGN, WORK,
      1                   IFAIL)
       INTEGER          M, N, IFAIL
       DOUBLE PRECISION X(M*N), Y(M*N), TRIGM(2*M), TRIGN(2*N),
      1                 WORK(2*M*N)
       CHARACTER*1      INIT
```

3. Description

This routine computes the two-dimensional discrete Fourier transform of a bivariate sequence of complex data values $z_{j_1 j_2}$, where $j_1=0,1,...,m-1$, $j_2=0,1,...,n-1$.

The discrete Fourier transform is here defined by:

$$\hat{z}_{k_1 k_2} = \frac{1}{\sqrt{mn}} \sum_{j_1=0}^{m-1} \sum_{j_2=0}^{n-1} z_{j_1 j_2} * \exp\left(-2(pi)i\left(\frac{j_1 k_1}{m} + \frac{j_2 k_2}{n}\right)\right),$$

where $k_1=0,1,...,m-1$, $k_2=0,1,...,n-1$.

(Note the scale factor of $\frac{1}{\sqrt{mn}}$ in this definition.)

```
                    \/mn

To compute the inverse discrete Fourier transform, defined with
exp(+2(pi)i(...)) in the above formula instead of exp(-
2(pi)i(...)), this routine should be preceded and followed by
calls of C06GCF to form the complex conjugates of the data values
and the transform.

This routine calls C06FRF to perform multiple one-dimensional
discrete Fourier transforms by the fast Fourier transform (FFT)
algorithm in Brigham [1]. It is designed to be particularly
efficient on vector processors.

4. References

[1]   Brigham E O (1973) The Fast Fourier Transform. Prentice-
      Hall.

[2]   Temperton C (1983) Self-sorting Mixed-radix Fast Fourier
      Transforms. J. Comput. Phys. 52 1--23.

5. Parameters

 1:  M -- INTEGER                                          Input
     On entry: the number of rows, m, of the arrays X and Y.
     Constraint: M >= 1.

 2:  N -- INTEGER                                          Input
     On entry: the number of columns, n, of the arrays X and Y.
     Constraint: N >= 1.

 3:  X(M,N) -- DOUBLE PRECISION array               Input/Output

 4:  Y(M,N) -- DOUBLE PRECISION array               Input/Output
     On entry: the real and imaginary parts of the complex data
     values must be stored in arrays X and Y respectively. If X
     and Y are regarded as two-dimensional arrays of dimension
     (0:M-1,0:N-1), then X(j ,j ) and Y(j ,j ) must contain the
                             1  2         1  2
     real and imaginary parts of z    . On exit: the real and
                                  j j
                                   1 2
     imaginary parts respectively of the corresponding elements
     of the computed transform.

 5:  INIT -- CHARACTER*1                                   Input
     On entry: if the trigonometric coefficients required to
     compute the transforms are to be calculated by the routine
     and stored in the arrays TRIGM and TRIGN, then INIT must be
```

```
     set equal to 'I', (Initial call).

     If INIT contains 'S', (Subsequent call), then the routine
     assumes that trigonometric coefficients for the specified
     values of m and n are supplied in the arrays TRIGM and
     TRIGN, having been calculated in a previous call to the
     routine.

     If INIT contains 'R', (Restart), then the routine assumes
     that trigonometric coefficients for the particular values of
     m and n are supplied in the arrays TRIGM and TRIGN, but does
     not check that the routine has previously been called. This
     option allows the TRIGM and TRIGN arrays to be stored in an
     external file, read in and re-used without the need for a
     call with INIT equal to 'I'. The routine carries out a
     simple test to check that the current values of m and n are
     compatible with the arrays TRIGM and TRIGN. Constraint: INIT
     = 'I', 'S' or 'R'.

 6:  TRIGM(2*M) -- DOUBLE PRECISION array            Input/Output

 7:  TRIGN(2*N) -- DOUBLE PRECISION array            Input/Output
     On entry: if INIT = 'S' or 'R',TRIGM and TRIGN must contain
     the required coefficients calculated in a previous call of
     the routine. Otherwise TRIGM and TRIGN need not be set.

     If m=n the same array may be supplied for TRIGM and TRIGN.
     On exit: TRIGM and TRIGN contain the required coefficients
     (computed by the routine if INIT = 'I').

 8:  WORK(2*M*N) -- DOUBLE PRECISION array              Workspace

 9:  IFAIL -- INTEGER                                Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

IFAIL= 1
     On entry M < 1.
```

```
IFAIL= 2
     On entry N < 1.

IFAIL= 3
     On entry INIT is not one of 'I', 'S' or 'R'.

IFAIL= 4
     On entry INIT = 'S', but C06FUF has not previously been
     called.

IFAIL= 5
     On entry INIT = 'S' or 'R', but at least one of the arrays
     TRIGM and TRIGN is inconsistent with the current value of M
     or N.

7. Accuracy

Some indication of accuracy can be obtained by performing a
subsequent inverse transform and comparing the results with the
original sequence (in exact arithmetic they would be identical).

8. Further Comments

The time taken by the routine is approximately proportional to
mn*log(mn), but also depends on the factorization of the
individual dimensions m and n. The routine is somewhat faster
than average if their only prime factors are 2, 3 or 5; and
fastest of all if they are powers of 2.

9. Example

This program reads in a bivariate sequence of complex data values
and prints the two-dimensional Fourier transform. It then
performs an inverse transform and prints the sequence so
obtained, which may be compared to the original data values.

The example program is not reproduced here. The source code for
all example programs is distributed with the NAG Foundation
Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Summation of Series

— nagc.ht —

```
\begin{page}{manpageXXc06gbf}{NAG Documentation: c06gbf}
\beginscroll
\begin{verbatim}



     C06GBF(3NAG)      Foundation Library (12/10/92)      C06GBF(3NAG)



          C06 -- Summation of Series                                 C06GBF
                  C06GBF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          C06GBF forms the complex conjugate of a Hermitian sequence of n
          data values.

          2. Specification

                 SUBROUTINE C06GBF (X, N, IFAIL)
                 INTEGER          N, IFAIL
                 DOUBLE PRECISION X(N)

          3. Description

          This is a utility routine for use in conjunction with C06EAF,
          C06EBF, C06FAF(*) or C06FBF(*) to calculate inverse discrete
          Fourier transforms (see the Chapter Introduction).

          4. References

          None.

          5. Parameters

           1:  X(N) -- DOUBLE PRECISION array                  Input/Output
               On entry: if the data values z  are written as x +iy  and
                                              j                 j   j
               if X is declared with bounds (0:N-1) in the (sub)program
```

```
     from which C06GBF is called, then for 0<=j<=n/2, X(j) must
     contain x (=x   ), while for n/2<j<=n-1, X(j) must contain
              j   n-j
     -y (=y   ). In other words, X must contain the Hermitian
       j   n-j
     sequence in Hermitian form. (See also Section 2.1.2 of the
     Chapter Introduction). On exit: the imaginary parts y  are
                                                          j
     negated. The real parts x  are not referenced.
                              j

 2:  N -- INTEGER                                          Input
     On entry: the number of data values, n. Constraint: N >= 1.

 3:  IFAIL -- INTEGER                               Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1
     N < 1.

7. Accuracy

Exact.

8. Further Comments

The time taken by the routine is negligible.

9. Example

This program reads in a sequence of real data values, calls
C06EAF followed by C06GBF to compute their inverse discrete
Fourier transform, and prints this after expanding it from
Hermitian form into a full complex sequence.

The example program is not reproduced here. The source code for
all example programs is distributed with the NAG Foundation
Library software and should be available on-line.

\end{verbatim}
\endscroll
```

```
\end{page}
```

———

Complex conjugate of a sequence of n data values

— **nagc.ht** —

```
\begin{page}{manpageXXc06gcf}{NAG Documentation: c06gcf}
\beginscroll
\begin{verbatim}



     C06GCF(3NAG)      Foundation Library (12/10/92)      C06GCF(3NAG)



          C06 -- Summation of Series                                 C06GCF
                  C06GCF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          C06GCF forms the complex conjugate of a sequence of n data
          values.

          2. Specification

                 SUBROUTINE C06GCF (Y, N, IFAIL)
                 INTEGER          N, IFAIL
                 DOUBLE PRECISION Y(N)

          3. Description

          This is a utility routine for use in conjunction with C06ECF or
          C06FCF(*) to calculate inverse discrete Fourier transforms (see
          the Chapter Introduction).

          4. References

          None.
```

```
          5. Parameters

           1:  Y(N) -- DOUBLE PRECISION array                  Input/Output
               On entry: if Y is declared with bounds (0:N-1) in the (sub)
               program which C06GCF is called, then Y(j) must contain the
               imaginary part of the jth data value, for 0<=j<=n-1. On
               exit: these values are negated.

           2:  N -- INTEGER                                           Input
               On entry: the number of data values, n. Constraint: N >= 1.

           3:  IFAIL -- INTEGER                                Input/Output
               On entry: IFAIL must be set to 0, -1 or 1. For users not
               familiar with this parameter (described in the Essential
               Introduction) the recommended value is 0.

               On exit: IFAIL = 0 unless the routine detects an error (see
               Section 6).

          6. Error Indicators and Warnings

          Errors detected by the routine:

          IFAIL= 1
               N < 1.

          7. Accuracy

          Exact.

          8. Further Comments

          The time taken by the routine is negligible.

          9. Example

          This program reads in a sequence of complex data values and
          prints their inverse discrete Fourier transform as computed by
          calling C06GCF, followed by C06ECF and C06GCF again.

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Complex conjugates of m Hermitian sequences

— nagc.ht —

```
\begin{page}{manpageXXc06gqf}{NAG Documentation: c06gqf}
\beginscroll
\begin{verbatim}



     C06GQF(3NAG)      Foundation Library (12/10/92)      C06GQF(3NAG)



          C06 -- Summation of Series                                C06GQF
                  C06GQF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          C06GQF forms the complex conjugates of m Hermitian sequences,
          each containing n data values.

          2. Specification

                 SUBROUTINE C06GQF (M, N, X, IFAIL)
                 INTEGER          M, N, IFAIL
                 DOUBLE PRECISION X(M*N)

          3. Description

          This is a utility routine for use in conjunction with C06FPF and
          C06FQF to calculate inverse discrete Fourier transforms (see the
          Chapter Introduction).

          4. References

          None.

          5. Parameters

           1:  M -- INTEGER                                            Input
               On entry: the number of Hermitian sequences to be
               conjugated, m. Constraint: M >= 1.
```

2: N -- INTEGER Input
On entry: the number of data values in each Hermitian sequence, n. Constraint: N >= 1.

3: X(M,N) -- DOUBLE PRECISION array Input/Output
On entry: the data must be stored in array X as if in a two-dimensional array of dimension (1:M,0:N-1); each of the m sequences is stored in a row of the array in Hermitian form. If the n data values z_j^p are written as $x_j^p+iy_j^p$, then for 0<=j<=n/2, x_j^p is contained in X(p,j), and for 1<=j<=(n-1)/2, y_j^p is contained in X(p,n-j). (See also Section 2.1.2 of the Chapter Introduction.) On exit: the imaginary parts y_j^p are negated. The real parts x_j^p are not referenced.

4: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL= 1
On entry M < 1.

IFAIL= 2
On entry N < 1.

7. Accuracy

Exact.

8. Further Comments

None.

```
        9. Example

        This program reads in sequences of real data values which are
        assumed to be Hermitian sequences of complex data stored in
        Hermitian form. The sequences are expanded into full complex form
        using C06GSF and printed. The sequences are then conjugated
        (using C06GQF) and the conjugated sequences are expanded into
        complex form using C06GSF and printed out.

        The example program is not reproduced here. The source code for
        all example programs is distributed with the NAG Foundation
        Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Form real and imaginary parts of m Hermitian sequences

— nagc.ht —

```
\begin{page}{manpageXXc06gsf}{NAG Documentation: c06gsf}
\beginscroll
\begin{verbatim}



     C06GSF(3NAG)      Foundation Library (12/10/92)      C06GSF(3NAG)



        C06 -- Summation of Series                                C06GSF
                C06GSF -- NAG Foundation Library Routine Document

        Note: Before using this routine, please read the Users' Note for
        your implementation to check implementation-dependent details.
        The symbol (*) after a NAG routine name denotes a routine that is
        not included in the Foundation Library.

        1. Purpose

        C06GSF takes m Hermitian sequences, each containing n data
        values, and forms the real and imaginary parts of the m
        corresponding complex sequences.
```

2. Specification

```
SUBROUTINE C06GSF (M, N, X, U, V, IFAIL)
INTEGER          M, N, IFAIL
DOUBLE PRECISION X(M*N), U(M*N), V(M*N)
```

3. Description

This is a utility routine for use in conjunction with C06FPF and C06FQF (see the Chapter Introduction).

4. References

None.

5. Parameters

1: M -- INTEGER Input
On entry: the number of Hermitian sequences, m, to be converted into complex form. Constraint: M >= 1.

2: N -- INTEGER Input
On entry: the number of data values, n, in each sequence. Constraint: N >= 1.

3: X(M,N) -- DOUBLE PRECISION array Input
On entry: the data must be stored in X as if in a two-dimensional array of dimension (1:M,0:N-1); each of the m sequences is stored in a row of the array in Hermitian form. If the n data values z_j^p are written as $x_j^p+iy_j^p$, then for $0<=j<=n/2$, x_j^p is contained in X(p,j), and for $1<=j<=(n-1)/2$, y_j^p is contained in X(p,n-j). (See also Section 2.1.2 of the Chapter Introduction.)

4: U(M,N) -- DOUBLE PRECISION array Output

5: V(M,N) -- DOUBLE PRECISION array Output
On exit: the real and imaginary parts of the m sequences of length n, are stored in U and V respectively, as if in two-dimensional arrays of dimension (1:M,0:N-1); each of the m sequences is stored as if in a row of each array. In other words, if the real parts of the pth sequence are denoted by
p

```
     x , for j=0,1,...,n-1 then the mn elements of the array U
      j
     contain the values
          1  2      m   1  2      m       1    2        m
         x ,x ,...,x , x ,x ,...,x ,..., x   ,x   ,...,x
          0  0      0   1  1      1       n-1  n-1      n-1

 6:  IFAIL -- INTEGER                                Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

IFAIL= 1
     On entry M < 1.

IFAIL= 2
     On entry N < 1.

7. Accuracy

Exact.

8. Further Comments

None.

9. Example

This program reads in sequences of real data values which are
assumed to be Hermitian sequences of complex data stored in
Hermitian form. The sequences are then expanded into full complex
form using C06GSF and printed.

The example program is not reproduced here. The source code for
all example programs is distributed with the NAG Foundation
Library software and should be available on-line.
```

```
\end{verbatim}
\endscroll
\end{page}
```

———-

22.3 nagd.ht

Quadrature

— **nagd.ht** —

```
\begin{page}{manpageXXd01}{NAG Documentation: d01}
\beginscroll
\begin{verbatim}



     D01(3NAG)         Foundation Library (12/10/92)         D01(3NAG)



          D01 -- Quadrature                             Introduction -- D01
                                    Chapter D01
                                    Quadrature

          1. Scope of the Chapter

          This chapter provides routines for the numerical evaluation of
          definite integrals in one or more dimensions and for evaluating
          weights and abscissae of integration rules.

          2. Background to the Problems

          The routines in this chapter are designed to estimate:

          (a)   the value of a one-dimensional definite integral of the
                form:
                                        b
                                        /
                                        |f(x)dx                          (1)
                                        /
                                        a
                where f(x) is defined by the user, either at a set of
                points (x ,f(x )), for i=1,2,...,n where a=x <x <...<x =b,
                         i    i                             1  2      n
                or in the form of a function; and the limits of integration
                a,b may be finite or infinite.
```

Some methods are specially designed for integrands of the form

$$f(x)=w(x)g(x) \tag{2}$$

which contain a factor w(x), called the weight-function, of a specific form. These methods take full account of any peculiar behaviour attributable to the w(x) factor.

(b) the value of a multi-dimensional definite integral of the form:

$$\int_{R_n} f(x_1,x_2,\ldots,x_n)dx_n \ldots dx_2\, dx_1 \tag{3}$$

where $f(x_1,x_2,\ldots,x_n)$ is a function defined by the user and R_n is some region of n-dimensional space.

The simplest form of R_n is the n-rectangle defined by:

$$a_i \le x_i \le b_i, \quad i=1,2,\ldots,n \tag{4}$$

where a_i and b_i are constants. When a_i and b_i are functions of x_j (j<i), the region can easily be transformed to the rectangular form (see Davis and Rabinowitz [1] page 266). Some of the methods described incorporate the transformation procedure.

2.1. One-dimensional Integrals

To estimate the value of a one-dimensional integral, a quadrature rule uses an approximation in the form of a weighted sum of integrand values, i.e.,

$$\int_a^b f(x)dx \simeq \sum_{i=1}^{N} w_i f(x_i). \tag{5}$$

The points x_i within the interval [a,b] are known as the abscissae, and the w_i are known as the weights.

More generally, if the integrand has the form (2), the corresponding formula is

```
b             N
/             --
|w(x)g(x)dx~= >  w g(x ).                    (6)
/             --  i   i
a             i=1
```

If the integrand is known only at a fixed set of points, these points must be used as the abscissae, and the weighted sum is calculated using finite-difference methods. However, if the functional form of the integrand is known, so that its value at any abscissa is easily obtained, then a wide variety of quadrature rules are available, each characterised by its choice of abscissae and the corresponding weights.

The appropriate rule to use will depend on the interval [a,b] - whether finite or otherwise - and on the form of any w(x) factor in the integrand. A suitable value of N depends on the general behaviour of f(x); or of g(x), if there is a w(x) factor present.

Among possible rules, we mention particularly the Gaussian formulae, which employ a distribution of abscissae which is optimal for f(x) or g(x) of polynomial form.

The choice of basic rules constitutes one of the principles on which methods for one-dimensional integrals may be classified. The other major basis of classification is the implementation strategy, of which some types are now presented.

(a) Single rule evaluation procedures

A fixed number of abscissae, N, is used. This number and the particular rule chosen uniquely determine the weights and abscissae. No estimate is made of the accuracy of the result.

(b) Automatic procedures

The number of abscissae, N, within [a,b] is gradually increased until consistency is achieved to within a level of accuracy (absolute or relative) requested by the user. There are essentially two ways of doing this; hybrid forms of these two methods are also possible:

(i) whole interval procedures (non-adaptive)

A series of rules using increasing values of N are successively applied over the whole interval [a,b]. It is clearly more economical if abscissae already used for a lower value of N can be used again as part of a higher-order formula. This principle is known as optimal extension. There is no overlap between the abscissae used in Gaussian formulae of different orders. However, the Kronrod formulae are designed to give an optimal (2N+1)-point formula by adding (N+1) points to an N-point Gauss formula. Further extensions have been developed by Patterson.

(ii) adaptive procedures

The interval [a,b] is repeatedly divided into a number of sub-intervals, and integration rules are applied separately to each sub-interval. Typically, the subdivision process will be carried further in the neighbourhood of a sharp peak in the integrand, than where the curve is smooth. Thus, the distribution of abscissae is adapted to the shape of the integrand.

Subdivision raises the problem of what constitutes an acceptable accuracy in each sub-interval. The usual global acceptability criterion demands that the sum of the absolute values of the error estimates in the sub-intervals should meet the conditions required of the error over the whole interval. Automatic extrapolation over several levels of subdivision may eliminate the effects of some types of singularities.

An ideal general-purpose method would be an automatic method which could be used for a wide variety of integrands, was efficient (i.e., required the use of as few abscissae as possible), and was reliable (i.e., always gave results within the requested accuracy). Complete reliability is unobtainable, and generally higher reliability is obtained at the expense of efficiency, and vice versa. It must therefore be emphasised that the automatic routines in this chapter cannot be assumed to be 100% reliable. In general, however, the reliability is very high.

2.2. Multi-dimensional Integrals

A distinction must be made between cases of moderately low dimensionality (say, up to 4 or 5 dimensions), and those of higher dimensionality. Where the number of dimensions is limited, a one-dimensional method may be applied to each dimension, according to some suitable strategy, and high accuracy may be obtainable (using product rules). However, the number of integrand evaluations rises very rapidly with the number of dimensions, so that the accuracy obtainable with an acceptable amount of computational labour is limited; for example a product of 3-point rules in 20 dimensions would require more than 10^9

integrand evaluations. Special techniques such as the Monte Carlo methods can be used to deal with high dimensions.

(a) Products of one-dimensional rules

Using a two-dimensional integral as an example, we have

$$\int_{a_1}^{b_1}\int_{a_2}^{b_2} f(x,y)\,dy\,dx \simeq \sum_{i=1}^{N} w_i \left[\int_{a_2}^{b_2} f(x_i,y)\,dy\right] \qquad (7)$$

$$\int_{a_1}^{b_1}\int_{a_2}^{b_2} f(x,y)\,dy\,dx \simeq \sum_{i=1}^{N}\sum_{j=1}^{N} w_i v_j f(x_i,y_j) \qquad (8)$$

where (w_i,x_i) and (v_i,y_i) are the weights and abscissae of the rules used in the respective dimensions.

A different one-dimensional rule may be used for each dimension, as appropriate to the range and any weight function present, and a different strategy may be used, as appropriate to the integrand behaviour as a function of each independent variable.

For a rule-evaluation strategy in all dimensions, the formula (8) is applied in a straightforward manner. For automatic strategies (i.e., attempting to attain a requested accuracy), there is a problem in deciding what accuracy must be requested in the inner integral(s). Reference to formula (7) shows that the presence of a limited but random error in the y-integration for different values of x_i can produce a 'jagged' function of x, which may be difficult to integrate to the desired accuracy and for this reason products of automatic one-dimensional routines should be used with caution (see also Lyness [3]).

(b) Monte Carlo methods

These are based on estimating the mean value of the integrand sampled at points chosen from an appropriate statistical distribution function. Usually a variance reducing procedure is incorporated to combat the fundamentally slow rate of convergence of the rudimentary form of the technique. These methods can be effective by comparison with alternative methods when the integrand contains singularities or is erratic in some way, but they are of quite limited accuracy.

(c) Number theoretic methods

These are based on the work of Korobov and Conroy and operate by exploiting implicitly the properties of the Fourier expansion of the integrand. Special rules, constructed from so-called optimal coefficients, give a particularly uniform distribution of the points throughout n-dimensional space and from their number theoretic properties minimize the error on a prescribed class of integrals. The method can be combined with the Monte Carlo procedure.

(d) Sag-Szekeres method

By transformation this method seeks to induce properties into the integrand which make it accurately integrable by the trapezoidal rule. The transformation also allows effective control over the number of integrand evaluations.

(e) Automatic adaptive procedures

An automatic adaptive strategy in several dimensions normally involves division of the region into subregions, concentrating the divisions in those parts of the region where the integrand is worst behaved. It is difficult to arrange with any generality for variable limits in the inner integral(s). For this reason, some methods use a region where all the limits are constants; this is called a hyper-rectangle. Integrals over regions defined by variable or infinite limits may be handled by transformation to a hyper-rectangle. Integrals over regions so irregular that such a transformation is not feasible may be handled by surrounding the region by an appropriate hyper-rectangle and defining the integrand to be zero outside the desired region. Such a technique should always be followed by a Monte Carlo method for integration.

The method used locally in each subregion produced by the adaptive subdivision process is usually one of three types: Monte Carlo, number theoretic or deterministic. Deterministic methods are usually the most rapidly convergent but are often expensive to use for high dimensionality and not as robust as the other techniques.

2.3. References

Comprehensive reference:

[1] Davis P J and Rabinowitz P (1975) Methods of Numerical Integration. Academic Press.

Special topics:

[2] Gladwell I (1986) Vectorisation of one dimensional quadrature codes. Techincal Report. TR7/86 NAG.

[3] Lyness J N (1983) When not to use an automatic quadrature routine. SIAM Review. 25 63--87.

[4] Piessens R, De Doncker-Kapenga E, Uberhuber C and Kahaner D (1983) QUADPACK, A Subroutine Package for Automatic Integration. Springer-Verlag.

[5] Sobol I M (1974) The Monte Carlo Method. The University of Chicago Press.

[6] Stroud A H (1971) Approximate Calculation of Multiple Integrals. Prentice-Hall.

3. Recommendations on Choice and Use of Routines

The following three sub-sections consider in turn routines for: one-dimensional integrals over a finite interval, and over a semi-infinite or an infinite interval; and multi-dimensional integrals. Within each sub-section, routines are classified by the type of method, which ranges from simple rule evaluation to automatic adaptive algorithms. The recommendations apply particularly when the primary objective is simply to compute the value of one or more integrals, and in these cases the automatic adaptive routines are generally the most convenient and reliable, although also the most expensive in computing time.

Note however that in some circumstances it may be counter-productive to use an automatic routine. If the results of the quadrature are to be used in turn as input to a further computation (e.g. an 'outer' quadrature or an optimization problem), then this further computation may be adversely affected by the 'jagged performance profile' of an automatic routine; a simple rule-evaluation routine may provide much better overall performance. For further guidance, the article Lyness [3] is recommended.

3.1. One-dimensional Integrals over a Finite Interval

```
(a)    Integrand defined as a set of points

       If f(x) is defined numerically at four or more points, then
       the Gill-Miller finite difference method (D01GAF) should be
       used. The interval of integration is taken to coincide with
       the range of x-values of the points supplied. It is in the
       nature of this problem that any routine may be unreliable.
       In order to check results independently and so as to
       provide an alternative technique the user may fit the
       integrand by Chebyshev series using E02ADF and then use
       routines E02AJF and E02AKF to evaluate its integral (which
       need not be restricted to the range of the integration
       points, as is the case for D01GAF). A further alternative
       is to fit a cubic spline to the data using E02BAF and then
       to evaluate its integral using E02BDF.

(b)    Integrand defined as a function

       If the functional form of f(x) is known, then one of the
       following approaches should be taken. They are arranged in
       the order from most specific to most general, hence the
       first applicable procedure in the list will be the most
       efficient.  However, if the user does not wish to make any
       assumptions about the integrand, the most reliable routine
       to use will be D01AJF, although this will in general be
       less efficient for simple integrals.
       (i)    Rule-evaluation routines

              If f(x) is known to be sufficiently well-behaved
              (more precisely, can be closely approximated by a
              polynomial of moderate degree), a Gaussian routine
              with a suitable number of abscissae may be used.

              D01BBF may be used if it is required to examine the
              weights and abscissae. In this case, the user should
              write the code for the evaluation of quadrature
              summation (6).

       (ii)   Automatic adaptive routines

              Firstly, several routines are available for
              integrands of the form w(x)g(x) where g(x) is a '
              smooth' function (i.e., has no singularities, sharp
              peaks or violent oscillations in the interval of
              integration) and w(x) is a weight function of one of
              the following forms:

                           (alpha)     (beta)          k          l
              if w(x)=(b-x)       (x-a)      (log(b-x)) (log(x-a))
```

where k,l=0 or 1, (alpha),(beta)>-1: use D01APF;

if w(x)=1/(x-c): use D01AQF (this integral is called the Hilbert transform of g);

if w(x)=cos((omega)x) or sin((omega)x): use D01ANF (this routine can also handle certain types of singularities in g(x)).

Secondly, there are some routines for general f(x). If f(x) is known to be free of singularities, though it may be oscillatory, D01AKF may be used.

The most powerful of the finite interval integration routine is D01AJF (which can cope with singularities of several types). It may be used if none of the more specific situations described above applies. D01AJF is very reliable, particularly where the integrand has singularities other than at an end-point, or has discontinuities or cusps, and is therefore recommended where the integrand is known to be badly-behaved, or where its nature is completely unknown.

Most of the routines in this chapter require the user to supply a function or subroutine to evaluate the integrand at a single point.

If f(x) has singularities of certain types, discontinuities or sharp peaks occurring at known points, the integral should be evaluated separately over each of the subranges or D01ALF may be used.

3.2. One-dimensional Integrals over a Semi-infinite or Infinite Interval

(a) Integrand defined as a set of points

If f(x) is defined numerically at four or more points, and the portion of the integral lying outside the range of the points supplied may be neglected, then the Gill-Miller finite difference method, D01GAF, should be used.

(b) Integrand defined as a function

(i) Rule evaluation routines

If f(x) behaves approximately like a polynomial in x, apart from a weight function of the form

```
 -(beta)x
e                    (beta)>0 (semi-infinite interval,
                              lower limit finite);
```

```
       -(beta)x
 or   e                    (beta)<0 (semi-infinite interval,
                                     upper limit finite);

                                 2
        -(beta)(x-(alpha))
 or    e                   (beta)>0 (infinite interval);

               or if f(x) behaves approximately like a
                                       -1
               polynomial in (x+B)   (semi-infinite range);
               then the Gaussian routines may be used.

               D01BBF may be used if it is required to
               examine the weights and abscissae. In this
               case, the user should write the code for the
               evaluation of quadrature summation (6).

(ii)  Automatic adaptive routines

      D01AMF may be used, except for integrands which decay
      slowly towards an infinite end-point, and oscillate
      in sign over the entire range. For this class, it may
      be possible to calculate the integral by integrating
      between the zeros and invoking some extrapolation
      process.

      D01ASF may be used for integrals involving weight
      functions of the form cos((omega)x) and sin((omega)x)
      over a semi-infinite interval (lower limit finite).

      The following alternative procedures are mentioned
      for completeness, though their use will rarely be
      necessary:
      (1)   If the integrand decays rapidly towards an
            infinite end-point, a finite cut-off may be
            chosen, and the finite range methods applied.

      (2)   If the only irregularities occur in the finite
            part (apart from a singularity at the finite
            limit, with which D01AMF can cope), the range
            may be divided, with D01AMF used on the
            infinite part.

      (3)   A transformation to finite range may be
            employed, e.g.
                             1-t
                          x= ---  or x=-log t
                              t            e
```

will transform (0,infty) to (1,0) while for infinite ranges we have

$$\int_{-\infty}^{+\infty} f(x)dx = \int_{0}^{\infty} [f(x)+f(-x)]dx.$$

If the integrand behaves badly on (-infty,0) and well on (0,infty) or vice versa it is better to compute it as

$$\int_{-\infty}^{0} f(x)dx + \int_{0}^{\infty} f(x)dx.$$

This saves computing unnecessary function values in the semi-infinite range where the function is well behaved.

3.3. Multi-dimensional Integrals

A number of techniques are available in this area and the choice depends to a large extent on the dimension and the required accuracy. It can be advantageous to use more than one technique as a confirmation of accuracy particularly for high dimensional integrations. Many of the routines incorporate the transformation procedure REGION which allows general product regions to be easily dealt with in terms of conversion to the standard n-cube region.

(a) Products of one-dimensional rules (suitable for up to about 5 dimensions)

If $f(x_1,x_2,...,x_n)$ is known to be a sufficiently well-behaved function of each variable x_i, apart possibly from weight functions of the types provided, a product of Gaussian rules may be used. These are provided by D01BBF. In this case, the user should write the code for the evaluation of quadrature summation (6). Rules for finite, semi-infinite and infinite ranges are included.

The one-dimensional routines may also be used recursively. For example, the two-dimensional integral

$$I=\int^{b_1}\int^{b_2} f(x,y)dydx$$

$$a_1 \quad a_2$$

may be expressed as

$$I=\int_{a_1}^{b_1} F(x)dx,$$

where

$$F(x)=\int_{a_2}^{b_2} f(x,y)dy.$$

The user segment to evaluate F(x) will call the integration routine for the y-integration, which will call another user segment for f(x,y) as a function of y (x being effectively a constant). Note that, as Fortran is not a recursive language, a different library integration routine must be used for each dimension. Apart from this restriction, the full range of one-dimensional routines are available, for finite/infinite intervals, constant/variable limits, rule evaluation/automatic strategies etc.

(b) Automatic routines (D01GBF and D01FCF)

Both routines are for integrals of the form

$$\int_{a_1}^{b_1}\int_{a_2}^{b_2}\cdots\int_{a_n}^{b_n} f(x_1,x_2,\ldots,x_n)dx_n dx_{n-1}\ldots dx_1.$$

D01GBF is an adaptive Monte Carlo routine. This routine is usually slow and not recommended for high accuracy work. It is a robust routine that can often be used for low accuracy results with highly irregular integrands or when n is large.

D01FCF is an adaptive deterministic routine. Convergence is fast for well-behaved integrands. Highly accurate results can often be obtained for n between 2 and 5, using significantly fewer integrand evaluations than would be required by D01GBF. The routine will usually work when the integrand is mildly singular and for n<=10 should be used before D01GBF. If it is known in advance that the integrand is highly irregular, it is best to compare results from at least two different routines.

There are many problems for which one or both of the routines will require large amounts of computing time to obtain even moderately accurate results. The amount of computing time is controlled by the number of integrand evaluations allowed by the user, and users should set this parameter carefully, with reference to the time available and the accuracy desired.

3.4. Decision Trees

(i) One-dimensional integrals over a finite interval. (If in doubt, follow the downward branch.)

Please see figure in printed Reference Manual

(ii) One-dimensional integrals over a semi-infinite or infinite interval. (If in doubt, follow the downward branch.)

Please see figure in printed Reference Manual

D01 -- Quadrature Contents -- D01
Chapter D01

Quadrature

D01AJF 1-D quadrature, adaptive, finite interval, strategy due to Piessens and de Doncker, allowing for badly-behaved integrands

D01AKF 1-D quadrature, adaptive, finite interval, method suitable for oscillating functions

D01ALF 1-D quadrature, adaptive, finite interval, allowing for singularities at user-specified break-points

D01AMF 1-D quadrature, adaptive, infinite or semi-infinite interval

D01ANF 1-D quadrature, adaptive, finite interval, weight

```
              function cos((omega)x) or sin((omega)x)

      D01APF  1-D quadrature, adaptive, finite interval, weight
              function with end-point singularities of algebraico-
              logarithmic type

      D01AQF  1-D quadrature, adaptive, finite interval, weight
              function 1/(x-c), Cauchy principal value (Hilbert
              transform)

      D01ASF  1-D quadrature, adaptive, semi-infinite interval, weight
              function cos((omega)x) or sin((omega)x)

      D01BBF  Weights and abscissae for Gaussian quadrature rules

      D01FCF  Multi-dimensional adaptive quadrature over hyper-
              rectangle

      D01GAF  1-D quadrature, integration of function defined by data
              values, Gill-Miller method

      D01GBF  Multi-dimensional quadrature over hyper-rectangle, Monte
              Carlo method

\end{verbatim}
\endscroll
\end{page}
```

Approximation of the integral over a finite interval

— nagd.ht —

```
\begin{page}{manpageXXd01ajf}{NAG Documentation: d01ajf}
\beginscroll
\begin{verbatim}



     D01AJF(3NAG)      Foundation Library (12/10/92)      D01AJF(3NAG)



          D01 -- Quadrature                                        D01AJF
                  D01AJF -- NAG Foundation Library Routine Document
```

Note: Before using this routine, please read the Users' Note for your implementation to check implementation-dependent details. The symbol (*) after a NAG routine name denotes a routine that is not included in the Foundation Library.

1. Purpose

D01AJF is a general-purpose integrator which calculates an approximation to the integral of a function f(x) over a finite interval [a,b]:

```
       b
       /
   I= |f(x)dx.
       /
       a
```

2. Specification

```
       SUBROUTINE D01AJF (F, A, B, EPSABS, EPSREL, RESULT,
      1                   ABSERR, W, LW, IW, LIW, IFAIL)
       INTEGER          LW, IW(LIW), LIW, IFAIL
       DOUBLE PRECISION F, A, B, EPSABS, EPSREL, RESULT, ABSERR, W
      1                 (LW)
       EXTERNAL         F
```

3. Description

D01AJF is based upon the QUADPACK routine QAGS (Piessens et al [3]). It is an adaptive routine, using the Gauss 10-point and Kronrod 21-point rules. The algorithm, described by de Doncker [1], incorporates a global acceptance criterion (as defined by Malcolm and Simpson [2]) together with the (epsilon)-algorithm (Wynn [4]) to perform extrapolation. The local error estimation is described by Piessens et al [3].

The routine is suitable as a general purpose integrator, and can be used when the integrand has singularities, especially when these are of algebraic or logarithmic type.

D01AJF requires the user to supply a function to evaluate the integrand at a single point.

The routine D01ATF(*) uses an identical algorithm but requires the user to supply a subroutine to evaluate the integrand at an array of points. Therefore D01ATF(*) will be more efficient if the evaluation can be performed in vector mode on a vector-processing machine.

4. References

[1] De Doncker E (1978) An Adaptive Extrapolation Algorithm for Automatic Integration. Signum Newsletter. 13 (2) 12--18.

[2] Malcolm M A and Simpson R B (1976) Local Versus Global Strategies for Adaptive Quadrature. ACM Trans. Math. Softw. 1 129--146.

[3] Piessens R, De Doncker-Kapenga E, Uberhuber C and Kahaner D (1983) QUADPACK, A Subroutine Package for Automatic Integration. Springer-Verlag.

[4] Wynn P (1956) On a Device for Computing the $e_m(S_n)$ Transformation. Math. Tables Aids Comput. 10 91--96.

5. Parameters

1: F -- DOUBLE PRECISION FUNCTION, supplied by the user.
External Procedure
F must return the value of the integrand f at a given point.

Its specification is:

```
DOUBLE PRECISION FUNCTION F (X)
DOUBLE PRECISION X
```

1: X -- DOUBLE PRECISION Input
On entry: the point at which the integrand f must be evaluated.

F must be declared as EXTERNAL in the (sub)program from which D01AJF is called. Parameters denoted as Input must not be changed by this procedure.

2: A -- DOUBLE PRECISION Input
On entry: the lower limit of integration, a.

3: B -- DOUBLE PRECISION Input
On entry: the upper limit of integration, b. It is not necessary that a<b.

4: EPSABS -- DOUBLE PRECISION Input
On entry: the absolute accuracy required. If EPSABS is negative, the absolute value is used. See Section 7.

5: EPSREL -- DOUBLE PRECISION Input
On entry: the relative accuracy required. If EPSREL is negative, the absolute value is used. See Section 7.

6: RESULT -- DOUBLE PRECISION Output

```
    On exit: the approximation to the integral I.

 7: ABSERR -- DOUBLE PRECISION                          Output
    On exit: an estimate of the modulus of the absolute error,
    which should be an upper bound for |I-RESULT|.

 8: W(LW) -- DOUBLE PRECISION array                     Output
    On exit: details of the computation, as described in
    Section 8.

 9: LW -- INTEGER                                        Input
    On entry:
    the dimension of the array W as declared in the (sub)program
    from which D01AJF is called.
    The value of LW (together with that of LIW below) imposes a
    bound on the number of sub-intervals into which the interval
    of integration may be divided by the routine. The number of
    sub-intervals cannot exceed LW/4. The more difficult the
    integrand, the larger LW should be. Suggested value: a value
    in the range 800 to 2000 is adequate for most problems.
    Constraint: LW >= 4.

10: IW(LIW) -- INTEGER array                            Output
    On exit: IW(1) contains the actual number of sub-intervals
    used. The rest of the array is used as workspace.

11: LIW -- INTEGER                                       Input
    On entry:
    the dimension of the array IW as declared in the
    (sub)program from which D01AJF is called.
    The number of sub-intervals into which the interval of
    integration may be divided cannot exceed LIW. Suggested
    value: LIW = LW/4. Constraint: LIW >= 1.

12: IFAIL -- INTEGER                              Input/Output
    On entry: IFAIL must be set to 0, -1 or 1. Users who are
    unfamiliar with this parameter should refer to the Essential
    Introduction for details.

    On exit: IFAIL = 0 unless the routine detects an error or
    gives a warning (see Section 6).

    For this routine, because the values of output parameters
    may be useful even if IFAIL /=0 on exit, users are
    recommended to set IFAIL to -1 before entry. It is then
    essential to test the value of IFAIL on exit.

6. Error Indicators and Warnings

Errors or warnings specified by the routine:
```

```
If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

IFAIL= 1
     The maximum number of subdivisions allowed with the given
     workspace has been reached without the accuracy requirements
     being achieved. Look at the integrand in order to determine
     the integration difficulties. If the position of a local
     difficulty within the interval can be determined (e.g. a
     singularity of the integrand or its derivative, a peak, a
     discontinuity, etc) you will probably gain from splitting up
     the interval at this point and calling the integrator on the
     subranges. If necessary, another integrator, which is
     designed for handling the type of difficulty involved, must
     be used. Alternatively, consider relaxing the accuracy
     requirements specified by EPSABS and EPSREL, or increasing
     the amount of workspace.

IFAIL= 2
     Round-off error prevents the requested tolerance from being
     achieved. The error may be under-estimated. Consider
     requesting less accuracy.

IFAIL= 3
     Extremely bad local integrand behaviour causes a very strong
     subdivision around one (or more) points of the interval.

IFAIL= 4
     The requested tolerance cannot be achieved, because the
     extrapolation does not increase the accuracy satisfactorily;
     the returned result is the best which can be obtained. The
     same advice applies as in the case of IFAIL = 1.

IFAIL= 5
     The integral is probably divergent, or slowly convergent.
     Please note that divergence can occur with any non-zero
     value of IFAIL.

IFAIL= 6
     On entry LW < 4,

     or       LIW < 1.

7. Accuracy

The routine cannot guarantee, but in practice usually achieves,
the following accuracy:

                        |I-RESULT|<=tol,
```

where

$$tol=\max\{|EPSABS|,|EPSREL|*|I|\},$$

and EPSABS and EPSREL are user-specified absolute and relative error tolerance. Moreover it returns the quantity ABSERR which, in normal circumstances, satisfies

$$|I-RESULT|<=ABSERR<=tol.$$

8. Further Comments

The time taken by the routine depends on the integrand and the accuracy required.

If IFAIL /= 0 on exit, then the user may wish to examine the contents of the array W, which contains the end-points of the sub-intervals used by D01AJF along with the integral contributions and error estimates over the sub-intervals.

Specifically, for i=1,2,...,n, let r_i denote the approximation to the value of the integral over the sub-interval $[a_i,b_i]$ in the partition of [a,b] and e_i be the corresponding absolute error estimate.

Then, $\int_{a_i}^{b_i} f(x)dx \simeq r_i$ and $RESULT=\sum_{i=1}^{n} r_i$, unless D01AJF terminates while testing for divergence of the integral (see Piessens et al [3], Section 3.4.3). In this case, RESULT (and ABSERR) are taken to be the values returned from the extrapolation process. The value of n is returned in IW(1), and the values a_i, b_i, e_i and r_i are stored consecutively in the array W, that is:

$a_i=W(i)$,

$b_i=W(n+i)$,

$e_i=W(2n+i)$ and

```
               i

              r =W(3n+i).
               i

         9. Example

         To compute
                             2(pi)
                             /         x sin(30x)
                             |     ----------------dx.
                             /
                             0        /(  (  x  )2)
                                     / (1-(-----) )
                                   \/  (  (2(pi)) )

         The example program is not reproduced here. The source code for
         all example programs is distributed with the NAG Foundation
         Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

———-

Adaptive integration over a finite integral

— **nagd.ht** —

```
\begin{page}{manpageXXd01akf}{NAG Documentation: d01akf}
\beginscroll
\begin{verbatim}



     D01AKF(3NAG)      Foundation Library (12/10/92)      D01AKF(3NAG)



         D01 -- Quadrature                                        D01AKF
                 D01AKF -- NAG Foundation Library Routine Document

         Note: Before using this routine, please read the Users' Note for
         your implementation to check implementation-dependent details.
         The symbol (*) after a NAG routine name denotes a routine that is
         not included in the Foundation Library.
```

1. Purpose

D01AKF is an adaptive integrator, especially suited to oscillating, non-singular integrands, which calculates an approximation to the integral of a function f(x) over a finite interval [a,b]:

$$I = \int_a^b f(x)\,dx.$$

2. Specification

```
       SUBROUTINE D01AKF (F, A, B, EPSABS, EPSREL, RESULT,
      1                   ABSERR, W, LW, IW, LIW, IFAIL)
       INTEGER          LW, IW(LIW), LIW, IFAIL
       DOUBLE PRECISION F, A, B, EPSABS, EPSREL, RESULT, ABSERR, W
      1                 (LW)
       EXTERNAL         F
```

3. Description

D01AKF is based upon the QUADPACK routine QAG (Piessens et al [3]). It is an adaptive routine, using the Gauss 30-point and Kronrod 61-point rules. A 'global' acceptance criterion (as defined by Malcolm and Simpson [1]) is used. The local error estimation is described in Piessens et al [3].

Because this routine is based on integration rules of high order, it is especially suitable for non-singular oscillating integrands.

D01AKF requires the user to supply a function to evaluate the integrand at a single point.

The routine D01AUF(*) uses an identical algorithm but requires the user to supply a subroutine to evaluate the integrand at an array of points. Therefore D01AUF(*) will be more efficient if the evaluation can be performed in vector mode on a vector-processing machine.

D01AUF(*) also has an additional parameter KEY which allows the user to select from six different Gauss-Kronrod rules.

4. References

[1] Malcolm M A and Simpson R B (1976) Local Versus Global

```
     Strategies for Adaptive Quadrature. ACM Trans. Math. Softw.
     1 129--146.

[2]  Piessens R (1973) An Algorithm for Automatic Integration.
     Angewandte Informatik. 15 399--401.

[3]  Piessens R, De Doncker-Kapenga E, Uberhuber C and Kahaner D
     (1983) QUADPACK, A Subroutine Package for Automatic
     Integration. Springer-Verlag.
```


```
5. Parameters

 1:  F -- DOUBLE PRECISION FUNCTION, supplied by the user.
                                          External Procedure
    F must return the value of the integrand f at a given point.

    Its specification is:

           DOUBLE PRECISION FUNCTION F (X)
           DOUBLE PRECISION X

     1:  X -- DOUBLE PRECISION                             Input
         On entry: the point at which the integrand f must be
         evaluated.
    F must be declared as EXTERNAL in the (sub)program from
    which D01AKF is called. Parameters denoted as Input
    must not be changed by this procedure.

 2:  A -- DOUBLE PRECISION                                 Input
    On entry: the lower limit of integration, a.

 3:  B -- DOUBLE PRECISION                                 Input
    On entry: the upper limit of integration, b. It is not
    necessary that a<b.

 4:  EPSABS -- DOUBLE PRECISION                            Input
    On entry: the absolute accuracy required. If EPSABS is
    negative, the absolute value is used. See Section 7.

 5:  EPSREL -- DOUBLE PRECISION                            Input
    On entry: the relative accuracy required. If EPSREL is
    negative, the absolute value is used. See Section 7.

 6:  RESULT -- DOUBLE PRECISION                           Output
    On exit: the approximation to the integral I.

 7:  ABSERR -- DOUBLE PRECISION                           Output
    On exit: an estimate of the modulus of the absolute error,
    which should be an upper bound |I-RESULT|.
```

```
 8:  W(LW) -- DOUBLE PRECISION array                       Output
     On exit: details of the computation, as described in
     Section 8.

 9:  LW -- INTEGER                                          Input
     On entry: the dimension of W, as declared in the (sub)
     program from which D01AKF is called. The value of LW
     (together with that of LIW below) imposes a bound on the
     number of sub-intervals into which the interval of
     integration may be divided by the routine. The number of
     sub-intervals cannot exceed LW/4. The more difficult the
     integrand, the larger LW should be. Suggested value: a value
     in the range 800 to 2000 is adequate for most problems.
     Constraint: LW >= 4. See IW below.

10:  IW(LIW) -- INTEGER array                               Output
     On exit: IW(1) contains the actual number of sub-intervals
     used. The rest of the array is used as workspace.

11:  LIW -- INTEGER                                         Input
     On entry:
     the dimension of the array IW as declared in the
     (sub)program from which D01AKF is called.
     The number of sub-intervals into which the interval of
     integration may be divided cannot exceed LIW. Suggested
     value: LIW = LW/4. Constraint: LIW >= 1.

12:  IFAIL -- INTEGER                                Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. Users who are
     unfamiliar with this parameter should refer to the Essential
     Introduction for details.

     On exit: IFAIL = 0 unless the routine detects an error or
     gives a warning (see Section 6).

     For this routine, because the values of output parameters
     may be useful even if IFAIL /=0 on exit, users are
     recommended to set IFAIL to -1 before entry. It is then
     essential to test the value of IFAIL on exit.

6. Error Indicators and Warnings

Errors or warnings specified by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

IFAIL= 1
     The maximum number of subdivisions allowed with the given
     workspace has been reached without the accuracy requirements
```

being achieved. Look at the integrand in order to determine the integration difficulties. Probably another integrator which is designed for handling the type of difficulty involved must be used. Alternatively, consider relaxing the accuracy requirements specified by EPSABS and EPSREL, or increasing the amount of workspace.

IFAIL= 2

Round-off error prevents the requested tolerance from being achieved. Consider requesting less accuracy.

IFAIL= 3

Extremely bad local integrand behaviour causes a very strong subdivision around one (or more) points of the interval. The same advice applies as in the case of IFAIL = 1.

IFAIL= 4

On entry LW < 4,

or LIW < 1.

7. Accuracy

The routine cannot guarantee, but in practice usually achieves, the following accuracy:

$$|I-RESULT|<=tol,$$

where

$$tol=max\{|EPSABS|,|EPSREL|*|I|\},$$

and EPSABS and EPSREL are user-specified absolute and relative error tolerances. Moreover it returns the quantity ABSERR which, in normal circumstances satisfies

$$|I-RESULT|<=ABSERR<=tol.$$

8. Further Comments

The time taken by the routine depends on the integrand and the accuracy required.

If IFAIL /= 0 on exit, then the user may wish to examine the contents of the array W, which contains the end-points of the sub-intervals used by D01AKF along with the integral contributions and error estimates over these sub-intervals.

Specifically, for i=1,2,...,n, let r_i denote the approximation to

```
         the value of the integral over the sub-interval [a ,b ] in the
                                                           i  i
         partition of [a,b] and e  be the corresponding absolute error
                                 i
                          b
                           i                       n
                          /                        --
         estimate. Then,  | f(x)dx~=r  and RESULT= >  r . The value of n
                          /          i             --  i
                          a                        i=1
                           i
         is returned in IW(1), and the values a , b , e  and r  are stored
                                               i   i   i      i
         consecutively in the array W, that is:

              a =W(i),
               i

              b =W(n+i),
               i

              e =W(2n+i) and
               i

              r =W(3n+i).
               i

         9. Example

         To compute

                                2(pi)
                                /
                                |    x sin(30x) cosx dx.
                                /
                                0

         The example program is not reproduced here. The source code for
         all example programs is distributed with the NAG Foundation
         Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Approximate integration with local singular points

— nagd.ht —

```
\begin{page}{manpageXXd01alf}{NAG Documentation: d01alf}
\beginscroll
\begin{verbatim}



     D01ALF(3NAG)      Foundation Library (12/10/92)      D01ALF(3NAG)



          D01 -- Quadrature                                       D01ALF
                  D01ALF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          D01ALF is a general purpose integrator which calculates an
          approximation to the integral of a function f(x) over a finite
          interval [a,b]:

                                         b
                                         /
                                      I= |f(x)dx
                                         /
                                         a

          where the integrand may have local singular behaviour at a finite
          number of points within the integration interval.

          2. Specification

                 SUBROUTINE D01ALF (F, A, B, NPTS, POINTS, EPSABS, EPSREL,
                1                   RESULT, ABSERR, W, LW, IW, LIW, IFAIL)
                 INTEGER          NPTS, LW, IW(LIW), LIW, IFAIL
                 DOUBLE PRECISION F, A, B, POINTS(*), EPSABS, EPSREL,
                1                 RESULT, ABSERR, W(LW)
                 EXTERNAL         F

          3. Description

          D01ALF is based upon the QUADPACK routine QAGP (Piessens et al
```

[3]). It is very similar to D01AJF, but allows the user to supply difficult. It is an adaptive routine, using the Gauss 10-point and Kronrod 21-point rules. The algorithm described by de Doncker [1], incorporates a global acceptance criterion (as defined by Malcolm and Simpson [2]) together with the (epsilon)-algorithm (Wynn [4]) to perform extrapolation. The user-supplied 'break-points'always occur as the end-points of some sub-interval during the adaptive process. The local error estimation is described by Piessens et al [3].

4. References

[1] De Doncker E (1978) An Adaptive Extrapolation Algorithm for Automatic Integration. Signum Newsletter. 13 (2) 12--18.

[2] Malcolm M A and Simpson R B (1976) Local Versus Global Strategies for Adaptive Quadrature. ACM Trans. Math. Softw. 1 129--146.

[3] Piessens R, De Doncker-Kapenga E, Uberhuber C and Kahaner D (1983) QUADPACK, A Subroutine Package for Automatic Integration. Springer-Verlag.

[4] Wynn P (1956) On a Device for Computing the $e_m(S_n)$ Transformation. Math. Tables Aids Comput. 10 91--96.

5. Parameters

1: F -- DOUBLE PRECISION FUNCTION, supplied by the user. External Procedure

F must return the value of the integrand f at a given point.

Its specification is:

```
DOUBLE PRECISION FUNCTION F (X)
DOUBLE PRECISION X
```

1: X -- DOUBLE PRECISION Input

On entry: the point at which the integrand f must be evaluated.

F must be declared as EXTERNAL in the (sub)program from which D01ALF is called. Parameters denoted as Input must not be changed by this procedure.

2: A -- DOUBLE PRECISION Input

On entry: the lower limit of integration, a.

3: B -- DOUBLE PRECISION Input

On entry: the upper limit of integration, b. It is not

```
    necessary that a<b.

 4: NPTS -- INTEGER                                        Input
    On entry: the number of user-supplied break-points within
    the integration interval. Constraint: NPTS >= 0.

 5: POINTS(NPTS) -- DOUBLE PRECISION array                 Input
    On entry: the user-specified break-points. Constraint: the
    break-points must all lie within the interval of integration
    (but may be supplied in any order).

 6: EPSABS -- DOUBLE PRECISION                             Input
    On entry: the absolute accuracy required. If EPSABS is
    negative, the absolute value is used. See Section 7.

 7: EPSREL -- DOUBLE PRECISION                             Input
    On entry: the relative accuracy required. If EPSREL is
    negative, the absolute value is used. See Section 7.

 8: RESULT -- DOUBLE PRECISION                             Input
    On entry: the approximation to the integral I.

 9: ABSERR -- DOUBLE PRECISION                            Output
    On exit: an estimate of the modulus of the absolute error,
    which should be an upper bound for |I-RESULT|.

10: W(LW) -- DOUBLE PRECISION array                       Output
    On exit: details of the computation, as described in
    Section 8.

11: LW -- INTEGER                                          Input
    On entry:
    the dimension of the array W as declared in the (sub)program
    from which D01ALF is called.
    The value of LW (together with that of LIW below) imposes a
    bound on the number of sub-intervals into which the interval
    of integration may be divided by the routine. The number of
    sub-intervals cannot exceed (LW-2*NPTS-4)/4. The more
    difficult the integrand, the larger LW should be. Suggested
    value: a value in the range 800 to 2000 is adequate for most
    problems. Constraint: LW>=2*NPTS+8.

12: IW(LIW) -- INTEGER array                              Output
    On exit: IW(1) contains the actual number of sub-intervals
    used. The rest of the array is used as workspace.

13: LIW -- INTEGER                                         Input
    On entry:
    the dimension of the array IW as declared in the
    (sub)program from which D01ALF is called.
```

```
     The number of sub-intervals into which the interval of
     integration may be divided cannot exceed (LIW-NPTS-2)/2.
     Suggested value: LIW = LW/2. Constraint: LIW >= NPTS + 4.

14:  IFAIL -- INTEGER                                Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. Users who are
     unfamiliar with this parameter should refer to the Essential
     Introduction for details.

     On exit: IFAIL = 0 unless the routine detects an error or
     gives a warning (see Section 6).

     For this routine, because the values of output parameters
     may be useful even if IFAIL /=0 on exit, users are
     recommended to set IFAIL to -1 before entry. It is then
     essential to test the value of IFAIL on exit.

6. Error Indicators and Warnings

Errors or warnings specified by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

IFAIL= 1
     The maximum number of subdivisions allowed with the given
     workspace has been reached, without the accuracy
     requirements being achieved. Look at the integrand in order
     to determine the integration difficulties. If the position
     of a local difficulty within the interval can be determined
     (e.g. a singularity of the integrand or its derivative, a
     peak, a discontinuity, etc) it should be supplied to the
     routine as an element of the vector POINTS. If necessary,
     another integrator should be used, which is designed for
     handling the type of difficulty involved. Alternatively,
     consider relaxing the accuracy requirements specified by
     EPSABS and EPSREL, or increasing the amount of workspace.

IFAIL= 2
     Round-off error prevents the requested tolerance from being
     achieved. The error may be under-estimated. Consider
     requesting less accuracy.

IFAIL= 3
     Extremely bad local integrand behaviour causes a very strong
     subdivision around one (or more) points of the interval.
     The same advice applies as in the case of IFAIL = 1.

IFAIL= 4
     The requested tolerance cannot be achieved, because the
```

extrapolation does not increase the accuracy satisfactorily; the result returned is the best which can be obtained. The same advice applies as in the case IFAIL = 1.

IFAIL= 5
The integral is probably divergent, or slowly convergent. Please note that divergence can also occur with any other non-zero value of IFAIL.

IFAIL= 6
The input is invalid: break-points are specified outside the integration range, NPTS > LIMIT or NPTS < 0. RESULT and ABSERR are set to zero.

IFAIL= 7
On entry LW<2*NPTS+8,

or LIW<NPTS+4.

7. Accuracy

The routine cannot guarantee, but in practice usually achieves, the following accuracy:

|I-RESULT|<=tol,

where

tol=max{|EPSABS|,|EPSREL|*|I|},

and EPSABS and EPSREL are user-specified absolute and relative error tolerances. Moreover it returns the quantity ABSERR which, in normal circumstances, satisfies

|I-RESULT|<=ABSERR<=tol.

8. Further Comments

The time taken by the routine depends on the integrand and on the accuracy required.

If IFAIL /= 0 on exit, then the user may wish to examine the contents of the array W, which contains the end-points of the sub-intervals used by D01ALF along with the integral contributions and error estimates over these sub-intervals.

Specifically, for i=1,2,...,n, let r_i denote the approximation to the value of the integral over the sub-interval $[a_i,b_i]$ in the partition of [a,b] and e_i be the corresponding absolute error

```
                                  i
                           b
                            i                       n
                           /                        --
          estimate. Then,  | f(x)dx~=r  and RESULT= >  r  unless D01ALF
                           /          i             --  i
                           a                        i=1
                            i
          terminates while testing for divergence of the integral (see
          Piessens et al [3] Section 3.4.3). In this case, RESULT (and
          ABSERR) are taken to be the values returned from the
          extrapolation process. The value of n is returned in IW(1), and
          the values a , b , e  and r  are stored consecutively in the
                      i   i   i      i
          array W, that is:

               a =W(i),
                i

               b =W(n+i),
                i

               e =W(2n+i) and
                i

               r =W(3n+i).
                i

          9. Example

          To compute

                                    1
                                    /     1
                                    | ---------dx.
                                    /
                                    0 \/|x-1/7|

          A break-point is specified at x=1/7, at which point the integrand
          is infinite. (For definiteness the function FST returns the value
          0.0 at this point.)

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

———-

Approximate integration over a (semi-)infinite interval

— nagd.ht —

```
\begin{page}{manpageXXd01amf}{NAG Documentation: d01amf}
\beginscroll
\begin{verbatim}



     D01AMF(3NAG)      Foundation Library (12/10/92)      D01AMF(3NAG)



          D01 -- Quadrature                                          D01AMF
                  D01AMF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          D01AMF calculates an approximation to the integral of a function
          f(x) over an infinite or semi-infinite interval [a,b]:

                                        b
                                        /
                                     I= |f(x)dx
                                        /
                                        a

          2. Specification

                 SUBROUTINE D01AMF (F, BOUND, INF, EPSABS, EPSREL, RESULT,
                1                   ABSERR, W, LW, IW, LIW, IFAIL)
                 INTEGER          INF, LW, IW(LIW), LIW, IFAIL
                 DOUBLE PRECISION F, BOUND, EPSABS, EPSREL, RESULT, ABSERR,
                1                 W(LW)
                 EXTERNAL         F

          3. Description

          D01AMF is based on the QUADPACK routine QAGI (Piessens et al [3])
          [0,1] using one of the identities:
```

$$\int_{-\infty}^{a} f(x)dx = \int_{0}^{1} f\left(a-\frac{1-t}{t}\right)\frac{1}{t^2}dt$$

$$\int_{a}^{\infty} f(x)dx = \int_{0}^{1} f\left(a+\frac{1-t}{t}\right)\frac{1}{t^2}dt$$

$$\int_{-\infty}^{\infty} f(x)dx = \int_{0}^{\infty} (f(x)+f(-x))dx = \int_{0}^{1}\left[f\left(\frac{1-t}{t}\right)+f\left(\frac{-1+t}{t}\right)\right]\frac{1}{t^2}dt$$

where a represents a finite integration limit. An adaptive procedure, based on the Gauss seven-point and Kronrod 15-point rules, is then employed on the transformed integral. The algorithm, described by de Doncker [1], incorporates a global acceptance criterion (as defined by Malcolm and Simpson [2]) together with the (epsilon)-algorithm (Wynn [4]) to perform extrapolation. The local error estimation is described by Piessens et al [3].

4. References

[1] De Doncker E (1978) An Adaptive Extrapolation Algorithm for Automatic Integration. Signum Newsletter. 13 (2) 12--18.

[2] Malcolm M A and Simpson R B (1976) Local Versus Global Strategies for Adaptive Quadrature. ACM Trans. Math. Softw. 1 129--146.

[3] Piessens R, De Doncker-Kapenga E, Uberhuber C and Kahaner D (1983) QUADPACK, A Subroutine Package for Automatic Integration. Springer-Verlag.

[4] Wynn P (1956) On a Device for Computing the $e_m(S_n)$ Transformation. Math. Tables Aids Comput. 10 91--96.

5. Parameters

1: F -- DOUBLE PRECISION FUNCTION, supplied by the user.
External Procedure

On entry: the point at which the integrand f must be

```
evaluated.

Its specification is:

       DOUBLE PRECISION FUNCTION F (X)
       DOUBLE PRECISION X

 1:  X -- DOUBLE PRECISION                             Input
     On entry: the point at which the integrand f must be
     evaluated.
F must be declared as EXTERNAL in the (sub)program from
which D01AMF is called. Parameters denoted as Input
must not be changed by this procedure.

2:  BOUND -- DOUBLE PRECISION                          Input
    On entry: the finite limit of the integration range (if
    present). BOUND is not used if the interval is doubly
    infinite.

3:  INF -- INTEGER                                     Input
    On entry: indicates the kind of integration range:
         if INF =1, the range is [BOUND, +infty)

         if INF =-1, the range is (-infty, BOUND]

         if INF =+2, the range is (-infty, +infty).
     Constraint: INF =-1, 1 or 2.

4:  EPSABS -- DOUBLE PRECISION                         Input
    On entry: the absolute accuracy required. If EPSABS is
    negative, the absolute value is used. See Section 7.

5:  EPSREL -- DOUBLE PRECISION                         Input
    On entry: the relative accuracy required. If EPSREL is
    negative, the absolute value is used. See Section 7.

6:  RESULT -- DOUBLE PRECISION                        Output
    On exit: the approximation to the integral I.

7:  ABSERR -- DOUBLE PRECISION                        Output
    On exit: an estimate of the modulus of the absolute error,
    which should be an upper bound for |I-RESULT|.

8:  W(LW) -- DOUBLE PRECISION array                   Output
    On exit: details of the computation, as described in
    Section 8.

9:  LW -- INTEGER                                      Input
    On entry:
    the dimension of the array W as declared in the (sub)program
```

```
     from which D01AMF is called.
     The value of LW (together with that of LIW below) imposes a
     bound on the number of sub-intervals into which the interval
     of integration may be divided by the routine. The number of
     sub-intervals cannot exceed LW/4. The more difficult the
     integrand, the larger LW should be. Suggested value: a value
     in the range 800 to 2000 is adequate for most problems.
     Constraint: LW >= 4.

10:  IW(LIW) -- INTEGER array                              Output
     On exit: IW(1) contains the actual number of sub-intervals
     used. The rest of the array is used as workspace.

11:  LIW -- INTEGER                                         Input
     On entry:
     the dimension of the array IW as declared in the
     (sub)program from which D01AMF is called.
     The number of sub-intervals into which the interval of
     integration may be divided cannot exceed LIW. Suggested
     value: LIW = LW/4. Constraint: LIW >= 1.

12:  IFAIL -- INTEGER                                Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. Users who are
     unfamiliar with this parameter should refer to the Essential
     Introduction for details.
     On exit: IFAIL = 0 unless the routine detects an error or
     gives a warning (see Section 6).

     For this routine, because the values of output parameters
     may be useful even if IFAIL /=0 on exit, users are
     recommended to set IFAIL to -1 before entry. It is then
     essential to test the value of IFAIL on exit.

6. Error Indicators and Warnings

Errors or warnings specified by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

IFAIL= 1
     The maximum number of subdivisions allowed with the given
     workspace has been reached without the requested accuracy
     requirements being achieved. Look at the integrand in order
     to determine the integration difficulties. If the position
     of a local difficulty within the interval can be determined
     (e.g. a singularity of the integrand or its derivative, a
     peak, a discontinuity, etc) you will probably gain from
     splitting up the interval at this point and calling D01AMF
     on the infinite subrange and an appropriate integrator on
```

the finite subrange. Alternatively, consider relaxing the accuracy requirements specified by EPSABS and EPSREL, or increasing the amount of workspace.

IFAIL= 2

Round-off error prevents the requested tolerance from being achieved. The error may be underestimated. Consider requesting less accuracy.

IFAIL= 3

Extremely bad local integrand behaviour causes a very strong subdivision around one (or more) points of the interval. The same advice applies as in the case of IFAIL = 1.

IFAIL= 4

The requested tolerance cannot be achieved, because the extrapolation does not increase the accuracy satisfactorily; the returned result is the best which can be obtained. The same advice applies as in the case of IFAIL = 1.

IFAIL= 5

The integral is probably divergent, or slowly convergent. It must be noted that divergence can also occur with any other non-zero value of IFAIL.

IFAIL= 6

On entry LW < 4,

or LIW < 1,

or INF /= -1, 1 or 2.

Please note that divergence can occur with any non-zero value of IFAIL.

7. Accuracy

The routine cannot guarantee, but in practice usually achieves, the following accuracy:

|I-RESULT|<=tol,

where

tol=max{|EPSABS|,|EPSREL|*|I|},

and EPSABS and EPSREL are user-specified absolute and relative error tolerances. Moreover it returns the quantity ABSERR, which, in normal circumstances, satisfies

|I-RESULT|<=ABSERR<=tol.

8. Further Comments

The time taken by the routine depends on the integrand and the accuracy required.

If IFAIL /= 0 on exit, then the user may wish to examine the contents of the array W, which contains the end-points of the sub-intervals used by D01AMF along with the integral contributions and error estimates over these sub-intervals.

Specifically, for i=1,2,...,n, let r_i denote the approximation to the value of the integral over the sub-interval $[a_i,b_i]$ in the partition of [a,b] and e_i be the corresponding absolute error estimate. Then, $\int_{a_i}^{b_i} f(x)dx \simeq r_i$ and RESULT$=\sum_{i=1}^{n} r_i$ unless D01AMF terminates while testing for divergence of the integral (see Piessens et al [3] Section 3.4.3). In this case, RESULT (and ABSERR) are taken to be the values returned from the extrapolation process. The value of n is returned in IW(1), and the values a_i, b_i, e_i and r_i are stored consecutively in the array W, that is:

$$a_i=W(i), b_i=W(n+i), e_i=W(2n+i) \text{ and } r_i=W(3n+i).$$

Note: that this information applies to the integral transformed to (0,1) as described in Section 3, not to the original integral.

9. Example

To compute

$$\int_0^{\infty} \frac{1}{(x+1)\sqrt{x}} dx.$$

The exact answer is (pi).

```
         The example program is not reproduced here. The source code for
         all example programs is distributed with the NAG Foundation
         Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Approximate sine or cosine transform over finite interval

— nagd.ht —

```
\begin{page}{manpageXXd01anf}{NAG Documentation: d01anf}
\beginscroll
\begin{verbatim}



     D01ANF(3NAG)      Foundation Library (12/10/92)      D01ANF(3NAG)



          D01 -- Quadrature                                       D01ANF
                  D01ANF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          D01ANF calculates an approximation to the sine or the cosine
          transform of a function g over [a,b]:

             b                                  b
             /                                  /
          I= |g(x)||,sin((omega)x)||,dx  or  I= |g(x)||,cos((omega)x)||,dx
             /                                  /
             a                                  a

          (for a user-specified value of (omega)).

          2. Specification
```

```
      SUBROUTINE D01ANF (G, A, B, OMEGA, KEY, EPSABS, EPSREL,
     1                   RESULT, ABSERR, W, LW, IW, LIW, IFAIL)
      INTEGER          KEY, LW, IW(LIW), LIW, IFAIL
      DOUBLE PRECISION G, A, B, OMEGA, EPSABS, EPSREL, RESULT,
     1                 ABSERR, W(LW)
      EXTERNAL         G
```

3. Description

D01ANF is based upon the QUADPACK routine QFOUR (Piessens et al [3]). It is an adaptive routine, designed to integrate a function of the form g(x)w(x), where w(x) is either sin((omega)x) or cos((omega)x). If a sub-interval has length

$$L=|b-a|2^{-l}$$

then the integration over this sub-interval is performed by means of a modified Clenshaw-Curtis procedure (Piessens and Branders [2]) if L(omega)>4 and l<=20. In this case a Chebyshev-series approximation of degree 24 is used to approximate g(x), while an error estimate is computed from this approximation together with that obtained using Chebyshev-series of degree 12. If the above conditions do not hold then Gauss 7-point and Kronrod 15-point rules are used. The algorithm, described in [3], incorporates a global acceptance criterion (as defined in Malcolm and Simpson [1]) together with the (epsilon)-algorithm Wynn [4] to perform extrapolation. The local error estimation is described in [3].

4. References

[1] Malcolm M A and Simpson R B (1976) Local Versus Global Strategies for Adaptive Quadrature. ACM Trans. Math. Softw. 1 129--146.

[2] Piessens R and Branders M (1975) Algorithm 002. Computation of Oscillating Integrals. J. Comput. Appl. Math. 1 153--164.

[3] Piessens R, De Doncker-Kapenga E, Uberhuber C and Kahaner D (1983) QUADPACK, A Subroutine Package for Automatic Integration. Springer-Verlag.

[4] Wynn P (1956) On a Device for Computing the $e_m(S_n)$ Transformation. Math. Tables Aids Comput. 10 91--96.

5. Parameters

1: G -- DOUBLE PRECISION FUNCTION, supplied by the user.
External Procedure

```
G must return the value of the function g at a given point.

Its specification is:

       DOUBLE PRECISION FUNCTION G (X)
       DOUBLE PRECISION X

 1:  X -- DOUBLE PRECISION                             Input
     On entry: the point at which the function g must be
     evaluated.
G must be declared as EXTERNAL in the (sub)program from
which D01ANF is called. Parameters denoted as Input
must not be changed by this procedure.
```

```
 2:  A -- DOUBLE PRECISION                                  Input
     On entry: the lower limit of integration, a.

 3:  B -- DOUBLE PRECISION                                  Input
     On entry: the upper limit of integration, b. It is not
     necessary that a<b.

 4:  OMEGA -- DOUBLE PRECISION                              Input
     On entry: the parameter (omega) in the weight function of
     the transform.

 5:  KEY -- INTEGER                                         Input
     On entry: indicates which integral is to be computed:
          if KEY = 1, w(x)=cos((omega)x);

          if KEY = 2, w(x)=sin((omega)x).
      Constraint: KEY = 1 or 2.

 6:  EPSABS -- DOUBLE PRECISION                             Input
     On entry: the absolute accuracy required. If EPSABS is
     negative, the absolute value is used. See Section 7.

 7:  EPSREL -- DOUBLE PRECISION                             Input
     On entry: the relative accuracy required. If EPSREL is
     negative, the absolute value is used. See Section 7.

 8:  RESULT -- DOUBLE PRECISION                            Output
     On exit: the approximation to the integral I.

 9:  ABSERR -- DOUBLE PRECISION                            Output
     On exit: an estimate of the modulus of the absolute error,
     which should be an upper bound for |I-RESULT|.

10:  W(LW) -- DOUBLE PRECISION array                       Output
     On exit: details of the computation, as described in
     Section 8.
```

```
11:  LW -- INTEGER                                         Input
     On entry:
     the dimension of the array W as declared in the (sub)program
     from which D01ANF is called.
     The value of LW (together with that of LIW below) imposes a
     bound on the number of sub-intervals into which the interval
     of integration may be divided by the routine. The number of
     sub-intervals cannot exceed LW/4. The more difficult the
     integrand, the larger LW should be. Suggested value: a value
     in the range 800 to 2000 is adequate for most problems.
     Constraint: LW >= 4.

12:  IW(LIW) -- INTEGER array                             Output
     On exit: IW(1) contains the actual number of sub-intervals
     used. The rest of the array is used as workspace.

13:  LIW -- INTEGER                                        Input
     On entry:
     the dimension of the array IW as declared in the
     (sub)program from which D01ANF is called.
     The number of sub-intervals into which the interval of
     integration may be divided cannot exceed LIW/2. Suggested
     value: LIW = LW/2. Constraint: LIW >= 2.

14:  IFAIL -- INTEGER                               Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. Users who are
     unfamiliar with this parameter should refer to the Essential
     Introduction for details.

     On exit: IFAIL = 0 unless the routine detects an error or
     gives a warning (see Section 6).

     For this routine, because the values of output parameters
     may be useful even if IFAIL /=0 on exit, users are
     recommended to set IFAIL to -1 before entry. It is then
     essential to test the value of IFAIL on exit.

6. Error Indicators and Warnings

Errors or warnings specified by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

IFAIL= 1
     The maximum number of subdivisions allowed with the given
     workspace has been reached without the accuracy requested
     being achieved. Look at the integrand in order to determine
     the integration difficulties. If the position of a local
```

```
     difficulty within the interval can be determined (e.g. a
     singularity of the integrand or its derivative, a peak, a
     discontinuity, etc) you will probably gain from splitting up
     the interval at this point and calling the integrator on the
     subranges. If necessary, another integrator, which is
     designed for handling the type of difficulty involved, must
     be used. Alternatively consider relaxing the accuracy
     requirements specified by EPSABS and EPSREL, or increasing
     amount of workspace.

IFAIL= 2
     Round-off error prevents the requested tolerance from being
     achieved. The error may be underestimated. Consider
     requesting less accuracy.

IFAIL= 3
     Extremely bad local behaviour of g(x) causes a very strong
     subdivision around one (or more) points of the interval.
     The same advice applies as in the case of IFAIL = 1.

IFAIL= 4
     The requested tolerance cannot be achieved because the
     extrapolation does not increase the accuracy satisfactorily;
     the returned result is the best which can be obtained.  The
     same advice applies as in the case of IFAIL = 1.

IFAIL= 5
     The integral is probably divergent, or slowly convergent. It
     must be noted that divergence can occur with any non-zero
     value of IFAIL.

IFAIL= 6
     On entry KEY < 1,

     or       KEY > 2.

IFAIL= 7
     On entry LW < 4,

     or       LIW < 2.

7. Accuracy

The routine cannot guarantee, but in practice usually achieves,
the following accuracy:

                        |I-RESULT|<=tol,

where
```

$$tol=\max\{|EPSABS|,|EPSREL|*|I|\},$$

and EPSABS and EPSREL are user-specified absolute and relative tolerances. Moreover it returns the quantity ABSERR, which, in normal circumstances, satisfies

$$|I-RESULT|<=ABSERR<=tol.$$

8. Further Comments

The time taken by the routine depends on the integrand and on the accuracy required.

If IFAIL /= 0 on exit, then the user may wish to examine the contents of the array W, which contains the end-points of the sub-intervals used by D01ANF along with the integral contributions and error estimates over these sub-intervals.

Specifically, for i=1,2,...,n, let r_i denote the approximation to the value of the integral over the sub-interval $[a_i,b_i]$ in the partition of [a,b] and e_i be the corresponding absolute error estimate. Then, $\int_{a_i}^{b_i} g(x)w(x)dx \simeq r_i$ and $RESULT=\sum_{i=1}^{n} r_i$ unless D01ANF terminates while testing for divergence of the integral (see Piessens et al [3] Section 3.4.3). In this case, RESULT (and ABSERR) are taken to be the values returned from the extrapolation process. The value of n is returned in IW(1), and the values a_i, b_i, e_i and r_i are stored consecutively in the array W, that is:

a_i=W(i),

b_i=W(n+i),

e_i=W(2n+i) and

r_i=W(3n+i).

```
         9. Example

         To compute

                                   1
                                   /
                                   |ln(x)sin(10(pi)x)dx.
                                   /
                                   0

         The example program is not reproduced here. The source code for
         all example programs is distributed with the NAG Foundation
         Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Adaptive integration of weighted function over an interval

— nagd.ht —

```
\begin{page}{manpageXXd01apf}{NAG Documentation: d01apf}
\beginscroll
\begin{verbatim}



     D01APF(3NAG)      Foundation Library (12/10/92)      D01APF(3NAG)



         D01 -- Quadrature                                          D01APF
                 D01APF -- NAG Foundation Library Routine Document

         Note: Before using this routine, please read the Users' Note for
         your implementation to check implementation-dependent details.
         The symbol (*) after a NAG routine name denotes a routine that is
         not included in the Foundation Library.

         1. Purpose

         D01APF is an adaptive integrator which calculates an
         approximation to the integral of a function g(x)w(x) over a
```

finite interval [a,b]:

```
    b
    /
I= |g(x)w(x)dx
    /
    a
```

where the weight function w has end-point singularities of algebraico-logarithmic type.

2. Specification

```
       SUBROUTINE D01APF (G, A, B, ALFA, BETA, KEY, EPSABS,
      1                   EPSREL, RESULT, ABSERR, W, LW, IW, LIW,
      2                   IFAIL)
       INTEGER          KEY, LW, IW(LIW), LIW, IFAIL
       DOUBLE PRECISION G, A, B, ALFA, BETA, EPSABS, EPSREL,
      1                 RESULT, ABSERR, W(LW)
       EXTERNAL         G
```

3. Description

D01APF is based upon the QUADPACK routine QAWSE (Piessens et al [3]) and integrates a function of the form g(x)w(x), where the weight function w(x) may have algebraico-logarithmic singularities at the end-points a and/or b. The strategy is a modification of that in D01AKF. We start by bisecting the original interval and applying modified Clenshaw-Curtis integration of orders 12 and 24 to both halves. Clenshaw-Curtis integration is then used on all sub-intervals which have a or b as one of their end-points (Piessens et al [2]). On the other sub-intervals Gauss-Kronrod (7-15 point) integration is carried out.

A 'global' acceptance criterion (as defined by Malcolm and Simpson [1]) is used. The local error estimation control is described by Piessens et al [3].

4. References

[1] Malcolm M A and Simpson R B (1976) Local Versus Global Strategies for Adaptive Quadrature. ACM Trans. Math. Softw. 1 129--146.

[2] Piessens R, Mertens I and Branders M (1974) Integration of Functions having End-point Singularities. Angewandte Informatik. 16 65--68.

[3] Piessens R, De Doncker-Kapenga E, Uberhuber C and Kahaner D

(1983) QUADPACK, A Subroutine Package for Automatic
Integration. Springer-Verlag.

5. Parameters

```
 1:  G -- DOUBLE PRECISION FUNCTION, supplied by the user.
                                              External Procedure
     G must return the value of the function g at a given point
     X.

     Its specification is:

            DOUBLE PRECISION FUNCTION G (X)
            DOUBLE PRECISION X

      1:  X -- DOUBLE PRECISION                             Input
          On entry: the point at which the function g must be
          evaluated.
     G must be declared as EXTERNAL in the (sub)program from
     which D01APF is called. Parameters denoted as Input
     must not be changed by this procedure.

 2:  A -- DOUBLE PRECISION                                  Input
     On entry: the lower limit of integration, a.

 3:  B -- DOUBLE PRECISION                                  Input
     On entry: the upper limit of integration, b. Constraint: B
     > A.

 4:  ALFA -- DOUBLE PRECISION                               Input
     On entry: the parameter (alpha) in the weight function.
     Constraint: ALFA>-1.

 5:  BETA -- DOUBLE PRECISION                               Input
     On entry: the parameter (beta) in the weight function.
     Constraint: BETA>-1.

 6:  KEY -- INTEGER                                         Input
     On entry: indicates which weight function is to be used:
                              (alpha)     (beta)
          if KEY = 1, w(x)=(x-a)       (b-x)

                              (alpha)     (beta)
          if KEY = 2, w(x)=(x-a)       (b-x)      ln(x-a)

                              (alpha)     (beta)
          if KEY = 3, w(x)=(x-a)       (b-x)      ln(b-x)

                              (alpha)     (beta)
          if KEY = 4, w(x)=(x-a)       (b-x)     ln(x-a)ln(b-x)
```

Constraint: KEY = 1, 2, 3 or 4

7: EPSABS -- DOUBLE PRECISION Input
On entry: the absolute accuracy required. If EPSABS is negative, the absolute value is used. See Section 7.

8: EPSREL -- DOUBLE PRECISION Input
On entry: the relative accuracy required. If EPSREL is negative, the absolute value is used. See Section 7.

9: RESULT -- DOUBLE PRECISION Output
On exit: the approximation to the integral I.

10: ABSERR -- DOUBLE PRECISION Output
On exit: an estimate of the modulus of the absolute error, which should be an upper bound for |I-RESULT|.

11: W(LW) -- DOUBLE PRECISION array Output
On exit: details of the computation, as described in Section 8.

12: LW -- INTEGER Input
On entry:
the dimension of the array W as declared in the (sub)program from which D01APF is called.
The value of LW (together with that of LIW below) imposes a bound on the number of sub-intervals into which the interval of integration may be divided by the routine. The number of sub-intervals cannot exceed LW/4. The more difficult the integrand, the larger LW should be. Suggested value: LW = 800 to 2000 is adequate for most problems. Constraint: LW >= 8.

13: IW(LIW) -- INTEGER array Output
On exit: IW(1) contains the actual number of sub-intervals used. The rest of the array is used as workspace.

14: LIW -- INTEGER Input
On entry:
the dimension of the array IW as declared in the (sub)program from which D01APF is called.
The number of sub-intervals into which the interval of integration may be divided cannot exceed LIW. Suggested value: LIW = LW/4. Constraint: LIW >= 2.

15: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. Users who are unfamiliar with this parameter should refer to the Essential Introduction for details.

```
     On exit: IFAIL = 0 unless the routine detects an error or
     gives a warning (see Section 6).

     For this routine, because the values of output parameters
     may be useful even if IFAIL /=0 on exit, users are
     recommended to set IFAIL to -1 before entry. It is then
     essential to test the value of IFAIL on exit.

6. Error Indicators and Warnings

Errors or warnings specified by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

IFAIL= 1
     The maximum number of subdivisions allowed with the given
     workspace has been reached without the accuracy requirements
     being achieved. Look at the integrand in order to determine
     the integration difficulties. If the position of a
     discontinuity or a singularity of algebraico-logarithmic
     type within the interval can be determined, the interval
     must be split up at this point and the integrator called on
     the subranges. If necessary, another integrator, which is
     designed for handling the difficulty involved, must be used.
     Alternatively consider relaxing the accuracy requirements
     specified by EPSABS and EPSREL, or increasing the amount of
     workspace.

IFAIL= 2
     Round-off error prevents the requested tolerance from being
     achieved. Consider  requesting less accuracy.

IFAIL= 3
     Extremely bad local integrand behaviour causes a very strong
     subdivision around one (or more) points of the interval.
     The same advice applies as in the case of IFAIL = 1.

IFAIL= 4
     On entry B <= A,

     or        ALFA <= -1,

     or        BETA <= -1,

     or        KEY < 1,

     or        KEY > 4.
```

IFAIL= 5
On entry LW < 8,

or LIW < 2.

7. Accuracy

The routine cannot guarantee, but in practice usually achieves, the following accuracy:

$$|I-RESULT|<=tol,$$

where

$$tol=max\{|EPSABS|,|EPSREL|*|I|\},$$

and EPSABS and EPSREL are user-specified absolute and relative error tolerances.

Moreover it returns the quantity ABSERR which, in normal circumstances, satisfies:

$$|I-RESULT|<=ABSERR<=tol.$$

8. Further Comments

The time taken by the routine depends on the integrand and on the accuracy required.

If IFAIL /= 0 on exit, then the user may wish to examine the contents of the array W, which contains the end-points of the sub-intervals used by D01APF along with the integral contributions and error estimates over these sub-intervals.

Specifically, for i=1,2,...,n, let r_i denote the approximation to the value of the integral over the sub-interval $[a_i,b_i]$ in the partition of [a,b] and e_i be the corresponding absolute error estimate. Then, $\int_{a_i}^{b_i} f(x)w(x)dx \simeq r_i$ and RESULT= $\sum_{i=1}^{n} r_i$. The value of n is returned in IW(1), and the values a_i, b_i, e_i and r_i are stored consecutively in the array W, that is:

```
          a =W(i),
           i

          b =W(n+i),
           i

          e =W(2n+i),
           i

          r =W(3n+i).
           i

     9. Example

     To compute:

                         1
                         /
                         |ln(x)cos(10(pi)x)dx  and
                         /
                         0

                              1
                              / sin(10x)
                              | --------dx.
                              /
                              0 \/x(1-x)

     The example program is not reproduced here. The source code for
     all example programs is distributed with the NAG Foundation
     Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Hilbert transform over finite interval

— nagd.ht —

```
\begin{page}{manpageXXd01aqf}{NAG Documentation: d01aqf}
\beginscroll
\begin{verbatim}
```

D01AQF(3NAG) Foundation Library (12/10/92) D01AQF(3NAG)

D01 -- Quadrature D01AQF
D01AQF -- NAG Foundation Library Routine Document

Note: Before using this routine, please read the Users' Note for your implementation to check implementation-dependent details. The symbol (*) after a NAG routine name denotes a routine that is not included in the Foundation Library.

1. Purpose

D01AQF calculates an approximation to the Hilbert transform of a function g(x) over [a,b]:

$$I=\int_a^b \frac{g(x)}{x-c}dx$$

for user-specified values of a, b and c.

2. Specification

```
       SUBROUTINE D01AQF (G, A, B, C, EPSABS, EPSREL, RESULT,
      1                   ABSERR, W, LW, IW, LIW, IFAIL)
       INTEGER          LW, IW(LIW), LIW, IFAIL
       DOUBLE PRECISION G, A, B, C, EPSABS, EPSREL, RESULT,
      1                 ABSERR, W(LW)
       EXTERNAL         G
```

3. Description

D01AQF is based upon the QUADPACK routine QAWC (Piessens et al [3]) and integrates a function of the form g(x)w(x), where the weight function

$$w(x)=\frac{1}{x-c}$$

is that of the Hilbert transform. (If a<c<b the integral has to be interpreted in the sense of a Cauchy principal value.) It is an adaptive routine which employs a 'global' acceptance criterion (as defined by Malcolm and Simpson [1]). Special care is taken to

ensure that c is never the end-point of a sub-interval (Piessens et al[2]). On each sub-interval (c_1, c_2) modified Clenshaw-Curtis integration of orders 12 and 24 is performed if $c_1-d<=c<=c_2+d$ where $d=(c_2-c_1)/20$. Otherwise the Gauss 7-point and Kronrod 15-point rules are used. The local error estimation is described by Piessens et al [3].

4. References

[1] Malcolm M A and Simpson R B (1976) Local Versus Global Strategies for Adaptive Quadrature. ACM Trans. Math. Softw. 1 129--146.

[2] Piessens R, Van Roy-Branders M and Mertens I (1976) The Automatic Evaluation of Cauchy Principal Value Integrals. Angewandte Informatik. 18 31--35.

[3] Piessens R, De Doncker-Kapenga E, Uberhuber C and Kahaner D (1983) QUADPACK, A Subroutine Package for Automatic Integration. Springer-Verlag.

5. Parameters

1: G -- DOUBLE PRECISION FUNCTION, supplied by the user. External Procedure

G must return the value of the function g at a given point.

Its specification is:

```
DOUBLE PRECISION FUNCTION G (X)
DOUBLE PRECISION X
```

1: X -- DOUBLE PRECISION Input

On entry: the point at which the function g must be evaluated.

G must be declared as EXTERNAL in the (sub)program from which D01AQF is called. Parameters denoted as Input must not be changed by this procedure.

2: A -- DOUBLE PRECISION Input

On entry: the lower limit of integration, a.

3: B -- DOUBLE PRECISION Input

On entry: the upper limit of integration, b. It is not necessary that a<b.

4: C -- DOUBLE PRECISION Input

```
     On entry: the parameter c in the weight function.
     Constraint: C must not equal A or B.

 5:  EPSABS -- DOUBLE PRECISION                          Input
     On entry: the absolute accuracy required. If EPSABS is
     negative, the absolute value is used. See Section 7.

 6:  EPSREL -- DOUBLE PRECISION                          Input
     On entry: the relative accuracy required. If EPSREL is
     negative, the absolute value is used. See Section 7.

 7:  RESULT -- DOUBLE PRECISION                         Output
     On exit: the approximation to the integral I.

 8:  ABSERR -- DOUBLE PRECISION                         Output
     On exit: an estimate of the modulus of the absolute error,
     which should be an upper bound for |I-RESULT|.

 9:  W(LW) -- DOUBLE PRECISION array                    Output
     On exit: details of the computation, as described in
     Section 8.

10:  LW -- INTEGER                                       Input
     On entry:
     the dimension of the array W as declared in the (sub)program
     from which D01AQF is called.
     The value of LW (together with that of LIW below) imposes a
     bound on the number of sub-intervals into which the interval
     of integration may be divided by the routine. The number of
     sub-intervals cannot exceed LW/4. The more difficult the
     integrand, the larger LW should be. Suggested value: LW =
     800 to 2000 is adequate for most problems. Constraint: LW >=
     4.

11:  IW(LIW) -- INTEGER array                           Output
     On exit: IW(1) contains the actual number of sub-intervals
     used. The rest of the array is used as workspace.

12:  LIW -- INTEGER                                      Input
     On entry:
     the dimension of the array IW as declared in the
     (sub)program from which D01AQF is called.
     The number of sub-intervals into which the interval of
     integration may be divided cannot exceed LIW. Suggested
     value: LIW = LW/4. Constraint: LIW >= 1.

13:  IFAIL -- INTEGER                             Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. Users who are
     unfamiliar with this parameter should refer to the Essential
     Introduction for details.
```

On exit: IFAIL = 0 unless the routine detects an error or gives a warning (see Section 6).

For this routine, because the values of output parameters may be useful even if IFAIL /=0 on exit, users are recommended to set IFAIL to -1 before entry. It is then essential to test the value of IFAIL on exit.

6. Error Indicators and Warnings

Errors or warnings specified by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL= 1

The maximum number of subdivisions allowed with the given workspace has been reached without the accuracy requirements being achieved. Look at the integrand in order to determine the integration difficulties. Another integrator which is designed for handling the type of difficulty involved, must be used. Alternatively consider relaxing the accuracy requirements specified by EPSABS and EPSREL, or increasing the workspace.

IFAIL= 2

Round-off error prevents the requested tolerance from being achieved. Consider requesting less accuracy.

IFAIL= 3

Extremely bad local behaviour of g(x) causes a very strong subdivision around one (or more) points of the interval. The same advice applies as in the case of IFAIL = 1.

IFAIL= 4

On entry C = A or C = B.

IFAIL= 5

On entry LW < 4,

or LIW < 1.

7. Accuracy

The routine cannot guarantee, but in practice usually achieves, the following accuracy:

|I-RESULT|<=tol,

where

$$tol=max\{|EPSABS|,|EPSREL|*|I|\},$$

and EPSABS and EPSREL are user-specified absolute and relative error tolerances. Moreover it returns the quantity ABSERR which, in normal circumstances satisfies:

$$|I-RESULT|<=ABSERR<=tol.$$

8. Further Comments

The time taken by the routine depends on the integrand and on the accuracy required.

If IFAIL /= 0 on exit, then the user may wish to examine the contents of the array W, which contains the end-points of the sub-intervals used by D01AQF along with the integral contributions and error estimates over these sub-intervals.

Specifically, for i=1,2,...,n, let r_i denote the approximation to the value of the integral over the sub-interval $[a_i,b_i]$ in the partition of [a,b] and e_i be the corresponding absolute error estimate. Then, $\int_{a_i}^{b_i} g(x)w(x)dx \simeq r_i$ and $RESULT=\sum_{i=1}^{n} r_i$. The value of n is returned in IW(1), and the values a_i, b_i, e_i and r_i are stored consecutively in the array W, that is:

a_i=W(i),

b_i=W(n+i),

e_i=W(2n+i) and

r_i=W(3n+i).

9. Example

```
        To compute the Cauchy principal value of

                               1
                               /         dx
                               |  ----------------.
                               /    2     2     1
                               -1 (x +0.01 )(x- -)
                                                2

        The example program is not reproduced here. The source code for
        all example programs is distributed with the NAG Foundation
        Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

———-

Approximate Sine or Cosine over $[a,\infty]$

— nagd.ht —

```
\begin{page}{manpageXXd01asf}{NAG Documentation: d01asf}
\beginscroll
\begin{verbatim}



     D01ASF(3NAG)      Foundation Library (12/10/92)      D01ASF(3NAG)



          D01 -- Quadrature                                      D01ASF
                  D01ASF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          D01ASF calculates an approximation to the sine or the cosine
          transform of a function g over [a,infty):
```

```
   infty
   /
I= |    g(x)||,sin((omega)x)dx  or  I=
   /
   a
 infty
 /
 |    g(x)||,cos((omega)x)dx
 /
 a
```

(for a user-specified value of (omega)).

2. Specification

```
       SUBROUTINE D01ASF (G, A, OMEGA, KEY, EPSABS, RESULT,
      1                   ABSERR, LIMLST, LST, ERLST, RSLST,
      2                   IERLST, W, LW, IW, LIW, IFAIL)
       INTEGER          KEY, LIMLST, LST, IERLST(LIMLST), LW, IW
      1                 (LIW), LIW, IFAIL
       DOUBLE PRECISION G, A, OMEGA, EPSABS, RESULT, ABSERR, ERLST
      1                 (LIMLST), RSLST(LIMLST), W(LW)
       EXTERNAL         G
```

3. Description

D01ASF is based upon the QUADPACK routine QAWFE (Piessens et al [2]). It is an adaptive routine, designed to integrate a function of the form g(x)w(x) over a semi-infinite interval, where w(x) is either sin((omega)x) or cos((omega)x). Over successive intervals

```
               C =[a+(k-1)c,a+kc],   k=1,2,...,LST
                k
```

integration is performed by the same algorithm as is used by D01ANF. The intervals C are of constant length
                       k

```
          c={2[|(omega)|]+1}(pi)/|(omega)|,  (omega)/=0
```

where [|(omega)|] represents the largest integer less than or equal to |(omega)|. Since c equals an odd number of half periods, the integral contributions over succeeding intervals will alternate in sign when the function g is positive and monotonically decreasing over [a,infty). The algorithm, described by [2], incorporates a global acceptance criterion (as defined by Malcolm and Simpson [1]) together with the (epsilon)-algorithm (Wynn [3]) to perform extrapolation. The local error estimation is described by Piessens et al [2].

If (omega)=0 and KEY = 1, the routine uses the same algorithm as D01AMF (with EPSREL = 0.0).

In contrast to the other routines in Chapter D01, D01ASF works only with a user-specified absolute error tolerance (EPSABS). Over the interval C_k it attempts to satisfy the absolute accuracy requirement

$$EPSA_k = U_k * EPSABS$$

where $U_k = (1-p)p^{k-1}$, for k=1,2,... and p=0.9.

However, when difficulties occur during the integration over the kth sub-interval C_k such that the error flag IERLST(k) is non-zero, the accuracy requirement over subsequent intervals is relaxed. See Piessens et al [2] for more details.

4. References

[1] Malcolm M A and Simpson R B (1976) Local Versus Global Strategies for Adaptive Quadrature. ACM Trans. Math. Softw. 1 129--146.

[2] Piessens R, De Doncker-Kapenga E, Uberhuber C and Kahaner D (1983) QUADPACK, A Subroutine Package for Automatic Integration. Springer-Verlag.

[3] Wynn P (1956) On a Device for Computing the $e_m(S_n)$ Transformation. Math. Tables Aids Comput. 10 91--96.

5. Parameters

1: G -- DOUBLE PRECISION FUNCTION, supplied by the user. External Procedure

G must return the value of the function g at a given point.

Its specification is:

```
      DOUBLE PRECISION FUNCTION G (X)
      DOUBLE PRECISION X
```

1: X -- DOUBLE PRECISION Input

On entry: the point at which the function g must be evaluated.

G must be declared as EXTERNAL in the (sub)program from

```
    which D01ASF is called. Parameters denoted as Input
    must not be changed by this procedure.

 2: A -- DOUBLE PRECISION                                  Input
    On entry: the lower limit of integration, a.

 3: OMEGA -- DOUBLE PRECISION                              Input
    On entry: the parameter (omega) in the weight function of
    the transform.

 4: KEY -- INTEGER                                         Input
    On entry: indicates which integral is to be computed:
         if KEY = 1, w(x)=cos((omega)x);

         if KEY = 2, w(x)=sin((omega)x).
     Constraint: KEY = 1 or 2.

 5: EPSABS -- DOUBLE PRECISION                             Input
    On entry: the absolute accuracy requirement. If EPSABS is
    negative, the absolute value is used. See Section 7.

 6: RESULT -- DOUBLE PRECISION                            Output
    On exit: the approximation to the integral I.

 7: ABSERR -- DOUBLE PRECISION                            Output
    On exit: an estimate of the modulus of the absolute error,
    which should be an upper bound for |I-RESULT|.

 8: LIMLST -- INTEGER                                      Input
    On entry: an upper bound on the number of intervals C
                                                          k
    needed for the integration. Suggested value: LIMLST = 50 is
    adequate for most problems. Constraint: LIMLST >= 3.

 9: LST -- INTEGER                                        Output
    On exit: the number of intervals C  actually used for the
                                      k
    integration.

10: ERLST(LIMLST) -- DOUBLE PRECISION array               Output
    On exit: ERLST(k) contains the error estimate corresponding
    to the integral contribution over the interval C , for
                                                    k
    k=1,2,...,LST.

11: RSLST(LIMLST) -- DOUBLE PRECISION array               Output
    On exit: RSLST(k) contains the integral contribution over
    the interval C  for k=1,2,...,LST.
                  k
```

```
12:  IERLST(LIMLST) -- INTEGER array                       Output
     On exit: IERLST(k) contains the error flag corresponding to
     RSLST(k), for k=1,2,...,LST. See Section 6.

13:  W(LW) -- DOUBLE PRECISION array                    Workspace

14:  LW -- INTEGER                                          Input
     On entry:
     the dimension of the array W as declared in the (sub)program
     from which D01ASF is called.
     The value of LW (together with that of LIW below) imposes a
     bound on the number of sub-intervals into which each
     interval C  may be divided by the routine. The number of
               k
     sub-intervals cannot exceed LW/4. The more difficult the
     integrand, the larger LW should be. Suggested value: a value
     in the range 800 to 2000 is adequate for most problems.
     Constraint: LW >= 4.

15:  IW(LIW) -- INTEGER array                              Output
     On exit: IW(1) contains the maximum number of sub-intervals
     actually used for integrating over any of the intervals C .
                                                             k
     The rest of the array is used as workspace.

16:  LIW -- INTEGER                                         Input
     On entry:
     the dimension of the array IW as declared in the
     (sub)program from which D01ASF is called.
     The number of sub-intervals into which each interval C  may
                                                          k
     be divided cannot exceed LIW/2. Suggested value: LIW = LW/2.
     Constraint: LIW >= 2.

17:  IFAIL -- INTEGER                                Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. Users who are
     unfamiliar with this parameter should refer to the Essential
     Introduction for details.

     On exit: IFAIL = 0 unless the routine detects an error or
     gives a warning (see Section 6).

     For this routine, because the values of output parameters
     may be useful even if IFAIL /=0 on exit, users are
     recommended to set IFAIL to -1 before entry. It is then
     essential to test the value of IFAIL on exit.

6. Error Indicators and Warnings

Errors or warnings specified by the routine:
```

If on entry IFAIL = 0 or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL= 1

The maximum number of subdivisions allowed with the given workspace has been reached without the accuracy requirements being achieved. Look at the integrand in order to determine the integration difficulties. If the position of a local difficulty within the interval can be determined (e.g. a singularity of the integrand or its derivative, a peak, a discontinuity, etc) you will probably gain from splitting up the interval at this point and calling D01ASF on the infinite subrange and an appropriate integrator on the finite subrange. Alternatively, consider relaxing the accuracy requirements specified by EPSABS or increasing the amount of workspace.

IFAIL= 2

Round-off error prevents the requested tolerance from being achieved. The error may be underestimated. Consider requesting less accuracy.

IFAIL= 3

Extremely bad local integrand behaviour causes a very strong subdivision around one (or more) points of the interval. The same advice applies as in the case of IFAIL = 1.

IFAIL= 4

The requested tolerance cannot be achieved, because the extrapolation does not increase the accuracy satisfactorily; the returned result is the best which can be obtained. The same advice applies as in the case of IFAIL = 1.

IFAIL= 5

The integral is probably divergent, or slowly convergent. Please note that divergence can occur with any non-zero value of IFAIL.

IFAIL= 6

On entry KEY < 1,

or KEY > 2,

or LIMLST < 3.

IFAIL= 7

Bad integration behaviour occurs within one or more of the intervals C_k. Location and type of the difficulty involved can be determined from the vector IERLST (see below).

IFAIL= 8

Maximum number of intervals C_k (= LIMLST) allowed has been achieved. Increase the value of LIMLST to allow more cycles.

IFAIL= 9

The extrapolation table constructed for convergence acceleration of the series formed by the integral contribution over the intervals C_k, does not converge to the required accuracy.

IFAIL= 10

On entry LW < 4,

or LIW < 2.

In the cases IFAIL = 7, 8 or 9, additional information about the cause of the error can be obtained from the array IERLST, as follows:

IERLST(k)=1

The maximum number of subdivisions = min(LW/4,LIW/2) has been achieved on the kth interval.

IERLST(k)=2

Occurrence of round-off error is detected and prevents the tolerance imposed on the kth interval from being achieved.

IERLST(k)=3

Extremely bad integrand behaviour occurs at some points of the kth interval.

IERLST(k)=4

The integration procedure over the kth interval does not converge (to within the required accuracy) due to round-off in the extrapolation procedure invoked on this interval. It is assumed that the result on this interval is the best which can be obtained.

IERLST(k)=5

The integral over the kth interval is probably divergent or slowly convergent. It must be noted that divergence can occur with any other value of IERLST(k).

7. Accuracy

The routine cannot guarantee, but in practice usually achieves,

```
          the following accuracy:

                              |I-RESULT|<=|EPSABS|,

          where EPSABS is the user-specified absolute error tolerance.
          Moreover, it returns the quantity ABSERR, which, in normal
          circumstances, satisfies

                          |I-RESULT|<=ABSERR<=|EPSABS|.

          8. Further Comments

          None.

          9. Example

          To compute

                              infty
                              /      1
                              |     ---cos((pi)x/2)dx.
                              /
                              0     \/x

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

———-

Weights and abscissae for Gaussian quadrature formula

— nagd.ht —

```
\begin{page}{manpageXXd01bbf}{NAG Documentation: d01bbf}
\beginscroll
\begin{verbatim}



     D01BBF(3NAG)                 D01BBF                D01BBF(3NAG)
```

```
D01 -- Quadrature                                          D01BBF
        D01BBF -- NAG Foundation Library Routine Document

Note: Before using this routine, please read the Users' Note for
your implementation to check implementation-dependent details.
The symbol (*) after a NAG routine name denotes a routine that is
not included in the Foundation Library.

Note for users via the Axiom system: the interface to this routine
has been enhanced for use with Axiom and is slightly different to
that offered in the standard version of the Foundation Library.

1. Purpose

D01BBF returns the weights and abscissae appropriate to a
Gaussian quadrature formula with a specified number of abscissae.
The formulae provided are Gauss-Legendre, Gauss-Rational, Gauss-
Laguerre and Gauss-Hermite.

2. Specification

       SUBROUTINE D01BBF (A, B, ITYPE, N, WEIGHT, ABSCIS, GTYPE,
      1                   IFAIL)
       INTEGER          ITYPE, N, GTYPE, IFAIL
       DOUBLE PRECISION A, B, WEIGHT(N), ABSCIS(N)

3. Description

This routine returns the weights and abscissae for use in the
Gaussian quadrature of a function f(x). The quadrature takes the
form

                             n
                             --
                          S= >  w f(x )
                             --  i   i
                             i=1

where w  are the weights and x  are the abscissae (see Davis and
       i                       i
Rabinowitz [1], Froberg [2], Ralston [3] or Stroud and Secrest
[4]).

Weights and abscissae are available for Gauss-Legendre, Gauss-
Rational, Gauss-Laguerre and Gauss-Hermite quadrature, and for a
selection of values of n (see Section 5).

(a)   Gauss-Legendre Quadrature:
                                  b
```

$$\tilde{S}=\int_{a} f(x)dx$$

where a and b are finite and it will be exact for any function of the form

$$f(x)=\sum_{i=0}^{2n-1} c_i x^i$$

(b) Gauss-Rational quadrature:

$$\tilde{S}=\int_{a}^{\infty} f(x)dx \quad (a+b>0) \quad \text{or} \quad \tilde{S}=\int_{-\infty}^{a} f(x)dx \quad (a+b<0)$$

and will be exact for any function of the form

$$f(x)=\sum_{i=2}^{2n+1} \frac{c_i}{(x+b)^i}=\frac{\sum_{i=0}^{2n-1} c_{2n+1-i}(x+b)^i}{(x+b)^{2n+1}}$$

(c) Gauss-Laguerre quadrature, adjusted weights option:

$$\tilde{S}=\int_{a}^{\infty} f(x)dx \quad (b>0<) \quad \text{or} \quad \tilde{S}=\int_{-\infty}^{a} f(x)dx \quad (b<0)$$

and will be exact for any function of the form

$$f(x)=e^{-bx}\sum_{i=0}^{2n-1} c_i x^i$$

(d) Gauss-Hermite quadrature, adjusted weights option:

$$\tilde{S}=\int_{-\infty}^{+\infty} f(x)dx$$

and will be exact for any function of the form

$$f(x)=e^{-b(x-a)^2}\sum_{i=0}^{2n-1} c_i x^i \quad (b>0)$$

(e) Gauss-Laguerre quadrature, normal weights option:

$$\tilde{S}=\int_{a}^{\infty} e^{-bx} f(x)dx \quad (b>0) \quad \text{or} \quad \tilde{S}=\int_{-\infty}^{a} e^{-bx} f(x)dx$$

(b<0)

and will be exact for any function of the form

$$f(x)=\sum_{i=0}^{2n-1} c_i x^i$$

(f) Gauss-Hermite quadrature, normal weights option:

$$\tilde{S}=\int_{-\infty}^{+\infty} e^{-b(x-a)^2} f(x)dx$$

and will be exact for any function of the form:

$$f(x)=\sum_{i=0}^{2n-1} c_i x^i$$

Note: that the Gauss-Legendre abscissae, with a=-1, b=+1, are the zeros of the Legendre polynomials; the Gauss-Laguerre abscissae, with a=0, b=1, are the zeros of the Laguerre polynomials; and the Gauss-Hermite abscissae, with a=0, b=1, are the zeros of the Hermite polynomials.

4. References

[1] Davis P J and Rabinowitz P (1967) Numerical Integration. Blaisdell Publishing Company. 33--52.

[2] Froberg C E (1965) Introduction to Numerical Analysis. Addison-Wesley. 181--187.

[3] Ralston A (1965) A First Course in Numerical Analysis. McGraw-Hill. 87--90.

[4] Stroud A H and Secrest D (1966) Gaussian Quadrature Formulas. Prentice-Hall.

5. Parameters

1: A -- DOUBLE PRECISION Input

2: B -- DOUBLE PRECISION Input
On entry: the quantities a and b as described in the appropriate subsection of Section 3.

3: ITYPE -- INTEGER Input
On entry: indicates the type of weights for Gauss-Laguerre or Gauss-Hermite quadrature (see Section 3):

if ITYPE = 1, adjusted weights will be returned;

if ITYPE = 0, normal weights will be returned.

Constraint: ITYPE = 0 or 1.

For Gauss-Legendre or Gauss-Rational quadrature, this parameter is not used.

4: N -- INTEGER Input
On entry: the number of weights and abscissae to be returned, n. Constraint: N = 1,2,3,4,5,6,8,10,12,14,16,20,24,32,48 or 64.

5: WEIGHT(N) -- DOUBLE PRECISION array Output
On exit: the N weights. For Gauss-Laguerre and Gauss-Hermite quadrature, these will be the adjusted weights if ITYPE = 1, and the normal weights if ITYPE = 0.

6: ABSCIS(N) -- DOUBLE PRECISION array Output
On exit: the N abscissae.

7: GTYPE -- INTEGER Input
On entry: The value of GTYPE indicates which quadrature formula are to be used:

GTYPE = 0 for Gauss-Legendre weights and abscissae;

GTYPE = 1 for Gauss-Rational weights and abscissae;

GTYPE = 2 for Gauss-Laguerre weights and abscissae;

GTYPE = 3 for Gauss-Hermite weights and abscissae.

8: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1
The N-point rule is not among those stored. If the soft fail option is used, the weights and abscissae returned will be those for the largest valid value of N less than the requested value, and the excess elements of WEIGHT and ABSCIS (i.e., up to the requested N) will be filled with zeros.

IFAIL= 2
The value of A and/or B is invalid.
Gauss-Rational: A + B = 0

Gauss-Laguerre: B = 0

Gauss-Hermite: B <= 0
If the soft fail option is used the weights and abscissae are returned as zero.

IFAIL= 3
Laguerre and Hermite normal weights only: underflow is occurring in evaluating one or more of the normal weights. If the soft fail option is used, the underflowing weights are returned as zero. A smaller value of N must be used; or adjusted weights should be used (ITYPE = 1). In the latter case, take care that underflow does not occur when evaluating the integrand appropriate for adjusted weights.

IFAIL=4
GTYPE < 0 or GTYPE > 3

7. Accuracy

The weights and abscissae are stored for standard values of A and B to full machine accuracy.

```
     8. Further Comments

     Timing is negligible.

     9. Example

     This example program returns the abscissae and (adjusted) weights
     for the six-point Gauss-Laguerre formula.

     The example program is not reproduced here. The source code for
     all example programs is distributed with the NAG Foundation
     Library software and should be available on-line.
\end{verbatim}
\endscroll
\end{page}
```

Multidimensional integrals with finite limits

— nagd.ht —

```
\begin{page}{manpageXXd01fcf}{NAG Documentation: d01fcf}
\beginscroll
\begin{verbatim}



     D01FCF(3NAG)      Foundation Library (12/10/92)      D01FCF(3NAG)



          D01 -- Quadrature                                       D01FCF
                  D01FCF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          D01FCF attempts to evaluate a multi-dimensional integral (up to
          15 dimensions), with constant and finite limits, to a specified
          relative accuracy, using an adaptive subdivision strategy.
```

2. Specification

```
       SUBROUTINE D01FCF (NDIM, A, B, MINPTS, MAXPTS, FUNCTN,
      1                   EPS, ACC, LENWRK, WRKSTR, FINVAL, IFAIL)
       INTEGER          NDIM, MINPTS, MAXPTS, LENWRK, IFAIL
       DOUBLE PRECISION A(NDIM), B(NDIM), FUNCTN, EPS, ACC, WRKSTR
      1                 (LENWRK), FINVAL
       EXTERNAL         FUNCTN
```

3. Description

The routine returns an estimate of a multi-dimensional integral over a hyper-rectangle (i.e., with constant limits), and also an estimate of the relative error. The user sets the relative accuracy required, supplies the integrand as a function subprogram (FUNCTN), and also sets the minimum and maximum acceptable number of calls to FUNCTN (in MINPTS and MAXPTS).

The routine operates by repeated subdivision of the hyper-rectangular region into smaller hyper-rectangles. In each subregion, the integral is estimated using a seventh-degree rule, and an error estimate is obtained by comparison with a fifth-degree rule which uses a subset of the same points. The fourth differences of the integrand along each co-ordinate axis are evaluated, and the subregion is marked for possible future subdivision in half along that co-ordinate axis which has the largest absolute fourth difference.

If the estimated errors, totalled over the subregions, exceed the requested relative error (or if fewer than MINPTS calls to FUNCTN have been made), further subdivision is necessary, and is performed on the subregion with the largest estimated error, that subregion being halved along the appropriate co-ordinate axis.

The routine will fail if the requested relative error level has not been attained by the time MAXPTS calls to FUNCTN have been made; or, if the amount LENWRK of working storage is insufficient. A formula for the recommended value of LENWRK is given in Section 5. If a smaller value is used, and is exhausted in the course of execution, the routine switches to a less efficient mode of operation; only if this mode also breaks down is insufficient storage reported.

D01FCF is based on the HALF subroutine developed by van Dooren and de Ridder [1]. It uses a different basic rule, described by Genz and Malik [2].

4. References

[1] Van Dooren P and De Ridder L (1976) An Adaptive Algorithm

for Numerical Integration over an N-dimensional Cube. J. Comput. Appl. Math. 2 207--217.

[2] Genz A C and Malik A A (1980) An Adaptive Algorithm for Numerical Integration over an N-dimensional Rectangular Region. J. Comput. Appl. Math. 6 295--302.

5. Parameters

1: NDIM -- INTEGER Input
On entry: the number of dimensions of the integral, n.
Constraint: 2 <= NDIM <= 15.

2: A(NDIM) -- DOUBLE PRECISION array Input
On entry: the lower limits of integration, a_i, for i=1,2,...,n.

3: B(NDIM) -- DOUBLE PRECISION array Input
On entry: the upper limits of integration, b_i, for i=1,2,...,n.

4: MINPTS -- INTEGER Input/Output
On entry: MINPTS must be set to the minimum number of integrand evaluations to be allowed. On exit: MINPTS contains the actual number of integrand evaluations used by D01FCF.

5: MAXPTS -- INTEGER Input
On entry: the maximum number of integrand evaluations to be allowed.
Constraints:

MAXPTS >= MINPTS

MAXPTS >= (alpha),

where $(alpha)=2^{NDIM}+2*NDIM^2+2*NDIM+1$.

6: FUNCTN -- DOUBLE PRECISION FUNCTION, supplied by the user.
External Procedure
FUNCTN must return the value of the integrand f at a given point.

Its specification is:

```
      DOUBLE PRECISION FUNCTION FUNCTN (NDIM,Z)
      INTEGER          NDIM
```

```
          DOUBLE PRECISION Z(NDIM)

      1:  NDIM -- INTEGER                                  Input
          On entry: the number of dimensions of the integral, n.

      2:  Z(NDIM) -- DOUBLE PRECISION array                Input
          On entry: the co-ordinates of the point at which the
          integrand must be evaluated.
     FUNCTN must be declared as EXTERNAL in the (sub)program
     from which D01FCF is called. Parameters denoted as
     Input must not be changed by this procedure.

 7:  EPS -- DOUBLE PRECISION                               Input
     On entry: the relative error acceptable to the user. When
     the solution is zero or very small relative accuracy may not
     be achievable but the user may still set EPS to a reasonable
     value and check for the error exit IFAIL = 2. Constraint:
     EPS > 0.0.

 8:  ACC -- DOUBLE PRECISION                              Output
     On exit: the estimated relative error in FINVAL.

 9:  LENWRK -- INTEGER                                     Input
     On entry:
     the dimension of the array WRKSTR as declared in the
     (sub)program from which D01FCF is called.
     Suggested value: for maximum efficiency, LENWRK >=
     (NDIM+2)*(1+MAXPTS/(alpha)) (see parameter MAXPTS for
     (alpha)).

     If LENWRK is less than this, the routine will usually run
     less efficiently and may fail. Constraint: LENWRK>=2*NDIM+4.

10:  WRKSTR(LENWRK) -- DOUBLE PRECISION array          Workspace

11:  FINVAL -- DOUBLE PRECISION                           Output
     On exit: the best estimate obtained for the integral.

12:  IFAIL -- INTEGER                               Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. Users who are
     unfamiliar with this parameter should refer to the Essential
     Introduction for details.
     On exit: IFAIL = 0 unless the routine detects an error or
     gives a warning (see Section 6).

     For this routine, because the values of output parameters
     may be useful even if IFAIL /=0 on exit, users are
     recommended to set IFAIL to -1 before entry. It is then
     essential to test the value of IFAIL on exit. To suppress
     the output of an error message when soft failure occurs, set
```

IFAIL to 1.

6. Error Indicators and Warnings

Errors or warnings specified by the routine:

IFAIL= 1
On entry NDIM < 2,

or NDIM > 15,

or MAXPTS is too small,

or LENWRK<2*NDIM+4,

or EPS <= 0.0.

IFAIL= 2
MAXPTS was too small to obtain the required relative accuracy EPS. On soft failure, FINVAL and ACC contain estimates of the integral and the relative error, but ACC will be greater than EPS.

IFAIL= 3
LENWRK was too small. On soft failure, FINVAL and ACC contain estimates of the integral and the relative error, but ACC will be greater than EPS.

7. Accuracy

A relative error estimate is output through the parameter ACC.

8. Further Comments

Execution time will usually be dominated by the time taken to evaluate the integrand FUNCTN, and hence the maximum time that could be taken will be proportional to MAXPTS.

9. Example

This example program estimates the integral

```
                    2
1  1  1  1  4z z exp(2z z )
/  /  /  /    1 3     1 3
|  |  |  |  ---------------dz dz dz dz =0.575364.
/  /  /  /            2     4  3  2  1
0  0  0  0    (1+z +z )
                  2  4
```

```
          The accuracy requested is one part in 10,000.

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Third-order finite-difference integration

— nagd.ht —

```
\begin{page}{manpageXXd01gaf}{NAG Documentation: d01gaf}
\beginscroll
\begin{verbatim}



     D01GAF(3NAG)      Foundation Library (12/10/92)      D01GAF(3NAG)



          D01 -- Quadrature                                        D01GAF
                  D01GAF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          D01GAF integrates a function which is specified numerically at
          four or more points, over the whole of its specified range, using
          third-order finite-difference formulae with error estimates,
          according to a method due to Gill and Miller.

          2. Specification

                 SUBROUTINE D01GAF (X, Y, N, ANS, ER, IFAIL)
                 INTEGER          N, IFAIL
                 DOUBLE PRECISION X(N), Y(N), ANS, ER
```

3. Description

This routine evaluates the definite integral

$$I = \int_{x_1}^{x_n} y(x)\,dx,$$

where the function y is specified at the n-points $x_1,x_2,\ldots,x_n$, which should be all distinct, and in either ascending or descending order. The integral between successive points is calculated by a four-point finite-difference formula centred on the interval concerned, except in the case of the first and last intervals, where four-point forward and backward difference formulae respectively are employed. If n is less than 4, the routine fails. An approximation to the truncation error is integrated and added to the result. It is also returned separately to give an estimate of the uncertainty in the result. The method is due to Gill and Miller.

4. References

[1] Gill P E and Miller G F (1972) An Algorithm for the Integration of Unequally Spaced Data. Comput. J. 15 80--83.

5. Parameters

1: X(N) -- DOUBLE PRECISION array Input
On entry: the values of the independent variable, i.e., the $x_1,x_2,\ldots,x_n$. Constraint: either X(1) < X(2) <... < X(N) or X(1) > X(2) >... > X(N).

2: Y(N) -- DOUBLE PRECISION array Input
On entry: the values of the dependent variable y_i at the points x_i, for i=1,2,...,n.

3: N -- INTEGER Input
On entry: the number of points, n. Constraint: N >= 4.

4: ANS -- DOUBLE PRECISION Output
On exit: the estimate of the integral.

```
 5:  ER -- DOUBLE PRECISION                                Output
     On exit: an estimate of the uncertainty in ANS.

 6:  IFAIL -- INTEGER                                Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1
     Indicates that fewer than four-points have been supplied to
     the routine.

IFAIL= 2
     Values of X are neither strictly increasing nor strictly
     decreasing.

IFAIL= 3
     Two points have the same X-value.

No error is reported arising from the relative magnitudes of ANS
and ER on return, due to the difficulty when the true answer is
zero.

7. Accuracy

No accuracy level is specified by the user before calling the
routine but on return ABS(ER) is an approximation to, but not
necessarily a bound for, |I-ANS|. If on exit IFAIL > 0, both ANS
and ER are returned as zero.

8. Further Comments

The time taken by the routine depends on the number of points
supplied, n.

In their paper, Gill and Miller [1] do not add the quantity ER to
ANS before return. However, extensive tests have shown that a
dramatic reduction in the error often results from such addition.
In other cases, it does not make an improvement, but these tend
to be cases of low accuracy in which the modified answer is not
significantly inferior to the unmodified one. The user has the
option of recovering the Gill-Miller answer by subtracting ER
from ANS on return from the routine.
```

```
          9. Example

          The example program evaluates the integral

                                    1
                                    /  4
                                    | ----dx=(pi)
                                    /    2
                                    0 1+x

          reading in the function values at 21 unequally-spaced points.

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Monte Carlo integration over hyper-rectangular regions

— nagd.ht —

```
\begin{page}{manpageXXd01gbf}{NAG Documentation: d01gbf}
\beginscroll
\begin{verbatim}



     D01GBF(3NAG)      Foundation Library (12/10/92)      D01GBF(3NAG)



          D01 -- Quadrature                                         D01GBF
                  D01GBF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose
```

D01GBF returns an approximation to the integral of a function over a hyper-rectangular region, using a Monte Carlo method. An approximate relative error estimate is also returned. This routine is suitable for low accuracy work.

2. Specification

```
       SUBROUTINE D01GBF (NDIM, A, B, MINCLS, MAXCLS, FUNCTN,
      1                   EPS, ACC, LENWRK, WRKSTR, FINEST, IFAIL)
       INTEGER          NDIM, MINCLS, MAXCLS, LENWRK, IFAIL
       DOUBLE PRECISION A(NDIM), B(NDIM), FUNCTN, EPS, ACC, WRKSTR
      1                 (LENWRK), FINEST
       EXTERNAL         FUNCTN
```

3. Description

D01GBF uses an adaptive Monte Carlo method based on the algorithm described by Lautrup [1]. It is implemented for integrals of the form:

$$\int_{a_1}^{b_1}\int_{a_2}^{b_2}\ldots\int_{a_n}^{b_n} f(x_1,x_2,\ldots,x_n)dx_n\ldots dx_2 dx_1 .$$

Upon entry, unless LENWRK has been set to the minimum value 10*NDIM, the routine subdivides the integration region into a number of equal volume subregions. Inside each subregion the integral and the variance are estimated by means of pseudo-random sampling. All contributions are added together to produce an estimate for the whole integral and total variance. The variance along each co-ordinate axis is determined and the routine uses this information to increase the density and change the widths of the sub-intervals along each axis, so as to reduce the total variance. The total number of subregions is then increased by a factor of two and the program recycles for another iteration. The program stops when a desired accuracy has been reached or too many integral evaluations are needed for the next cycle.

4. References

[1] Lautrup B (1971) An Adaptive Multi-dimensional Integration Procedure. Proc. 2nd Coll. on Advanced Methods in Theoretical Physics, Marseille.

5. Parameters

1: NDIM -- INTEGER Input
On entry: the number of dimensions of the integral, n.
Constraint: NDIM >= 1.

2: A(NDIM) -- DOUBLE PRECISION array Input
On entry: the lower limits of integration, a_i, for i=1,2,...,n.

3: B(NDIM) -- DOUBLE PRECISION array Input
On entry: the upper limits of integration, b_i, for i=1,2,...,n.

4: MINCLS -- INTEGER Input/Output
On entry: MINCLS must be set:

either to the minimum number of integrand evaluations to be allowed, in which case MINCLS >= 0;

or to a negative value. In this case the routine assumes that a previous call had been made with the same parameters NDIM, A and B and with either the same integrand (in which case D01GBF continues calculation) or a similar integrand (in which case D01GBF begins the calculation with the subdivision used in the last iteration of the previous call) . See also WRKSTR. On exit: MINCLS contains the number of integrand evaluations actually used by D01GBF.

5: MAXCLS -- INTEGER Input
On entry: the maximum number of integrand evaluations to be allowed. In the continuation case this is the number of new integrand evaluations to be allowed. These counts do not include zero integrand values.
Constraints:

MAXCLS > MINCLS,

MAXCLS >= 4*(NDIM+1).

6: FUNCTN -- DOUBLE PRECISION FUNCTION, supplied by the user.
External Procedure
FUNCTN must return the value of the integrand f at a given point.

Its specification is:

```
DOUBLE PRECISION FUNCTION FUNCTN (NDIM, X)
INTEGER          NDIM
DOUBLE PRECISION X(NDIM)
```

```
     1:  NDIM -- INTEGER                                Input
         On entry: the number of dimensions of the integral, n.

     2:  X(NDIM) -- DOUBLE PRECISION array              Input
         On entry: the co-ordinates of the point at which the
         integrand must be evaluated.
     FUNCTN must be declared as EXTERNAL in the (sub)program
     from which D01GBF is called. Parameters denoted as
     Input must not be changed by this procedure.

  7: EPS -- DOUBLE PRECISION                            Input
     On entry: the relative accuracy required. Constraint: EPS
     >= 0.0.

  8: ACC -- DOUBLE PRECISION                           Output
     On exit: the estimated relative accuracy of FINEST.

  9: LENWRK -- INTEGER                                  Input
     On entry:
     the dimension of the array WRKSTR as declared in the
     (sub)program from which D01GBF is called.
     For maximum efficiency, LENWRK should be about
                                   1/NDIM
                  3*NDIM*(MAXCLS/4)      +7*NDIM.
     If LENWRK is given the value 10*NDIM then the subroutine
     uses only one iteration of a crude Monte Carlo method with
     MAXCLS sample points. Constraint: LENWRK >= 10*NDIM.

 10: WRKSTR(LENWRK) -- DOUBLE PRECISION array        Input/Output
     On entry: if MINCLS<0, WRKSTR must be unchanged from the
     previous call of D01GBF - except that for a new integrand
     WRKSTR(LENWRK) must be set to 0.0. See also MINCLS. On
     exit: WRKSTR contains information about the current sub-
     interval structure which could be used in later calls of
     D01GBF. In particular, WRKSTR(j) gives the number of sub-
     intervals used along the jth co-ordinate axis.

 11: FINEST -- DOUBLE PRECISION                        Output
     On exit: the best estimate obtained for the integral.

 12: IFAIL -- INTEGER                            Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. Users who are
     unfamiliar with this parameter should refer to the Essential
     Introduction for details.

     On exit: IFAIL = 0 unless the routine detects an error or
     gives a warning (see Section 6).

     For this routine, because the values of output parameters
     may be useful even if IFAIL /=0 on exit, users are
```

```
     recommended to set IFAIL to -1 before entry. It is then
     essential to test the value of IFAIL on exit. To suppress
     the output of an error message when soft failure occurs, set
     IFAIL to 1.

6. Error Indicators and Warnings

Errors or warnings specified by the routine:

IFAIL= 1
     On entry NDIM < 1,

     or       MINCLS >= MAXCLS,

     or       LENWRK < 10*NDIM,

     or       MAXCLS < 4*(NDIM+1),

     or       EPS < 0.0.

IFAIL= 2
     MAXCLS was too small for D01GBF to obtain the required
     relative accuracy EPS. In this case D01GBF returns a value
     of FINEST with estimated relative error ACC, but ACC will be
     greater than EPS. This error exit may be taken before MAXCLS
     non-zero integrand evaluations have actually occurred, if
     the routine calculates that the current estimates could not
     be improved before MAXCLS was exceeded.

7. Accuracy

A relative error estimate is output through the parameter ACC.
The confidence factor is set so that the actual error should be
less than ACC 90% of the time. If a user desires a higher
confidence level then a smaller value of EPS should be used.

8. Further Comments

The running time for D01GBF will usually be dominated by the time
used to evaluate the integrand FUNCTN, so the maximum time that
could be used is approximately proportional to MAXCLS.

For some integrands, particularly those that are poorly behaved
in a small part of the integration region, D01GBF may terminate
with a value of ACC which is significantly smaller than the
actual relative error. This should be suspected if the returned
value of MINCLS is small relative to the expected difficulty of
the integral. Where this occurs, D01GBF should be called again,
but with a higher entry value of MINCLS (e.g. twice the returned
value) and the results compared with those from the previous
```

```
call.

The exact values of FINEST and ACC on return will depend (within
statistical limits) on the sequence of random numbers generated
within D01GBF by calls to G05CAF. Separate runs will produce
identical answers unless the part of the program executed prior
to calling D01GBF also calls (directly or indirectly) routines
from Chapter G05, and the series of such calls differs between
runs. If desired, the user may ensure the identity or difference
between runs of the results returned by D01GBF, by calling G05CBF
or G05CCF respectively, immediately before calling D01GBF.

9. Example

This example program calculates the integral

        1  1  1  1  4x x exp(2x x )
        /  /  /  /    1 3      1 3
        |  |  |  |  ---------------dx dx dx dx =0.575364.
        /  /  /  /             2     1  2  3  4
        0  0  0  0    (1+x +x )
                          2  4

The example program is not reproduced here. The source code for
all example programs is distributed with the NAG Foundation
Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Ordinary Differential Equations

— nagd.ht —

```
\begin{page}{manpageXXd02}{NAG Documentation: d02}
\beginscroll
\begin{verbatim}


     D02(3NAG)         Foundation Library (12/10/92)         D02(3NAG)
```

Chapter D02

Ordinary Differential Equations

1. Scope of the Chapter

This chapter is concerned with the numerical solution of ordinary differential equations. There are two main types of problem, those in which all boundary conditions are specified at one point (initial-value problems), and those in which the boundary conditions are distributed between two or more points (boundary-value problems and eigenvalue problems). Routines are available for initial-value problems, two-point boundary-value problems and Sturm-Liouville eigenvalue problems.

2. Background to the Problems

For most of the routines in this chapter a system of ordinary differential equations must be written in the form

$$y'_1 = f_1(x,y_1,y_2,...,y_n),$$

$$y'_2 = f_2(x,y_1,y_2,...,y_n),$$

$$.. ...$$

$$y'_n = f_n(x,y_1,y_2,...y_n),$$

that is the system must be given in first-order form. The n dependent variables (also, the solution) $y_1,y_2,...,y_n$ are functions of the independent variable x, and the differential equations give expressions for the first derivatives $y'_i=dy_i/dx$ in terms of x and $y_1,y_2,...,y_n$. For a system of n first-order equations, n associated boundary conditions are usually required to define the solution.

A more general system may contain derivatives of higher order, but such systems can almost always be reduced to the first-order form by introducing new variables. For example, suppose we have the third-order equation

$$z'''+zz''+k(l-z'^2)=0.$$

We write $y_1=z$, $y_2=z'$, $y_3=z''$, and the third order equation may then be written as the system of first-order equations

$$y'_1 = y_2$$

$$y'_2 = y_3$$

$$y'_3 = -y_1 y_3 -k(1-y_2^2).$$

For this system n = 3 and we require 3 boundary conditions in order to define the solution. These conditions must specify values of the dependent variables at certain points. For example, we have an initial-value problem if the conditions are:

$y_1 = 0$ at x=0

$y_2 = 0$ at x=0

$y_3 = 0.1$ at x=0.

These conditions would enable us to integrate the equations numerically from the point x=0 to some specified end-point. We have a boundary-value problem if the conditions are:

$y_1 = 0$ at x=0

$y_2 = 0$ at x=0

$y_2 = 1$ at x=10.

These conditions would be sufficient to define a solution in the range 0<=x<=10, but the problem could not be solved by direct integration (see Section 2.2). More general boundary conditions are permitted in the boundary-value case.

2.1. Initial-value Problems

To solve first-order systems, initial values of the dependent variables y_i, for i=1,2,...,n must be supplied at a given point, i

a. Also a point, b, at which the values of the dependent variables are required, must be specified. The numerical solution is then obtained by a step-by-step calculation which approximates values of the variables y_i, for i=1,2,...,n at finite intervals over the required range [a,b]. The routines in this chapter adjust the step length automatically to meet specified accuracy tolerances. Although the accuracy tests used are reliable over each step individually, in general an accuracy requirement cannot be guaranteed over a long range. For many problems there may be no serious accumulation of error, but for unstable systems small perturbations of the solution will often lead to rapid divergence of the calculated values from the true values. A simple check for stability is to carry out trial calculations with different tolerances; if the results differ appreciably the system is probably unstable. Over a short range, the difficulty may possibly be overcome by taking sufficiently small tolerances, but over a long range it may be better to try to reformulate the problem.

A special class of initial-value problems are those for which the solutions contain rapidly decaying transient terms. Such problems are called stiff; an alternative way of describing them is to say that certain eigenvalues of the Jacobian matrix (ddf_i/ddy_j) have large negative real parts when compared to others. These problems require special methods for efficient numerical solution; the methods designed for non-stiff problems when applied to stiff problems tend to be very slow, because they need small step lengths to avoid numerical instability. A full discussion is given in Hall and Watt [6] and a discussion of the methods for stiff problems is given in Berzins, Brankin and Gladwell [1].

2.2. Boundary-value Problems

A full discussion of the design of the methods and codes for boundary-value problems is given in Gladwell [4]. In general, a system of nonlinear differential equations with boundary conditions given at two or more points cannot be guaranteed to have a solution. The solution has to be determined iteratively (if it exists). Finite-difference equations are set up on a mesh of points and estimated values for the solution at the grid points are chosen. Using these estimated values as starting values a Newton iteration is used to solve the finite-difference equations. The accuracy of the solution is then improved by deferred corrections or the addition of points to the mesh or a combination of both. Good initial estimates of the solution may be required in some cases but results may be difficult to compute when the solution varies very rapidly over short ranges. A discussion is given in Chapters 9 and 11 of Gladwell and Sayers [5] and Chapter 4 of Childs et al [2].

2.3. Eigenvalue Problems

Sturm-Liouville problems of the form

(p(x)y')'+q(x,(lambda))y=0

with appropriate boundary conditions given at two points, can be solved by a Scaled Pruefer method. In this method the differential equation is transformed to another which can be solved for a specified eigenvalue by a shooting method. A discussion is given in Chapter 11 of Gladwell and Sayers [5] and a complete description is given in Pryce [7].

2.6. References

[1] Berzins M, Brankin R W and Gladwell I (1987) Design of the Stiff Integrators in the NAG Library. Technical Report. TR14/87 NAG.

[2] (1979) Codes for Boundary-value Problems in Ordinary Differential Equations. Lecture Notes in Computer Science. (ed Childs B, Scott M, Daniel J W, Denman E and Nelson P) 76 Springer-Verlag.

[3] Gladwell I (1979) Initial Value Routines in the NAG Library. ACM Trans Math Softw. 5 386--400.

[4] Gladwell I (1987) The NAG Library Boundary Value Codes. Numerical Analysis Report. 134 Manchester University.

[5] (1980) Computational Techniques for Ordinary Differential Equations. (Gladwell I and Sayers D K) Academic Press.

[6] Hall G and Watt J M (eds) (1976) Modern Numerical Methods for Ordinary Differential Equations. Clarendon Press.

[7] Pryce J D (1986) Error Estimation for Phase-function Shooting Methods for Sturm-Liouville Problems. J. Num. Anal. 6 103--123.

3. Recommendations on Choice and Use of Routines

There are no routines which deal directly with COMPLEX equations. These may however be transformed to larger systems of real equations of the required form. Split each equation into its real and imaginary parts and solve for the real and imaginary parts of each component of the solution. Whilst this process doubles the size of the system and may not always be appropriate it does make available for use the full range of routines provided presently.

3.1. Initial-value Problems

For simple first-order problems with low accuracy requirements, that is problems on a short range of integration, with derivative functions f_i which are inexpensive to calculate and where only a few correct figures are required, the best routines to use are likely to be the Runge-Kutta-Merson (RK) routines, D02BBF and D02BHF. For larger problems, over long ranges or with high accuracy requirements the variable-order, variable-step Adams routine D02CJF should usually be preferred. For stiff equations, that is those with rapidly decaying transient solutions, the Backward Differentiation Formula (BDF) variable-order, variable-step routine D02EJF should be used.

There are four routines for initial-value problems, two of which use the Runge-Kutta-Merson method:

D02BBF integrates a system of first order ordinary differential equations over a range with intermediate output and a choice of error control

D02BHF integrates a system of first order ordinary differential equations with a choice of error control until a position is determined where a function of the solution is zero.

one uses an Adams method:

D02CJF combines the functionality of D02BBF and D02BHF

and one uses a BDF method:

D02EJF combines the functionality of D02BBF and D02BHF.

3.2. Boundary-value Problems

D02GAF may be used for simple boundary-value problems with assigned boundary values. The user may find that convergence is difficult to achieve using D02GAF since only specifying the unknown boundary values and the position of the finite-difference mesh is permitted. In such cases the user may use D02RAF which permits specification of an initial estimate for the solution at all mesh points and allows the calculation to be influenced in other ways too. D02RAF is designed to solve a general nonlinear two-point boundary value problem with nonlinear boundary conditions.

```
A routine, D02GBF, is also supplied specifically for the general
linear two-point boundary-value problem written in a standard

The user is advised to use interpolation routines from the E01
Chapter to obtain solution values at points not on the final
mesh.

3.3. Eigenvalue Problems

There is one general purpose routine for eigenvalue problems,
D02KEF. It may be used to solve regular or singular second-order
Sturm-Liouville problems on a finite or infinite range.
Discontinous coefficient functions can be treated and
eigenfunctions can be computed.


D02 -- Ordinary Differential Equations            Contents -- D02
Chapter D02

Ordinary Differential Equations

D02BBF  ODEs, IVP, Runge-Kutta-Merson method, over a range,
        intermediate output

D02BHF  ODEs, IVP, Runge-Kutta-Merson method, until function of
        solution is zero

D02CJF  ODEs, IVP, Adams method, until function of solution is
        zero, intermediate output

D02EJF  ODEs, stiff IVP, BDF method, until function of solution
        is zero, intermediate output

D02GAF  ODEs, boundary value problem, finite difference technique
        with deferred correction, simple nonlinear problem

D02GBF  ODEs, boundary value problem, finite difference technique
        with deferred correction, general linear problem

D02KEF  2nd order Sturm-Liouville problem, regular/singular
        system, finite/infinite range, eigenvalue and
        eigenfunction, user-specified break-points

D02RAF  ODEs, general nonlinear boundary value problem, finite
        difference technique with deferred correction,
        continuation facility

\end{verbatim}
\endscroll
```

```
\end{page}
```

———-

First-order ODE over an interval with initial conditions

— **nagd.ht** —

```
\begin{page}{manpageXXd02bbf}{NAG Documentation: d02bbf}
\beginscroll
\begin{verbatim}



     D02BBF(3NAG)                 D02BBF                 D02BBF(3NAG)



     D02 -- Ordinary Differential Equations                    D02BBF
             D02BBF -- NAG Foundation Library Routine Document

     Note: Before using this routine, please read the Users' Note for
     your implementation to check implementation-dependent details.
     The symbol (*) after a NAG routine name denotes a routine that is
     not included in the Foundation Library.

     Note for users via the Axiom system: the interface to this routine
     has been enhanced for use with Axiom and is slightly different to
     that offered in the standard version of the Foundation Library.

     1. Purpose

     D02BBF integrates a system of first-order ordinary differential
     equations over an interval with suitable initial conditions,
     using a Runge-Kutta-Merson method, and returns the solution at
     points specified by the user.

     2. Specification

            SUBROUTINE D02BBF (X, XEND, M, N, Y, TOL, IRELAB, RESULT,
           1                   FCN, OUTPUT, W, IFAIL)
            INTEGER          M, N, IRELAB, IFAIL
            DOUBLE PRECISION X, XEND, Y(N), TOL, W(N,7), RESULT(M,N)
            EXTERNAL         FCN, OUTPUT

     3. Description
```

The routine integrates a system of ordinary differential equations

$$y'_i = f_i(x, y_1, y_2, \ldots, y_n) \quad i=1,2,\ldots,n$$

from x = X to x = XEND using a Merson form of the Runge-Kutta method. The system is defined by a subroutine FCN supplied by the user, which evaluates f_i in terms of x and $y_1, y_2, \ldots, y_n$, and the values of $y_1, y_2, \ldots, y_n$ must be given at x = X.

The solution is returned via the user-supplied routine OUTPUT at a set of points specified by the user. This solution is obtained by quintic Hermite interpolation on solution values produced by the Runge-Kutta method.

The accuracy of the integration and, indirectly, the interpolation is controlled by the parameters TOL and IRELAB.

For a description of Runge-Kutta methods and their practical implementation see Hall and Watt [1].

4. References

[1] Hall G and Watt J M (eds) (1976) Modern Numerical Methods for Ordinary Differential Equations. Clarendon Press.

5. Parameters

1: X -- DOUBLE PRECISION Input/Output
On entry: X must be set to the initial value of the independent variable x. On exit: XEND, unless an error has occurred, when it contains the value of x at the error.

2: XEND -- DOUBLE PRECISION Input
On entry: the final value of the independent variable. If XEND < X on entry, integration will proceed in a negative direction.

3: M -- INTEGER Input
On entry: the first dimension of the array RESULT. This will usually be equal to the number of points at which the solution is required.
Constraint: M > 0.

4: N -- INTEGER Input
On entry: the number of differential equations.
Constraint: N > 0.

5: Y(N) -- DOUBLE PRECISION array Input/Output
On entry: the initial values of the solution $y_1,y_2,\ldots,y_n$.
On exit: the computed values of the solution at the final value of X.

6: TOL -- DOUBLE PRECISION Input/Output
On entry: TOL must be set to a positive tolerance for controlling the error in the integration.

D02BBF has been designed so that, for most problems, a reduction in TOL leads to an approximately proportional reduction in the error in the solution at XEND. The relation between changes in TOL and the error at intermediate output points is less clear, but for TOL small enough the error at intermediate output points should also be approximately proportional to TOL. However, the actual relation between TOL and the accuracy achieved cannot be guaranteed. The user is strongly recommended to call D02BBF with more than one value for TOL and to compare the results obtained to estimate their accuracy. In the absence of any prior knowledge, the user might compare the results obtained by calling D02BBF with $TOL=10.0^{-p}$ and $TOL=10.0^{-p-1}$ if p correct decimal digits in the solution are required. Constraint: TOL > 0.0. On exit: normally unchanged. However if the range X to XEND is so short that a small change in TOL is unlikely to make any change in the computed solution then, on return, TOL has its sign changed. This should be treated as a warning that the computed solution is likely to be more accurate than would be produced by using the same value of TOL on a longer range of integration.

7: IRELAB -- INTEGER Input
On entry: IRELAB determines the type of error control. At each step in the numerical solution an estimate of the local error, EST, is made. For the current step to be accepted the following condition must be satisfied:
IRELAB = 0
$EST=10.0<=TOL*\max\{1.0,|y_1|,|y_2|,\ldots,|y_n|\}$;

IRELAB = 1
EST <= TOL;

IRELAB = 2
$EST<=TOL*\max\{(\epsilon),|y_1|,|y_2|,\ldots,|y_n|\}$, where (epsilon) is machine precision.

If the appropriate condition is not satisfied, the step size is reduced and the solution is recomputed on the current step.

If the user wishes to measure the error in the computed solution in terms of the number of correct decimal places, then IRELAB should be given the value 1 on entry, whereas if the error requirement is in terms of the number of correct significant digits, then IRELAB should be given the value 2. Where there is no preference in the choice of error test IRELAB = 0 will result in a mixed error test. Constraint: 0 <= IRELAB <= 2.

8: RESULT(M,N) -- DOUBLE PRECISION array Output
On exit: the computed values of the solution at the points given by OUTPUT.

9: FCN -- SUBROUTINE, supplied by the user.
External Procedure
FCN must evaluate the functions f_i (i.e., the derivatives y'_i) for given values of its arguments $x,y_1,...,y_n$.

Its specification is:

```
SUBROUTINE FCN (X, Y, F)
DOUBLE PRECISION X, Y(n), F(n)
```

where n is the actual value of N in the call of D02BBF.

1: X -- DOUBLE PRECISION Input
On entry: the value of the argument x.

2: Y(*) -- DOUBLE PRECISION array Input
On entry: the value of the argument y_i, for i=1,2,...,n.

3: F(*) -- DOUBLE PRECISION array Output
On exit: the value of f_i, for i=1,2,...,n.

FCN must be declared as EXTERNAL in the (sub)program from which D02BBF is called. Parameters denoted as Input must not be changed by this procedure.

10: OUTPUT -- SUBROUTINE, supplied by the user.
External Procedure
OUTPUT allows the user to have access to intermediate values of the computed solution at successive points specified by the

```
        user. These solution values may be returned to the user via
        the array RESULT if desired (this is a non-standard feature
        added for use with the Axiom system).  OUTPUT is initially
        called by D02BBF with XSOL = X (the initial value of x). The
        user must reset XSOL to the next point where OUTPUT is to be
        called, and so on at each call to OUTPUT. If, after a call
        to OUTPUT, the reset point XSOL is beyond XEND, D02BBF will
        integrate to XEND with no further calls to OUTPUT; if a call
        to OUTPUT is required at the point XSOL = XEND, then XSOL
        must be given precisely the value XEND.

        Its specification is:

             SUBROUTINE OUTPUT(XSOL,Y,COUNT,M,N,RESULT)
             DOUBLE PRECISION Y(N),RESULT(M,N),XSOL
             INTEGER M,N,COUNT

         1:  XSOL -- DOUBLE PRECISION                   Input/Output
             On entry: the current value of the independent
             variable x. On exit: the next value of x at which
             OUTPUT is to be called.

         2:  Y(N) -- DOUBLE PRECISION array                    Input
             On entry: the computed solution at the point XSOL.

         3:  COUNT -- INTEGER                           Input/Output
             On entry: Zero if OUTPUT has not been called before, or
             the previous value of COUNT.
             On exit: A new value of COUNT: this can be used to keep
             track of the number of times OUTPUT has been called.

         4:  M -- INTEGER                                      Input
             On entry: The first dimension of RESULT.

         5:  N -- INTEGER                                      Input
             On entry: The dimension of Y.

         6:  RESULT(M,N) -- DOUBLE PRECISION array      Input/Output
             On entry: the previous contents of RESULT.
             On exit: RESULT may be used to return the values of the
             intermediate solutions to the user.

        OUTPUT must be declared as EXTERNAL in the (sub)program
        from which D02BBF is called. Parameters denoted as
        Input must not be changed by this procedure.

   11:  W(N,7) -- DOUBLE PRECISION array                   Workspace

   12:  IFAIL -- INTEGER                                Input/Output
        On entry: IFAIL must be set to 0, -1 or 1. For users not
```

```
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1
     On entry TOL <= 0,

     or       N <= 0,

     or       IRELAB /= 0, 1 or 2.

IFAIL= 2
     With the given value of TOL, no further progress can be made
     across the integration range from the current point x = X,
     or the dependence of the error on TOL would be lost if
     further progress across the integration range were attempted
     (see Section 8 for a discussion of this error exit). The
     components Y(1),Y(2),...,Y(n) contain the computed values of
     the solution at the current point x = X.

IFAIL= 3
     TOL is too small for the routine to take an initial step
     (see Section 8). X and Y(1),Y(2),...,Y(n) retain their
     initial values.

IFAIL= 4
     X = XEND and XSOL /= X after the initial call to OUTPUT.

IFAIL= 5
     A value of XSOL returned by OUTPUT lies behind the previous
     value of XSOL in the direction of integration.

IFAIL= 6
     A serious error has occurred in an internal call to
     D02PAF(*). Check all subroutine calls and array dimensions.
     Seek expert help.

IFAIL= 7
     A serious error has occurred in an internal call to
     D02XAF(*). Check all subroutine calls and array dimensions.
     Seek expert help.

7. Accuracy
```

The accuracy depends on TOL, on the mathematical properties of the differential system, on the length of the range of integration and on the method. It can be controlled by varying TOL but the approximate proportionality of the error to TOL holds only for a restricted range of values of TOL. For TOL too large, the underlying theory may break down and the result of varying TOL may be unpredictable. For TOL too small, rounding errors may affect the solution significantly and an error exit with IFAIL = 2 or IFAIL = 3 is possible.

At the intermediate output points the same remarks apply. For large values of TOL it is possible that the errors at some intermediate output points may be much larger than at XEND. In any case, it must not be expected that the error will have the same size at all output points. At any point, it is a combination of the errors arising from the integration of the differential equation and the interpolation. The effect of combining these errors will vary, though in most cases the integration error will dominate.

The user who requires a more reliable estimate of the accuracy achieved than can be obtained by varying TOL, is recommended to call D02BDF(*) where both the solution and a global error estimate are computed.

8. Further Comments

The time taken by the routine depends on the complexity and mathematical properties of the system of differential equations defined by FCN, on the range, the tolerance and the number of calls to OUTPUT. There is also an overhead of the form a+b*n where a and b are machine-dependent computing times.

If the routine fails with IFAIL = 3, then it can be called again with a larger value of TOL (if this has not already been tried). If the accuracy requested is really needed and cannot be obtained with this routine, the system may be very stiff (see below) or so badly scaled that it cannot be solved to the required accuracy.

If the routine fails with IFAIL = 2, it is probable that it has been called with a value of TOL which is so small that the solution cannot be obtained on the range X to XEND. This can happen for well-behaved systems and very small values of TOL. The user should, however, consider whether there is a more fundamental difficulty. For example:

(a) in the region of a singularity (infinite value) of the solution, the routine will usually stop with IFAIL = 2, unless overflow occurs first. If overflow occurs using D02BBF, D02PAF(*) can be used instead to trap the

```
      increasing solution before overflow occurs. In any case,
      numerical integration cannot be continued through a
      singularity, and analytic treatment should be considered;

(b)   for 'stiff' equations, where the solution contains rapidly
      decaying components, the routine will use very small steps
      in x (internally to D02BBF) to preserve stability. This
      will usually exhibit itself by making the computing time
      excessively long, or occasionally by an exit with IFAIL =
      2. Merson's method is not efficient in such cases, and the
      user should try using D02EBF(*) (Backward Differentiation
      Formula). To determine whether a problem is stiff,
      D02BDF(*) may be used.

For well-behaved systems with no difficulties such as stiffness
or singularities, the Merson method should work well for low
accuracy calculations (three or four figures). For higher
accuracy calculations or where FCN is costly to evaluate,
Merson's method may not be appropriate and a computationally less
expensive method may be D02CBF(*) which uses an Adams method.

Users with problems for which D02BBF is not sufficiently general
should consider using D02PAF(*) with D02XAF(*). D02PAF(*) is a
more general Merson routine with many facilities including more
general error control options and several criteria for
interrupting the calculations. D02XAF(*) interpolates on values
produced by D02PAF(*).

9. Example

To integrate the following equations (for a projectile)

                           y'=tan(phi)

                      -0.032tan(phi)   0.02v
                  v'= --------------- --------
                             v         cos(phi)


                                 -0.032
                         (phi)'= ------
                                    2
                                   v

over an interval X = 0.0 to XEND = 8.0, starting with values
y=0.0, v=0.5 and (phi)=(pi)/5 and printing the solution at steps
of 1.0. We write y=Y(1), v=Y(2) and (phi)=Y(3), and we set
TOL=1.0E-4 and TOL=1.0E-5 in turn so that we may compare the
solutions. The value of (pi) is obtained by using X01AAF(*).
```

```
     Note the careful construction of routine OUT to ensure that the
     value of XEND is printed.

     The example program is not reproduced here. The source code for
     all example programs is distributed with the NAG Foundation
     Library software and should be available on-line.
\end{verbatim}
\endscroll
\end{page}
```

———-

First-order ODE with initial conditions and user function

— nagd.ht —

```
\begin{page}{manpageXXd02bhf}{NAG Documentation: d02bhf}
\beginscroll
\begin{verbatim}



     D02BHF(3NAG)      Foundation Library (12/10/92)      D02BHF(3NAG)



          D02 -- Ordinary Differential Equations                     D02BHF
                  D02BHF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          D02BHF integrates a system of first-order ordinary differential
          equations over an interval with suitable initial conditions,
          using a Runge-Kutta-Merson method, until a user-specified
          function of the solution is zero.

          2. Specification

                 SUBROUTINE D02BHF (X, XEND, N, Y, TOL, IRELAB, HMAX, FCN,
                1                   G, W, IFAIL)
                 INTEGER          N, IRELAB, IFAIL
                 DOUBLE PRECISION X, XEND, Y(N), TOL, HMAX, G, W(N,7)
```

```
      EXTERNAL        FCN, G
```

3. Description

The routine advances the solution of a system of ordinary differential equations

$$y'_i = f_i(x, y_1, y_2, \ldots, y_n), \quad i=1,2,\ldots,n,$$

from x = X towards x = XEND using a Merson form of the Runge-Kutta method. The system is defined by a subroutine FCN supplied by the user, which evaluates f_i in terms of x and $y_1, y_2, \ldots, y_n$ (see Section 5), and the values of $y_1, y_2, \ldots, y_n$ must be given at x = X.

As the integration proceeds, a check is made on the function g(x,y) specified by the user, to determine an interval where it changes sign. The position of this sign change is then determined accurately by interpolating for the solution and its derivative. It is assumed that g(x,y) is a continuous function of the variables, so that a solution of g(x,y) = 0 can be determined by searching for a change in sign in g(x,y).

The accuracy of the integration and, indirectly, of the determination of the position where g(x,y) = 0, is controlled by the parameter TOL.

For a description of Runge-Kutta methods and their practical implementation see Hall and Watt [1].

4. References

[1] Hall G and Watt J M (eds) (1976) Modern Numerical Methods for Ordinary Differential Equations. Clarendon Press.

5. Parameters

1: X -- DOUBLE PRECISION Input/Output
On entry: X must be set to the initial value of the independent variable x. On exit: the point where g(x,y) = 0. 0 unless an error has occurred, when it contains the value of x at the error. In particular, if g(x,y)/=0.0 anywhere on the range X to XEND, it will contain XEND on exit.

2: XEND -- DOUBLE PRECISION Input
On entry: the final value of the independent variable x.

If XEND < X on entry, integration proceeds in a negative direction.

3: N -- INTEGER Input
On entry: the number of differential equations, n.
Constraint: N > 0.

4: Y(N) -- DOUBLE PRECISION array Input/Output
On entry: the initial values of the solution $y_1, y_2, ..., y_n$.

On exit: the computed values of the solution at the final point x = X.

5: TOL -- DOUBLE PRECISION Input/Output
On entry: TOL must be set to a positive tolerance for controlling the error in the integration and in the determination of the position where g(x,y) = 0.0.

D02BHF has been designed so that, for most problems, a reduction in TOL leads to an approximately proportional reduction in the error in the solution obtained in the integration. The relation between changes in TOL and the error in the determination of the position where g(x,y) = 0.0 is less clear, but for TOL small enough the error should be approximately proportional to TOL. However, the actual relation between TOL and the accuracy cannot be guaranteed. The user is strongly recommended to call D02BHF with more than one value for TOL and to compare the results obtained to estimate their accuracy. In the absence of any prior knowledge the user might compare results obtained by calling D02BHF with $TOL=10.0^{-p}$ and $TOL=10.0^{-p-1}$ if p correct decimal digits in the solution are required. Constraint: TOL > 0.0.

On exit: normally unchanged. However if the range from x = X to the position where g(x,y) = 0.0 (or to the final value of x if an error occurs) is so short that a small change in TOL is unlikely to make any change in the computed solution, then TOL is returned with its sign changed. To check results returned with TOL < 0.0, D02BHF should be called again with a positive value of TOL whose magnitude is considerably smaller than that of the previous call.

6: IRELAB -- INTEGER Input
On entry: IRELAB determines the type of error control. At each step in the numerical solution an estimate of the local error, EST, is made. For the current step to be accepted the following condition must be satisfied:
IRELAB = 0

EST<=TOL*max{1.0,|y |,|y |,...,|y |};

```
                        1    2        n

   IRELAB = 1
         EST <= TOL;

   IRELAB = 2
         EST<=TOL*max{(epsilon),|y |,|y |,...,|y |},
                                  1    2        n

         where (epsilon) is machine precision.
   If the appropriate condition is not satisfied, the step size
   is reduced and the solution recomputed on the current step.

   If the user wishes to measure the error in the computed
   solution in terms of the number of correct decimal places,
   then IRELAB should be given the value 1 on entry, whereas if
   the error requirement is in terms of the number of correct
   significant digits, then IRELAB should be given the value 2.
   Where there is no preference in the choice of error test,
   IRELAB = 0 will result in a mixed error test. It should be
   borne in mind that the computed solution will be used in
   evaluating g(x,y). Constraint: 0 <= IRELAB <= 2.

7: HMAX -- DOUBLE PRECISION                               Input
   On entry: if HMAX = 0.0, no special action is taken.

   If HMAX /= 0.0, a check is made for a change in sign of
   g(x,y) at steps not greater than |HMAX|. This facility
   should be used if there is any chance of 'missing' the
   change in sign by checking too infrequently. For example, if
   two changes of sign of g(x,y) are expected within a distance
   h, say, of each other, then a suitable value for HMAX might
   be HMAX = h/2. If only one change of sign in g(x,y) is
   expected on the range X to XEND, then the choice HMAX = 0.0
   is most appropriate.

8: FCN -- SUBROUTINE, supplied by the user.
                                           External Procedure
   FCN must evaluate the functions f  (i.e., the derivatives
                                    i
   y' ) for given values of its arguments x,y ,...,y .
     i                                       1      n

   Its specification is:

          SUBROUTINE FCN (X, Y, F)
          DOUBLE PRECISION X, Y(n), F(n)

    1:  X -- DOUBLE PRECISION                             Input
        On entry: the value of the argument x.
```

```
     2:  Y(*) -- DOUBLE PRECISION array                    Input
         On entry: the value of the argument y , for
                                              i
         i=1,2,...,n.

     3:  F(*) -- DOUBLE PRECISION array                   Output
         On exit: the value of f , for i=1,2,...,n.
                                i
    FCN must be declared as EXTERNAL in the (sub)program
    from which D02BHF is called. Parameters denoted as
    Input must not be changed by this procedure.

 9:  G -- DOUBLE PRECISION FUNCTION, supplied by the user.
                                          External Procedure
    G must evaluate the function g(x,y) at a specified point.

    Its specification is:

           DOUBLE PRECISION FUNCTION G (X, Y)
           DOUBLE PRECISION X, Y(n)
    where n is the actual value of N in the call of D02BHF.

     1:  X -- DOUBLE PRECISION                             Input
         On entry: the value of the independent variable x.

     2:  Y(*) -- DOUBLE PRECISION array                    Input
         On entry: the value of y , for i=1,2,...,n.
                                 i
    G must be declared as EXTERNAL in the (sub)program from
    which D02BHF is called. Parameters denoted as Input
    must not be changed by this procedure.

10:  W(N,7) -- DOUBLE PRECISION array                  Workspace

11:  IFAIL -- INTEGER                               Input/Output
    On entry: IFAIL must be set to 0, -1 or 1. For users not
    familiar with this parameter (described in the Essential
    Introduction) the recommended value is 0.

    On exit: IFAIL = 0 unless the routine detects an error (see
    Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1
     On entry TOL <= 0.0,
```

```
     or       N <= 0,

     or       IRELAB /= 0, 1 or 2.

IFAIL= 2
     With the given value of TOL, no further progress can be made
     across the integration range from the current point x = X,
     or dependence of the error on TOL would be lost if further
     progress across the integration range were attempted (see
     Section 8 for a discussion of this error exit). The
     components Y(1),Y(2),...,Y(n) contain the computed values of
     the solution at the current point x = X. No point at which
     g(x,y) changes sign has been located up to the point x = X.

IFAIL= 3
     TOL is too small for the routine to take an initial step
     (see Section 8). X and Y(1),Y(2),...,Y(n) retain their
     initial values.

IFAIL= 4
     At no point in the range X to XEND did the function g(x,y)
     change sign. It is assumed that g(x,y) = 0.0 has no
     solution.

IFAIL= 5
     A serious error has occurred in an internal call to
     C05AZF(*). Check all subroutine calls and array dimensions.
     Seek expert help.

IFAIL= 6
     A serious error has occurred in an internal call to
     D02PAF(*). Check all subroutine calls and array dimensions.
     Seek expert help.

IFAIL= 7
     A serious error has occurred in an internal call to
     D02XAF(*). Check all subroutine calls and array dimensions.
     Seek expert help.

7. Accuracy

The accuracy depends on TOL, on the mathematical properties of
the differential system, on the position where g(x,y) = 0.0 and
on the method. It can be controlled by varying TOL but the
approximate proportionality of the error to TOL holds only for a
restricted range of values of TOL. For TOL too large, the
underlying theory may break down and the result of varying TOL
may be unpredictable. For TOL too small, rounding error may
affect the solution significantly and an error exit with IFAIL =
2 or IFAIL = 3 is possible.
```

The accuracy may also be restricted by the properties of g(x,y). The user should try to code G without introducing any unnecessary cancellation errors.

8. Further Comments

The time taken by the routine depends on the complexity and mathematical properties of the system of differential equations defined by FCN, the complexity of G, on the range, the position of the solution and the tolerance. There is also an overhead of the form a+b*n where a and b are machine-dependent computing times.

For some problems it is possible that D02BHF will return IFAIL = 4 because of inaccuracy of the computed values Y, leading to inaccuracy in the computed values of g(x,y) used in the search for the solution of g(x,y) = 0.0. This difficulty can be overcome by reducing TOL sufficiently, and if necessary, by choosing HMAX sufficiently small. If possible, the user should choose XEND well beyond the expected point where g(x,y) = 0.0; for example make |XEND-X| about 50 larger than the expected range. As a simple check, if, with XEND fixed, a change in TOL does not lead to a significant change in Y at XEND, then inaccuracy is not a likely source of error.

If the routine fails with IFAIL = 3, then it could be called again with a larger value of TOL if this has not already been tried. If the accuracy requested is really needed and cannot be obtained with this routine, the system may be very stiff (see below) or so badly scaled that it cannot be solved to the required accuracy.

If the routine fails with IFAIL = 2, it is likely that it has been called with a value of TOL which is so small that a solution cannot be obtained on the range X to XEND. This can happen for well-behaved systems and very small values of TOL. The user should, however, consider whether there is a more fundamental difficulty. For example:

(a) in the region of a singularity (infinite value) of the solution, the routine will usually stop with IFAIL = 2, unless overflow occurs first. If overflow occurs using D02BHF, D02PAF(*) can be used instead to trap the increasing solution, before overflow occurs. In any case, numerical integration cannot be continued through a singularity, and analytical treatment should be considered;

(b) for 'stiff' equations, where the solution contains rapidly decaying components, the routine will compute in very small steps in x (internally to D02BHF) to preserve stability. This will usually exhibit itself by making the computing time excessively long, or occasionally by an exit with IFAIL = 2. Merson's method is not efficient in such cases, and the user should try D02EHF(*) which uses a Backward Differentiation Formula method. To determine whether a problem is stiff, D02BDF(*) may be used.

For well-behaved systems with no difficulties such as stiffness or singularities, the Merson method should work well for low accuracy calculations (three or four figures). For high accuracy calculations or where FCN is costly to evaluate, Merson's method may not be appropriate and a computationally less expensive method may be D02CHF(*) which uses an Adams method.

For problems for which D02BHF is not sufficiently general, the user should consider D02PAF(*). D02PAF(*) is a more general Merson routine with many facilities including more general error control options and several criteria for interrupting the calculations. D02PAF(*) can be combined with the rootfinder C05AZF(*) and the interpolation routine D02XAF(*) to solve equations involving $y_1,y_2,...,y_n$ and their derivatives.

D02BHF can also be used to solve an equation involving x, $y_1,y_2,...,y_n$ and the derivatives of $y_1,y_2,...,y_n$. For example in Section 9, D02BHF is used to find a value of X > 0.0 where Y(1) = 0.0. It could instead be used to find a turning-point of y_1 by replacing the function g(x,y) in the program by:

```
      DOUBLE PRECISION FUNCTION G(X,Y)
      DOUBLE PRECISION X,Y(3),F(3)
      CALL FCN(X,Y,F)
      G = F(1)
      RETURN
      END
```

This routine is only intended to locate the first zero of g(x,y). If later zeros are required, users are strongly advised to construct their own more general root finding routines as discussed above.

9. Example

To find the value X > 0.0 at which y=0.0, where y, v, (phi) are defined by

y'=tan(phi)

```
          -0.032tan(phi)   0.02v
      v'= --------------- --------
                v        cos(phi)

                    0.032
           (phi)'=- -----
                      2
                     v

and where at X = 0.0 we are given y=0.5, v=0.5 and (phi)=(pi)/5.
We write y=Y(1), v=Y(2) and (phi)=Y(3) and we set TOL=1.0E-4 and
TOL=1.0E-5 in turn so that we can compare the solutions. We
expect the solution X ~= 7.3 and so we set XEND = 10.0 to avoid
determining the solution of y=0.0 too near the end of the range
of integration. The value of (pi) is obtained by using X01AAF(*).


The example program is not reproduced here. The source code for
all example programs is distributed with the NAG Foundation
Library software and should be available on-line.
```

```
\end{verbatim}
\endscroll
\end{page}
```

First-order ODE with variable-order, variable-step

— nagd.ht —

```
\begin{page}{manpageXXd02cjf}{NAG Documentation: d02cjf}
\beginscroll
\begin{verbatim}



     D02CJF(3NAG)                 D02CJF                 D02CJF(3NAG)



     D02 -- Ordinary Differential Equations                    D02CJF
             D02CJF -- NAG Foundation Library Routine Document

     Note: Before using this routine, please read the Users' Note for
     your implementation to check implementation-dependent details.
```

The symbol (*) after a NAG routine name denotes a routine that is not included in the Foundation Library.

Note for users via the Axiom system: the interface to this routine has been enhanced for use with Axiom and is slightly different to that offered in the standard version of the Foundation Library.

1. Purpose

D02CJF integrates a system of first-order ordinary differential equations over a range with suitable initial conditions, using a variable-order, variable-step Adams method until a user-specified function, if supplied, of the solution is zero, and returns the solution at points specified by the user, if desired.

2. Specification

```
       SUBROUTINE D02CJF (X, XEND, M, N, Y, FCN, TOL, RELABS,
      1                   RESULT, OUTPUT, G, W, IFAIL)
       INTEGER          M, N, IFAIL
       DOUBLE PRECISION X, XEND, Y(N), TOL, G, W(28+21*N), RESULT(M,N)
       CHARACTER*1      RELABS
       EXTERNAL         FCN, OUTPUT, G
```

3. Description

The routine advances the solution of a system of ordinary differential equations

$$y'_i=f_i(x,y_1,y_2,\ldots,y_n),\quad i=1,2,\ldots,n,$$

from x = X to x = XEND using a variable-order, variable-step Adams method. The system is defined by a subroutine FCN supplied by the user, which evaluates f_i in terms of x and $y_1,y_2,\ldots,y_n$. The initial values of $y_1,y_2,\ldots,y_n$ must be given at x = X.

The solution is returned via the user-supplied routine OUTPUT at points specified by the user, if desired: this solution is obtained by C^1 interpolation on solution values produced by the method. As the integration proceeds a check can be made on the user-specified function g(x,y) to determine an interval where it changes sign. The position of this sign change is then determined accurately by C^1 interpolation to the solution. It is assumed that g(x,y) is a continuous function of the variables, so that a solution of g(x,y)=0.0 can be determined by searching for a change in sign in g(x,y). The accuracy of the integration, the interpolation and, indirectly, of the determination of the position where g(x,y)=0.0, is controlled by the parameters TOL and RELABS.

For a description of Adams methods and their practical implementation see Hall and Watt [1].

4. References

[1] Hall G and Watt J M (eds) (1976) Modern Numerical Methods for Ordinary Differential Equations. Clarendon Press.

5. Parameters

1: X -- DOUBLE PRECISION Input/Output
On entry: the initial value of the independent variable x. Constraint: X /= XEND. On exit: if g is supplied by the user, it contains the point where g(x,y)=0.0, unless g(x,y)/=0.0 anywhere on the range X to XEND, in which case, X will contain XEND. If g is not supplied by the user it contains XEND, unless an error has occurred, when it contains the value of x at the error.

2: XEND -- DOUBLE PRECISION Input
On entry: the final value of the independent variable. If XEND < X, integration proceeds in the negative direction. Constraint: XEND /= X.

3: M -- INTEGER Input
On entry: the first dimension of the array RESULT. This will usually be equal to the number of points at which the solution is required.
Constraint: M > 0.

4: N -- INTEGER Input
On entry: the number of differential equations.
Constraint: N >= 1.

5: Y(N) -- DOUBLE PRECISION array Input/Output
On entry: the initial values of the solution $y_1,y_2,...,y_n$ at x = X. On exit: the computed values of the solution at the final point x = X.

6: FCN -- SUBROUTINE, supplied by the user. External Procedure
FCN must evaluate the functions f_i (i.e., the derivatives y'_i) for given values of their arguments $x,y_1,y_2,...,y_n$.

```
  i                                         1  2      n

Its specification is:

       SUBROUTINE FCN (X, Y, F)
       DOUBLE PRECISION X, Y(n), F(n)
where n is the actual value of N in the call of D02CJF.

 1:  X -- DOUBLE PRECISION                              Input
     On entry: the value of the independent variable x.

 2:  Y(*) -- DOUBLE PRECISION array                     Input
     On entry: the value of the variable y , for
                                          i
     i=1,2,...,n.

 3:  F(*) -- DOUBLE PRECISION array                    Output
     On exit: the value of f , for i=1,2,...,n.
                             i
FCN must be declared as EXTERNAL in the (sub)program
from which D02CJF is called. Parameters denoted as
Input must not be changed by this procedure.
```

```
7:  TOL -- DOUBLE PRECISION                             Input
    On entry: a positive tolerance for controlling the error in
    the integration. Hence TOL affects the determination of the
    position where g(x,y)=0.0, if g is supplied.

    D02CJF has been designed so that, for most problems, a
    reduction in TOL leads to an approximately proportional
    reduction in the error in the solution. However, the actual
    relation between TOL and the accuracy achieved cannot be
    guaranteed. The user is strongly recommended to call D02CJF
    with more than one value for TOL and to compare the results
    obtained to estimate their accuracy. In the absence of any
    prior knowledge, the user might compare the results obtained
                                -p              -p-1
    by calling D02CJF with TOL=10.0   and TOL=10.0     where p
    correct decimal digits are required in the solution.
    Constraint: TOL > 0.0.

8:  RELABS -- CHARACTER*1                               Input
    On entry: the type of error control. At each step in the
    numerical solution an estimate of the local error, EST, is
    made. For the current step to be accepted the following
    condition must be satisfied:

                 ------------------------------
                / n
               /  --                           2
       EST=   /   >  (e /((tau) *|y |+(tau) )) <=1.0
```

```
             /    --   i       r   i       a
           \/     i=1
where (tau)  and (tau)  are defined by
           r          a
     RELABS   (tau)    (tau)
                   r        a

     'M'      TOL      TOL

     'A'      0.0      TOL

     'R'      TOL      (epsilon)

     'D'      TOL      TOL
where (epsilon) is a small machine-dependent number and e
                                                        i
is an estimate of the local error at y , computed
                                      i
internally. If the appropriate condition is not satisfied,
the step size is reduced and the solution is recomputed on
the current step. If the user wishes to measure the error in
the computed solution in terms of the number of correct
decimal places, then RELABS should be set to 'A' on entry,
whereas if the error requirement is in terms of the number
of correct significant digits, then RELABS should be set to
'R'. If the user prefers a mixed error test, then RELABS
should be set to 'M', otherwise if the user has no
preference, RELABS should be set to the default 'D'. Note
that in this case 'D' is taken to be 'M'. Constraint: RELABS
= 'M', 'A', 'R', 'D'.

 9: RESULT(M,N) -- DOUBLE PRECISION array                Output
    On exit: the computed values of the solution at the points
    given by OUTPUT.

10: OUTPUT -- SUBROUTINE, supplied by the user.
                                                 External Procedure
    OUTPUT allows the user to have access to intermediate values
    of the computed solution at successive points specified by the
    user. These solution values may be returned to the user via
    the array RESULT if desired (this is a non-standard feature
    added for use with the Axiom system).  OUTPUT is initially
    called by D02CJF with XSOL = X (the initial value of x). The
    user must reset XSOL to the next point where OUTPUT is to be
    called, and so on at each call to OUTPUT. If, after a call
    to OUTPUT, the reset point XSOL is beyond XEND, D02CJF will
    integrate to XEND with no further calls to OUTPUT; if a call
    to OUTPUT is required at the point XSOL = XEND, then XSOL
    must be given precisely the value XEND.
```

```
Its specification is:

     SUBROUTINE OUTPUT(XSOL,Y,COUNT,M,N,RESULT)
     DOUBLE PRECISION Y(N),RESULT(M,N),XSOL
     INTEGER M,N,COUNT

 1:  XSOL -- DOUBLE PRECISION                   Input/Output
     On entry: the current value of the independent
     variable x. On exit: the next value of x at which
     OUTPUT is to be called.

 2:  Y(N) -- DOUBLE PRECISION array                    Input
     On entry: the computed solution at the point XSOL.

 3:  COUNT -- INTEGER                           Input/Output
     On entry: Zero if OUTPUT has not been called before, or
     the previous value of COUNT.
     On exit: A new value of COUNT: this can be used to keep
     track of the number of times OUTPUT has been called.

 4:  M -- INTEGER                                      Input
     On entry: The first dimension of RESULT.

 5:  N -- INTEGER                                      Input
     On entry: The dimension of Y.

 6:  RESULT(M,N) -- DOUBLE PRECISION array      Input/Output
     On entry: the previous contents of RESULT.
     On exit: RESULT may be used to return the values of the
     intermediate solutions to the user.

OUTPUT must be declared as EXTERNAL in the (sub)program
from which D02CJF is called. Parameters denoted as
Input must not be changed by this procedure.

11:  G -- DOUBLE PRECISION FUNCTION, supplied by the user.
                                        External Procedure
     G must evaluate the function g(x,y) for specified values x,y
     . It specifies the function g for which the first position x
     where g(x,y)=0 is to be found.

     If the user does not require the root finding option, the
     actual argument G  mustbe the dummy routine D02CJW. (D02CJW
     is included in the NAG Foundation Library and so need not be
     supplied by the user).

     Its specification is:

        DOUBLE PRECISION FUNCTION G (X, Y)
        DOUBLE PRECISION X, Y(n)
```

```
     where n is the actual value of N in the call of D02CJF.

      1:  X -- DOUBLE PRECISION                            Input
          On entry: the value of the independent variable x.

      2:  Y(*) -- DOUBLE PRECISION array                   Input
          On entry: the value of the variable y , for
                                                i
          i=1,2,...,n.
     G must be declared as EXTERNAL in the (sub)program from
     which D02CJF is called. Parameters denoted as Input
     must not be changed by this procedure.

12:  W(28+21*N) -- DOUBLE PRECISION array              Workspace

13:  IFAIL -- INTEGER                               Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

IFAIL= 1
     On entry TOL <= 0.0,

     or       N <= 0,

     or       RELABS /= 'M', 'A', 'R' or 'D'.

     or       X = XEND.

IFAIL= 2
     With the given value of TOL, no further progress can be made
     across the integration range from the current point x = X.
     (See Section 8 for a discussion of this error exit.) The
     components Y(1),Y(2),...,Y(N) contain the computed values of
     the solution at the current point x = X. If the user has
     supplied g, then no point at which g(x,y) changes sign has
     been located up to the point x = X.

IFAIL= 3
     TOL is too small for D02CJF to take an initial step. X and Y
```

(1),Y(2),...,Y(N) retain their initial values.

IFAIL= 4
XSOL has not been reset or XSOL lies behind X in the direction of integration, after the initial call to OUTPUT, if the OUTPUT option was selected.

IFAIL= 5
A value of XSOL returned by OUTPUT has not been reset or lies behind the last value of XSOL in the direction of integration, if the OUTPUT option was selected.

IFAIL= 6
At no point in the range X to XEND did the function g(x,y) change sign, if g was supplied. It is assumed that g(x,y)=0 has no solution.

IFAIL= 7
A serious error has occurred in an internal call. Check all subroutine calls and array sizes. Seek expert help.

7. Accuracy

The accuracy of the computation of the solution vector Y may be controlled by varying the local error tolerance TOL. In general, a decrease in local error tolerance should lead to an increase in accuracy. Users are advised to choose RELABS = 'M' unless they have a good reason for a different choice.

If the problem is a root-finding one, then the accuracy of the root determined will depend on the properties of g(x,y). The user should try to code G without introducing any unnecessary cancellation errors.

8. Further Comments

If more than one root is required then D02QFF(*) should be used.

If the routine fails with IFAIL = 3, then it can be called again with a larger value of TOL if this has not already been tried. If the accuracy requested is really needed and cannot be obtained with this routine, the system may be very stiff (see below) or so badly scaled that it cannot be solved to the required accuracy.

If the routine fails with IFAIL = 2, it is probable that it has been called with a value of TOL which is so small that a solution cannot be obtained on the range X to XEND. This can happen for well-behaved systems and very small values of TOL. The user should, however, consider whether there is a more fundamental difficulty. For example:

```
     (a)   in the region of a singularity (infinite value) of the
           solution, the routine will usually stop with IFAIL = 2,
           unless overflow occurs first. Numerical integration cannot
           be continued through a singularity, and analytic treatment
           should be considered;

     (b)   for 'stiff' equations where the solution contains rapidly
           decaying components, the routine will use very small steps
           in x (internally to D02CJF) to preserve stability. This
           will exhibit itself by making the computing time
           excessively long, or occasionally by an exit with IFAIL =
           2. Adams methods are not efficient in such cases, and the
           user should try D02EJF.

     9. Example

     We illustrate the solution of four different problems. In each
     case the differential system (for a projectile) is

                               y'=tan(phi)

                            -0.032tan(phi)   0.02v
                        v'= --------------- --------
                                  v         cos(phi)

                                      -0.032
                               (phi)'= ------
                                         2
                                        v

     over an interval X = 0.0 to XEND = 10.0 starting with values
     y=0.5, v=0.5 and (phi)=(pi)/5. We solve each of the following
     problems with local error tolerances 1.0E-4 and 1.0E-5.

     (i)   To integrate to x=10.0 producing output at intervals of 2.0
           until a point is encountered where y=0.0.

     (ii)  As (i) but with no intermediate output.

     (iii) As (i) but with no termination on a root-finding condition.

     (iv)  As (i) but with no intermediate output and no root-finding
           termination condition.

     The example program is not reproduced here. The source code for
     all example programs is distributed with the NAG Foundation
     Library software and should be available on-line.
\end{verbatim}
\endscroll
```

```
\end{page}
```

———-

Stiff First-order ODE with variable order and step

— nagd.ht —

```
\begin{page}{manpageXXd02ejf}{NAG Documentation: d02ejf}
\beginscroll
\begin{verbatim}



     D02EJF(3NAG)                 D02EJF                 D02EJF(3NAG)



     D02 -- Ordinary Differential Equations                     D02EJF
             D02EJF -- NAG Foundation Library Routine Document

     Note: Before using this routine, please read the Users' Note for
     your implementation to check implementation-dependent details.
     The symbol (*) after a NAG routine name denotes a routine that is
     not included in the Foundation Library.

     Note for users via the Axiom system: the interface to this routine
     has been enhanced for use with Axiom and is slightly different to
     that offered in the standard version of the Foundation Library.

     1. Purpose

     D02EJF integrates a stiff system of first-order ordinary
     differential equations over an interval with suitable initial
     conditions, using a variable-order, variable-step method
     implementing the Backward Differentiation Formulae (BDF), until a
     user-specified function, if supplied, of the solution is zero,
     and returns the solution at points specified by the user, if
     desired.

     2. Specification

            SUBROUTINE D02EJF (X, XEND, M, N, Y, FCN, PEDERV, TOL,
           1                   RELABS, OUTPUT, G, W, IW, RESULT, IFAIL)
            INTEGER          M, N, IW, IFAIL
            DOUBLE PRECISION X, XEND, Y(N), TOL, G, W(IW), RESULT(M,N)
            CHARACTER*1      RELABS
```

```
      EXTERNAL        FCN, PEDERV, OUTPUT, G
```

3. Description

The routine advances the solution of a system of ordinary differential equations

$$y'_i = f_i(x, y_1, y_2, \ldots, y_n), \quad i=1,2,\ldots,n,$$

from x = X to x = XEND using a variable-order, variable-step method implementing the BDF. The system is defined by a subroutine FCN supplied by the user, which evaluates f_i in terms of x and $y_1, y_2, \ldots, y_n$ (see Section 5). The initial values of $y_1, y_2, \ldots, y_n$ must be given at x = X.

The solution is returned via the user-supplied routine OUTPUT at points specified by the user, if desired: this solution is obtained by C^1 interpolation on solution values produced by the method. As the integration proceeds a check can be made on the user-specified function g(x,y) to determine an interval where it changes sign. The position of this sign change is then determined accurately by C^1 interpolation to the solution. It is assumed that g(x,y) is a continuous function of the variables, so that a solution of g(x,y) = 0.0 can be determined by searching for a change in sign in g(x,y). The accuracy of the integration, the interpolation and, indirectly, of the determination of the position where g(x,y) = 0.0, is controlled by the parameters TOL and RELABS. The Jacobian of the system y'=f(x,y) may be supplied in routine PEDERV, if it is available.

For a description of BDF and their practical implementation see Hall and Watt [1].

4. References

[1] Hall G and Watt J M (eds) (1976) Modern Numerical Methods for Ordinary Differential Equations. Clarendon Press.

5. Parameters

1: X -- DOUBLE PRECISION Input/Output
On entry: the initial value of the independent variable x. Constraint: X /= XEND On exit: if G is supplied by the user, X contains the point where g(x,y) = 0.0, unless g(x,y) /= 0.

0 anywhere on the range X to XEND, in which case, X will contain XEND. If G is not supplied X contains XEND, unless an error has occured, when it contains the value of x at the error.

2: XEND -- DOUBLE PRECISION Input
On entry: the final value of the independent variable. If XEND < X, integration proceeds in the negative direction.
Constraint: XEND /= X.

3: M -- INTEGER Input
On entry: the first dimension of the array RESULT. This will usually be equal to the number of points at which the solution is required.
Constraint: M > 0.

4: N -- INTEGER Input
On entry: the number of differential equations, n.
Constraint: N >= 1.

5: Y(N) -- DOUBLE PRECISION array Input/Output
On entry: the initial values of the solution $y_1,y_2,...,y_n$ at x = X. On exit: the computed values of the solution at the final point x = X.

6: FCN -- SUBROUTINE, supplied by the user.
External Procedure
FCN must evaluate the functions f_i (i.e., the derivatives y'_i) for given values of their arguments $x,y_1,y_2,...,y_n$.

Its specification is:

```
       SUBROUTINE FCN (X, Y, F)
       DOUBLE PRECISION X, Y(n), F(n)
```

where n is the actual value of N in the call of D02EJF.

1: X -- DOUBLE PRECISION Input
On entry: the value of the independent variable x.

2: Y(*) -- DOUBLE PRECISION array Input
On entry: the value of the variable y_i, for i=1,2,...,n.

3: F(*) -- DOUBLE PRECISION array Output
On exit: the value of f_i, for i=1,2,...,n.

FCN must be declared as EXTERNAL in the (sub)program from which D02EJF is called. Parameters denoted as Input must not be changed by this procedure.

7: PEDERV -- SUBROUTINE, supplied by the user. External Procedure

PEDERV must evaluate the Jacobian of the system (that is, the partial derivatives $\frac{\partial f_i}{\partial y_j}$) for given values of the variables $x, y_1, y_2, \ldots, y_n$.

Its specification is:

```
      SUBROUTINE PEDERV (X, Y, PW)
      DOUBLE PRECISION X, Y(n), PW(n,n)
```

where n is the actual value of N in the call of D02EJF.

1: X -- DOUBLE PRECISION Input

On entry: the value of the independent variable x.

2: Y(*) -- DOUBLE PRECISION array Input

On entry: the value of the variable y_i, for $i=1,2,\ldots,n$.

3: PW(n,*) -- DOUBLE PRECISION array Output

On exit: the value of $\frac{\partial f_i}{\partial y_j}$, for $i,j=1,2,\ldots,n$.

If the user does not wish to supply the Jacobian, the actual argument PEDERV must be the dummy routine D02EJY . (D02EJY is included in the NAG Foundation Library and so need not be supplied by the user. The name may be implementation dependent: see the User's Note for your implementation for details).
PEDERV must be declared as EXTERNAL in the (sub)program from which D02EJF is called. Parameters denoted as Input must not be changed by this procedure.

8: TOL -- DOUBLE PRECISION Input/Output

On entry: TOL must be set to a positive tolerance for controlling the error in the integration. Hence TOL affects the determination of the position where g(x,y) = 0.0, if G is supplied.

D02EJF has been designed so that, for most problems, a reduction in TOL leads to an approximately proportional reduction in the error in the solution. However, the actual relation between TOL and the accuracy achieved cannot be guaranteed. The user is strongly recommended to call D02EJF with more than one value for TOL and to compare the results obtained to estimate their accuracy. In the absence of any prior knowledge, the user might compare the results obtained by calling D02EJF with $TOL=10^{-p}$ and $TOL=10^{-p-1}$ if p correct decimal digits are required in the solution. Constraint: TOL > 0.0. On exit: normally unchanged. However if the range X to XEND is so short that a small change in TOL is unlikely to make any change in the computed solution, then, on return, TOL has its sign changed.

9: RELABS -- CHARACTER*1 Input

On entry: the type of error control. At each step in the numerical solution an estimate of the local error, EST, is made. For the current step to be accepted the following condition must be satisfied:

$$EST=\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(e_i/((\tau)_r*|y_i|+(\tau)_a)\right)^2}\le 1.0$$

where $(\tau)_r$ and $(\tau)_a$ are defined by

RELABS	$(\tau)_r$	$(\tau)_a$
'M'	TOL	TOL
'A'	0.0	TOL
'R'	TOL	ϵ
'D'	TOL	ϵ

where ϵ is a small machine-dependent number and e_i is an estimate of the local error at y_i, computed internally. If the appropriate condition is not satisfied, the step size is reduced and the solution is recomputed on the current step. If the user wishes to measure the error in the computed solution in terms of the number of correct decimal places, then RELABS should be set to 'A' on entry, whereas if the error requirement is in terms of the number

```
     of correct significant digits, then RELABS should be set to
     'R'. If the user prefers a mixed error test, then RELABS
     should be set to 'M', otherwise if the user has no
     preference, RELABS should be set to the default 'D'. Note
     that in this case 'D' is taken to be 'R'. Constraint: RELABS
     = 'A', 'M', 'R' or 'D'.

 10: OUTPUT -- SUBROUTINE, supplied by the user.
                                              External Procedure
     OUTPUT allows the user to have access to intermediate values
     of the computed solution at successive points specified by the
     user. These solution values may be returned to the user via
     the array RESULT if desired (this is a non-standard feature
     added for use with the Axiom system).  OUTPUT is initially
     called by D02EJF with XSOL = X (the initial value of x). The
     user must reset XSOL to the next point where OUTPUT is to be
     called, and so on at each call to OUTPUT. If, after a call
     to OUTPUT, the reset point XSOL is beyond XEND, D02EJF will
     integrate to XEND with no further calls to OUTPUT; if a call
     to OUTPUT is required at the point XSOL = XEND, then XSOL
     must be given precisely the value XEND.

     Its specification is:

          SUBROUTINE OUTPUT(XSOL,Y,COUNT,M,N,RESULT)
          DOUBLE PRECISION Y(N),RESULT(M,N),XSOL
          INTEGER M,N,COUNT

      1:  XSOL -- DOUBLE PRECISION                    Input/Output
          On entry: the current value of the independent
          variable x. On exit: the next value of x at which
          OUTPUT is to be called.

      2:  Y(N) -- DOUBLE PRECISION array                     Input
          On entry: the computed solution at the point XSOL.

      3:  COUNT -- INTEGER                            Input/Output
          On entry: Zero if OUTPUT has not been called before, or
          the previous value of COUNT.
          On exit: A new value of COUNT: this can be used to keep
          track of the number of times OUTPUT has been called.

      4:  M -- INTEGER                                       Input
          On entry: The first dimension of RESULT.

      5:  N -- INTEGER                                       Input
          On entry: The dimension of Y.

      6:  RESULT(M,N) -- DOUBLE PRECISION array      Input/Output
          On entry: the previous contents of RESULT.
```

```
     On exit: RESULT may be used to return the values of the
     intermediate solutions to the user.

OUTPUT must be declared as EXTERNAL in the (sub)program
from which D02EJF is called. Parameters denoted as
Input must not be changed by this procedure.
```

```
11:  G -- DOUBLE PRECISION FUNCTION, supplied by the user.
                                          External Procedure
     G must evaluate the function g(x,y) for specified values x,y
     . It specifies the function g for which the first position x
     where g(x,y) = 0 is to be found.

     Its specification is:

            DOUBLE PRECISION FUNCTION G (X, Y)
            DOUBLE PRECISION X, Y(n)
     where n is the actual value of N in the call of D02EJF.

      1:  X -- DOUBLE PRECISION                             Input
          On entry: the value of the independent variable x.

      2:  Y(*) -- DOUBLE PRECISION array                    Input
          On entry: the value of the variable y , for
                                                i
          i=1,2,...,n.
     If the user does not require the root finding option,
     the actual argument G must be the dummy routine D02EJW.
     (D02EJW is included in the NAG Foundation Library and
     so need not be supplied by the user).
     G must be declared as EXTERNAL in the (sub)program from
     which D02EJF is called. Parameters denoted as Input
     must not be changed by this procedure.

12:  W(IW) -- DOUBLE PRECISION array                    Workspace

13:  IW -- INTEGER                                          Input
     On entry:
     the dimension of the array W as declared in the (sub)program
     from which D02EJF is called.
     Constraint: IW>=(12+N)*N+50.

14:  RESULT(M,N) -- DOUBLE PRECISION array                 Output
     On exit: the computed values of the solution at the points
     given by OUTPUT.

15:  IFAIL -- INTEGER                                Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.
```

```
     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

IFAIL= 1
     On entry TOL <= 0.0,

     or       X = XEND,

     or       N <= 0,

     or       RELABS /= 'M', 'A', 'R', 'D'.

     or       IW<(12+N)*N+50.

IFAIL= 2
     With the given value of TOL, no further progress can be made
     across the integration range from the current point x = X.
     (See Section 5 for a discussion of this error test.) The
     components Y(1),Y(2),...,Y(n) contain the computed values of
     the solution at the current point x = X. If the user has
     supplied G, then no point at which g(x,y) changes sign has
     been located up to the point x = X.

IFAIL= 3
     TOL is too small for D02EJF to take an initial step. X and Y
     (1),Y(2),...,Y(n) retain their initial values.

IFAIL= 4
     XSOL lies behind X in the direction of integration, after
     the initial call to OUTPUT, if the OUTPUT option was
     selected.

IFAIL= 5
     A value of XSOL returned by OUTPUT lies behind the last
     value of XSOL in the direction of integration, if the OUTPUT
     option was selected.

IFAIL= 6
     At no point in the range X to XEND did the function g(x,y)
     change sign, if G was supplied. It is assumed that g(x,y) =
     0 has no solution.
```

IFAIL= 7
A serious error has occurred in an internal call to C05AZF(*). Check all subroutine calls and array dimensions. Seek expert help.

IFAIL= 8
A serious error has occurred in an internal call to D02XKF(*). Check all subroutine calls and array dimensions. Seek expert help.

IFAIL= 9
A serious error has occurred in an internal call to D02NMF(*). Check all subroutine calls and array dimensions. Seek expert help.

7. Accuracy

The accuracy of the computation of the solution vector Y may be controlled by varying the local error tolerance TOL. In general, a decrease in local error tolerance should lead to an increase in accuracy. Users are advised to choose RELABS = 'R' unless they have a good reason for a different choice. It is particularly appropriate if the solution decays.

If the problem is a root-finding one, then the accuracy of the root determined will depend strongly on $\frac{ddg}{ddx}$ and $\frac{ddg}{ddy_i}$, for i=1,2,...,n. Large values for these quantities may imply large errors in the root.

8. Further Comments

If more than one root is required, then to determine the second and later roots D02EJF may be called again starting a short distance past the previously determined roots. Alternatively the user may construct his own root finding code using D02QDF(*) (or the routines of the subchapter D02M-D02N), D02XKF(*) and C05AZF(*).

If it is easy to code, the user should supply the routine PEDERV. However, it is important to be aware that if PEDERV is coded incorrectly, a very inefficient integration may result and possibly even a failure to complete the integration (IFAIL = 2).

9. Example

We illustrate the solution of five different problems. In each case the differential system is the well-known stiff Robertson

```
    problem.

                                    4
                   a'= -0.04a-10 bc
                                    4        7 2
                   b'=  0.04a-10 bc -3*10 b
                                             7 2
                   c'=              3*10 b

    with initial conditions a=1.0, b=c=0.0 at x=0.0. We solve each of
    the following problems with local error tolerances 1.0E-3 and 1.0
    E-4.

    (i)   To integrate to x=10.0 producing output at intervals of 2.0
          until a point is encountered where a=0.9. The Jacobian is
          calculated numerically.

    (ii)  As (i) but with the Jacobian calculated analytically.

    (iii) As (i) but with no intermediate output.

    (iv)  As (i) but with no termination on a root-finding condition.

    (v)   Integrating the equations as in (i) but with no
          intermediate output and no root-finding termination
          condition.

    The example program is not reproduced here. The source code for
    all example programs is distributed with the NAG Foundation
    Library software and should be available on-line.
\end{verbatim}
\endscroll
\end{page}
```

Two-point boundary-value ODE

— nagd.ht —

```
\begin{page}{manpageXXd02gaf}{NAG Documentation: d02gaf}
\beginscroll
\begin{verbatim}


    D02GAF(3NAG)      Foundation Library (12/10/92)      D02GAF(3NAG)
```

```
D02 -- Ordinary Differential Equations                     D02GAF
        D02GAF -- NAG Foundation Library Routine Document

Note: Before using this routine, please read the Users' Note for
your implementation to check implementation-dependent details.
The symbol (*) after a NAG routine name denotes a routine that is
not included in the Foundation Library.

1. Purpose

D02GAF solves the two-point boundary-value problem with assigned
boundary values for a system of ordinary differential equations,
using a deferred correction technique and a Newton iteration.

2. Specification

       SUBROUTINE D02GAF (U, V, N, A, B, TOL, FCN, MNP, X, Y, NP,
      1                   W, LW, IW, LIW, IFAIL)
       INTEGER          N, MNP, NP, LW, IW(LIW), LIW, IFAIL
       DOUBLE PRECISION U(N,2), V(N,2), A, B, TOL, X(MNP), Y
      1                 (N,MNP), W(LW)
       EXTERNAL         FCN

3. Description

D02GAF solves a two-point boundary-value problem for a system of
n differential equations in the interval [a,b]. The system is
written in the form

            y' =f (x,y ,y ,...,y ) , i=1,2,...,n              (1)
              i  i    1  2      n

and the derivatives are evaluated by a subroutine FCN supplied by
the user. Initially, n boundary values of the variables y  must
                                                         i
be specified (assigned), some at a and some at b. The user also
supplies estimates of the remaining n boundary values and all the
boundary values are used in constructing an initial approximation
to the solution. This approximate solution is corrected by a
finite-difference technique with deferred correction allied with
a Newton iteration to solve the finite-difference equations. The
technique used is described fully in Pereyra [1]. The Newton
                                      ddf
                                         i
iteration requires a Jacobian matrix  ---- and this is calculated
                                      ddy
                                         j
```

by numerical differentiation using an algorithm described in Curtis et al [2].

The user supplies an absolute error tolerance and may also supply an initial mesh for the construction of the finite-difference equations (alternatively a default mesh is used). The algorithm constructs a solution on a mesh defined by adding points to the initial mesh. This solution is chosen so that the error is everywhere less than the user's tolerance and so that the error is approximately equidistributed on the final mesh. The solution is returned on this final mesh.

If the solution is required at a few specific points then these should be included in the initial mesh. If on the other hand the solution is required at several specific points then the user should use the interpolation routines provided in Chapter E01 if these points do not themselves form a convenient mesh.

4. References

[1] Pereyra V (1979) PASVA3: An Adaptive Finite-Difference Fortran Program for First Order Nonlinear, Ordinary Boundary Problems. Codes for Boundary Value Problems in Ordinary Differential Equations. Lecture Notes in Computer Science. (ed B Childs, M Scott, J W Daniel, E Denman and P Nelson) 76 Springer-Verlag.

[2] Curtis A R, Powell M J D and Reid J K (1974) On the Estimation of Sparse Jacobian Matrices. J. Inst. Maths Applics. 13 117--119.

5. Parameters

1: U(N,2) -- DOUBLE PRECISION array Input
On entry: U(i,1) must be set to the known (assigned) or estimated values of y_i at a and U(i,2) must be set to the known or estimated values of y_i at b, for i=1,2,...,n.

2: V(N,2) -- DOUBLE PRECISION array Input
On entry: V(i,j) must be set to 0.0 if U(i,j) is a known (assigned) value and to 1.0 if U(i,j) is an estimated value, i=1,2,...,n; j=1,2. Constraint: precisely N of the V(i,j) must be set to 0.0, i.e., precisely N of the U(i,j) must be known values, and these must not be all at a or all at b.

3: N -- INTEGER Input
On entry: the number of equations. Constraint: N >= 2.

4: A -- DOUBLE PRECISION Input
On entry: the left-hand boundary point, a.

5: B -- DOUBLE PRECISION Input
On entry: the right-hand boundary point, b. Constraint: B > A.

6: TOL -- DOUBLE PRECISION Input
On entry: a positive absolute error tolerance. If

$$a=x_1 <x_2 <...<x_{NP} =b$$

is the final mesh, $z_j(x_i)$ is the jth component of the approximate solution at x_i, and $y_j(x)$ is the jth component of the true solution of equation (1) (see Section 3) and the boundary conditions, then, except in extreme cases, it is expected that

$$|z_j(x_i)-y_j(x_i)|<=TOL , i=1,2,...,NP;j=1,2,...,n \qquad (2)$$

Constraint: TOL > 0.0.

7: FCN -- SUBROUTINE, supplied by the user.
External Procedure
FCN must evaluate the functions f_i (i.e., the derivatives y'_i) at the general point x.

Its specification is:

```
SUBROUTINE FCN (X, Y, F)
DOUBLE PRECISION X, Y(n), F(n)
```

where n is the actual value of N in the call of D02GAF.

1: X -- DOUBLE PRECISION Input
On entry: the value of the argument x.

2: Y(*) -- DOUBLE PRECISION array Input
On entry: the value of the argument y_i, for i=1,2,...,n.

3: F(*) -- DOUBLE PRECISION array Output
On exit: the values of f_i, for i=1,2,...,n.

FCN must be declared as EXTERNAL in the (sub)program from which D02GAF is called. Parameters denoted as Input must not be changed by this procedure.

```
 8:  MNP -- INTEGER                                          Input
     On entry: the maximum permitted number of mesh points.
     Constraint: MNP >= 32.

 9:  X(MNP) -- DOUBLE PRECISION array                 Input/Output
     On entry: if NP >= 4 (see NP below), the first NP elements
     must define an initial mesh. Otherwise the elements of X
     need not be set. Constraint:
                A=X(1)<X(2)<...<X(NP)=B for NP>=4              (3)
     On exit: X(1),X(2),...,X(NP) define the final mesh (with
     the returned value of NP) satisfying the relation (3).

10:  Y(N,MNP) -- DOUBLE PRECISION array                     Output
     On exit: the approximate solution z (x ) satisfying (2), on
                                        j  i
     the final mesh, that is
              Y(j,i)=z (x ) , i=1,2,...,NP;j=1,2,...,n,
                      j  i
     where NP is the number of points in the final mesh.

     The remaining columns of Y are not used.

11:  NP -- INTEGER                                    Input/Output
     On entry: determines whether a default or user-supplied
     mesh is used. If NP = 0, a default value of 4 for NP and a
     corresponding equispaced mesh X(1),X(2),...,X(NP) are used.
     If NP >= 4, then the user must define an initial mesh using
     the array X as described. Constraint: NP = 0 or 4 <= NP <=
     MNP. On exit: the number of points in the final (returned)
     mesh.

12:  W(LW) -- DOUBLE PRECISION array                     Workspace

13:  LW -- INTEGER                                           Input
     On entry: the length of the array W as declared in the
                                                      2         2
     calling (sub)program. Constraint: LW>=MNP*(3N +6N+2)+4N +4N

14:  IW(LIW) -- INTEGER array                            Workspace

15:  LIW -- INTEGER                                          Input
     On entry: the length of the array IW as declared in the
                                                            2
     calling (sub)program. Constraint: LIW>=MNP*(2N+1)+N +4N+2.

16:  IFAIL -- INTEGER                                 Input/Output
     For this routine, the normal use of IFAIL is extended to
     control the printing of error and warning messages as well
     as specifying hard or soft failure (see the Essential
     Introduction).
```

```
     Before entry, IFAIL must be set to a value with the decimal
     expansion cba, where each of the decimal digits c, b and a
     must have a value of 0 or 1.
     a=0   specifies hard failure, otherwise soft failure;

     b=0   suppresses error messages, otherwise error messages
           will be printed (see Section 6);

     c=0   suppresses warning messages, otherwise warning
           messages will be printed (see Section 6).
     The recommended value for inexperienced users is 110 (i.e.,
     hard failure with all messages printed).

     Unless the routine detects an error (see Section 6), IFAIL
     contains 0 on exit.

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

IFAIL= 1
     One or more of the parameters N, TOL, NP, MNP, LW or LIW has
     been incorrectly set, or B <= A, or the condition (3) on X
     is not satisfied, or the number of known boundary values
     (specified by V) is not N.

IFAIL= 2
     The Newton iteration has failed to converge. This could be
     due to there being too few points in the initial mesh or to
     the initial approximate solution being too inaccurate. If
     this latter reason is suspected the user should use
     subroutine D02RAF instead. If the warning 'Jacobian matrix
     is singular' is printed this could be due to specifying zero
     estimated boundary values and these should be varied. This
     warning could also be printed in the unlikely event of the
     Jacobian matrix being calculated inaccurately. If the user
     cannot make changes to prevent the warning then subroutine
     D02RAF should be used.

IFAIL= 3
     The Newton iteration has reached round-off level. It could
     be, however, that the answer returned is satisfactory. This
     error might occur if too much accuracy is requested.

IFAIL= 4
     A finer mesh is required for the accuracy requested; that is
```

MNP is not large enough.

IFAIL= 5
A serious error has occurred in a call to D02GAF. Check all array subscripts and subroutine parameter lists in calls to D02GAF. Seek expert help.

7. Accuracy

The solution returned by the routine will be accurate to the user's tolerance as defined by the relation (2) except in extreme circumstances. If too many points are specified in the initial mesh, the solution may be more accurate than requested and the error may not be approximately equidistributed.

8. Further Comments

The time taken by the routine depends on the difficulty of the problem, the number of mesh points used (and the number of different meshes used), the number of Newton iterations and the number of deferred corrections.

The user is strongly recommended to set IFAIL to obtain self-explanatory error messages, and also monitoring information about the course of the computation. The user may select the channel numbers on which this output is to appear by calls of X04AAF (for error messages) or X04ABF (for monitoring information) - see Section 9 for an example. Otherwise the default channel numbers will be used, as specified in the implementation document.

A common cause of convergence problems in the Newton iteration is the user specifying too few points in the initial mesh. Although the routine adds points to the mesh to improve accuracy it is unable to do so until the solution on the initial mesh has been calculated in the Newton iteration.

If the user specifies zero known and estimated boundary values, the routine constructs a zero initial approximation and in many cases the Jacobian is singular when evaluated for this approximation, leading to the breakdown of the Newton iteration.

The user may be unable to provide a sufficiently good choice of initial mesh and estimated boundary values, and hence the Newton iteration may never converge. In this case the continuation facility provided in D02RAF is recommended.

In the case where the user wishes to solve a sequence of similar problems, the final mesh from solving one case is strongly recommended as the initial mesh for the next.

```
          9. Example

          We solve the differential equation

                                                        2
                                y'''=-yy''-(beta)(1-y' )

          with boundary conditions

                                      y(0)=y'(0)=0,

                                        y'(10)=1

          for (beta)=0.0 and (beta)=0.2 to an accuracy specified by TOL=1.0
          E-3. We solve first the simpler problem with (beta)=0.0 using an
          equispaced mesh of 26 points and then we solve the problem with
          (beta)=0.2 using the final mesh from the first problem.

          Note the call to X04ABF prior to the call to D02GAF.

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Two-point boundary value ODE with deferred correction

— nagd.ht —

```
\begin{page}{manpageXXd02gbf}{NAG Documentation: d02gbf}
\beginscroll
\begin{verbatim}



     D02GBF(3NAG)      Foundation Library (12/10/92)      D02GBF(3NAG)



          D02 -- Ordinary Differential Equations                    D02GBF
                  D02GBF -- NAG Foundation Library Routine Document
```

Note: Before using this routine, please read the Users' Note for your implementation to check implementation-dependent details. The symbol (*) after a NAG routine name denotes a routine that is not included in the Foundation Library.

1. Purpose

D02GBF solves a general linear two-point boundary value problem for a system of ordinary differential equations using a deferred correction technique.

2. Specification

```
       SUBROUTINE D02GBF (A, B, N, TOL, FCNF, FCNG, C, D, GAM,
      1                   MNP, X, Y, NP, W, LW, IW, LIW, IFAIL)
       INTEGER          N, MNP, NP, LW, IW(LIW), LIW, IFAIL
       DOUBLE PRECISION A, B, TOL, C(N,N), D(N,N), GAM(N), X(MNP),
      1                 Y(N,MNP), W(LW)
       EXTERNAL         FCNF, FCNG
```

3. Description

D02GBF solves the linear two-point boundary value problem for a system of n ordinary differential equations in the interval

[a,b]. The system is written in the form

y'=F(x)y+g(x) (1)

and the boundary conditions are written in the form

Cy(a)+Dy(b)=(gamma) (2)

Here F(x), C and D are n by n matrices, and g(x) and (gamma) are n-component vectors. The approximate solution to (1) and (2) is found using a finite-difference method with deferred correction. The algorithm is a specialisation of that used in subroutine D02RAF which solves a nonlinear version of (1) and (2). The nonlinear version of the algorithm is described fully in Pereyra [1].

The user supplies an absolute error tolerance and may also supply an initial mesh for the construction of the finite-difference equations (alternatively a default mesh is used). The algorithm constructs a solution on a mesh defined by adding points to the initial mesh. This solution is chosen so that the error is everywhere less than the user's tolerance and so that the error is approximately equidistributed on the final mesh. The solution is returned on this final mesh.

If the solution is required at a few specific points then these should be included in the initial mesh. If, on the other hand, the solution is required at several specific points, then the user should use the interpolation routines provided in Chapter E01 if these points do not themselves form a convenient mesh.

4. References

[1] Pereyra V (1979) PASVA3: An Adaptive Finite-Difference Fortran Program for First Order Nonlinear, Ordinary Boundary Problems. Codes for Boundary Value Problems in Ordinary Differential Equations. Lecture Notes in Computer Science. (ed B Childs, M Scott, J W Daniel, E Denman and P Nelson) 76 Springer-Verlag.

5. Parameters

1: A -- DOUBLE PRECISION Input

On entry: the left-hand boundary point, a.

2: B -- DOUBLE PRECISION Input

On entry: the right-hand boundary point, b. Constraint: B > A.

3: N -- INTEGER Input

On entry: the number of equations; that is n is the order of system (1). Constraint: N >= 2.

4: TOL -- DOUBLE PRECISION Input

On entry: a positive absolute error tolerance. If

$$a=x_1<x_2<\ldots<x_{NP}=b$$

is the final mesh, z(x) is the approximate solution from D02GBF and y(x) is the true solution of equations (1) and (2) then, except in extreme cases, it is expected that

$$||z-y||<=TOL \qquad (3)$$

where

$$||u||=\max_{1<=i<=N} \max_{1<=j<=NP} |u_i(x_j)|.$$

Constraint: TOL > 0.0.

5: FCNF -- SUBROUTINE, supplied by the user. External Procedure

FCNF must evaluate the matrix F(x) in (1) at a general point x.

Its specification is:

```
SUBROUTINE FCN (X, F)
DOUBLE PRECISION X, F(n,n)
```

```
    where n is the actual value of N in the call of D02GBF.

     1:  X -- DOUBLE PRECISION                             Input
         On entry: the value of the independent variable x.

     2:  F(n,n) -- DOUBLE PRECISION array                 Output
         On exit: the (i,j)th element of the matrix F(x), for
         i,j=1,2,...,n. (See Section 9 for an example.)
    FCN must be declared as EXTERNAL in the (sub)program
    from which D02GBF is called. Parameters denoted as
    Input must not be changed by this procedure.

6:  FCNG -- SUBROUTINE, supplied by the user.
                                          External Procedure
    FCNG must evaluate the vector g(x) in (1) at a general point
    x.

    Its specification is:

           SUBROUTINE FCNG (X, G)
           DOUBLE PRECISION X, G(n)
    where n is the actual value of N in the call of D02GBF.

     1:  X -- DOUBLE PRECISION                             Input
         On entry: the value of the independent variable x.

     2:  G(*) -- DOUBLE PRECISION array                   Output
         On exit: the ith element of the vector g(x), for
         i=1,2,...,n. (See Section Section 9 for an example.)
    FCNG must be declared as EXTERNAL in the (sub)program
    from which D02GBF is called. Parameters denoted as
    Input must not be changed by this procedure.

7:  C(N,N) -- DOUBLE PRECISION array              Input/Output

8:  D(N,N) -- DOUBLE PRECISION array              Input/Output

9:  GAM(N) -- DOUBLE PRECISION array              Input/Output
    On entry: the arrays C and D must be set to the matrices C
    and D in (2). GAM must be set to the vector (gamma) in (2).
    On exit: the rows of C and D and the components of GAM are
    re-ordered so that the boundary conditions are in the order:
      (i)   conditions on y(a) only;

      (ii)  condition involving y(a) and y(b); and

      (iii) conditions on y(b) only.

    The routine will be slightly more efficient if the arrays C,
    D and GAM are ordered in this way before entry, and in this
```

event they will be unchanged on exit.

Note that the problems (1) and (2) must be of boundary value type, that is neither C nor D may be identically zero. Note also that the rank of the matrix [C,D] must be n for the problem to be properly posed. Any violation of these conditions will lead to an error exit.

10: MNP -- INTEGER Input
On entry: the maximum permitted number of mesh points.
Constraint: MNP >= 32.

11: X(MNP) -- DOUBLE PRECISION array Input/Output
On entry: if NP >= 4 (see NP below), the first NP elements must define an initial mesh. Otherwise the elements of x need not be set. Constraint:

$$A=X(1)<X(2)<...<X(NP)=B, \text{ for } NP>=4. \qquad (4)$$

On exit: X(1),X(2),...,X(NP) define the final mesh (with the returned value of NP) satisfying the relation (4).

12: Y(N,MNP) -- DOUBLE PRECISION array Output
On exit: the approximate solution z(x) satisfying (3), on the final mesh, that is

$$Y(j,i)=z_j(x_i) \text{ , } i=1,2,...,NP;j=1,2,...,n$$

where NP is the number of points in the final mesh.

The remaining columns of Y are not used.

13: NP -- INTEGER Input/Output
On entry: determines whether a default mesh or user-supplied mesh is used. If NP = 0, a default value of 4 for NP and a corresponding equispaced mesh X(1),X(2),...,X(NP) are used. If NP >= 4, then the user must define an initial mesh X as in (4) above. On exit: the number of points in the final (returned) mesh.

14: W(LW) -- DOUBLE PRECISION array Workspace

15: LW -- INTEGER Input
On entry: the length of the array W, Constraint:
$LW>=MNP*(3N^2+5N+2)+3N^2+5N$.

16: IW(LIW) -- INTEGER array Workspace

17: LIW -- INTEGER Input
On entry: the length of the array IW. Constraint:
LIW>=MNP*(2N+1)+N.

```
18:  IFAIL -- INTEGER                                Input/Output
     For this routine, the normal use of IFAIL is extended to
     control the printing of error and warning messages as well
     as specifying hard or soft failure (see the Essential
     Introduction).

     Before entry, IFAIL must be set to a value with the decimal
     expansion cba, where each of the decimal digits c, b and a
     must have a value of 0 or 1.
     a=0   specifies hard failure, otherwise soft failure;

     b=0   suppresses error messages, otherwise error messages
           will be printed (see Section 6);

     c=0   suppresses warning messages, otherwise warning
           messages will be printed (see Section 6).
     The recommended value for inexperienced users is 110 (i.e.,
     hard failure with all messages printed).

     Unless the routine detects an error (see Section 6), IFAIL
     contains 0 on exit.

6. Error Indicators and Warnings

Errors detected by the routine:

For each error, an explanatory error message is output on the
current error message unit (as defined by X04AAF), unless
suppressed by the value of IFAIL on entry.

IFAIL= 1
     One or more of the parameters N, TOL, NP, MNP, LW or LIW is
     incorrectly set, B <= A or the condition (4) on X is not
     satisfied.

IFAIL= 2
     There are three possible reasons for this error exit to be
     taken:
       (i)  one of the matrices C or D is identically zero (that
            is the problem is of initial value and not boundary
            value type). In this case, IW(1) = 0 on exit;

       (ii) a row of C and the corresponding row of D are
            identically zero (that is the boundary conditions are
            rank deficient). In this case, on exit IW(1) contains
            the index of the first such row encountered; and

       (iii more than n of the columns of the n by 2n matrix [C,D
       )    ] are identically zero (that is the boundary
            conditions are rank deficient). In this case, on exit
```

IW(1) contains minus the number of non-identically zero columns.

IFAIL= 3

The routine has failed to find a solution to the specified accuracy. There are a variety of possible reasons including:

(i) the boundary conditions are rank deficient, which may be indicated by the message that the Jacobian is singular. However this is an unlikely explanation for the error exit as all rank deficient boundary conditions should lead instead to error exits with either IFAIL = 2 or IFAIL = 5; see also (iv) below;

(ii) not enough mesh points are permitted in order to attain the required accuracy. This is indicated by NP = MNP on return from a call to D02GBF. This difficulty may be aggravated by a poor initial choice of mesh points;

(iii) the accuracy requested cannot be attained on the computer being used; and

(iv) an unlikely combination of values of F(x) has led to a singular Jacobian. The error should not persist if more mesh points are allowed.

IFAIL= 4

A serious error has occurred in a call to D02GBF. Check all array subscripts and subroutine parameter lists in calls to D02GBF. Seek expert help.

IFAIL= 5

There are two possible reasons for this error exit which occurs when checking the rank of the boundary conditions by reduction to a row echelon form:

(i) at least one row of the n by 2n matrix [C,D] is a linear combination of the other rows and hence the boundary conditions are rank deficient. The index of the first such row encountered is given by IW(1) on exit; and

(ii) as (i) but the rank deficiency implied by this error exit has only been determined up to a numerical tolerance. Minus the index of the first such row encountered is given by IW(1) on exit.

In case (ii) above there is some doubt as to the rank deficiency of the boundary conditions. However even if the boundary conditions are not rank deficient they are not posed in a suitable form for use with this routine.

```
     For example, if
                                                  ((gamma) )
            (1 0         )     (1 0)              (       1)
         C=(1 (epsilon)) , D=(1 0) , (gamma)=((gamma) )
                                                  (       2)
     and (epsilon) is small enough, this error exit is likely to
     be taken. A better form for the boundary conditions in this
     case would be
                                 ((gamma)                      )
                                 (       1                     )
       (1 0)     (1 0)           (         -1                  )
     C=(0 1) , D=(0 0) , (gamma)=((epsilon)  ((gamma) -(gamma) ))
                                 (                  2        1 )
```

7. Accuracy

The solution returned by the routine will be accurate to the user's tolerance as defined by the relation (3) except in extreme circumstances. If too many points are specified in the initial mesh, the solution may be more accurate than requested and the error may not be approximately equidistributed.

8. Further Comments

The time taken by the routine depends on the difficulty of the problem, the number of mesh points (and meshes) used and the number of deferred corrections.

The user is strongly recommended to set IFAIL to obtain self-explanatory error messages, and also monitoring information about the course of the computation. The user may select the channel numbers on which this output is to appear by calls of X04AAF (for error messages) or X04ABF (for monitoring information) - see Section 9 for an example. Otherwise the default channel numbers will be used, as specified in the implementation document.

In the case where the user wishes to solve a sequence of similar problems, the use of the final mesh from one case is strongly recommended as the initial mesh for the next.

9. Example

We solve the problem (written as a first order system)

```
                         (epsilon)y''+y'=0
```

with boundary conditions

```
                          y(0)=0,  y(1)=1
```

```
                                   -1                  -2
          for the cases (epsilon)=10   and (epsilon)=10   using the default
          initial mesh in the first case, and the final mesh of the first
          case as initial mesh for the second (more difficult) case. We
          give the solution and the error at each mesh point to illustrate
          the accuracy of the method given the accuracy request TOL=1.0E-3.

          Note the call to X04ABF prior to the call to D02GBF.

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Eignevalue of regular singular 2nd-order Sturm-Liouville

— nagd.ht —

```
\begin{page}{manpageXXd02kef}{NAG Documentation: d02kef}
\beginscroll
\begin{verbatim}



D02KEF(3NAG)                 D02KEF                  D02KEF(3NAG)



     D02 -- Ordinary Differential Equations                     D02KEF
             D02KEF -- NAG Foundation Library Routine Document

     Note: Before using this routine, please read the Users' Note for
     your implementation to check implementation-dependent details.
     The symbol (*) after a NAG routine name denotes a routine that is
     not included in the Foundation Library.

     1. Purpose

     D02KEF finds a specified eigenvalue of a regular singular second-
     order Sturm-Liouville system on a finite or infinite range, using
     a Pruefer transformation and a shooting method. It also reports
     values of the eigenfunction and its derivatives. Provision is
```

made for discontinuities in the coefficient functions or their derivatives.

2. Specification

```
       SUBROUTINE D02KEF (XPOINT, M, MATCH, COEFFN, BDYVAL, K,
      1                   TOL, ELAM, DELAM, HMAX, MAXIT, MAXFUN,
      2                   MONIT, REPORT, IFAIL)
       INTEGER          M, MATCH, K, MAXIT, MAXFUN, IFAIL
       DOUBLE PRECISION XPOINT(M), TOL, ELAM, DELAM, HMAX(2,M)
       EXTERNAL         COEFFN, BDYVAL, MONIT, REPORT
```

3. Description

D02KEF has essentially the same purpose as D02KDF(*) with minor modifications to enable values of the eigenfunction to be obtained after convergence to the eigenvalue has been achieved.

```
                                       ~~~~~~~~
It first finds a specified eigenvalue (lambda) of a Sturm-
```

Liouville system defined by a self-adjoint differential equation of the second-order

```
                (p(x)y')'+q(x;(lambda))y=0, a<x<b
```

together with the appropriate boundary conditions at the two (finite or infinite) end-points a and b. The functions p and q, which are real-valued, must be defined by a subroutine COEFFN. The boundary conditions must be defined by a subroutine BDYVAL, and, in the case of a singularity at a or b, take the form of an asymptotic formula for the solution near the relevant end-point.

```
                                  ~~~~~~~~
When the final estimate (lambda)=(lambda) of the eigenvalue has
```

been found, the routine integrates the differential equation once more with that value of (lambda), and with initial conditions chosen so that the integral

```
                  b
                  /    2    ddq
               S= |y(x)  ----------(x;(lambda))dx
                  /      dd(lambda)
                  a
```

is approximately one. When q(x;(lambda)) is of the form (lambda)w(x)+q(x), which is the most common case, S represents the square of the norm of y induced by the inner product

```
                            b
                            /
```

```
                    <f,g>= |f(x)g(x)w(x)dx,
                           /
                           a
```

with respect to which the eigenfunctions are mutually orthogonal. This normalisation of y is only approximate, but experience shows that S generally differs from unity by only one or two per cent.

During this final integration the REPORT routine supplied by the user is called at each integration mesh point x. Sufficient information is returned to permit the user to compute y(x) and y'(x) for printing or plotting. For reasons described in Section 8.2, D02KEF passes across to REPORT, not y and y', but the Pru efer variables (beta), (phi) and (rho) on which the numerical method is based. Their relationship to y and y' is given by the equations

```
         ______   ( (rho))   ( (phi))
p(x)y'=\/(beta)exp( -----)cos( -----);
                  (   2  )   (   2  )

           1       ( (rho))   ( (phi))
     y= --------exp( -----)sin( -----).
          ______   (   2  )   (   2  )
        \/(beta)
```

For the theoretical basis of the numerical method to be valid, the following conditions should hold on the coefficient functions:

(a)   p(x) must be non-zero and of one sign throughout the interval (a,b); and,

```
          ddq
(b)    ---------- must be of one sign throughout (a,b) for all
       dd(lambda)
      relevant values of (lambda), and must not be identically
      zero as x varies, for any (lambda).
```

Points of discontinuity in the functions p and q or their derivatives are allowed, and should be included as 'break-points' in the array XPOINT.

A good account of the theory of Sturm-Liouville systems, with some description of Pruefer transformations, is given in Birkhoff and Rota [4], Chapter X. An introduction for the user of Pruefer transformations for the numerical solution of eigenvalue problems arising from physics and chemistry is Bailey [2].

The scaled Pruefer method is fairly recent, and is described in a short note by Pryce [6] and in some detail in the technical report [5].

4. References

[1] Abramowitz M and Stegun I A (1968) Handbook of Mathematical Functions. Dover Publications.

[2] Bailey P B (1966) Sturm-Liouville Eigenvalues via a Phase Function. SIAM J. Appl. Math . 14 242--249.

[3] Banks D O and Kurowski I (1968) Computation of Eigenvalues of Singular Sturm-Liouville Systems. Math. Computing. 22 304--310.

[4] Birkhoff G and Rota G C (1962) Ordinary Differential Equations. Ginn & Co., Boston and New York.

[5] Pryce J D (1981) Two codes for Sturm-Liouville problems. Technical Report CS-81-01. Dept of Computer Science, Bristol University .

[6] Pryce J D and Hargrave B A (1977) The Scale Pruefer Method for one-parameter and multi-parameter eigenvalue problems in ODEs. Inst. Math. Appl., Numerical Analysis Newsletter. 1(3)

5. Parameters

1: XPOINT(M) -- DOUBLE PRECISION array Input

On entry: the points where the boundary conditions computed by BDYVAL are to be imposed, and also any break-points, i.e., XPOINT(1) to XPOINT(m) must contain values $x_1,...,x_m$ such that

$$x_1 <= x_2 < x_3 < ... < x_{m-1} <= x_m$$

with the following meanings:

(a) x_1 and x_m are the left and right end-points, a and b, of the domain of definition of the Sturm-Liouville system if these are finite. If either a or b is infinite, the corresponding value x_1 or x_m may be a more-or-less arbitrarily 'large' number of appropriate sign.

(b) x_2 and x_{m-1} are the Boundary Matching Points (BMP's), that is the points at which the left and right boundary conditions computed in BDYVAL are imposed.

If the left-hand end-point is a regular point then the user should set $x_2=x_1$ (=a), while if it is a singular point the user must set $x_2>x_1$. Similarly $x_{m-1}=x_m$ (=b) if the right-hand end-point is regular, and $x_{m-1}<x_m$ if it is singular.

(c) The remaining m-4 points $x_3,...,x_{m-2}$, if any, define 'break-points' which divide the interval $[x_2,x_{m-1}]$ into m-3 sub-intervals

$$i_1=[x_2,x_3],...,i_{m-3}=[x_{m-2},x_{m-1}]$$

Numerical integration of the differential equation is stopped and restarted at each break-point. In simple cases no break-points are needed. However if p(x) or q(x;(lambda)) are given by different formulae in different parts of the range, then integration is more efficient if the range is broken up by break-points in the appropriate way. Similarly points where any jumps occur in p(x) or q(x;(lambda)), or in their derivatives up to the fifth order, should appear as break-points.

Constraint: X(1) <= X(2) < ... < X(M-1) <= X(M).

2: M -- INTEGER Input
On entry: the number of points in the array XPOINT.
Constraint: M >= 4.

3: MATCH -- INTEGER Input/Output
On entry: MATCH must be set to the index of the 'break-point ' to be used as the matching point (see Section 8.3). If MATCH is set to a value outside the range [2,m-1] then a default value is taken, corresponding to the break-point nearest the centre of the interval [XPOINT(2),XPOINT(m-1)].
On exit: the index of the break-point actually used as the matching point.

4: COEFFN -- SUBROUTINE, supplied by the user.
External Procedure
COEFFN must compute the values of the coefficient functions p(x) and q(x;(lambda)) for given values of x and (lambda). Section 3 states conditions which p and q must satisfy.

Its specification is:

```
SUBROUTINE COEFFN (P, Q, DQDL, X, ELAM, JINT)
DOUBLE PRECISION P, Q, DQDL, X, ELAM
INTEGER          JINT
```

1: P -- DOUBLE PRECISION Output
On exit: the value of p(x) for the current value of x.

2: Q -- DOUBLE PRECISION Output
On exit: the value of q(x;(lambda)) for the current value of x and the current trial value of (lambda).

3: DQDL -- DOUBLE PRECISION Output
On exit: the value of $\frac{ddq}{dd(lambda)}(x;(lambda))$ for the current value of x and the current trial value of (lambda). However DQDL is only used in error estimation and an approximation (say to within 20%) will suffice.

4: X -- DOUBLE PRECISION Input
On entry: the current value of x.

5: ELAM -- DOUBLE PRECISION Input
On entry: the current trial value of the eigenvalue parameter (lambda).

6: JINT -- INTEGER Input
On entry: the index j of the sub-interval i_j (see specification of XPOINT) in which x lies.

See Section 8.4 and Section 9 for examples.

COEFFN must be declared as EXTERNAL in the (sub)program from which D02KEF is called. Parameters denoted as Input must not be changed by this procedure.

5: BDYVAL -- SUBROUTINE, supplied by the user. External Procedure
BDYVAL must define the boundary conditions. For each end-point, BDYVAL must return (in YL or YR) values of y(x) and p(x)y'(x) which are consistent with the boundary conditions at the end-points; only the ratio of the values matters. Here x is a given point (XL or XR) equal to, or close to, the end-point.

For a regular end-point (a, say), x=a; and a boundary condition of the form

$$c_1 y(a)+c_2 y'(a)=0$$

can be handled by returning constant values in YL, e.g. YL(1)=c_2 and YL(2)=$-c_1$ p(a).

For a singular end-point however, YL(1) and YL(2) will in general be functions of XL and ELAM, and YR(1) and YR(2) functions of XR and ELAM, usually derived analytically from a power-series or asymptotic expansion. Examples are given in Section 8.5 Section 9.

Its specification is:

```
      SUBROUTINE BDYVAL (XL, XR, ELAM, YL, YR))
      DOUBLE PRECISION XL, XR, ELAM, YL(3), YR(3)
```

1: XL -- DOUBLE PRECISION Input
On entry: if a is a regular end-point of the system (so that $a=x_1=x_2$), then XL contains a. If a is a singular point (so that $a<=x_1<x_2$), then XL contains a point x such that $x_1<x<=x_2$).

2: XR -- DOUBLE PRECISION Input
On entry: if b is a regular end-point of the system (so that $x_{m-1}=x_m=b$), then XR contains b. If b is a singular point (so that $x_{m-1}<x_m<=b$), then XR contains a point x such that $x_{m-1}<=x<x_m$.

3: ELAM -- DOUBLE PRECISION Input
On entry: the current trial value of (lambda).

4: YL(3) -- DOUBLE PRECISION array Output
On exit: YL(1) and YL(2) should contain values of y(x) and p(x)y'(x) respectively (not both zero) which are consistent with the boundary condition at the left-hand end-point, given by x = XL. YL(3) should not be set.

5: YR(3) -- DOUBLE PRECISION array Output
On exit: YR(1) and YR(2) should contain values of y(x) and p(x)y'(x) respectively (not both zero) which are consistent with the boundary condition at the right-hand end-point, given by x = XR. YR(3) should not be set.

BDYVAL must be declared as EXTERNAL in the (sub)program

from which D02KEF is called. Parameters denoted as
Input must not be changed by this procedure.

```
6:  K -- INTEGER                                         Input
    On entry: the index k of the required eigenvalue when the
    eigenvalues are ordered
    (lambda) <(lambda) <(lambda) <...<(lambda) <....
            0        1        2            k
    Constraint: K >= 0.

7:  TOL -- DOUBLE PRECISION                              Input
    On entry: the tolerance parameter which determines the
    accuracy of the computed eigenvalue. The error estimate held
    in DELAM on exit satisfies the mixed absolute/relative error
    test
                  DELAM<=TOL*max(1.0,|ELAM|)                (*)
    where ELAM is the final estimate of the eigenvalue. DELAM is
    usually somewhat smaller than the right-hand side of (*) but
    not several orders of magnitude smaller. Constraint: TOL >
    0.0.

8:  ELAM -- DOUBLE PRECISION                      Input/Output
                                                  ~~~~~~~~

    On entry: an initial estimate of the eigenvalue (lambda).
    On exit: the final computed estimate, whether or not an
    error occurred.

9:  DELAM -- DOUBLE PRECISION                     Input/Output
    On entry: an indication of the scale of the problem in the
    (lambda)-direction. DELAM holds the initial 'search step'
    (positive or negative). Its value is not critical but the
    first two trial evaluations are made at ELAM and ELAM +
    DELAM, so the routine will work most efficiently if the
    eigenvalue lies between these values. A reasonable choice
    (if a closer bound is not known) is half the distance
    between adjacent eigenvalues in the neighbourhood of the one
    sought. In practice, there will often be a problem, similar
    to the one in hand but with known eigenvalues, which will
    help one to choose initial values for ELAM and DELAM.

    If DELAM = 0.0 on entry, it is given the default value of
    0.25*max(1.0,|ELAM|). On exit: with IFAIL = 0, DELAM holds
    an estimate of the absolute error in the computed
                       ~~~~~~~~
    eigenvalue, that is |(lambda)-ELAM|~=DELAM. (In Section 8.2
    we discuss the assumptions under which this is true.) The
    true error is rarely more than twice, or less than a tenth,
    of the estimated error.

    With IFAIL /= 0, DELAM may hold an estimate of the error, or
```

its initial value, depending on the value of IFAIL. See Section 6 for further details.

10: HMAX(2,M) -- DOUBLE PRECISION array Input/Output
On entry: HMAX(1,j) a maximum step size to be used by the differential equation code in the jth sub-interval i_j (as described in the specification of parameter XPOINT), for j=1,2,...,m-3. If it is zero the routine generates a maximum step size internally.

It is recommended that HMAX(1,j) be set to zero unless the coefficient functions p and q have features (such as a narrow peak) within the jth sub-interval that could be ' missed' if a long step were taken. In such a case HMAX(1,j) should be set to about half the distance over which the feature should be observed. Too small a value will increase the computing time for the routine. See Section 8 for further suggestions.

The rest of the array is used as workspace. On exit: HMAX(1,m-1) and HMAX(1,m) contain the sensitivity coefficients $(sigma)_l$,$(sigma)_r$, described in Section 8.6. Other entries contain diagnostic output in case of an error (see Section 6).

11: MAXIT -- INTEGER Input/Output
On entry: a bound on n_r, the number of root-finding iterations allowed, that is the number of trial values of (lambda) that are used. If MAXIT <= 0, no such bound is assumed. (See also under MAXFUN.) Suggested value: MAXIT = 0. On exit: MAXIT will have been decreased by the number of iterations actually performed, whether or not it was positive on entry.

12: MAXFUN -- INTEGER Input
On entry: a bound on n_f, the number of calls to COEFFN made in any one root-finding iteration. If MAXFUN <= 0, no such bound is assumed. Suggested value: MAXFUN = 0.

MAXFUN and MAXIT may be used to limit the computational cost of a call to D02KEF, which is roughly proportional to n_r*n_f.

13: MONIT -- SUBROUTINE, supplied by the user.
External Procedure
MONIT is called by D02KEF at the end of each root-finding

```
iteration and allows the user to monitor the course of the
computation by printing out the parameters (see Section 8
for an example).
If no monitoring is required, the dummy subroutine D02KAY
may be used. (D02KAY is included in the NAG Foundation
Library).

Its specification is:

       SUBROUTINE MONIT (MAXIT, IFLAG, ELAM, FINFO)
       INTEGER          MAXIT, IFLAG
       DOUBLE PRECISION ELAM, FINFO(15)

 1:  MAXIT -- INTEGER                                  Input
     On entry: the current value of the parameter MAXIT of
     D02KEF; this is decreased by one at each iteration.

 2:  IFLAG -- INTEGER                                  Input
     On entry: IFLAG describes what phase the computation is
     in, as follows:
     IFLAG < 0
           an error occurred in the computation of the '
           miss-distance' at this iteration;

           an error exit from D02KEF with IFAIL =-IFLAG will
           follow.

     IFLAG = 1
           the routine is trying to bracket the eigenvalue
           ~~~~~~~~
           (lambda).

     IFLAG = 2
           the routine is converging to the eigenvalue
           ~~~~~~~~
           (lambda) (having already bracketed it).

 3:  ELAM -- DOUBLE PRECISION                          Input
     On entry: the current trial value of (lambda).

 4:  FINFO(15) -- DOUBLE PRECISION array               Input
     On entry: information about the behaviour of the
     shooting method, and diagnostic information in the case
     of errors. It should not normally be printed in full if
     no error has occurred (that is, if IFLAG > 0), though
     the first few components may be of interest to the
     user. In case of an error (IFLAG < 0) all the
     components of FINFO should be printed. The contents of
     FINFO are as follows:
```

FINFO(1): the current value of the 'miss-distance' or ' residual' function f((lambda)) on which the shooting method is based. FINFO(1) is set to zero if IFLAG < 0.

FINFO(2): an estimate of the quantity dd(lambda) defined as follows. Consider the perturbation in the miss-distance f((lambda)) that would result if the local error, in the solution of the differential equation, were always positive and equal to its maximum permitted value. Then dd(lambda) is the perturbation in (lambda) that would have the same effect on f((lambda)) . Thus, at the zero of f((lambda)),|dd(lambda)| is an approximate bound on the perturbation of the zero (that is the eigenvalue) caused by errors in numerical solution. If dd(lambda) is very large then it is possible that there has been a programming error in COEFFN such that q is independent of (lambda). If this is the case, an error exit with IFAIL = 5 should follow. FINFO(2) is set to zero if IFLAG < 0.

FINFO(3): the number of internal iterations, using the same value of (lambda) and tighter accuracy tolerances, needed to bring the accuracy (that is the value of dd(lambda)) to an acceptable value. Its value should normally be 1.0, and should almost never exceed 2.0.

FINFO(4): the number of calls to COEFFN at this iteration.

FINFO(5): the number of successful steps taken by the internal differential equation solver at this iteration. A step is successful if it is used to advance the integration (cf. COUT(8) in specification of D02PAF(*)).

FINFO(6): the number of unsuccessful steps used by the internal integrator at this iteration (cf. COUT(9) in specification of D02PAF(*)).

FINFO(7): the number of successful steps at the maximum step size taken by the internal integrator at this iteration (cf. COUT(3) in specification of D02PAF(*)).

FINFO(8): is not used.

FINFO(9) to FINFO(15): set to zero, unless IFLAG < 0 in which case they hold the following values describing the point of failure:

FINFO(9): contains the index of the sub-interval where failure occurred, in the range 1 to m-3. In case of an error in BDYVAL, it is set to 0 or m-2 depending on whether the left or right boundary condition caused the error.

FINFO(10): the value f the independent variable x, the point at which error occurred. In case of an error in BDYVAL, it is set to the value of XL or XR as appropriate (see the specification of BDYVAL).

FINFO(11), FINFO(12), FINFO(13): the current values of the Pruefer dependent variables (beta), (phi) and (rho) respectively. These are set to zero in case of an error in BDYVAL.

FINFO(14): the local-error tolerance being used by the internal integrator at the point of failure. This is set to zero in the case of an error in BDYVAL.

FINFO(15): the last integration mesh point. This is set to zero in the case of an error in BDYVAL.

MONIT must be declared as EXTERNAL in the (sub)program from which D02KEF is called. Parameters denoted as Input must not be changed by this procedure.

14: REPORT -- SUBROUTINE, supplied by the user.
External Procedure

This routine provides the means by which the user may compute the eigenfunction y(x) and its derivative at each integration mesh point x. (See Section 8 for an example).

Its specification is:

```
SUBROUTINE REPORT (X, V, JINT)
INTEGER          JINT
DOUBLE PRECISION X, V(3)
```

1: X -- DOUBLE PRECISION Input

On entry: the current value of the independent variable x. See Section 8.3 for the order in which values of x are supplied.

2: V(3) -- DOUBLE PRECISION array Input

On entry: V(1), V(2), V(3) hold the current values of the Pruefer variables (beta), (phi), (rho) respectively.

3: JINT -- INTEGER Input

On entry: JINT indicates the sub-interval between break-points in which X lies exactly as for the routine

COEFFN, except that at the extreme left end-point (when x = XPOINT(2)) JINT is set to 0 and at the extreme right end-point (when $x=x_r$ =XPOINT(m-1)) JINT is set to m-2.

REPORT must be declared as EXTERNAL in the (sub)program from which D02KEF is called. Parameters denoted as Input must not be changed by this procedure.

15: IFAIL -- INTEGER Input/Output

On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1

A parameter error. All parameters (except IFAIL) are left unchanged. The reason for the error is shown by the value of HMAX(2,1) as follows:

HMAX(2,1) = 1: M < 4;

HMAX(2,1) = 2: K < 0;

HMAX(2,1) = 3: TOL <= 0.0;

HMAX(2,1) = 4: XPOINT(1) to XPOINT(m) are not in ascending order.

HMAX(2,2) gives the position i in XPOINT where this was detected.

IFAIL= 2

At some call to BDYVAL, invalid values were returned, that is, either YL(1) = YL(2) = 0.0, or YR(1) = YR(2) = 0.0 (a programming error in BDYVAL). See the last call of MONIT for details.

This error exit will also occur if p(x) is zero at the point where the boundary condition is imposed. Probably BDYVAL was called with XL equal to a singular end-point a or with XR equal to a singular end-point b.

IFAIL= 3

At some point between XL and XR the value of p(x) computed

```
     by COEFFN became zero or changed sign. See the last call of
     MONIT for details.

IFAIL= 4
     MAXIT > 0 on entry, and after MAXIT iterations the
     eigenvalue had not been found to the required accuracy.

IFAIL= 5
     The 'bracketing' phase (with parameter IFLAG of MONIT equal
     to 1) failed to bracket the eigenvalue within ten
     iterations. This is caused by an error in formulating the
     problem (for example, q is independent of (lambda)), or by
     very poor initial estimates of ELAM, DELAM.

     On exit ELAM and ELAM + DELAM give the end-points of the
     interval within which no eigenvalue was located by the
     routine.

IFAIL= 6
     MAXFUN > 0 on entry, and the last iteration was terminated
     because more than MAXFUN calls to COEFFN were used. See the
     last call of MONIT for details.

IFAIL= 7
     To obtain the desired accuracy the local error tolerance was
     set so small at the start of some sub-interval that the
     differential equation solver could not choose an initial
     step size large enough to make significant progress. See the
     last call of MONIT for diagnostics.

IFAIL= 8
     At some point inside a sub-interval the step size in the
     differenital equation solver was reduced to a value too
     small to make significant progress (for the same reasons as
     with IFAIL = 7). This could be due to pathological behaviour
     of p(x) and q(x;(lambda)) or to an unreasonable accuracy
     requirement or to the current value of (lambda) making the
     equations 'stiff'. See the last call of MONIT for details.

IFAIL= 9
     TOL is too small for the problem being solved and the
     machine precision is being used. The final value of ELAM
     should be a very good approximation to the eigenvalue.

IFAIL= 10
     C05AZF(*), called by D02KEF, has terminated with the error
     exit corresponding to a pole of the residual function
     f((lambda)). This error exit should not occur, but if it
     does, try solving the problem again with a smaller value for
     TOL.
```

IFAIL= 11
A serious error has occurred in an internal call to D02KDY. Check all subroutine calls and array dimensions. Seek expert help.

IFAIL= 12
A serious error has occurred in an internal call to C05AZF(*). Check all subroutine calls and array dimensions. Seek expert help.

HMAX(2,1) holds the failure exit number from the routine where the failure occurred. In the case of a failure in C05AZF(*), HMAX(2,2) holds the value of parameter IND of C05AZF(*).

Note: error exits with IFAIL = 2, 3, 6, 7, 8, 11 are caused by being unable to set up or solve the differential equation at some iteration, and will be immediately preceded by a call of MONIT giving diagnostic information. For other errors, diagnostic information is contained in HMAX(2,j), for j=1,2,...,m, where appropriate.

7. Accuracy

See the discussion in Section 8.2.

8. Further Comments

8.1. Timing

The time taken by the routine depends on the complexity of the coefficient functions, whether they or their derivatives are rapidly changing, the tolerance demanded, and how many iterations are needed to obtain convergence. The amount of work per iteration is roughly doubled when TOL is divided by 16. To make the most economical use of the routine, one should try to obtain good initial values for ELAM and DELAM, and, where appropriate, good asymptotic formulae. The boundary matching points should not be set unnecessarily close to singular points. The extra time needed to compute the eigenfunction is principally the cost of one additional integration once the eigenvalue has been found.

8.2. General Description of the Algorithm

A shooting method, for differential equation problems containing unknown parameters, relies on the construction of a 'miss-distance function', which for given trial values of the parameters measures how far the conditions of the problem are from being met. The problem is then reduced to one of finding the values of the parameters for which the miss-distance function is zero, that is to a root-finding process. Shooting methods differ mainly in how the miss-distance is defined.

This routine defines a miss-distance f((lambda)) based on the rotation around the origin of the point P(x)=(p(x)y'(x),y(x)) in the Phase Plane as the solution proceeds from a to b. The boundary-conditions define the ray (i.e., two-sided line through the origin) on which p(x) should start, and the ray on which it should finish. The eigenvalue index k defines the total number of half-turns it should make. Numerical solution is actually done by matching point x=c. Then f((lambda)) is taken as the angle between the rays to the two resulting points $P_a(c)$ and $P_b(c)$. A relative scaling of the py' and y axes, based on the behaviour of the coefficient functions p and q, is used to improve the numerical behaviour.

Please see figure in printed Reference Manual

```
The resulting function f((lambda)) is monotonic over -
                                       ddq
infty<(lambda)<infty, increasing if  ---------->0 and decreasing
                                     dd(lambda)
       ddq
if  ----------<0, with a unique zero at the desired eigenvalue
    dd(lambda)
~~~~~~~~
(lambda). The routine measures f((lambda)) in units of a half-
```

turn. This means that as (lambda) increases, f((lambda)) varies by about 1 as each eigenvalue is passed. (This feature implies that the values of f((lambda)) at successive iterations - especially in the early stages of the iterative process - can be used with suitable extrapolation or interpolation to help the choice of initial estimates for eigenvalues near to the one currently being found.)

```
The routine actually computes a value for f((lambda)) with
errors, arising from the local errors of the differential
equation code and from the asymptotic formulae provided by the
user if singular points are involved. However, the error estimate
output in DELAM is usually fairly realistic, in that the actual
       ~~~~~~~~
error |(lambda)-ELAM| is within an order of magnitude of DELAM.
```

We pass the values of (beta), (phi), (rho) across through REPORT rather than converting them to values of y, y' inside D02KEF, for the following reasons. First, there may be cases where auxiliary quantities can be more accurately computed from the Pruefer variables than from y and y'. Second, in singular problems on an infinite interval y and y' may underflow towards the end of the range, whereas the Pruefer variables remain well-behaved. Third, with high-order eigenvalues (and therefore highly oscillatory eigenfunctions) the eigenfunction may have a complete oscillation (or more than one oscillation) between two mesh points, so that values of y and y' at mesh points give a very poor representation of the curve. The probable behaviour of the Pruefer variables in this case is that (beta) and (rho) vary slowly whilst (phi) increases quickly: for all three Pruefer variables linear interpolation between the values at adjacent mesh points is probably sufficiently accurate to yield acceptable intermediate values of (beta), (phi), (rho) (and hence of y,y') for graphical purposes.

Similar considerations apply to the exponentially decaying 'tails Here (phi) has approximately constant value whilst (rho) increases rapidly in the direction of integration, though the step length is generally fairly small over such a range.

If the solution is output through REPORT at x-values which are too widely spaced, the step length can be controlled by choosing HMAX suitably, or, preferably, by reducing TOL. Both these choices will lead to more accurate eigenvalues and eigenfunctions but at some computational cost.

8.3. The Position of the Shooting Matching Point c

This point is always one of the values x_i in array XPOINT. It may be specified using the parameter MATCH. The default value is chosen to be the value of that x_i,2<=i<=m-1, that lies closest to the middle of the interval $[x_2,x_{m-1}]$. If there is a tie, the rightmost candidate is chosen. In particular if there are no break-points then $c=x_{m-1}$ $(=x_3)$ - that is the shooting is from left to right in this case. A break-point may be inserted purely to move c to an interior point of the interval, even though the form of the equations does not require it. This often speeds up convergence especially with singular problems.

Note that the shooting method used by the code integrates first from the left-hand end x_l, then from the right-hand end x_r, to meet at the matching point c in the middle. This will of course be reflected in printed or graphical output. The diagram shows a possible sequence of nine mesh points (tau) through (tau) in

1 9
the order in which they appear, assuming there are just two sub-intervals (so m=5).

Figure 1
Please see figure in printed Reference Manual

Since the shooting method usually fails to match up the two 'legs p(x)y' or both, at the matching point c. The code in fact 'shares large jump does not imply an inaccurate eigenvalue, but implies either

(a) a badly chosen matching point: if q(x;(lambda)) has a ' humped' shape, c should be chosen near the maximum value of q, especially if q is negative at the ends of the interval.

(b) An inherently ill-conditioned problem, typically one where another eigenvalue is pathologically close to the one being sought. In this case it is extremely difficult to obtain an accurate eigenfunction.

In Section 9 below, we find the 11th eigenvalue and corresponding eigenfunction of the equation

$$y''+((\lambda)-x-2/x^2)y=0 \quad \text{on} \quad 0<x<\infty$$

the boundary conditions being that y should remain bounded as x tends to 0 and x tends to infty. The coding of this problem is discussed in detail in Section 8.5.

The choice of matching point c is open. If we choose c=30.0 as in the D02KDF(*) example program we find that the exponentially increasing component of the solution dominates and we get extremely inaccurate values for the eigenfunction (though the eigenvalue is determined accurately). The values of the eigenfunction calculated with c=30.0 are given schematically in Figure 2.

Figure 2
Please see figure in printed Reference Manual

If we choose c as the maximum of the hump in q(x;(lambda)) (see (a) above) we instead obtain the accurate results given in Figure 3.

Figure 3

Please see figure in printed Reference Manual

8.4. Examples of Coding the COEFFN Routine

Coding COEFFN is straightforward except when break-points are needed. The examples below show:

(a) a simple case,

(b) a case in which discontinuities in the coefficient functions or their derivatives necessitate break-points, and

(c) a case where break-points together with the HMAX parameter are an efficient way to deal with a coefficient function that is well-behaved except over one short interval.

Example A

The modified Bessel equation

$$x(xy')'+((\lambda)x^2-(\nu)^2)y=0$$

Assuming the interval of solution does not contain the origin, dividing through by x, we have p(x)=x, $q(x;(\lambda))=(\lambda)x-(\nu)^2/x$. The code could be

```
      SUBROUTINE COEFFN(P,Q,DQDL,X,ELAM,JINT)
      P = X
      Q = ELAM*X + NU*NU/X
      DQDL = X
      RETURN
      END
```

where NU (standing for (nu)) is a real variable that might be defined in a DATA statement, or might be in user-declared COMMON so that its value could be set in the main program.

Example B

The Schroedinger equation

$$y''+((\lambda)+q(x))y=0$$

$$\{x^2-10 \quad (|x|<=4),$$

where q(x)={6/|x| (|x|>4),

over some interval 'approximating to (-infty,infty)', say [-20, 20]. Here we need break-points at +- 4, forming three sub-intervals i_1=[-20,-4], i_2=[-4,4], i_3=[4,20]. The code could be

```
      SUBROUTINE COEFFN(P,Q,DQDL,X,ELAM,JINT)
      IF (JINT.EQ.2) THEN
        Q = ELAM + X*X - 10.0E0
      ELSE
        Q = ELAM + 6.0E0/ABS(X)
      ENDIF
      P = 1.0E0
      DQDL = 1.0E0
      RETURN
      END
```

The array XPOINT would contain the values x_1, -20.0, -4.0, +4.0, +20.0, x_6 and m would be 6. The choice of appropriate values for x_1 and x_6 depends on the form of the asymptotic formula computed by BDYVAL and the technique is discussed in the next subsection.

Example C

$$y''+(\lambda)(1-2e^{-100x^2})y=0, \text{ over } -10<=x<=10$$

Here q(x;(lambda)) is nearly constant over the range except for a sharp inverted spike over approximately -0.1<=x<=0.1. There is a danger that the routine will build up to a large step size and ' step over' the spike without noticing it. By using break-points - say at +- 0.5 - one can restrict the step size near the spike without impairing the efficiency elsewhere.

The code for COEFFN could be

```
      SUBROUTINE COEFFN(P,Q,DQDL,X,ELAM,JINT)
      P = 1.0E0
      DQDL = 1.0E0 - 2.0E0*EXP(-100.0E0*X*X)
      Q = ELAM*DQDL
      RETURN
```

```
      END
```

XPOINT might contain -0.0, -10.0, -0.5, 0.5, 10.0, 10.0 (assuming +- 10 are regular points) and m would be 6. HMAX(1,j), j=1,2,3 might contain 0.0, 0.1 and 0.0.

8.5. Examples of Boundary Conditions at Singular Points

Quoting from Bailey [2] page 243: 'Usually... the differential equation has two essentially different types of solution near a singular point, and the boundary condition there merely serves to distinguish one kind from the other. This is the case in all the standard examples of mathematical physics.'

In most cases the behaviour of the ratio p(x)y'/y near the point is quite different for the two types of solution. Essentially what the user provides through his BDYVAL routine is an approximation to this ratio, valid as x tends to the singular point (SP).

The user must decide (a) how accurate to make this approximation or asymptotic formula, for example how many terms of a series to use, and (b) where to place the boundary matching point (BMP) at which the numerical solution of the differential equation takes over from the asymptotic formula. Taking the BMP closer to the SP will generally improve the accuracy of the asymptotic formula, but will make the computation more expensive as the Pruefer differential equations generally become progressively more ill-behaved as the SP is approached. The user is strongly recommended to experiment with placing the BMPs. In many singular problems quite crude asymptotic formulae will do. To help the user avoid needlessly accurate formulae, D02KEF outputs two 'sensitivity coefficients' $(\sigma)_l$,$(\sigma)_r$ which estimate how much the errors at the BMP's affect the computed eigenvalue. They are described in detail below, see Section 8.6.

Example of coding BDYVAL:

The example below illustrates typical situations:

$$y''+\left(\lambda-x-\frac{2}{x^2}\right)y=0, \text{ for } 0<x<\infty$$

the boundary conditions being that y should remain bounded as x tends to 0 and x tends to infty.

```
                                                     2
At the end x=0 there is one solution that behaves like x  and
                           -1
another that behaves like x  . For the first of these solutions
p(x)y'/y is asymptotically 2/x while for the second it is
asymptotically -1/x. Thus the desired ratio is specified by
setting

                     YL(1)=x and YL(2)=2.0.

At the end x=infty the equation behaves like Airy's equation
shifted through (lambda), i.e., like y''-ty=0 where t=x-(lambda),
so again there are two types of solution. The solution we require
behaves as

                           (  2 3/2) 4 _
                        exp(- -t   )/\/t
                           (  3    )

and the other as

                           (  2 3/2) 4 _
                        exp(+ -t   )/\/t
                           (  3    )

                                                  _
once, the desired solution has p(x)y'/y~-\/t so that we could set
                                ----------
YR(1) = 1.0 and YR(2)=-\/x-(lambda). The complete subroutine
might thus be


      SUBROUTINE BDYVAL(XL,XR,ELAM,YL,YR)
      real XL, XR, ELAM, YL(3), YR(3)
      YL(1) = XL
      YL(2) = 2.0E0
      YR(1) = 1.0E0
      YR(2) = -SQRT(XR - ELAM)
      RETURN
      END


Clearly for this problem it is essential that any value given by
D02KEF to XR is well to the right of the value of ELAM, so that
the user must vary the right-hand BMP with the eigenvalue index k
                           k
function Ai(x), so there is no problem in estimating ELAM.

More accurate asymptotic formulae are easily found - near x=0 by
the standard Frobenius method, and near x=infty by using standard
```

asymptotics for Ai(x), Ai'(x) (see [1], p. 448). For example, by the Frobenius method the solution near x=0 has the expansion

$$y=x^2(c_0+c_1x+c_2x^2+...)$$

with

$$c_0=1, c_1=0, c_2=-\frac{(\lambda)}{10}, c_3=\frac{1}{18}, ..., c_n=\frac{c_{n-3}-(\lambda)c_{n-2}}{n(n+3)}$$

This yields

$$\frac{p(x)y'}{y}=\frac{2-\frac{2}{5}(\lambda)x^2+...}{x\left(1-\frac{(\lambda)}{10}x^2+...\right)}.$$

8.6. The Sensitivity Parameters $(\sigma)_l$ and $(\sigma)_r$

The sensitivity parameters $(\sigma)_l$, $(\sigma)_r$ (held in HMAX(1,m-1) and HMAX(1,m) on output) estimate the effect of errors in the boundary conditions. For sufficiently small errors (Delta)y, (Delta)py' in y and py' respectively, the relations

$$(\Delta)(\lambda)\tilde{=}(y.(\Delta)py'-py'.(\Delta)y)_l(\sigma)_l$$

$$(\Delta)(\lambda)\tilde{=}(y.(\Delta)py'-py'.(\Delta)y)_r(\sigma)_r$$

are satisfied where the subscripts l, r denote errors committed at left- and right-hand BMP's respectively, and (Delta)(lambda) denotes the consequent error in the computed eigenvalue.

8.7. Missed Zeros

This is a pitfall to beware of at a singular point. If the BMP is chosen so far from the SP that a zero of the desired eigenfunction lies in between them, then the routine will fail to number of zeros of its eigenfunction, the result will be that:

```
     (a)    The wrong eigenvalue will be computed for the given index k
            - in fact some (lambda)     will be found where k'>=1.
                                   k+k'

     (b)    The same index k can cause convergence to any of several
            eigenvalues depending on the initial values of ELAM and
            DELAM.

     It is up to the user to take suitable precautions - for instance
     by varying the position of the BMP's in the light of his
     knowledge of the asymptotic behaviour of the eigenfunction at
     different eigenvalues.

     9. Example

     To find the 11th eigenvalue and eigenfunction of the example of
     Section 8.5, using the simple asymptotic formulae for the
     boundary conditions.

     Comparison of the results from this example program with the
     corresponding results from D02KDF(*) example program shows that
     similar output is produced from the routine MONIT, followed by
     the eigenfunction values from REPORT, and then a further line of
     information from MONIT (corresponding to the integration to find
     the eigenfunction). Final information is printed within the
     example program exactly as with D02KDF(*).

                                                          3 _
     Note the discrepancy at the matching point c(=\/4 , the maximum
     of q(x;(lambda)), in this case) between the solutions obtained by
     integrations from left and right end-points.

     The example program is not reproduced here. The source code for
     all example programs is distributed with the NAG Foundation
     Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

———

Two-point boundary-value ODE equation systems

— nagd.ht —

```
\begin{page}{manpageXXd02raf}{NAG Documentation: d02raf}
```

```
\beginscroll
\begin{verbatim}


     D02RAF(3NAG)      Foundation Library (12/10/92)      D02RAF(3NAG)



          D02 -- Ordinary Differential Equations                     D02RAF
                  D02RAF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          D02RAF solves the two-point boundary-value problem with general
          boundary conditions for a system of ordinary differential
          equations, using a deferred correction technique and Newton
          iteration.

          2. Specification

                 SUBROUTINE D02RAF (N, MNP, NP, NUMBEG, NUMMIX, TOL, INIT,
                1                   X, Y, IY, ABT, FCN, G, IJAC, JACOBF,
                2                   JACOBG, DELEPS, JACEPS, JACGEP, WORK,
                3                   LWORK, IWORK, LIWORK, IFAIL)
                 INTEGER          N, MNP, NP, NUMBEG, NUMMIX, INIT, IY,
                1                 IJAC, LWORK, IWORK(LIWORK), LIWORK, IFAIL
                 DOUBLE PRECISION TOL, X(MNP), Y(IY,MNP), ABT(N), DELEPS,
                1                 WORK(LWORK)
                 EXTERNAL         FCN, G, JACOBF, JACOBG, JACEPS, JACGEP

          3. Description

          D02RAF solves a two-point boundary-value problem for a system of
          n ordinary differential equations in the interval (a,b) with b>a.
          The system is written in the form

                      y '=f (x,y ,y ,...,y ) , i=1,2,...,n               (1)
                       i   i    1  2      n

          and the derivatives f  are evaluated by a subroutine FCN supplied
                               i
          by the user. With the differential equations (1) must be given a
          system of n (nonlinear) boundary conditions
```

$$g_i(y(a),y(b))=0\ ,\ i=1,2,\ldots,n$$

where

$$y(x)=[y_1(x),y_2(x),\ldots,y_n(x)]^T\ . \qquad (2)$$

The functions g_i are evaluated by a subroutine G supplied by the user. The solution is computed using a finite-difference technique with deferred correction allied to a Newton iteration to solve the finite-difference equations. The technique used is described fully in Pereyra [1].

The user must supply an absolute error tolerance and may also supply an initial mesh for the finite-difference equations and an initial approximate solution (alternatively a default mesh and approximation are used). The approximate solution is corrected using Newton iteration and deferred correction. Then, additional points are added to the mesh and the solution is recomputed with the aim of making the error everywhere less than the user's tolerance and of approximately equidistributing the error on the final mesh. The solution is returned on this final mesh.

If the solution is required at a few specific points then these should be included in the initial mesh. If, on the other hand, the solution is required at several specific points then the user should use the interpolation routines provided in Chapter E01 if these points do not themselves form a convenient mesh.

The Newton iteration requires Jacobian matrices

$$\left(\frac{\partial f_i}{\partial y_j}\right),\ \left(\frac{\partial g_i}{\partial y_j(a)}\right) \text{ and } \left(\frac{\partial g_i}{\partial y_j(b)}\right).$$

These may be supplied by the user through subroutines JACOBF for $\left(\frac{\partial f_i}{\partial y_j}\right)$ and JACOBG for the others. Alternatively the Jacobians may be calculated by numerical differentiation using the algorithm described in Curtis et al [2].

For problems of the type (1) and (2) for which it is difficult to determine an initial approximation from which the Newton iteration will converge, a continuation facility is provided. The user must set up a family of problems

$$y'=f(x,y,\epsilon), \quad g(y(a),y(b),\epsilon)=0, \qquad (3)$$

where $f=[f_1,f_2,\ldots,f_n]^T$ etc, and where ϵ is a continuation parameter. The choice $\epsilon=0$ must give a problem (3) which is easy to solve and $\epsilon=1$ must define the problem whose solution is actually required. The routine solves a sequence of problems with ϵ values

$$0=\epsilon_1<\epsilon_2<\ldots<\epsilon_p=1. \qquad (4)$$

The number p and the values ϵ_i are chosen by the routine so that each problem can be solved using the solution of its predecessor as a starting approximation. Jacobians $\frac{\partial f}{\partial \epsilon}$ and $\frac{\partial g}{\partial \epsilon}$ are required and they may be supplied by the user via routines JACEPS and JACGEP respectively or may be computed by numerical differentiation.

4. References

[1] Pereyra V (1979) PASVA3: An Adaptive Finite-Difference Fortran Program for First Order Nonlinear, Ordinary Boundary Problems. Codes for Boundary Value Problems in Ordinary Differential Equations. Lecture Notes in Computer Science. (ed B Childs, M Scott, J W Daniel, E Denman and P Nelson) 76 Springer-Verlag.

[2] Curtis A R, Powell M J D and Reid J K (1974) On the Estimation of Sparse Jacobian Matrices. J. Inst. Maths Applics. 13 117--119.

5. Parameters

1: N -- INTEGER Input
On entry: the number of differential equations, n.
Constraint: N > 0.

2: MNP -- INTEGER Input
On entry: MNP must be set to the maximum permitted number

of points in the finite-difference mesh. If LWORK or LIWORK (see below) is too small then internally MNP will be replaced by the maximum permitted by these values. (A warning message will be output if on entry IFAIL is set to obtain monitoring information.) Constraint: MNP >= 32.

3: NP -- INTEGER Input/Output
On entry: NP must be set to the number of points to be used in the initial mesh. Constraint: 4 <= NP <= MNP. On exit: the number of points in the final mesh.

4: NUMBEG -- INTEGER Input
On entry: the number of left-hand boundary conditions (that is the number involving y(a) only). Constraint: 0 <= NUMBEG < N.

5: NUMMIX -- INTEGER Input
On entry: the number of coupled boundary conditions (that is the number involving both y(a) and y(b)). Constraint: 0 <= NUMMIX <= N - NUMBEG.

6: TOL -- DOUBLE PRECISION Input
On entry: a positive absolute error tolerance. If

$$a=x_1<x_2<\ldots<x_{NP}=b$$

is the final mesh, $z_j(x_i)$ is the jth component of the approximate solution at x_i, and $y_j(x)$ is the jth component of the true solution of (1) and (2), then, except in extreme circumstances, it is expected that

$$|z_j(x_i)-y_j(x_i)|<=TOL , i=1,2,...,NP ; j=1,2,...,n. \quad (5)$$

Constraint: TOL > 0.0.

7: INIT -- INTEGER Input
On entry: indicates whether the user wishes to supply an initial mesh and approximate solution (INIT /= 0) or whether default values are to be used, (INIT = 0).

8: X(MNP) -- DOUBLE PRECISION array Input/Output
On entry: the user must set X(1) = a and X(NP) = b. If INIT = 0 on entry a default equispaced mesh will be used, otherwise the user must specify a mesh by setting $X(i)=x_i$, for i = 2,3,...NP-1. Constraints:

X(1) < X(NP), if INIT = 0,

X(1) < X(2) <... < X(NP), if INIT /= 0.

On exit: X(1),X(2),...,X(NP) define the final mesh (with

```
    the returned value of NP) and X(1) = a and X(NP) = b.

 9: Y(IY,MNP) -- DOUBLE PRECISION array             Input/Output
    On entry: if INIT = 0, then Y need not be set.

    If INIT /= 0, then the array Y must contain an initial
    approximation to the solution such that Y(j,i) contains an
    approximation to
                y (x ) , i=1,2,...,NP ; j=1,2,...,n.
                 j  i
    On exit: the approximate solution z (x ) satisfying (5) on
                                       j  i
    the final mesh, that is
            Y(j,i)=z (x ) , i=1,2,...,NP ; j=1,2,...,n,
                    j  i
    where NP is the number of points in the final mesh. If an
    error has occurred then Y contains the latest approximation
    to the solution. The remaining columns of Y are not used.

10: IY -- INTEGER                                          Input
    On entry:
    the first dimension of the array Y as declared in the
    (sub)program from which D02RAF is called.
    Constraint: IY >= N.

11: ABT(N) -- DOUBLE PRECISION array                      Output
    On exit: ABT(i), for i=1,2,...,n, holds the largest
    estimated error (in magnitude) of the ith component of the
    solution over all mesh points.

12: FCN -- SUBROUTINE, supplied by the user.
                                              External Procedure
    FCN must evaluate the functions f  (i.e., the derivatives
                                     i
    y' ) at a general point x for a given value of (epsilon),
      i
    the continuation parameter (see Section 3).

    Its specification is:

           SUBROUTINE FCN (X, EPS, Y, F, N)
           INTEGER          N
           DOUBLE PRECISION X, EPS, Y(N), F(N)

     1:  X -- DOUBLE PRECISION                             Input
         On entry: the value of the argument x.

     2:  EPS -- DOUBLE PRECISION                           Input
         On entry: the value of the continuation parameter,
         (epsilon). This is 1 if continuation is not being used.
```

```
 3:  Y(N) -- DOUBLE PRECISION array                   Input
     On entry: the value of the argument y , for
                                          i
     i=1,2,...,n.

 4:  F(N) -- DOUBLE PRECISION array                  Output
     On exit: the values of f , for i=1,2,...,n.
                             i

 5:  N -- INTEGER                                     Input
     On entry: the number of equations.
FCN must be declared as EXTERNAL in the (sub)program
from which D02RAF is called. Parameters denoted as
Input must not be changed by this procedure.

13:  G -- SUBROUTINE, supplied by the user.
                                     External Procedure
G must evaluate the boundary conditions in equation (3) and
place them in the array BC.

Its specification is:

       SUBROUTINE G (EPS, YA, YB, BC, N)
       INTEGER          N
       DOUBLE PRECISION EPS, YA(N), YB(N), BC(N)

 1:  EPS -- DOUBLE PRECISION                          Input
     On entry: the value of the continuation parameter,
     (epsilon). This is 1 if continuation is not being used.

 2:  YA(N) -- DOUBLE PRECISION array                  Input
     On entry: the value y (a), for i=1,2,...,n.
                          i

 3:  YB(N) -- DOUBLE PRECISION array                  Input
     On entry: the value y (b), for i=1,2,...,n.
                          i

 4:  BC(N) -- DOUBLE PRECISION array                 Output
     On exit: the values g (y(a),y(b),(epsilon)), for
                          i
     i=1,2,...,n. These must be ordered as follows:
       (i)   first, the conditions involving only y(a) (see
             NUMBEG description above);

       (ii)  next, the NUMMIX coupled conditions involving
             both y(a) and y(b) (see NUMMIX description
             above); and,
```

```
       (iii) finally, the conditions involving only y(b) (N-
             NUMBEG-NUMMIX).

 5:  N -- INTEGER                                     Input
     On entry: the number of equations, n.
G must be declared as EXTERNAL in the (sub)program from
which D02RAF is called. Parameters denoted as Input
must not be changed by this procedure.

14:  IJAC -- INTEGER                                  Input
     On entry: indicates whether or not the user is supplying
     Jacobian evaluation routines. If IJAC /= 0 then the user
     must supply routines JACOBF and JACOBG and also, when
     continuation is used, routines JACEPS and JACGEP. If IJAC =
     0 numerical differentiation is used to calculate the
     Jacobian and the routines D02GAZ, D02GAY, D02GAZ and D02GAX
                                      ( ddf )
                                      (    i)
     JACOBF must evaluate the Jacobian ( ----) for i,j=1,2,...,n,
                                      ( ddy )
                                      (    j)
     given x and y , for j=1,2,...,n.
                  j

     Its specification is:

            SUBROUTINE JACOBF (X, EPS, Y, F, N)
            INTEGER          N
            DOUBLE PRECISION X, EPS, Y(N), F(N,N)

 1:  X -- DOUBLE PRECISION                            Input
     On entry: the value of the argument x.

 2:  EPS -- DOUBLE PRECISION                          Input
     On entry: the value of the continuation parameter
     (epsilon). This is 1 if continuation is not being used.

 3:  Y(N) -- DOUBLE PRECISION array                   Input
     On entry: the value of the argument y , for
                                          i
     i=1,2,...,n.

 4:  F(N,N) -- DOUBLE PRECISION array                Output
                                                 ddf
                                                    i
     On exit: F(i,j) must be set to the value of   ----,
                                                 ddy
                                                    j
     evaluated at the point (x,y), for i,j=1,2,...,n.
```

```
 5:  N -- INTEGER                                      Input
     On entry: the number of equations, n.
JACOBF must be declared as EXTERNAL in the (sub)program
from which D02RAF is called. Parameters denoted as
Input must not be changed by this procedure.

16:  JACOBG -- SUBROUTINE, supplied by the user.
                                           External Procedure
                                    (  ddg   )     (  ddg   )
                                    (     i  )     (     i  )
JACOBG must evaluate the Jacobians ( -------) and ( -------)
                                    ( ddy (a))     ( ddy (b))
                                    (    j   )     (    j   )
The ordering of the rows of AJ and BJ must correspond to
the ordering of the boundary conditions described in the
specification of subroutine G above.

Its specification is:

       SUBROUTINE JACOBG (EPS, YA, YB, AJ, BJ, N)
       INTEGER          N
       DOUBLE PRECISION EPS, YA(N), YB(N), AJ(N,N), BJ
      1                 (N,N)

 1:  EPS -- DOUBLE PRECISION                           Input
     On entry: the value of the continuation parameter,
     (epsilon). This is 1 if continuation is not being used.

 2:  YA(N) -- DOUBLE PRECISION array                   Input
     On entry: the value y (a), for i=1,2,...,n.
                          i

 3:  YB(N) -- DOUBLE PRECISION array                   Input
     On entry: the value y (b), for i=1,2,...,n.
                         i

 4:  AJ(N,N) -- DOUBLE PRECISION array                Output
                                                 ddg
                                                    i
     On exit: AJ(i,j) must be set to the value  -------,
                                                ddy (a)
                                                   j
     for i,j=1,2,...,n.

 5:  BJ(N,N) -- DOUBLE PRECISION array                Output
                                                 ddg
                                                    i
     On exit: BJ(i,j) must be set to the value  -------,
                                                ddy (b)
                                                   j
```

```
         for i,j=1,2...,n.

      6:  N -- INTEGER                                      Input
          On entry: the number of equations, n.
     JACOBG must be declared as EXTERNAL in the (sub)program
     from which D02RAF is called. Parameters denoted as
     Input must not be changed by this procedure.

 17: DELEPS -- DOUBLE PRECISION                      Input/Output
     On entry: DELEPS must be given a value which specifies
     whether continuation is required. If DELEPS <= 0.0 or DELEPS
     >= 1.0 then it is assumed that continuation is not required.
     If 0.0 < DELEPS < 1.0 then it is assumed that continuation
     is required unless DELEPS < square root of machine precision
     when an error exit is taken. DELEPS is used as the increment
     (epsilon) -(epsilon)  (see (4)) and the choice DELEPS = 0.1
              2          1
     is recommended. On exit: an overestimate of the increment
     (epsilon) -(epsilon)    (in fact the value of the increment
              p          p-1
     which would have been tried if the restriction (epsilon) =1
                                                             p
     had not been imposed). If continuation was not requested
     then DELEPS = 0.0.

     If continuation is not requested then the parameters JACEPS
     and JACGEP may be replaced by dummy actual parameters in the
     call to D02RAF. (D02GAZ and D02GAX respectively may be used
     as the dummy parameters.)

 18: JACEPS -- SUBROUTINE, supplied by the user.
                                           External Procedure
                                         ddf
                                            i
     JACEPS must evaluate the derivative  ----------- given x and
                                         dd(epsilon)
     y if continuation is being used.

     Its specification is:

            SUBROUTINE JACEPS (X, EPS, Y, F, N)
            INTEGER          N
            DOUBLE PRECISION X, EPS, Y(N), F(N)

      1:  X -- DOUBLE PRECISION                             Input
          On entry: the value of the argument x.

      2:  EPS -- DOUBLE PRECISION                           Input
          On entry: the value of the continuation parameter,
          (epsilon).
```

3: Y(N) -- DOUBLE PRECISION array Input
On entry: the solution values y_i at the point x, for i=1,2,...,n.

4: F(N) -- DOUBLE PRECISION array Output
On exit: F(i) must contain the value $\frac{\partial f_i}{\partial \epsilon}$ at the point (x,y), for i=1,2,...,n.

5: N -- INTEGER Input
On entry: the number of equations, n.

JACEPS must be declared as EXTERNAL in the (sub)program from which D02RAF is called. Parameters denoted as Input must not be changed by this procedure.

19: JACGEP -- SUBROUTINE, supplied by the user. External Procedure

JACGEP must evaluate the derivatives $\frac{\partial g_i}{\partial \epsilon}$ if continuation is being used.

Its specification is:

```
SUBROUTINE JACGEP (EPS, YA, YB, BCEP, N)
INTEGER          N
DOUBLE PRECISION EPS, YA(N), YB(N), BCEP(N)
```

1: EPS -- DOUBLE PRECISION Input
On entry: the value of the continuation parameter, ϵ.

2: YA(N) -- DOUBLE PRECISION array Input
On entry: the value of $y_i(a)$, for i=1,2,...,n.

3: YB(N) -- DOUBLE PRECISION array Input
On entry: the value of $y_i(b)$, for i=1,2,...,n.

4: BCEP(N) -- DOUBLE PRECISION array Output
On exit: BCEP(i) must contain the value of $\frac{\partial g_i}{\partial \epsilon}$, for i=1,2,...,n.

```
      dd(epsilon)

 5:  N -- INTEGER                                      Input
     On entry: the number of equations, n.
JACGEP must be declared as EXTERNAL in the (sub)program
from which D02RAF is called. Parameters denoted as
Input must not be changed by this procedure.

20:  WORK(LWORK) -- DOUBLE PRECISION array             Workspace

21:  LWORK -- INTEGER                                  Input
     On entry:
     the dimension of the array WORK as declared in the
     (sub)program from which D02RAF is called.
                                  2         2
     Constraint: LWORK>=MNP*(3N +6N+2)+4N +3N.

22:  IWORK(LIWORK) -- INTEGER array                    Workspace

23:  LIWORK -- INTEGER                                 Input
     On entry:
     the dimension of the array IWORK as declared in the
     (sub)program from which D02RAF is called.
     Constraints:
          LIWORK>=MNP*(2*N+1)+N, if IJAC /= 0,

                                  2
          LIWORK>=MNP*(2*N+1)+N +4*N+2, if IJAC = 0.

24:  IFAIL -- INTEGER                                  Input/Output
     For this routine, the normal use of IFAIL is extended to
     control the printing of error and warning messages as well
     as specifying hard or soft failure (see the Essential
     Introduction).

     Before entry, IFAIL must be set to a value with the decimal
     expansion cba, where each of the decimal digits c, b and a
     must have a value of 0 or 1.
     a=0   specifies hard failure, otherwise soft failure;

     b=0   suppresses error messages, otherwise error messages
           will be printed (see Section 6);

     c=0   suppresses warning messages, otherwise warning
           messages will be printed (see Section 6).
     The recommended value for inexperienced users is 110 (i.e.,
     hard failure with all messages printed).

     Unless the routine detects an error (see Section 6), IFAIL
     contains 0 on exit.
```

```
6. Error Indicators and Warnings

Errors detected by the routine:

For each error, an explanatory error message is output on the
current error message unit (as defined by X04AAF), unless
suppressed by the value of IFAIL on entry.

IFAIL= 1
     One or more of the parameters N, MNP, NP, NUMBEG, NUMMIX,
     TOL, DELEPS, LWORK or LIWORK has been incorrectly set, or X
     (1) >= X(NP) or the mesh points X(i) are not in strictly
     ascending order.

IFAIL= 2
     A finer mesh is required for the accuracy requested; that is
     MNP is not large enough. This error exit normally occurs
     when the problem being solved is difficult (for example,
     there is a boundary layer) and high accuracy is requested. A
     poor initial choice of mesh points will make this error exit
     more likely.

IFAIL= 3
     The Newton iteration has failed to converge. There are
     several possible causes for this error:
     (i)   faulty coding in one of the Jacobian calculation
           routines;

     (ii)  if IJAC = 0 then inaccurate Jacobians may have been
           calculated numerically (this is a very unlikely
           cause); or,

     (iii) a poor initial mesh or initial approximate solution
           has been selected either by the user or by default or
           there are not enough points in the initial mesh.
           Possibly, the user should try the continuation
           facility.

IFAIL= 4
     The Newton iteration has reached round-off error level. It
     could be however that the answer returned is satisfactory.
     The error is likely to occur if too high an accuracy is
     requested.

IFAIL= 5
     The Jacobian calculated by JACOBG (or the equivalent matrix
     calculated by numerical differentiation) is singular. This
     may occur due to faulty coding of JACOBG or, in some
     circumstances, to a zero initial choice of approximate
```

```
     solution (such as is chosen when INIT = 0).

IFAIL= 6
     There is no dependence on (epsilon) when continuation is
     being used. This can be due to faulty coding of JACEPS or
     JACGEP or, in some circumstances, to a zero initial choice
     of approximate solution (such as is chosen when INIT = 0).

IFAIL= 7
     DELEPS is required to be less than machine precision for
     continuation to proceed. It is likely that either the
     problem (3) has no solution for some value near the current
     value of (epsilon) (see the advisory print out from D02RAF)
     or that the problem is so difficult that even with
     continuation it is unlikely to be solved using this routine.
     If the latter cause is suspected then using more mesh points
     initially may help.

IFAIL= 8
     Indicates that a serious error has occurred in a call to
     D02RAF. Check all array subscripts and subroutine parameter
     lists in calls to D02RAF. Seek expert help.

IFAIL= 9
     Indicates that a serious error has occurred in a call to
     D02RAR. Check all array subscripts and subroutine parameter
     lists in calls to D02RAF. Seek expert help.

7. Accuracy

The solution returned by the routine will be accurate to the
user's tolerance as defined by the relation (5) except in extreme
circumstances. The final error estimate over the whole mesh for
each component is given in the array ABT. If too many points are
specified in the initial mesh, the solution may be more accurate
than requested and the error may not be approximately
equidistributed.

8. Further Comments

There are too many factors present to quantify the timing. The
time taken by the routine is negligible only on very simple
problems.

The user is strongly recommended to set IFAIL to obtain self-
explanatory error messages, and also monitoring information about
the course of the computation.

In the case where the user wishes to solve a sequence of similar
problems, the use of the final mesh and solution from one case as
```

```
         the initial mesh is strongly recommended for the next.

         9. Example

         We solve the differential equation

                                                2
                           y'''=-yy''-2(epsilon)(1-y' )

         with (epsilon)=1 and boundary conditions

                              y(0)=y'(0)=0,  y'(10)=1

         to an accuracy specified by TOL=1.0E-4. The continuation facility
         is used with the continuation parameter (epsilon) introduced as
         in the differential equation above and with DELEPS = 0.1
         initially. (The continuation facility is not needed for this
         problem and is used here for illustration.)

         The example program is not reproduced here. The source code for
         all example programs is distributed with the NAG Foundation
         Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Partial differential equations

— nagd.ht —

```
\begin{page}{manpageXXd03}{NAG Documentation: d03}
\beginscroll
\begin{verbatim}



     D03(3NAG)        Foundation Library (12/10/92)        D03(3NAG)



         D03 -- Partial Differential Equations         Introduction -- D03
                                    Chapter D03
                           Partial Differential Equations
```

1. Scope of the Chapter

This chapter is concerned with the solution of partial differential equations.

2. Background to the Problems

The definition of a partial differential equation problem includes not only the equation itself but also the domain of interest and appropriate subsidiary conditions. Indeed, partial differential equations are usually classified as elliptic, hyperbolic or parabolic according to the form of the equation and the form of the subsidiary conditions which must be assigned to produce a well-posed problem. Ultimately it is hoped that this chapter will contain routines for the solution of equations of each of these types together with automatic mesh generation routines and other utility routines particular to the solution of partial differential equations. The routines in this chapter will often call upon routines from the Linear Algebra Chapter F04 -- Simultaneous Linear Equations.

The classification of partial differential equations is easily described in the case of linear equations of the second order in two independent variables, i.e., equations of the form

$$au_{xx}+2bu_{xy}+cu_{yy}+du_x+eu_y+fu+g=0, \qquad (1)$$

where a, b, c, d, e, f and g are functions of x and y only. Equation (1) is called elliptic, hyperbolic or parabolic according as $ac-b^2$ is positive, negative or zero. Useful definitions of the concepts of elliptic, hyperbolic and parabolic character can also be given for differential equations in more than two independent variables, for systems and for nonlinear differential equations.

For elliptic equations, of which Laplace's equation

$$u_{xx}+u_{yy}=0 \qquad (2)$$

is the simplest example of second order, the subsidiary conditions take the form of boundary conditions, i.e., conditions which provide information about the solution at all points of a closed boundary. For example, if equation (2) holds in a plane domain D bounded by a contour C, a solution u may be sought subject to the condition

$$u=f \quad \text{on } C, \qquad (3)$$

where f is a given function. The condition (3) is known as a Dirichlet boundary condition. Equally common is the Neumann boundary condition

$$u'=g \quad \text{on } C, \tag{4}$$

which is one form of a more general condition

$$u'+fu=g \quad \text{on } C, \tag{5}$$

where u' denotes the derivative of u normal to the contour C and f and g are given functions. Provided that f and g satisfy certain restrictions, condition (5) yields a well-posed boundary value problem for Laplace's equation. In the case of the Neumann problem, one further piece of information, e.g. the value of u at a particular point, is necessary for uniqueness of the solution. Boundary conditions similar to the above are applicable to more general second order elliptic equations, whilst two such conditions are required for equations of fourth order.

For hyperbolic equations, the wave equation

$$u_{tt}-u_{xx}=0 \tag{6}$$

is the simplest example of second order. It is equivalent to a first order system

$$u_t-v_x=0, \quad v_t-u_x=0. \tag{7}$$

The subsidiary conditions may take the form of initial conditions, i.e., conditions which provide information about the solution at points on a suitable open boundary. For example, if equation (6) is satisfied for t>0, a solution u may be sought such that

$$u(x,0)=f(x), \quad u_t(x,0)=g(x), \tag{8}$$

where f and g are given functions. This is an example of an initial value problem, sometimes known as Cauchy's problem.

For parabolic equations, of which the heat conduction equation

$$u_t-u_{xx}=0 \tag{9}$$

is the simplest example, the subsidiary conditions always include some of initial type and may also include some of boundary type. For example, if equation (9) is satisfied for $t>0$ and $0<x<1$, a solution u may be sought such that

$$u(x,0)=f(x), \quad 0<x<1, \tag{10}$$

and

$$u(0,t)=0, \quad u(1,t)=1, \quad t>0. \tag{11}$$

This is an example of a mixed initial/boundary value problem.

For all types of partial differential equations, finite difference methods (Mitchell and Griffiths [5]) and finite element methods (Wait and Mitchell [9]) are the most common means of solution and such methods obviously feature prominently either in this chapter or in the companion NAG Finite Element Library. Many of the utility routines in this chapter are concerned with the solution of the large sparse systems of equations which arise from the finite difference and finite element methods.

Alternative methods of solution are often suitable for special classes of problems. For example, the method of characteristics is the most common for hyperbolic equations involving time and one space dimension (Smith [7]). The method of lines (see Mikhlin and Smolitsky [4]) may be used to reduce a parabolic equation to a (stiff) system of ordinary differential equations, which may be solved by means of routines from Chapter D02 -- Ordinary Differential Equations. Similarly, integral equation or boundary element methods (Jaswon and Symm [3]) are frequently used for elliptic equations. Typically, in the latter case, the solution of a boundary value problem is represented in terms of certain boundary functions by an integral expression which satisfies the differential equation throughout the relevant domain. The boundary functions are obtained by applying the given boundary conditions to this representation. Implementation of this method necessitates discretisation of only the boundary of the domain, the dimensionality of the problem thus being effectively reduced by one. The boundary conditions yield a full system of simultaneous equations, as opposed to the sparse systems yielded by the finite difference and finite element methods, but the full system is usually of much lower order. Solution of this system yields the boundary functions, from which the solution of the problem may be obtained, by quadrature, as and where required.

2.1. References

[1] Ames W F (1977) Nonlinear Partial Differential Equations in Engineering. Academic Press (2nd Edition).

[2] Berzins M (1990) Developments in the NAG Library Software for Parabolic Equations. Scientific Software Systems. (ed J C Mason and M G Cox) Chapman and Hall. 59--72.

[3] Jaswon M A and Symm G T (1977) Integral Equation Methods in Potential Theory and Elastostatics. Academic Press.

[4] Mikhlin S G and Smolitsky K L (1967) Approximate Methods for the Solution of Differential and Integral Equations. Elsevier.

[5] Mitchell A R and Griffiths D F (1980) The Finite Difference Method in Partial Differential Equations. Wiley.

[6] Richtmyer R D and Morton K W (1967) Difference Methods for Initial-value Problems. Interscience (2nd Edition).

[7] Smith G D (1985) Numerical Solution of Partial Differential Equations: Finite Difference Methods. Oxford University Press (3rd Edition).

[8] Swarztrauber P N and Sweet R A (1979) Efficient Fortran Subprograms for the Solution of Separable Elliptic Partial Differential Equations. ACM Trans. Math. Softw. 5 352--364.

[9] Wait R and Mitchell A R (1985) Finite Element Analysis and Application. Wiley.

3. Recommendations on Choice and Use of Routines

The choice of routine will depend first of all upon the type of partial differential equation to be solved. At present no special allowances are made for problems with boundary singularities such as may arise at corners of domains or at points where boundary conditions change. For such problems results should be treated with caution.

Users may wish to construct their own partial differential equation solution software for problems not solvable by the routines described in Sections 3.1 to 3.4 below. In such cases users can employ appropriate routines from the Linear Algebra Chapters to solve the resulting linear systems; see Section 3.5 for further details.

3.1. Elliptic Equations

The routine D03EDF solves a system of seven-point difference equations in a rectangular grid (in two dimensions), using the multigrid iterative method. The equations are supplied by the user, and the seven-point form allows cross-derivative terms to be represented (see Mitchell and Griffiths [5]). The method is particularly efficient for large systems of equations with diagonal dominance.

The routine D03EEF discretises a second-order equation on a two-dimensional rectangular region using finite differences and a seven-point molecule. The routine allows for cross-derivative terms, Dirichlet, Neumann or mixed boundary conditions, and either central or upwind differences. The resulting seven-diagonal difference equations are in a form suitable for passing directly to the multigrid routine D03EDF, although other solution methods could easily be used.

The routine D03FAF, based on the routine HW3CRT from FISHPACK (Swarztrauber and Sweet [8]), solves the Helmholtz equation in a three-dimensional cuboidal region, with any combination of Dirichlet, Neumann or periodic boundary conditions. The method used is based on the fast Fourier transform algorithm, and is likely to be particularly efficient on vector-processing machines.

3.2. Hyperbolic Equations

There are no routines available yet for the solution of these equations.

3.3. Parabolic Equations

There are no routines available yet for the solution of these equations.

But problems in two space dimensions plus time may be treated as a succession of elliptic equations [1], [6] using appropriate D03E- routines or one may use codes from the NAG Finite Element Library.

3.4. Utility Routines

There are no utility routines available yet, but routines are available in the Linear Algebra Chapters for the direct and iterative solution of linear equations. Here we point to some of the routines that may be of use in solving the linear systems that arise from finite difference or finite element approximations to partial differential equation solutions. Chapters F01 and F04 should be consulted for further information and for the routine documents. Decision trees for the solution of linear systems are given in Section 3.5 of the F04 Chapter Introduction.

The following routines allow the direct solution of symmetric

```
positive-definite systems:

    Band               F04ACF

    Variable band      F01MCF and F04MCF
    (skyline)

    Tridiagonal        F04FAF

    Sparse             F01MAF* and F04MAF

(* the parameter DROPTL should be set to zero for F01MAF to give
a direct solution)

and the following routines allow the iterative solution of
symmetric positive-definite systems:

    Sparse (incomplete F01MAF and F04MBF
    Cholesky)

    Sparse (conjugate  F04MBF
    gradient)

The latter routine above allows the user to supply a pre-
conditioner and also allows the solution of indefinite symmetric
systems.

The following routines allow the direct solution of unsymmetric
systems:

    Band               F01LBF and F04LDF

    Almost block-      F01LHF and F04LHF
    diagonal

    Tridiagonal        F01LEF and F04LEF or F04EAF

    Sparse             F01BRF (and F01BSF) and F04AXF

and the following routine allows the iterative solution of
unsymmetric systems:

    Sparse             F04QAF

The above routine allows the user to supply a pre-conditioner and
also allows the solution of least-squares systems.

3.5. Index

Elliptic equations
```

```
        equations on rectangular grid (seven-point 2-D       D03EDF
        molecule)
        discretisation on rectangular grid (seven-point 2-D   D03EEF
        molecule)
        Helmholtz's equation in three dimensions             D03FAF



   D03 -- Partial Differential Equations            Contents -- D03
   Chapter D03

   Partial Differential Equations

   D03EDF  Elliptic PDE, solution of finite difference equations by
           a multigrid technique

   D03EEF  Discretize a 2nd order elliptic PDE on a rectangle

   D03FAF  Elliptic PDE, Helmholtz equation, 3-D Cartesian co-
           ordinates

\end{verbatim}
\endscroll
\end{page}
```

Discrete elliptic PDE on rectangular region

— nagd.ht —

```
\begin{page}{manpageXXd03edf}{NAG Documentation: d03edf}
\beginscroll
\begin{verbatim}



     D03EDF(3NAG)      Foundation Library (12/10/92)      D03EDF(3NAG)



   D03 -- Partial Differential Equations                      D03EDF
           D03EDF -- NAG Foundation Library Routine Document

   Note: Before using this routine, please read the Users' Note for
   your implementation to check implementation-dependent details.
   The symbol (*) after a NAG routine name denotes a routine that is
```

not included in the Foundation Library.

1. Purpose

D03EDF solves seven-diagonal systems of linear equations which arise from the discretization of an elliptic partial differential equation on a rectangular region. This routine uses a multigrid technique.

2. Specification

```
       SUBROUTINE D03EDF (NGX, NGY, LDA, A, RHS, UB, MAXIT, ACC,
      1                   US, U, IOUT, NUMIT, IFAIL)
       INTEGER          NGX, NGY, LDA, MAXIT, IOUT, NUMIT, IFAIL
       DOUBLE PRECISION A(LDA,7), RHS(LDA), UB(NGX*NGY), ACC, US
      1                 (LDA), U(LDA)
```

3. Description

D03EDF solves, by multigrid iteration, the seven-point scheme

$$A^{6}_{i,j} u_{i-1,j+1} + A^{7}_{i,j} u_{i,j+1}$$

$$+A^{3}_{i,j} u_{i-1,j} + A^{4}_{i,j} u_{ij} + A^{5}_{i,j} u_{i+1,j}$$

$$+A^{1}_{i,j} u_{i,j-1} + A^{2}_{i,j} u_{i+1,j-1} = f_{ij} ,$$

$$i=1,2,\ldots,n_x ; \; j=1,2,\ldots,n_y ,$$

which arises from the discretization of an elliptic partial differential equation of the form

$$(\alpha)(x,y)U_{xx} + (\beta)(x,y)U_{xy} + (\gamma)(x,y)U_{yy} + (\delta)(x,y)U_x$$

$$+(\epsilon)(x,y)U_y + (\phi)(x,y)U = (\psi)(x,y)$$

and its boundary conditions, defined on a rectangular region. This we write in matrix form as

$$Au=f$$

The algorithm is described in separate reports by Wesseling [2], [3] and McCarthy [1].

Systems of linear equations, matching the seven-point stencil defined above, are solved by a multigrid iteration. An initial estimate of the solution must be provided by the user. A zero guess may be supplied if no better approximation is available.

A 'smoother' based on incomplete Crout decomposition is used to eliminate the high frequency components of the error. A restriction operator is then used to map the system on to a sequence of coarser grids. The errors are then smoothed and prolongated (mapped onto successively finer grids). When the finest cycle is reached, the approximation to the solution is corrected. The cycle is repeated for MAXIT iterations or until the required accuracy, ACC, is reached.

D03EDF will automatically determine the number l of possible coarse grids, 'levels' of the multigrid scheme, for a particular problem. In other words, D03EDF determines the maximum integer l so that n_x and n_y can be expressed in the form

$$n_x=m2^{l-1}+1,\quad n_y=n2^{l-1}+1,\quad \text{with } m\geq 2 \text{ and } n\geq 2.$$

It should be noted that the rate of convergence improves significantly with the number of levels used (see McCarthy [1]), so that n_x and n_y should be carefully chosen so that n_x-1 and n_y-1 have factors of the form 2^l, with l as large as possible. For good convergence the integer l should be at least 2.

D03EDF has been found to be robust in application, but being an iterative method the problem of divergence can arise. For a strictly diagonally dominant matrix A

$$|A^4_{ij}|>\sum_{k\neq 4}|A^k_{ij}|$$

no such problem is foreseen. The diagonal dominance of A is not a necessary condition, but should this condition be strongly violated then divergence may occur. The quickest test is to try the routine.

4. References

[1] McCarthy G J (1983) Investigation into the Multigrid Code MGD1. Report AERE-R 10889. Harwell.

[2] Wesseling P (1982) MGD1 - A Robust and Efficient Multigrid Method. Multigrid Methods. Lecture Notes in Mathematics. 960 Springer-Verlag. 614--630.

[3] Wesseling P (1982) Theoretical Aspects of a Multigrid Method. SIAM J. Sci. Statist. Comput. 3 387--407.

5. Parameters

1: NGX -- INTEGER Input
On entry: the number of interior grid points in the x-direction, n_x. NGX-1 should preferably be divisible by as high a power of 2 as possible. Constraint: NGX >= 3.

2: NGY -- INTEGER Input
On entry: the number of interior grid points in the y-direction, n_y. NGY-1 should preferably be divisible by as high a power of 2 as possible. Constraint: NGY >= 3.

3: LDA -- INTEGER Input
On entry: the first dimension of the array A as declared in the (sub)program from which D03EDF is called, which must also be a lower bound for the dimensions of the arrays RHS, US and U. It is always sufficient to set LDA>=(4*(NGX+1)*(NGY+1))/3, but slightly smaller values may be permitted, depending on the values of NGX and NGY. If on entry, LDA is too small, an error message gives the minimum permitted value. (LDA must be large enough to allow space for the coarse-grid approximations).

4: A(LDA,7) -- DOUBLE PRECISION array Input/Output
On entry: A(i+(j-1)*NGX,k) must be set to A^k_{ij}, for i = 1,2,...,NGX; j = 1,2,...,NGY and k = 1,2,...,7. On exit: A is overwritten.

5: RHS(LDA) -- DOUBLE PRECISION array Input/Output
On entry: RHS(i+(j-1)*NGX) must be set to f_{ij}, for i = 1,2,...,NGX; j = 1,2,...,NGY. On exit: the first NGX*NGY

```
    elements are unchanged and the rest of the array is used as
    workspace.

 6: UB(NGX*NGY) -- DOUBLE PRECISION array           Input/Output
    On entry: UB(i+(j-1)*NGX) must be set to the initial
    estimate for the solution u  . On exit: the corresponding
                               ij
    component of the residual r=f-Au.

 7: MAXIT -- INTEGER                                       Input
    On entry: the maximum permitted number of multigrid
    iterations. If MAXIT = 0, no multigrid iterations are
    performed, but the coarse-grid approximations and incomplete
    Crout decompositions are computed, and may be output if IOUT
    is set accordingly. Constraint: MAXIT >= 0.

 8: ACC -- DOUBLE PRECISION                                Input
    On entry: the required tolerance for convergence of the
    residual 2-norm:


                              / NGX*NGY
                             /  --        2
                 ||r|| =   /   >      (r )
                      2   /    --       k
                        \/     k=1
    where r=f-Au and u is the computed solution. Note that the
    norm is not scaled by the number of equations. The routine
    will stop after fewer than MAXIT iterations if the residual
    2-norm is less than the specified tolerance. (If MAXIT > 0,
    at least one iteration is always performed.)

    If on entry ACC = 0.0, then the machine precision is used as
    a default value for the tolerance; if ACC > 0.0, but ACC is
    less than the machine precision, then the routine will stop
    when the residual 2-norm is less than the machine precision
    and IFAIL will be set to 4. Constraint: ACC >= 0.0.

 9: US(LDA) -- DOUBLE PRECISION array                     Output
    On exit: the residual 2-norm, stored in element US(1).

10: U(LDA) -- DOUBLE PRECISION array                      Output
    On exit: the computed solution u   is returned in U(i+(j-
                                    ij
    1)*NGX), for i = 1,2,...,NGX; j = 1,2,...,NGY.

11: IOUT -- INTEGER                                        Input
    On entry: controls the output of printed information to the
    advisory message unit as returned by X04ABF:
    IOUT = 0
```

No output.

IOUT = 1
The solution u_{ij}, for i = 1,2,...,NGX; j = 1,2,...,NGY.

IOUT = 2
The residual 2-norm after each iteration, with the reduction factor over the previous iteration.

IOUT = 3
As for IOUT = 1 and IOUT = 2.

IOUT = 4
As for IOUT = 3, plus the final residual (as returned in UB).

IOUT = 5
As for IOUT = 4, plus the initial elements of A and RHS.

IOUT = 6
As for IOUT = 5, plus the Galerkin coarse grid approximations.

IOUT = 7
As for IOUT = 6, plus the incomplete Crout decompositions.

IOUT = 8
As for IOUT = 7, plus the residual after each iteration.

The elements A(p,k), the Galerkin coarse grid approximations and the incomplete Crout decompositions are output in the format:

Y-index = j

X-index = i A(p,1) A(p,2) A(p,3) A(p,4) A(p,5) A(p,6) A(p,7)

where p=i+(j-1)*NGX, i = 1,2,...,NGX and j = 1,2,...,NGY.

The vectors U(p), UB(p), RHS(p) are output in matrix form with NGY rows and NGX columns. Where NGX > 10, the NGX values for a given j-value are produced in rows of 10. Values of IOUT > 4 may yield considerable amounts of output. Constraint: 0 <= IOUT <= 8.

12: NUMIT -- INTEGER Output

```
     On exit: the number of iterations performed.

13:  IFAIL -- INTEGER                                Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

IFAIL= 1
     On entry NGX < 3,

     or       NGY < 3,

     or       LDA is too small,

     or       ACC < 0.0,

     or       MAXIT < 0,

     or       IOUT < 0,

     or       IOUT > 8.

IFAIL= 2
     MAXIT iterations have been performed with the residual 2-
     norm decreasing at each iteration but the residual 2-norm
     has not been reduced to less than the specified tolerance
     (see ACC). Examine the progress of the iteration by setting
     IOUT >= 2.

IFAIL= 3
     As for IFAIL = 2, except that at one or more iterations the
     residual 2-norm did not decrease. It is likely that the
     method fails to converge for the given matrix A.

IFAIL= 4
     On entry ACC is less than the machine precision. The routine
     terminated because the residual norm is less than the
     machine precision.

7. Accuracy
```

See ACC (Section 5).

8. Further Comments

The rate of convergence of this routine is strongly dependent upon the number of levels, l, in the multigrid scheme, and thus the choice of NGX and NGY is very important. The user is advised to experiment with different values of NGX and NGY to see the effect they have on the rate of convergence; for example, using a value such as NGX = 65 ($=2^6+1$) followed by NGX = 64 (for which l = 1).

9. Example

The program solves the elliptic partial differential equation

$$U_{xx}-\alpha U_{xy}+U_{yy}=-4, \quad \alpha=1.7$$

on the unit square 0<=x,y<=1, with boundary conditions

$$U=0 \text{ on } \begin{cases} x=0, & (0\le y\le 1) \\ y=0, & (0\le x\le 1) \\ y=1, & (0\le x\le 1) \end{cases} \quad U=1 \text{ on } x=1, \quad 0\le y\le 1.$$

For the equation to be elliptic, (alpha) must be less than 2.

The equation is discretized on a square grid with mesh spacing h in both directions using the following approximations:

Please see figure in printed Reference Manual

$$U_{xx} \simeq \frac{1}{h^2}(U_E-2U_O+U_W)$$

$$U_{yy} \simeq \frac{1}{h^2}(U_N-2U_O+U_S)$$

$$U_{xy} \simeq \frac{1}{2h^2}(U_N-U_{NW}+U_E-2U_O+U_W-U_{SE}+U_S).$$

```
         Thus the following equations are solved:


             1                      1
             -(alpha)u        + (1- -(alpha))u
             2        i-1,j+1       2         i,j+1

              1                                           1

         +(1- -(alpha))u      + (-4+(alpha))u       + (1- -(alpha))u
              2         i-1,j                 ij          2        i+1,j

                                    1                  1
                              + (1- -(alpha))u      +  -(alpha)u
                                    2         i,j-1    2        i+1,j-1

                   2
               =-4h

         The example program is not reproduced here. The source code for
         all example programs is distributed with the NAG Foundation
         Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

———-

Discrete 2nd-order elliptic PDE on rectangular regions

— nagd.ht —

```
\begin{page}{manpageXXd03eef}{NAG Documentation: d03eef}
\beginscroll
\begin{verbatim}



     D03EEF(3NAG)      Foundation Library (12/10/92)      D03EEF(3NAG)



         D03 -- Partial Differential Equations                     D03EEF
                 D03EEF -- NAG Foundation Library Routine Document
```

Note: Before using this routine, please read the Users' Note for your implementation to check implementation-dependent details. The symbol (*) after a NAG routine name denotes a routine that is not included in the Foundation Library.

1. Purpose

D03EEF discretizes a second order elliptic partial differential equation (PDE) on a rectangular region.

2. Specification

```
      SUBROUTINE D03EEF (XMIN, XMAX, YMIN, YMAX, PDEF, BNDY,
     1                   NGX, NGY, LDA, A, RHS, SCHEME, IFAIL)
      INTEGER          NGX, NGY, LDA, IFAIL
      DOUBLE PRECISION XMIN, XMAX, YMIN, YMAX, A(LDA,7), RHS(LDA)
      CHARACTER*1      SCHEME
      EXTERNAL         PDEF, BNDY
```

3. Description

D03EEF discretizes a second order linear elliptic partial differential equation of the form

$$\alpha(x,y)\frac{\partial^2 U}{\partial x^2}+\beta(x,y)\frac{\partial^2 U}{\partial x\partial y}+\gamma(x,y)\frac{\partial^2 U}{\partial y^2}$$

$$+\delta(x,y)\frac{\partial U}{\partial x}+\epsilon(x,y)\frac{\partial U}{\partial y}+\phi(x,y)U=\psi(x,y) \qquad (1)$$

on a rectangular region

$$x_A\le x\le x_B$$

$$y_A\le y\le y_B$$

subject to boundary conditions of the form

$$a(x,y)U+b(x,y)\frac{\partial U}{\partial n}=c(x,y)$$

where $\frac{\partial U}{\partial n}$ denotes the outward pointing normal derivative on the boundary. Equation (1) is said to be elliptic if

$$4\alpha(x,y)\gamma(x,y)\geq(\beta(x,y))^2$$

for all points in the rectangular region. The linear equations produced are in a form suitable for passing directly to the multigrid routine D03EDF.

The equation is discretized on a rectangular grid, with n_x grid points in the x-direction and n_y grid points in the y-direction. The grid spacing used is therefore

$$h_x=(x_B-x_A)/(n_x-1)$$

$$h_y=(y_B-y_A)/(n_y-1)$$

and the co-ordinates of the grid points (x_i,y_j) are

$$x_i=x_A+(i-1)h_x, \quad i=1,2,\ldots,n_x,$$

$$y_j=y_A+(j-1)h_y, \quad j=1,2,\ldots,n_y.$$

At each grid point (x_i,y_j) six neighbouring grid points are used to approximate the partial differential equation, so that the equation is discretized on the following seven-point stencil:

Please see figure in printed Reference Manual

For convenience the approximation u_{ij} to the exact solution $U(x_i,y_j)$ is denoted by u_O, and the neighbouring approximations are labelled according to points of the compass as shown. Where numerical labels for the seven points are required, these are also shown above.

The following approximations are used for the second derivatives:

```
  2
dd U   1
----~= --(u -2u +u )
  2     2  E   O  W
dd x   h
        x

  2
dd U   1
----~= --(u -2u +u )
  2     2  N   O  S
dd y   h
        y

   2
 dd U    1
------~= -----(u -u  +u -2u +u -u  +u ).
ddxddy 2h h   N  NW  E   O  W  SE  S
         x y
```

Two possible schemes may be used to approximate the first
derivatives:

Central Differences

```
ddU     1
---~=  ---(u -u )
ddx    2h   W  E
         x

ddU     1
---~=  ---(u -u )
ddy    2h   N  S
         y
```

Upwind Differences

```
ddU    1
---~=  --(u -u )if (delta)(x,y)>0
ddx    h   E  O
        x

ddU    1
---~=  --(u -u )if (delta)(x,y)<0
ddx    h   O  W
        x

ddU    1
```

```
     ---~=  --(u -u )if (epsilon)(x,y)>0
     ddy    h   N  O
             y

     ddU    1
     ---~=  --(u -u )if (epsilon)(x,y)<0.
     ddy    h   O  S
             y
```

Central differences are more accurate than upwind differences, but upwind differences may lead to a more diagonally dominant matrix for those problems where the coefficients of the first derivatives are significantly larger than the coefficients of the second derivatives.

The approximations used for the first derivatives may be written in a more compact form as follows:

```
     ddU     1
     ---~=  ---((k -1)u -2k u +(k +1)u )
     ddx    2h (  x    W   x O   x    E)
              x

     ddU     1
     ---~=  ---((k -1)u -2k u +(k +1)u )
     ddy    2h (  y    S   y O   y    N)
              y
```

```
where k =sign (delta) and k =sign (epsilon) for upwind
       x                   y
differences, and k =k =0 for central differences.
                  x  y
```

At all points in the rectangular domain, including the boundary, the coefficients in the partial differential equation are evaluated by calling the user-supplied subroutine PDEF, and applying the approximations. This leads to a seven-diagonal system of linear equations of the form:

```
    6           7
   A  u        +A  u
    ij i-1,j+1   ij i,j+1
     3          4       5
   +A  u       +A  u   +A  u
     ij i-1,j   ij ij   ij i+1,j
       1          2
     +A  u      +A  u         =f  , i=1,2,...,n ;j=1,2,...,n ,
       ij i,j-1   ij i+1,j-1   ij             x            y
```

where the coefficients are given by

$$A^1_{ij} = (\beta)(x_i,y_j)\frac{1}{2h_x h_y} + (\gamma)(x_i,y_j)\frac{1}{h_y^2} + (\epsilon)(x_i,y_j)\frac{1}{2h_y}(k_y - 1)$$

$$A^2_{ij} = -(\beta)(x_i,y_j)\frac{1}{2h_x h_y}$$

$$A^3_{ij} = (\alpha)(x_i,y_j)\frac{1}{h_x^2} + (\beta)(x_i,y_j)\frac{1}{2h_x h_y} + (\delta)(x_i,y_j)\frac{1}{2h_x}(k_x - 1)$$

$$A^4_{ij} = -(\alpha)(x_i,y_j)\frac{2}{h_x^2} - (\beta)(x_i,y_j)\frac{1}{h_x h_y} - (\gamma)(x_i,y_j)\frac{2}{h_y^2}$$

$$-(\delta)(x_i,y_j)\frac{k_y}{h_x} - (\epsilon)(x_i,y_j)\frac{k_y}{h_y} - (\phi)(x_i,y_j)$$

$$A^5_{ij} = (\alpha)(x_i,y_j)\frac{1}{h_x^2} + (\beta)(x_i,y_j)\frac{1}{2h_x h_y} + (\delta)(x_i,y_j)\frac{1}{2h_x}(k_x + 1)$$

$$A^6_{ij} = -(\beta)(x_i,y_j)\frac{1}{2h_x h_y}$$

$$A^7_{ij} = (\beta)(x_i,y_j)\frac{1}{2h_x h_y} + (\gamma)(x_i,y_j)\frac{1}{h_y^2} + (\epsilon)(x_i,y_j)\frac{1}{2h_y}(k_y + 1)$$

$$f_{ij} = (\psi)(x_i,y_j)$$

These equations then have to be modified to take account of the boundary conditions. These may be Dirichlet (where the solution is given), Neumann (where the derivative of the solution is given), or mixed (where a linear combination of solution and derivative is given).

If the boundary conditions are Dirichlet, there are an infinity of possible equations which may be applied:

$$\mu u_{ij} = \mu f_{ij}, \quad \mu \neq 0. \qquad (2)$$

If D03EDF is used to solve the discretized equations, it turns out that the choice of (mu) can have a dramatic effect on the rate of convergence, and the obvious choice (mu)=1 is not the best. Some choices may even cause the multigrid method to fail altogether. In practice it has been found that a value of the same order as the other diagonal elements of the matrix is best, and the following value has been found to work well in practice:

$$\mu = \min_{ij}\left(-\left\{\frac{2}{h_x^2}+\frac{2}{h_y^2}\right\}, A_{ij}^4\right).$$

If the boundary conditions are either mixed or Neumann (i.e., B /= 0 on return from the user-supplied subroutine BNDY), then one of the points in the seven-point stencil lies outside the domain. In this case the normal derivative in the boundary conditions is used to eliminate the 'fictitious' point, $u_{outside}$:

$$\frac{\partial U}{\partial n} \sim= \frac{1}{2h}(u_{outside} - u_{inside}). \qquad (3)$$

It should be noted that if the boundary conditions are Neumann and (phi)(x,y)==0, then there is no unique solution. The routine returns with IFAIL = 5 in this case, and the seven-diagonal matrix is singular.

The four corners are treated separately. The user-supplied subroutine BNDY is called twice, once along each of the edges meeting at the corner. If both boundary conditions at this point are Dirichlet and the prescribed solution values agree, then this value is used in an equation of the form (2). If the prescribed solution is discontinuous at the corner, then the average of the two values is used. If one boundary condition is Dirichlet and the other is mixed, then the value prescribed by the Dirichlet condition is used in an equation of the form given above.

Finally, if both conditions are mixed or Neumann, then two 'fictitious' points are eliminated using two equations of the form (3).

It is possible that equations for which the solution is known at all points on the boundary, have coefficients which are not defined on the boundary. Since this routine calls the user-supplied subroutine PDEF at all points in the domain, including boundary points, arithmetic errors may occur in the user's routine PDEF which this routine cannot trap. If the user has an equation with Dirichlet boundary conditions (i.e., B = 0 at all points on the boundary), but with PDE coefficients which are singular on the boundary, then D03EDF could be called directly only using interior grid points with the user's own discretization.

After the equations have been set up as described above, they are checked for diagonal dominance. That is to say,

$$|A^4_{ij}| > \sum_{k\neq 4} |A^k_{ij}|, \quad i=1,2,\ldots,n_x ; \; j=1,2,\ldots,n_y .$$

If this condition is not satisfied then the routine returns with IFAIL = 6. The multigrid routine D03EDF may still converge in this case, but if the coefficients of the first derivatives in the partial differential equation are large compared with the coefficients of the second derivative, the user should consider using upwind differences (SCHEME = 'U').

Since this routine is designed primarily for use with D03EDF, this document should be read in conjunction with the document for that routine.

4. References

[1] Wesseling P (1982) MGD1 - A Robust and Efficient Multigrid Method. Multigrid Methods. Lecture Notes in Mathematics. 960 Springer-Verlag. 614--630.

5. Parameters

1: XMIN -- DOUBLE PRECISION Input

2: XMAX -- DOUBLE PRECISION Input

On entry: the lower and upper x co-ordinates of the rectangular region respectively, x_A and x_B. Constraint: XMIN < XMAX.

```
3:  YMIN -- DOUBLE PRECISION                              Input

4:  YMAX -- DOUBLE PRECISION                              Input
    On entry: the lower and upper y co-ordinates of the
    rectangular region respectively, y  and y . Constraint: YMIN
                                      A      B
    < YMAX.

5:  PDEF -- SUBROUTINE, supplied by the user.
                                            External Procedure
    PDEF must evaluate the functions (alpha)(x,y), (beta)(x,y),
    (gamma)(x,y), (delta)(x,y), (epsilon)(x,y), (phi)(x,y) and
    (psi)(x,y) which define the equation at a general point
    (x,y).

    Its specification is:

           SUBROUTINE PDEF (X, Y, ALPHA, BETA, GAMMA,
          1                 DELTA, EPSLON, PHI, PSI)
           DOUBLE PRECISION X, Y, ALPHA, BETA, GAMMA, DELTA,
          1                 EPSLON, PHI, PSI

     1:  X -- DOUBLE PRECISION                            Input

     2:  Y -- DOUBLE PRECISION                            Input
         On entry: the x and y co-ordinates of the point at
         which the coefficients of the partial differential
         equation are to be evaluated. 8

     3:  ALPHA -- DOUBLE PRECISION                       Output

     4:  BETA -- DOUBLE PRECISION                        Output

     5:  GAMMA -- DOUBLE PRECISION                       Output

     6:  DELTA -- DOUBLE PRECISION                       Output

     7:  EPSLON -- DOUBLE PRECISION                      Output

     8:  PHI -- DOUBLE PRECISION                         Output

     9:  PSI -- DOUBLE PRECISION                         Output
         On exit: ALPHA, BETA, GAMMA, DELTA, EPSLON, PHI and PSI
         must be set to the values of (alpha)(x,y), (beta)(x,y),
         (gamma)(x,y), (delta)(x,y), (epsilon)(x,y), (phi)(x,y)
         and (psi)(x,y) respectively at the point specified by X
         and Y.
    PDEF must be declared as EXTERNAL in the (sub)program
    from which D03EEF is called. Parameters denoted as
```

```
    Input must not be changed by this procedure.

6:  BNDY -- SUBROUTINE, supplied by the user.
                                              External Procedure
    BNDY must evaluate the functions a(x,y), b(x,y), and c(x,y)
    involved in the boundary conditions.

    Its specification is:

           SUBROUTINE BNDY (X, Y, A, B, C, IBND)
           INTEGER          IBND
           DOUBLE PRECISION X, Y, A, B, C

     1:  X -- DOUBLE PRECISION                             Input

     2:  Y -- DOUBLE PRECISION                             Input
         On entry: the x and y co-ordinates of the point at
         which the boundary conditions are to be evaluated.

     3:  A -- DOUBLE PRECISION                            Output

     4:  B -- DOUBLE PRECISION                            Output

     5:  C -- DOUBLE PRECISION                            Output
         On exit: A, B and C must be set to the values of the
         functions appearing in the boundary conditions.

     6:  IBND -- INTEGER                                   Input
         On entry: specifies on which boundary the point (X,Y)
         lies. IBND = 0, 1, 2 or 3 according as the point lies
         on the bottom, right, top or left boundary.
    BNDY must be declared as EXTERNAL in the (sub)program
    from which D03EEF is called. Parameters denoted as
    Input must not be changed by this procedure.

7:  NGX -- INTEGER                                         Input

8:  NGY -- INTEGER                                         Input
    On entry: the number of interior grid points in the x- and y
    -directions respectively, n  and n . If the seven-diagonal
                               x      y
    equations are to be solved by D03EDF, then NGX-1 and NGY-1
    should preferably be divisible by as high a power of 2 as
    possible. Constraint: NGX >= 3, NGY >= 3.

9:  LDA -- INTEGER                                         Input
    On entry:
    the first dimension of the array A as declared in the
    (sub)program from which D03EEF is called.
    Constraint: if only the seven-diagonal equations are
```

```
     required, then LDA >= NGX*NGY. If a call to this routine is
     to be followed by a call to D03EDF to solve the seven-
     diagonal linear equations, LDA >= (4*(NGX+1)*(NGY+1))/3.

     Note: this routine only checks the former condition. D03EDF,
     if called, will check the latter condition.

10:  A(LDA,7) -- DOUBLE PRECISION array                    Output
     On exit: A(i,j), for i=1,2,...,NGX*NGY; j = 1,2,...,7,
     contains the seven-diagonal linear equations produced by the
     discretization described above. If LDA > NGX*NGY, the
     remaining elements are not referenced by the routine, but if
     LDA >= (4*(NGX+1)*(NGY+1))/3 then the array A can be passed
     directly to D03EDF, where these elements are used as
     workspace.

11:  RHS(LDA) -- DOUBLE PRECISION array                    Output
     On exit: the first NGX*NGY elements contain the right-hand
     sides of the seven-diagonal linear equations produced by the
     discretization described above. If LDA > NGX*NGY, the
     remaining elements are not referenced by the routine, but if
     LDA >= (4*(NGY+1)*(NGY+1))/3 then the array RHS can be
     passed directly to D03EDF, where these elements are used as
     workspace.

12:  SCHEME -- CHARACTER*1                                  Input
     On entry: the type of approximation to be used for the first
     derivatives which occur in the partial differential
     equation.

     If SCHEME = 'C', then central differences are used.

     If SCHEME = 'U', then upwind differences are used.
     Constraint: SCHEME = 'C' or 'U'.

     Note: generally speaking, if at least one of the
     coefficients multiplying the first derivatives (DELTA or
     EPSLON as returned by PDEF) are large compared with the
     coefficients multiplying the second derivatives, then upwind
     differences may be more appropriate. Upwind differences are
     less accurate than central differences, but may result in
     more rapid convergence for strongly convective equations.
     The easiest test is to try both schemes.

13:  IFAIL -- INTEGER                                Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. Users who are
     unfamiliar with this parameter should refer to the Essential
     Introduction for details.

     On exit: IFAIL = 0 unless the routine detects an error or
```

gives a warning (see Section 6).

For this routine, because the values of output parameters may be useful even if IFAIL /=0 on exit, users are recommended to set IFAIL to -1 before entry. It is then essential to test the value of IFAIL on exit.

6. Error Indicators and Warnings

Errors or warnings specified by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL= 1

On entry XMIN >= XMAX,

or YMIN >= YMAX,

or NGX < 3,

or NGY < 3,

or LDA < NGX*NGY,

or SCHEME is not one of 'C' or 'U'.

IFAIL= 2

At some point on the boundary there is a derivative in the boundary conditions (B /= 0 on return from a BNDY) and there is a non-zero coefficient of the mixed derivative $\frac{\partial^2 U}{\partial x \partial y}$ (BETA /= 0 on return from PDEF).

IFAIL= 3

A null boundary has been specified, i.e., at some point both A and B are zero on return from a call to BNDY.

IFAIL= 4

The equation is not elliptic, i.e., $4*ALPHA*GAMMA<BETA^2$ after a call to PDEF. The discretization has been completed, but the convergence of D03EDF cannot be guaranteed.

IFAIL= 5

The boundary conditions are purely Neumann (only the derivative is specified) and there is, in general, no unique solution.

```
IFAIL= 6
     The equations were not diagonally dominant. (See Section 3).

7. Accuracy

Not applicable.

8. Further Comments

If this routine is used as a pre-processor to the multigrid
routine D03EDF it should be noted that the rate of convergence of
that routine is strongly dependent upon the number of levels in
the multigrid scheme, and thus the choice of NGX and NGY is very
important.

9. Example

The program solves the elliptic partial differential equation

                   2     2
                 dd U  dd U   { ddU  ddU}
                 ----+ ----+50{ ---+ ---}=f(x,y)
                    2     2   { ddx  ddy}
                 ddx   ddy

on the unit square 0<=x, y<=1, with boundary conditions

      ddU
      --- given on x=0 and y=0,
      ddn

     U given on x=1 and y=1.

The function f(x,y) and the exact form of the boundary conditions
are derived from the exact solution U(x,y)=sinxsiny.

The equation is first solved using central differences. Since the
coefficients of the first derivatives are large, the linear
equations are not diagonally dominated, and convergence is slow.
The equation is solved a second time with upwind differences,
showing that convergence is more rapid, but the solution is less
accurate.

The example program is not reproduced here. The source code for
all example programs is distributed with the NAG Foundation
Library software and should be available on-line.
```

```
\end{verbatim}
\endscroll
```

```
\end{page}
```

Helmholtz equation in 3 dimensions

— nagd.ht —

```
\begin{page}{manpageXXd03faf}{NAG Documentation: d03faf}
\beginscroll
\begin{verbatim}



     D03FAF(3NAG)      Foundation Library (12/10/92)      D03FAF(3NAG)



          D03 -- Partial Differential Equations                      D03FAF
                  D03FAF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          D03FAF solves the Helmholtz equation in Cartesian co-ordinates in
          three dimensions using the standard seven-point finite difference
          approximation. This routine is designed to be particularly
          efficient on vector processors.

          2. Specification

                 SUBROUTINE D03FAF (XS, XF, L, LBDCND, BDXS, BDXF, YS, YF,
                1                   M, MBDCND, BDYS, BDYF, ZS, ZF, N,
                2                   NBDCND, BDZS, BDZF, LAMBDA, LDIMF,
                3                   MDIMF, F, PERTRB, W, LWRK, IFAIL)
                 INTEGER          L, LBDCND, M, MBDCND, N, NBDCND, LDIMF,
                1                 MDIMF, LWRK, IFAIL
                 DOUBLE PRECISION XS, XF, BDXS(MDIMF,N+1), BDXF(MDIMF,N+1),
                1                 YS, YF, BDYS(LDIMF,N+1), BDYF(LDIMF,N+1),
                2                 ZS, ZF, BDZS(LDIMF,M+1), BDZF(LDIMF,M+1),
                3                 LAMBDA, F(LDIMF,MDIMF,N+1), PERTRB, W
                4                 (LWRK)
```

3. Description

D03FAF solves the three-dimensional Helmholtz equation in cartesian co-ordinates:

```
  2     2     2
dd u  dd u  dd u
----+ ----+ ----+(lambda)u=f(x,y,z)
  2     2     2
dd x  dd y  dd z
```

This subroutine forms the system of linear equations resulting from the standard seven-point finite difference equations, and then solves the system using a method based on the fast Fourier transform (FFT) described by Swarztrauber [1]. This subroutine is based on the routine HW3CRT from FISHPACK (see Swarztrauber and Sweet [2]).

More precisely, the routine replaces all the second derivatives by second-order central difference approximations, resulting in a block tridiagonal system of linear equations. The equations are modified to allow for the prescribed boundary conditions. Either the solution or the derivative of the solution may be specified on any of the boundaries, or the solution may be specified to be periodic in any of the three dimensions. By taking the discrete Fourier transform in the x- and y-directions, the equations are reduced to sets of tridiagonal systems of equations. The Fourier transforms required are computed using the multiple FFT routines found in Chapter C06 of the NAG Fortran Library.

4. References

[1] Swarztrauber P N (1984) Fast Poisson Solvers. Studies in Numerical Analysis. (ed G H Golub) Mathematical Association of America.

[2] Swarztrauber P N and Sweet R A (1979) Efficient Fortran Subprograms for the Solution of Separable Elliptic Partial Differential Equations. ACM Trans. Math. Softw. 5 352--364.

5. Parameters

1: XS -- DOUBLE PRECISION Input
On entry: the lower bound of the range of x, i.e., XS <= x <= XF. Constraint: XS < XF.

2: XF -- DOUBLE PRECISION Input
On entry: the upper bound of the range of x, i.e., XS <= x <= XF. Constraint: XS < XF.

3: L -- INTEGER Input

On entry: the number of panels into which the interval (XS,XF) is subdivided. Hence, there will be L+1 grid points in the x-direction given by x_i=XS+(i-1)*(delta)x, for i=1,2,...,L+1, where (delta)x=(XF-XS)/L is the panel width. Constraint: L >= 5.

4: LBDCND -- INTEGER Input

On entry: indicates the type of boundary conditions at x = XS and x = XF.

LBDCND = 0
  if the solution is periodic in x, i.e., u(XS,y,z)=u(XF,y,z).

LBDCND = 1
  if the solution is specified at x = XS and x = XF.

LBDCND = 2
  if the solution is specified at x = XS and the derivative of the solution with respect to x is specified at x = XF.

LBDCND = 3
  if the derivative of the solution with respect to x is specified at x = XS and x = XF.

LBDCND = 4
  if the derivative of the solution with respect to x is specified at x = XS and the solution is specified at x = XF.

Constraint: 0 <= LBDCND <= 4.

5: BDXS(MDIMF,N+1) -- DOUBLE PRECISION array Input

On entry: the values of the derivative of the solution with respect to x at x = XS. When LBDCND = 3 or 4, BDXS(j,k)=u_x(XS,y_j,z_k), for j=1,2,...,M+1; k=1,2,...,N+1.

When LBDCND has any other value, BDXS is not referenced.

6: BDXF(MDIMF,N+1) -- DOUBLE PRECISION array Input

On entry: the values of the derivative of the solution with respect to x at x = XF. When LBDCND = 2 or 3, BDXF(j,k) = u_x(XF,y_i,z_k), for j=1,2,...,M+1; k=1,2,...,N+1.

When LBDCND has any other value, BDXF is not referenced.

7: YS -- DOUBLE PRECISION Input

```
     On entry: the lower bound of the range of y, i.e., YS <= y
     <= YF. Constraint: YS < YF.

 8:  YF -- DOUBLE PRECISION                                 Input
     On entry: the upper bound of the range of y, i.e., YS <= y
     <= YF. Constraint: YS < YF.

 9:  M -- INTEGER                                           Input
     On entry: the number of panels into which the interval
     (YS,YF) is subdivided. Hence, there will be M+1 grid points
     in the y-direction given by y  = YS + (j-1)*(delta)y for
                                  j
     j=1,2,...,M+1, where (delta)y=(YF-YS)/M is the panel width.
     Constraint: M >= 5.

10:  MBDCND -- INTEGER                                      Input
     On entry: indicates the type of boundary conditions at y =
     YS and y = YF.
     MBDCND = 0
           if the solution is periodic in y, i.e.,
           u(x,YF,z)=u(x,YS,z).

     MBDCND = 1
           if the solution is specified at y = YS and y = YF.

     MBDCND = 2
           if the solution is specified at y = YS and the
           derivative of the solution with respect to y is
           specified at y = YF.

     MBDCND = 3
           if the derivative of the solution with respect to y is
           specified at y = YS and y = YF.

     MBDCND = 4
           if the derivative of the solution with respect to y is
           specified at y = YS and the solution is specified at y
           = YF.
      Constraint: 0 <= MBDCND <= 4.

11:  BDYS(LDIMF,N+1) -- DOUBLE PRECISION array              Input
     On entry: the values of the derivative of the solution with
     respect to y at y = YS. When MBDCND = 3 or 4, BDYS
     (i,k)=u (x ,y ,z ), for i=1,2,...,L+1; k=1,2,...,N+1.
            y  i  s  k

     When MBDCND has any other value, BDYS is not referenced.

12:  BDYF(LDIMF,N+1) -- DOUBLE PRECISION array              Input
     On entry: the values of the derivative of the solution with
```

```
     respect to y at y = YF. When MBDCND = 2 or 3, BDYF
     (i,k)=u (x ,YF,z ), for i=1,2,...,L+1; k=1,2,...,N+1.
            y  i     k

     When MBDCND has any other value, BDYF is not referenced.

13:  ZS -- DOUBLE PRECISION                                  Input
     On entry: the lower bound of the range of z, i.e., ZS <= z
     <= ZF. Constraint: ZS < ZF.

14:  ZF -- DOUBLE PRECISION                                  Input
     On entry: the upper bound of the range of z, i.e., ZS <= z
     <= ZF. Constraint: ZS < ZF.

15:  N -- INTEGER                                            Input
     On entry: the number of panels into which the interval
     (ZS,ZF) is subdivided. Hence, there will be N+1 grid points
     in the z-direction given by z =ZS+(k-1)*(delta)z, for
                                  k
     k=1,2,...,N+1, where (delta)z=(ZF-ZS)/N is the panel width.
     Constraint: N >= 5.

16:  NBDCND -- INTEGER                                       Input
     On entry: specifies the type of boundary conditions at z =
     ZS and z = ZF.
     NBDCND = 0
           if the solution is periodic in z, i.e.,
           u(x,y,ZF)=u(x,y,ZS).

     NBDCND = 1
           if the solution is specified at z = ZS and z = ZF.

     NBDCND = 2
           if the solution is specified at z = ZS and the
           derivative of the solution with respect to z is
           specified at z = ZF.

     NBDCND = 3
           if the derivative of the solution with respect to z is
           specified at z = ZS and z = ZF.

     NBDCND = 4
           if the derivative of the solution with respect to z is
           specified at z = ZS and the solution is specified at z
           = ZF.
      Constraint: 0 <= NBDCND <= 4.

17:  BDZS(LDIMF,M+1) -- DOUBLE PRECISION array               Input
     On entry: the values of the derivative of the solution with
     respect to z at z = ZS. When NBDCND = 3 or 4, BDZS
```

```
     (i,j)=u (x ,y ,ZS)=u(x,y,z), for i=1,2,...,L+1;
            z  i  j
     j=1,2,...,M+1.

     When NBDCND has any other value, BDZS is not referenced.

18:  BDZF(LDIMF,M+1) -- DOUBLE PRECISION array              Input
     On entry: the values of the derivative of the solution with
     respect to z at z = ZF. When NBDCND = 2 or 3, BDZF
     (i,j)=u (x ,y ,ZF)=u(x,y,z), for i=1,2,...,L+1;
            z  i  j
     j=1,2,...,M+1.

     When NBDCND has any other value, BDZF is not referenced.

19:  LAMBDA -- DOUBLE PRECISION                             Input
     On entry: the constant (lambda) in the Helmholtz equation.
     For certain positive values of (lambda) a solution to the
     differential equation may not exist, and close to these
     values the solution of the discretized problem will be
     extremely ill-conditioned. If (lambda)>0, then D03FAF will
     set IFAIL to 3, but will still attempt to find a solution.
     However, since in general the values of (lambda) for which
     no solution exists cannot be predicted a priori, the user is
     advised to treat any results computed with (lambda)>0 with
     great caution.

20:  LDIMF -- INTEGER                                       Input
     On entry:
     the first dimension of the arrays F, BDYS, BDYF, BDZS and
     BDZF as declared in the (sub)program from which D03FAF is
     called.
     Constraint: LDIMF >= L + 1.

21:  MDIMF -- INTEGER                                       Input
     On entry: the second dimension of the array F and
     the first dimension of the arrays BDXS and BDXF as declared
     in the (sub)program from which D03FAF is called.
     Constraint: MDIMF >= M + 1.

22:  F(LDIMF,MDIMF,N+1) -- DOUBLE PRECISION array    Input/Output
     On entry: the values of the right-side of the Helmholtz
     equation and boundary values (if any).

     F(i,j,k)=f(x ,y ,z ) i=2,3,...,L, j=2,3,...,M and k
                 i  j  k
     =2,3,...,N.

     On the boundaries F is defined by
     LBDCND    F(1,j,k)  F(L+1,j,k)
```

```
0         f(XS,y ,z )f(XS,y ,z )
                j  k      j  k

1         u(XS,y ,z )u(XF,y ,z )
                j  k      j  k

2         u(XS,y ,z )f(XF,y ,z )  j=1,2,...,M+1
                j  k      j  k

3         f(XS,y ,z )f(XF,y ,z )  k=1,2,...,N+1
                j  k      j  k

4         f(XS,y ,z )u(XF,y ,z )
                j  k      j  k


MBDCND    F(i,1,k)  F(
                    i,M+1,k)

0         f(x ,YS,z )f(x ,YS,z )
             i     k   i     k

1         u(x ,YS,z )u(x ,YF,z )
             i     k   i     k

2         u(x ,YS,z )f(x ,YF,z )  i=1,2,...,L+1
             i     k   i     k

3         f(x ,YS,z )f(x ,YF,z )  k=1,2,...,N+1
             i     k   i     k

4         f(x ,YS,z )u(x ,YF,z )
             i     k   i     k


NBDCND    F(i,j,1)  F(i,j,N+1
                    )

0         f(x ,y ,ZS)f(x ,y ,ZS)
             i  j      i  j

1         u(x ,y ,ZS)u(x ,y ,ZF)
             i  j      i  j

2         u(x ,y ,ZS)f(x ,y ,ZF)  i=1,2,...,L+1
             i  j      i  j
```

```
3         f(x ,y ,ZS)f(x ,y ,ZF)  j=1,2,...,M+1
             i  j     i  j

4         f(x ,y ,ZS)u(x ,y ,ZF)
             i  j     i  j
Note: if the table calls for both the solution u and the
right-hand side f on a boundary, then the solution must be
specified. On exit: F contains the solution u(i,j,k) of the
finite difference approximation for the grid point (
x ,y ,z ) for i=1,2,...,L+1, j=1,2,...,M+1 and k=1,2,...,N+1.
 i  j  k
```

```
23:  PERTRB -- DOUBLE PRECISION                            Output
     On exit: PERTRB = 0, unless a solution to Poisson's
     equation ((lambda)=0) is required with a combination of
     periodic or derivative boundary conditions (LBDCND, MBDCND
     and NBDCND = 0 or 3). In this case a solution may not exist.
     PERTRB is a constant, calculated and subtracted from the
     array F, which ensures that a solution exists. D03FAF then
     computes this solution, which is a least-squares solution to
     the original approximation. This solution is not unique and
     is unnormalised. The value of PERTRB should be small
     compared to the right-hand side F, otherwise a solution has
     been obtained to an essentially different problem. This
     comparison should always be made to insure that a meaningful
     solution has been obtained.

24:  W(LWRK) -- DOUBLE PRECISION array                  Workspace

25:  LWRK -- INTEGER                                        Input
     On entry:
     the dimension of the array W as declared in the (sub)program
     from which D03FAF is called.
     LRWK>=2*(N+1)*max(L,M)+3*L+3*M+4*N+6 is an upper bound on
     the required size of W. If LWRK is too small, the routine
     exits with IFAIL = 2, and if on entry IFAIL = 0 or IFAIL = -
     1, a message is output giving the exact value of LWRK
     required to solve the current problem.

26:  IFAIL -- INTEGER                                Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. Users who are
     unfamiliar with this parameter should refer to the Essential
     Introduction for details.

     On exit: IFAIL = 0 unless the routine detects an error or
     gives a warning (see Section 6).

     For this routine, because the values of output parameters
     may be useful even if IFAIL /=0 on exit, users are
     recommended to set IFAIL to -1 before entry. It is then
```

```
     essential to test the value of IFAIL on exit.

6. Error Indicators and Warnings

Errors or warnings specified by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

IFAIL= 1
     On entry XS >= XF,

     or       L < 5,

     or       LBDCND < 0,

     or       LBDCND > 4,

     or       YS >= YF,

     or       M < 5,

     or       MBDCND < 0,

     or       MBDCND > 4,

     or       ZS >= ZF,

     or       N < 5,

     or       NBDCND < 0,

     or       NBDCND > 4,

     or       LDIMF < L + 1 > 0,

     or       MDIMF < M + 1.

IFAIL= 2
     On entry LWRK is too small.

IFAIL= 3
     On entry (lambda) > 0.

7. Accuracy

None.

8. Further Comments
```

```
     The execution time is roughly proportional to
     L*M*N*(log L+log M+5), but also depends on input parameters
               2     2
     LBDCND and MBDCND.

     9. Example

     The example solves the Helmholz equation

                      2     2     2
                    dd u  dd u  dd u
                    ----+ ----+ ----+(lambda)u=f(x,y,z)
                       2     2     2
                    ddx   ddy   ddz

                                              (pi)
     for (x,y,z)is in [0,1]*[0,2(pi)]*[0, ----] where (lambda)=-2, and
                                               2
     f(x,y,z) is derived from the exact solution

                                    4
                         u(x,y,z)=x sin(y)cos(z).

     The equation is subject to the following boundary conditions,
     again derived from the exact solution given above.

          u(0,y,z) and u(1,y,z) are prescribed (i.e., LBDCND = 1).

          u(x,0,z)=u(x,2(pi),z) (i.e., MBDCND = 0).

                                (pi)
          u(x,y,0) and u (x,y, ----) are prescribed (i.e. NBDCND = 2).
                        x       2

     The example program is not reproduced here. The source code for
     all example programs is distributed with the NAG Foundation
     Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

22.4 nage.ht

Interpolation

— nage.ht —

```
\begin{page}{manpageXXe01}{NAG Documentation: e01}
\beginscroll
\begin{verbatim}



     E01(3NAG)        Foundation Library (12/10/92)        E01(3NAG)



          E01 -- Interpolation                          Introduction -- E01
                                      Chapter E01
                                     Interpolation

          1. Scope of the Chapter

          This chapter is concerned with the interpolation of a function of
          one or two variables. When provided with the value of the
          function (and possibly one or more of its lowest-order
          derivatives) at each of a number of values of the variable(s),
          the routines provide either an interpolating function or an
          interpolated value. For some of the interpolating functions,
          there are supporting routines to evaluate, differentiate or
          integrate them.

          2. Background to the Problems

          In motivation and in some of its numerical processes, this
          chapter has much in common with Chapter E02 (Curve and Surface
          Fitting). For this reason, we shall adopt the same terminology
          and refer to dependent variable and independent variable(s)
          instead of function and variable(s). Where there is only one
          independent variable, we shall denote it by x and the dependent
          variable by y. Thus, in the basic problem considered in this
          chapter, we are given a set of distinct values x ,x ,...,x  of x
                                                          1  2      m
          and a corresponding set of values y ,y ,...,y  of y, and we shall
                                             1  2      m
          describe the problem as being one of interpolating the data
          points (x ,y ), rather than interpolating a function. In modern
                   r  r
          usage, however, interpolation can have either of two rather
```

different meanings, both relevant to routines in this chapter. They are

(a) the determination of a function of x which takes the value y_r at $x=x_r$, for r=1,2,...,m (an interpolating function or interpolant),

(b) the determination of the value (interpolated value or interpolate) of an interpolating function at any given value, say $\hat{x}$, of x within the range of the x_r (so as to estimate the value at $\hat{x}$ of the function underlying the data).

The latter is the older meaning, associated particularly with the use of mathematical tables. The term 'function underlying the data', like the other terminology described above, is used so as to cover situations additional to those in which the data points have been computed from a known function, as with a mathematical table. In some contexts, the function may be unknown, perhaps representing the dependency of one physical variable on another, say temperature upon time.

Whether the underlying function is known or unknown, the object of interpolation will usually be to approximate it to acceptable accuracy by a function which is easy to evaluate anywhere in some range of interest. Piecewise polynomials such as cubic splines (see Section 2.2 of the E02 Chapter Introduction for definitions of terms in this case), being easy to evaluate and also capable of approximating a wide variety of functions, are the types of function mostly used in this chapter as interpolating functions.

Piecewise polynomials also, to a large extent, avoid the well-known problem of large unwanted fluctuations which can arise when interpolating a data set with a simple polynomial. Fluctuations can still arise but much less frequently and much less severely than with simple polynomials. Unwanted fluctuations are avoided altogether by a routine using piecewise cubic polynomials having only first derivative continuity. It is designed especially for monotonic data, but for other data still provides an interpolant which increases, or decreases, over the same intervals as the data.

The concept of interpolation can be generalised in a number of ways. For example, we may be required to estimate the value of the underlying function at a value $\hat{x}$ outside the range of the data. This is the process of extrapolation. In general, it is a good deal less accurate than interpolation and is to be avoided whenever possible.

Interpolation can also be extended to the case of two independent variables. We shall denote these by x and y, and the dependent variable by f. Methods used depend markedly on whether or not the data values of f are given at the intersections of a rectangular mesh in the (x,y)-plane. If they are, bicubic splines (see Section 2.3.2 of the E02 Chapter Introduction) are very suitable and usually very effective for the problem. For other cases, perhaps where the f values are quite arbitrarily scattered in the (x,y)-plane, polynomials and splines are not at all appropriate and special forms of interpolating function have to be employed. Many such forms have been devised and two of the most successful are in routines in this chapter. They both have continuity in first, but not higher, derivatives.

2.1. References

[1] Froberg C E (1970) Introduction to Numerical Analysis. Addison-Wesley (2nd Edition).

[2] Dahlquist G and Bjork A (1974) Numerical Methods. Prentice-Hall.

3. Recommendations on Choice and Use of Routines

3.1. General

Before undertaking interpolation, in other than the simplest cases, the user should seriously consider the alternative of using a routine from Chapter E02 to approximate the data by a polynomial or spline containing significantly fewer coefficients than the corresponding interpolating function. This approach is much less liable to produce unwanted fluctuations and so can often provide a better approximation to the function underlying the data.

When interpolation is employed to approximate either an underlying function or its values, the user will need to be satisfied that the accuracy of approximation achieved is adequate. There may be a means for doing this which is particular to the application, or the routine used may itself provide a means. In other cases, one possibility is to repeat the interpolation using one or more extra data points, if they are available, or otherwise one or more fewer, and to compare the results. Other possibilities, if it is an interpolating function which is determined, are to examine the function graphically, if that gives sufficient accuracy, or to observe the behaviour of the differences in a finite-difference table, formed from evaluations of the interpolating function at equally-spaced values of x over the range of interest. The spacing should be small enough to cause the typical size of the differences to decrease as the order of difference increases.

3.2. One Independent Variable

E01BAF computes an interpolating cubic spline, using a particular choice for the set of knots which has proved generally satisfactory in practice. If the user wishes to choose a different set, a cubic spline routine from Chapter E02, namely E02BAF, may be used in its interpolating mode, setting NCAP7 = M+ 4 and all elements of the parameter W to unity. These routines provide the interpolating function in B-spline form (see Section 2.2.2 in the E02 Chapter Introduction). Routines for evaluating, differentiating and integrating this form are discussed in Section 3.7 of the E02 Chapter Introduction.

The cubic spline does not always avoid unwanted fluctuations, especially when the data show a steep slope close to a region of small slope, or when the data inadequately represent the underlying curve. In such cases, E01BEF can be very useful. It derives a piecewise cubic polynomial (with first derivative continuity) which, between any adjacent pair of data points, either increases all the way, or decreases all the way (or stays constant). It is especially suited to data which are monotonic over their whole range.

In this routine, the interpolating function is represented simply by its value and first derivative at the data points. Supporting routines compute its value and first derivative elsewhere, as well as its definite integral over an arbitary interval.

3.3. Two Independent Variables

3.3.1. Data on a rectangular mesh

Given the value f_{qr} of the dependent variable f at the point (x_q,y_r) in the plane of the independent variables x and y, for each q=1,2,...,m and r=1,2,...,n (so that the points (x_q,y_r) lie at the m*n intersections of a rectangular mesh), E01DAF computes an interpolating bicubic spline, using a particular choice for each of the spline's knot-set. This choice, the same as in E01BAF , has proved generally satisfactory in practice. If, instead, the user wishes to specify his own knots, a routine from Chapter E02, namely E02DAF, may be adapted (it is more cumbersome for the purpose, however, and much slower for larger problems). Using m and n in the above sense, the parameter M must be set to m*n, PX and PY must be set to m+4 and n+4 respectively and all elements of W should be set to unity. The recommended value for EPS is zero.

3.3.2. Arbitrary data

As remarked at the end of Section 2, special types of interpolating are required for this problem, which can often be difficult to solve satisfactorily. Two of the most successful are employed in E01SAF and E01SEF, the two routines which (with their respective evaluation routines E01SBF and E01SFF) are provided for the problem. Definitions can be found in the routine documents. Both interpolants have first derivative continuity and are 'local', in that their value at any point depends only on data in the immediate neighbourhood of the point. This latter feature is necessary for large sets of data to avoid prohibitive computing time.

The relative merits of the two methods vary with the data and it is not possible to predict which will be the better in any particular case.

3.4. Index

E01 -- Interpolation Contents -- E01

```
          Chapter E01

          Interpolation

          E01BAF  Interpolating functions, cubic spline interpolant, one
                  variable

          E01BEF  Interpolating functions, monotonicity-preserving,
                  piecewise cubic Hermite, one variable

          E01BFF  Interpolated values, interpolant computed by E01BEF,
                  function only, one variable,

          E01BGF  Interpolated values, interpolant computed by E01BEF,
                  function and 1st derivative, one variable

          E01BHF  Interpolated values, interpolant computed by E01BEF,
                  definite integral, one variable

          E01DAF  Interpolating functions, fitting bicubic spline, data on
                  rectangular grid

          E01SAF  Interpolating functions, method of Renka and Cline, two
                  variables

          E01SBF  Interpolated values, evaluate interpolant computed by
                  E01SAF, two variables

          E01SEF  Interpolating functions, modified Shepard's method, two
                  variables

          E01SFF  Interpolated values, evaluate interpolant computed by
                  E01SEF, two variables

\end{verbatim}
\endscroll
\end{page}
```

Cubic spline interpolant

— nage.ht —

```
\begin{page}{manpageXXe01baf}{NAG Documentation: e01baf}
\beginscroll
\begin{verbatim}
```

E01 -- Interpolation   E01BAF

E01BAF -- NAG Foundation Library Routine Document

Note: Before using this routine, please read the Users' Note for your implementation to check implementation-dependent details. The symbol (*) after a NAG routine name denotes a routine that is not included in the Foundation Library.

1. Purpose

E01BAF determines a cubic spline interpolant to a given set of data.

2. Specification

```
      SUBROUTINE E01BAF (M, X, Y, LAMDA, C, LCK, WRK, LWRK,
     1                   IFAIL)
      INTEGER          M, LCK, LWRK, IFAIL
      DOUBLE PRECISION X(M), Y(M), LAMDA(LCK), C(LCK), WRK(LWRK)
```

3. Description

This routine determines a cubic spline $s(x)$, defined in the range $x_1 <= x <= x_m$, which interpolates (passes exactly through) the set of data points (x_i, y_i), for $i=1,2,...,m$, where $m>=4$ and $x_1 < x_2 < ... < x_m$ end conditions are not imposed. The spline interpolant chosen has $m-4$ interior knots $\lambda_5, \lambda_6, ..., \lambda_m$, which are set to the values of $x_3, x_4, ..., x_{m-2}$ respectively. This spline is represented in its B-spline form (see Cox [1]):

$$s(x)= \sum_{i=1}^{m} c_i N_i(x),$$

where $N_i(x)$ denotes the normalised B-Spline of degree 3, defined upon the knots (lambda) ,(lambda)   ,...,(lambda)   , and c i i+1 i+4 i

denotes its coefficient, whose value is to be determined by the routine.

The use of B-splines requires eight additional knots $(\lambda)_1$, $(\lambda)_2$, $(\lambda)_3$, $(\lambda)_4$, $(\lambda)_{m+1}$, $(\lambda)_{m+2}$, $(\lambda)_{m+3}$ and $(\lambda)_{m+4}$ to be specified; the routine sets the first four of these to x_1 and the last four to x_m.

The algorithm for determining the coefficients is as described in [1] except that QR factorization is used instead of LU decomposition. The implementation of the algorithm involves setting up appropriate information for the related routine E02BAF followed by a call of that routine. (For further details of E02BAF, see the routine document.)

Values of the spline interpolant, or of its derivatives or definite integral, can subsequently be computed as detailed in Section 8.

4. References

[1] Cox M G (1975) An Algorithm for Spline Interpolation. J. Inst. Math. Appl. 15 95--108.

[2] Cox M G (1977) A Survey of Numerical Methods for Data and Function Approximation. The State of the Art in Numerical Analysis. (ed D A H Jacobs) Academic Press. 627--668.

5. Parameters

1: M -- INTEGER Input
On entry: m, the number of data points. Constraint: M >= 4.

2: X(M) -- DOUBLE PRECISION array Input
On entry: X(i) must be set to x_i, the ith data value of the independent variable x, for i=1,2,...,m. Constraint: X(i) < X(i+1), for i=1,2,...,M-1.

3: Y(M) -- DOUBLE PRECISION array Input
On entry: Y(i) must be set to y_i, the ith data value of the dependent variable y, for i=1,2,...,m.

4: LAMDA(LCK) -- DOUBLE PRECISION array Output

```
    On exit: the value of (lambda) , the ith knot, for
                                  i
    i=1,2,...,m+4.

5:  C(LCK) -- DOUBLE PRECISION array                     Output
    On exit: the coefficient c  of the B-spline N (x), for
                              i                  i
    i=1,2,...,m. The remaining elements of the array are not
    used.

6:  LCK -- INTEGER                                        Input
    On entry:
    the dimension of the arrays LAMDA and C as declared in the
    (sub)program from which E01BAF is called.
    Constraint: LCK >= M + 4.

7:  WRK(LWRK) -- DOUBLE PRECISION array               Workspace

8:  LWRK -- INTEGER                                       Input
    On entry:
    the dimension of the array WRK as declared in the
    (sub)program from which E01BAF is called.
    Constraint: LWRK>=6*M+16.

9:  IFAIL -- INTEGER                               Input/Output
    On entry: IFAIL must be set to 0, -1 or 1. For users not
    familiar with this parameter (described in the Essential
    Introduction) the recommended value is 0.

    On exit: IFAIL = 0 unless the routine detects an error (see
    Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1
     On entry M < 4,

     or       LCK<M+4,

     or       LWRK<6*M+16.

IFAIL= 2
     The X-values fail to satisfy the condition

     X(1) < X(2) < X(3) <... < X(M).

7. Accuracy
```

The rounding errors incurred are such that the computed spline is an exact interpolant for a slightly perturbed set of ordinates $y_i+(delta)y_i$. The ratio of the root-mean-square value of the $(delta)y_i$ to that of the y_i is no greater than a small multiple of the relative machine precision.

8. Further Comments

The time taken by the routine is approximately proportional to m.

All the x_i are used as knot positions except x_2 and x_{m-1}. This choice of knots (see Cox [2]) means that s(x) is composed of m-3 cubic arcs as follows. If m=4, there is just a single arc space spanning the whole interval x_1 to x_4. If m>=5, the first and last arcs span the intervals x_1 to x_3 and x_{m-2} to x_m respectively. Additionally if m>=6, the ith arc, for i=2,3,...,m-4 spans the interval x_{i+1} to x_{i+2}.

After the call

```
CALL E01BAF (M, X, Y, LAMDA, C, LCK, WRK, LWRK, IFAIL)
```

the following operations may be carried out on the interpolant s(x).

The value of s(x) at x = XVAL can be provided in the real variable SVAL by the call

```
CALL E02BBF (M+4, LAMDA, C, XVAL, SVAL, IFAIL)
```

The values of s(x) and its first three derivatives at x = XVAL can be provided in the real array SDIF of dimension 4, by the call

```
CALL E02BCF (M+4, LAMDA, C, XVAL, LEFT, SDIF, IFAIL)
```

Here LEFT must specify whether the left- or right-hand value of the third derivative is required (see E02BCF for details).

The value of the integral of s(x) over the range x_1 to x_m can be provided in the real variable SINT by

```
CALL E02BDF (M+4, LAMDA, C, SINT, IFAIL)
```

```
         9. Example

         The example program sets up data from 7 values of the exponential
         function in the interval 0 to 1. E01BAF is then called to compute
         a spline interpolant to these data.

         The spline is evaluated by E02BBF, at the data points and at
         points halfway between each adjacent pair of data points, and the
                                         x
         spline values and the values of e  are printed out.

         The example program is not reproduced here. The source code for
         all example programs is distributed with the NAG Foundation
         Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Monoticity-preserving piecewise cubic Hermite interpolant

— nage.ht —

```
\begin{page}{manpageXXe01bef}{NAG Documentation: e01bef}
\beginscroll
\begin{verbatim}



     E01BEF(3NAG)      Foundation Library (12/10/92)      E01BEF(3NAG)



         E01 -- Interpolation                                     E01BEF
                 E01BEF -- NAG Foundation Library Routine Document

         Note: Before using this routine, please read the Users' Note for
         your implementation to check implementation-dependent details.
         The symbol (*) after a NAG routine name denotes a routine that is
         not included in the Foundation Library.

         1. Purpose

         E01BEF computes a monotonicity-preserving piecewise cubic Hermite
```

interpolant to a set of data points.

2. Specification

```
SUBROUTINE E01BEF (N, X, F, D, IFAIL)
INTEGER          N, IFAIL
DOUBLE PRECISION X(N), F(N), D(N)
```

3. Description

This routine estimates first derivatives at the set of data points (x_r, f_r), for r=1,2,...,n, which determine a piecewise cubic Hermite interpolant to the data, that preserves monotonicity over ranges where the data points are monotonic. If the data points are only piecewise monotonic, the interpolant will have an extremum at each point where monotonicity switches direction. The estimates of the derivatives are computed by a formula due to Brodlie, which is described in Fritsch and Butland [1], with suitable changes at the boundary points.

The routine is derived from routine PCHIM in Fritsch [2].

Values of the computed interpolant, and of its first derivative and definite integral, can subsequently be computed by calling E01BFF, E01BGF and E01BHF, as described in Section 8

4. References

[1] Fritsch F N and Butland J (1984) A Method for Constructing Local Monotone Piecewise Cubic Interpolants. SIAM J. Sci. Statist. Comput. 5 300--304.

[2] Fritsch F N (1982) PCHIP Final Specifications. Report UCID-30194. Lawrence Livermore National Laboratory.

5. Parameters

1: N -- INTEGER Input
On entry: n, the number of data points. Constraint: N >= 2.

2: X(N) -- DOUBLE PRECISION array Input
On entry: X(r) must be set to x_r, the rth value of the independent variable (abscissa), for r=1,2,...,n.
Constraint: X(r) < X(r+1).

3: F(N) -- DOUBLE PRECISION array Input
On entry: F(r) must be set to f_r, the rth value of the

```
     dependent variable (ordinate), for r=1,2,...,n.

 4:  D(N) -- DOUBLE PRECISION array                          Output
     On exit: estimates of derivatives at the data points. D(r)
     contains the derivative at X(r).

 5:  IFAIL -- INTEGER                                  Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

IFAIL= 1
     On entry N < 2.

IFAIL= 2
     The values of X(r), for r=1,2,...,N, are not in strictly
     increasing order.

7. Accuracy

The computational errors in the array D should be negligible in
most practical situations.

8. Further Comments

The time taken by the routine is approximately proportional to n.

The values of the computed interpolant at the points PX(i), for
i=1,2,...,M, may be obtained in the real array PF, of length at
least M, by the call:


        CALL E01BFF(N,X,F,D,M,PX,PF,IFAIL)

where N, X and F are the input parameters to E01BEF and D is the
output parameter from E01BEF.

The values of the computed interpolant at the points PX(i), for i
= 1,2,...,M, together with its first derivatives, may be obtained
in the real arrays PF and PD, both of length at least M, by the
```

```
         call:

                 CALL E01BGF(N,X,F,D,M,PX,PF,PD,IFAIL)

         where N, X, F and D are as described above.

         The value of the definite integral of the interpolant over the
         interval A to B can be obtained in the real variable PINT by the
         call:

                 CALL E01BHF(N,X,F,D,A,B,PINT,IFAIL)

         where N, X, F and D are as described above.

         9. Example

         This example program reads in a set of data points, calls E01BEF
         to compute a piecewise monotonic interpolant, and then calls
         E01BFF to evaluate the interpolant at equally spaced points.

         The example program is not reproduced here. The source code for
         all example programs is distributed with the NAG Foundation
         Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Piecewise cubic Hermite interpolant

— nage.ht —

```
\begin{page}{manpageXXe01bff}{NAG Documentation: e01bff}
\beginscroll
\begin{verbatim}



     E01BFF(3NAG)      Foundation Library (12/10/92)      E01BFF(3NAG)



         E01 -- Interpolation                                     E01BFF
                 E01BFF -- NAG Foundation Library Routine Document
```

Note: Before using this routine, please read the Users' Note for your implementation to check implementation-dependent details. The symbol (*) after a NAG routine name denotes a routine that is not included in the Foundation Library.

1. Purpose

E01BFF evaluates a piecewise cubic Hermite interpolant at a set of points.

2. Specification

```
SUBROUTINE E01BFF (N, X, F, D, M, PX, PF, IFAIL)
INTEGER          N, M, IFAIL
DOUBLE PRECISION X(N), F(N), D(N), PX(M), PF(M)
```

3. Description

This routine evaluates a piecewise cubic Hermite interpolant, as computed by E01BEF, at the points PX(i), for i=1,2,...,m. If any point lies outside the interval from X(1) to X(N), a value is extrapolated from the nearest extreme cubic, and a warning is returned.

The routine is derived from routine PCHFE in Fritsch [1].

4. References

[1] Fritsch F N (1982) PCHIP Final Specifications. Report UCID-30194. Lawrence Livermore National Laboratory.

5. Parameters

1: N -- INTEGER Input

2: X(N) -- DOUBLE PRECISION array Input

3: F(N) -- DOUBLE PRECISION array Input

4: D(N) -- DOUBLE PRECISION array Input
On entry: N, X, F and D must be unchanged from the previous call of E01BEF.

5: M -- INTEGER Input
On entry: m, the number of points at which the interpolant is to be evaluated. Constraint: M >= 1.

6: PX(M) -- DOUBLE PRECISION array Input
On entry: the m values of x at which the interpolant is to be evaluated.

```
 7:  PF(M) -- DOUBLE PRECISION array                      Output
     On exit: PF(i) contains the value of the interpolant
     evaluated at the point PX(i), for i=1,2,...,m.

 8:  IFAIL -- INTEGER                               Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

IFAIL= 1
     On entry N < 2.

IFAIL= 2
     The values of X(r), for r = 1,2,...,N, are not in strictly
     increasing order.

IFAIL= 3
     On entry M < 1.

IFAIL= 4
     At least one of the points PX(i), for i = 1,2,...,M, lies
     outside the interval [X(1),X(N)], and extrapolation was
     performed at all such points. Values computed at such points
     may be very unreliable.

7. Accuracy

The computational errors in the array PF should be negligible in
most practical situations.

8. Further Comments

The time taken by the routine is approximately proportional to
the number of evaluation points, m. The evaluation will be most
efficient if the elements of PX are in non-decreasing order (or,
more generally, if they are grouped in increasing order of the
intervals [X(r-1),X(r)]). A single call of E01BFF with m>1 is
more efficient than several calls with m=1.
```

```
         9. Example

         This example program reads in values of N, X, F and D, and then
         calls E01BFF to evaluate the interpolant at equally spaced
         points.

         The example program is not reproduced here. The source code for
         all example programs is distributed with the NAG Foundation
         Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Piecewise cubic Hermite interpolant and 1st deriv

— nage.ht —

```
\begin{page}{manpageXXe01bgf}{NAG Documentation: e01bgf}
\beginscroll
\begin{verbatim}



     E01BGF(3NAG)      Foundation Library (12/10/92)      E01BGF(3NAG)



         E01 -- Interpolation                                       E01BGF
                 E01BGF -- NAG Foundation Library Routine Document

         Note: Before using this routine, please read the Users' Note for
         your implementation to check implementation-dependent details.
         The symbol (*) after a NAG routine name denotes a routine that is
         not included in the Foundation Library.

         1. Purpose

         E01BGF evaluates a piecewise cubic Hermite interpolant and its
         first derivative at a set of points.

         2. Specification

                SUBROUTINE E01BGF (N, X, F, D, M, PX, PF, PD, IFAIL)
                INTEGER          N, M, IFAIL
```

```
      DOUBLE PRECISION X(N), F(N), D(N), PX(M), PF(M), PD(M)
```

3. Description

This routine evaluates a piecewise cubic Hermite interpolant, as computed by E01BEF, at the points PX(i), for i=1,2,...,m. The first derivatives at the points are also computed. If any point lies outside the interval from X(1) to X(N), values of the interpolant and its derivative are extrapolated from the nearest extreme cubic, and a warning is returned.

If values of the interpolant only, and not of its derivative, are required, E01BFF should be used.

The routine is derived from routine PCHFD in Fritsch [1].

4. References

[1] Fritsch F N (1982) PCHIP Final Specifications. Report UCID-30194. Lawrence Livermore National Laboratory.

5. Parameters

1: N -- INTEGER Input

2: X(N) -- DOUBLE PRECISION array Input

3: F(N) -- DOUBLE PRECISION array Input

4: D(N) -- DOUBLE PRECISION array Input
On entry: N, X, F and D must be unchanged from the previous call of E01BEF.

5: M -- INTEGER Input
On entry: m, the number of points at which the interpolant is to be evaluated. Constraint: M >= 1.

6: PX(M) -- DOUBLE PRECISION array Input
On entry: the m values of x at which the interpolant is to be evaluated.

7: PF(M) -- DOUBLE PRECISION array Output
On exit: PF(i) contains the value of the interpolant evaluated at the point PX(i), for i=1,2,...,m.

8: PD(M) -- DOUBLE PRECISION array Output
On exit: PD(i) contains the first derivative of the interpolant evaluated at the point PX(i), for i=1,2,...,m.

9: IFAIL -- INTEGER Input/Output

On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL= 1
On entry N < 2.

IFAIL= 2
The values of X(r), for r = 1,2,...,N, are not in strictly increasing order.

IFAIL= 3
On entry M < 1.

IFAIL= 4
At least one of the points PX(i), for i = 1,2,...,M, lies outside the interval [X(1),X(N)], and extrapolation was performed at all such points. Values computed at these points may be very unreliable.

7. Accuracy

The computational errors in the arrays PF and PD should be negligible in most practical situations.

8. Further Comments

The time taken by the routine is approximately proportional to the number of evaluation points, m. The evaluation will be most efficient if the elements of PX are in non-decreasing order (or, more generally, if they are grouped in increasing order of the intervals [X(r-1),X(r)]). A single call of E01BGF with m>1 is more efficient than several calls with m=1.

9. Example

This example program reads in values of N, X, F and D, and calls E01BGF to compute the values of the interpolant and its derivative at equally spaced points.

```
         The example program is not reproduced here. The source code for
         all example programs is distributed with the NAG Foundation
         Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Definite integral of piecewise cubic Hermite interpolant

— nage.ht —

```
\begin{page}{manpageXXe01bhf}{NAG Documentation: e01bhf}
\beginscroll
\begin{verbatim}



     E01BHF(3NAG)      Foundation Library (12/10/92)      E01BHF(3NAG)



         E01 -- Interpolation                                     E01BHF
                 E01BHF -- NAG Foundation Library Routine Document

         Note: Before using this routine, please read the Users' Note for
         your implementation to check implementation-dependent details.
         The symbol (*) after a NAG routine name denotes a routine that is
         not included in the Foundation Library.

         1. Purpose

         E01BHF evaluates the definite integral of a piecewise cubic
         Hermite interpolant over the interval [a,b].

         2. Specification

                SUBROUTINE E01BHF (N, X, F, D, A, B, PINT, IFAIL)
                INTEGER          N, IFAIL
                DOUBLE PRECISION X(N), F(N), D(N), A, B, PINT

         3. Description

         This routine evaluates the definite integral of a piecewise cubic
         Hermite interpolant, as computed by E01BEF, over the interval
```

[a,b].

If either a or b lies outside the interval from X(1) to X(N) computation of the integral involves extrapolation and a warning is returned.

The routine is derived from routine PCHIA in Fritsch [1].

4. References

[1] Fritsch F N (1982) PCHIP Final Specifications. Report UCID-30194. Lawrence Livermore National Laboratory .

5. Parameters

1: N -- INTEGER Input

2: X(N) -- DOUBLE PRECISION array Input

3: F(N) -- DOUBLE PRECISION array Input

4: D(N) -- DOUBLE PRECISION array Input
On entry: N, X, F and D must be unchanged from the previous call of E01BEF.

5: A -- DOUBLE PRECISION Input

6: B -- DOUBLE PRECISION Input
On entry: the interval [a,b] over which integration is to be performed.

7: PINT -- DOUBLE PRECISION Output
On exit: the value of the definite integral of the interpolant over the interval [a,b].

8: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

```
IFAIL= 1
     On entry N < 2.

IFAIL= 2
     The values of X(r), for r = 1,2,...,N, are not in strictly
     increasing order.

IFAIL= 3
     On entry at least one of A or B lies outside the interval [X
     (1),X(N)], and extrapolation was performed to compute the
     integral. The value returned is therefore unreliable.

7. Accuracy

The computational error in the value returned for PINT should be
negligible in most practical situations.

8. Further Comments

The time taken by the routine is approximately proportional to
the number of data points included within the interval [a,b].

9. Example

This example program reads in values of N, X, F and D. It then
reads in pairs of values for A and B, and evaluates the definite
integral of the interpolant over the interval [A,B] until end-of-
file is reached.

The example program is not reproduced here. The source code for
all example programs is distributed with the NAG Foundation
Library software and should be available on-line.
```

```
\end{verbatim}
\endscroll
\end{page}
```

Bicubic spline interpolated surface

— nage.ht —

```
\begin{page}{manpageXXe01daf}{NAG Documentation: e01daf}
\beginscroll
\begin{verbatim}
```

E01 -- Interpolation E01DAF

E01DAF -- NAG Foundation Library Routine Document

Note: Before using this routine, please read the Users' Note for your implementation to check implementation-dependent details. The symbol (*) after a NAG routine name denotes a routine that is not included in the Foundation Library.

1. Purpose

E01DAF computes a bicubic spline interpolating surface through a set of data values, given on a rectangular grid in the x-y plane.

2. Specification

```
       SUBROUTINE E01DAF (MX, MY, X, Y, F, PX, PY, LAMDA, MU, C,
      1                   WRK, IFAIL)
       INTEGER          MX, MY, PX, PY, IFAIL
       DOUBLE PRECISION X(MX), Y(MY), F(MX*MY), LAMDA(MX+4), MU(MX
      1                 +4), C(MX*MY), WRK((MX+6)*(MY+6))
```

3. Description

This routine determines a bicubic spline interpolant to the set of data points $(x_q,y_r,f_{q,r})$, for $q=1,2,\ldots,m_x$; $r=1,2,\ldots,m_y$. The spline is given in the B-spline representation

$$s(x,y)=\sum_{i=1}^{m_x}\sum_{j=1}^{m_y} c_{ij} M_i(x)N_j(y),$$

such that

$$s(x_q,y_r)=f_{q,r},$$

where $M_i(x)$ and $N_j(y)$ denote normalised cubic B-splines, the former defined on the knots $(\lambda)_i$ to $(\lambda)_{i+4}$ and the latter on the knots $(mu)_j$ to $(mu)_{j+4}$, and the c_{ij} are the spline coefficients. These knots, as well as the coefficients, are determined by the routine, which is derived from the routine B2IRE in Anthony et al[1]. The method used is described in Section 8.2.

For further information on splines, see Hayes and Halliday [4] for bicubic splines and de Boor [3] for normalised B-splines.

Values of the computed spline can subsequently be obtained by calling E02DEF or E02DFF as described in Section 8.3.

4. References

[1] Anthony G T, Cox M G and Hayes J G (1982) DASL - Data Approximation Subroutine Library. National Physical Laboratory.

[2] Cox M G (1975) An Algorithm for Spline Interpolation. J. Inst. Math. Appl. 15 95--108.

[3] De Boor C (1972) On Calculating with B-splines. J. Approx. Theory. 6 50--62.

[4] Hayes J G and Halliday J (1974) The Least-squares Fitting of Cubic Spline Surfaces to General Data Sets. J. Inst. Math. Appl. 14 89--103.

5. Parameters

1: MX -- INTEGER Input

2: MY -- INTEGER Input
On entry: MX and MY must specify m_x and m_y respectively, the number of points along the x and y axis that define the rectangular grid. Constraint: MX >= 4 and MY >= 4.

3: X(MX) -- DOUBLE PRECISION array Input

4: Y(MY) -- DOUBLE PRECISION array Input
On entry: X(q) and Y(r) must contain x_q, for $q=1,2,...,m_x$, and y_r, for $r=1,2,...,m_y$, respectively. Constraints:

$X(q) < X(q+1)$, for $q=1,2,...,m_x-1$,

$Y(r) < Y(r+1)$, for $r=1,2,...,m_y-1$.

y

5: F(MX*MY) -- DOUBLE PRECISION array Input

On entry: F(m_y*(q-1)+r) must contain $f_{q,r}$, for q=1,2,...,m_x; r=1,2,...,m_y.

6: PX -- INTEGER Output

7: PY -- INTEGER Output

On exit: PX and PY contain m_x+4 and m_y+4, the total number of knots of the computed spline with respect to the x and y variables, respectively.

8: LAMDA(MX+4) -- DOUBLE PRECISION array Output

9: MU(MY+4) -- DOUBLE PRECISION array Output

On exit: LAMDA contains the complete set of knots $(lambda)_i$ associated with the x variable, i.e., the interior knots LAMDA(5), LAMDA(6), ..., LAMDA(PX-4), as well as the additional knots LAMDA(1) = LAMDA(2) = LAMDA(3) = LAMDA(4) = X(1) and LAMDA(PX-3) = LAMDA(PX-2) = LAMDA(PX-1) = LAMDA(PX) = X(MX) needed for the B-spline representation. MU contains the corresponding complete set of knots $(mu)_i$ associated with the y variable.

10: C(MX*MY) -- DOUBLE PRECISION array Output

On exit: the coefficients of the spline interpolant. C(m_y*(i-1)+j) contains the coefficient c_{ij} described in Section 3.

11: WRK((MX+6)*(MY+6)) -- DOUBLE PRECISION array Workspace

12: IFAIL -- INTEGER Input/Output

On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL= 1
On entry MX < 4,

or MY < 4.

IFAIL= 2
On entry either the values in the X array or the values in the Y array are not in increasing order.

IFAIL= 3
A system of linear equations defining the B-spline coefficients was singular; the problem is too ill-conditioned to permit solution.

7. Accuracy

The main sources of rounding errors are in steps (2), (3), (6) and (7) of the algorithm described in Section 8.2. It can be shown (Cox [2]) that the matrix A_x formed in step (2) has elements differing relatively from their true values by at most a small multiple of 3(epsilon), where (epsilon) is the machine precision. A_x is 'totally positive', and a linear system with such a coefficient matrix can be solved quite safely by elimination without pivoting. Similar comments apply to steps (6) and (7). Thus the complete process is numerically stable.

8. Further Comments

8.1. Timing

The time taken by this routine is approximately proportional to $m_x m_y$.

8.2. Outline of method used

The process of computing the spline consists of the following steps:

(1) choice of the interior x-knots $(lambda)_5$, $(lambda)_6$,..., $(lambda)_{m_x}$ as $(lambda)_i=x_{i-2}$, for $i=5,6,...,m_x$,

(2) formation of the system

$$A_x E=F,$$

where A_x is a band matrix of order m_x and bandwidth 4, containing in its qth row the values at x_q of the B-splines in x, F is the m_x by m_y rectangular matrix of values $f_{q,r}$, and E denotes an m_x by m_y rectangular matrix of intermediate coefficients,

(3) use of Gaussian elimination to reduce this system to band triangular form,

(4) solution of this triangular system for E,

(5) choice of the interior y knots $(mu)_5$, $(mu)_6$,...,$(mu)_{m_y}$ as

$(mu)_i=y_{i-2}$, for i=5,6,...,m_y,

(6) formation of the system

$$A_y C^T=E^T,$$

where A_y is the counterpart of A_x for the y variable, and C denotes the m_x by m_y rectangular matrix of values of c_{ij},

(7) use of Gaussian elimination to reduce this system to band triangular form,

(8) solution of this triangular system for C^T and hence C.

For computational convenience, steps (2) and (3), and likewise steps (6) and (7), are combined so that the formation of A_x and A_y and the reductions to triangular form are carried out one row at a time.

8.3. Evaluation of Computed Spline

The values of the computed spline at the points (TX(r),TY(r)), for r = 1,2,...,N, may be obtained in the double precision array

```
FF, of length at least N, by the following call:


      IFAIL = 0
      CALL E02DEF(N,PX,PY,TX,TY,LAMDA,MU,C,FF,WRK,IWRK,IFAIL)

where PX, PY, LAMDA, MU and C are the output parameters of E01DAF
, WRK is a double precision workspace array of length at least
PY-4, and IWRK is an integer workspace array of length at least
PY-4.

To evaluate the computed spline on an NX by NY rectangular grid
of points in the x-y plane, which is defined by the x co-
ordinates stored in TX(q), for q = 1,2,...,NX, and the y co-
ordinates stored in TY(r), for r = 1,2,...,NY, returning the
results in the double precision array FG which is of length at
least NX*NY, the following call may be used:


       IFAIL = 0
       CALL E02DFF(NX,NY,PX,PY,TX,TY,LAMDA,MU,C,FG,WRK,LWRK,
      *            IWRK,LIWRK,IFAIL)

where PX, PY, LAMDA, MU and C are the output parameters of E01DAF
, WRK is a double precision workspace array of length at least
LWRK = min(NWRK1,NWRK2), NWRK1 = NX*4+PX, NWRK2 = NY*4+PY, and
IWRK is an integer workspace array of length at least LIWRK = NY
+ PY - 4 if NWRK1 > NWRK2, or NX + PX - 4 otherwise. The result
of the spline evaluated at grid point (q,r) is returned in
element (NY*(q-1)+r) of the array FG.

9. Example

This program reads in values of m , x  for q=1,2,...,m , m  and
                                 x   q                x   y
y  for r=1,2,...,m , followed by values of the ordinates f
 r               y                                        q,r
defined at the grid points (x ,y ). It then calls E01DAF to
                             q  r
compute a bicubic spline interpolant of the data values, and
prints the values of the knots and B-spline coefficients. Finally
it evaluates the spline at a small sample of points on a
rectangular grid.

The example program is not reproduced here. The source code for
all example programs is distributed with the NAG Foundation
Library software and should be available on-line.

\end{verbatim}
\endscroll
```

```
\end{page}
```

———-

Two-D surface interpolating a set of scattered data points

— nage.ht —

```
\begin{page}{manpageXXe01saf}{NAG Documentation: e01saf}
\beginscroll
\begin{verbatim}



     E01SAF(3NAG)      Foundation Library (12/10/92)      E01SAF(3NAG)



          E01 -- Interpolation                                     E01SAF
                  E01SAF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          E01SAF generates a two-dimensional surface interpolating a set of
          scattered data points, using the method of Renka and Cline.

          2. Specification

                 SUBROUTINE E01SAF (M, X, Y, F, TRIANG, GRADS, IFAIL)
                 INTEGER          M, TRIANG(7*M), IFAIL
                 DOUBLE PRECISION X(M), Y(M), F(M), GRADS(2,M)

          3. Description

          This routine constructs an interpolating surface F(x,y) through a
          set of m scattered data points (x ,y ,f ), for r=1,2,...,m, using
                                           r  r  r
          a method due to Renka and Cline. In the (x,y) plane, the data
          points must be distinct. The constructed surface is continuous
          and has continuous first derivatives.

          The method involves firstly creating a triangulation with all the
```

(x,y) data points as nodes, the triangulation being as nearly equiangular as possible (see Cline and Renka [1]). Then gradients in the x- and y-directions are estimated at node r, for r=1,2,...,m, as the partial derivatives of a quadratic function of x and y which interpolates the data value f_r, and which fits the data values at nearby nodes (those within a certain distance chosen by the algorithm) in a weighted least-squares sense. The weights are chosen such that closer nodes have more influence than more distant nodes on derivative estimates at node r. The computed partial derivatives, with the f_r values, at the three nodes of each triangle define a piecewise polynomial surface of a certain form which is the interpolant on that triangle. See Renka and Cline [4] for more detailed information on the algorithm, a development of that by Lawson [2]. The code is derived from Renka [3].

The interpolant F(x,y) can subsequently be evaluated at any point (x,y) inside or outside the domain of the data by a call to E01SBF. Points outside the domain are evaluated by extrapolation.

4. References

[1] Cline A K and Renka R L (1984) A Storage-efficient Method for Construction of a Thiessen Triangulation. Rocky Mountain J. Math. 14 119--139.

[2] Lawson C L (1977) Software for C^1 Surface Interpolation. Mathematical Software III. (ed J R Rice) Academic Press. 161--194.

[3] Renka R L (1984) Algorithm 624: Triangulation and Interpolation of Arbitrarily Distributed Points in the Plane. ACM Trans. Math. Softw. 10 440--442.

[4] Renka R L and Cline A K (1984) A Triangle-based C^1 Interpolation Method. Rocky Mountain J. Math. 14 223--237.

5. Parameters

1: M -- INTEGER Input
On entry: m, the number of data points. Constraint: M >= 3.

2: X(M) -- DOUBLE PRECISION array Input

3: Y(M) -- DOUBLE PRECISION array Input

```
 4:  F(M) -- DOUBLE PRECISION array                         Input
     On entry: the co-ordinates of the rth data point, for
     r=1,2,...,m. The data points are accepted in any order, but
     see Section 8. Constraint: The (x,y) nodes must not all be
     collinear, and each node must be unique.

 5:  TRIANG(7*M) -- INTEGER array                          Output
     On exit: a data structure defining the computed
     triangulation, in a form suitable for passing to E01SBF.

 6:  GRADS(2,M) -- DOUBLE PRECISION array                  Output
     On exit: the estimated partial derivatives at the nodes, in
     a form suitable for passing to E01SBF. The derivatives at
     node r with respect to x and y are contained in GRADS(1,r)
     and GRADS(2,r) respectively, for r=1,2,...,m.

 7:  IFAIL -- INTEGER                                Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

IFAIL= 1
     On entry M < 3.

IFAIL= 2
     On entry all the (X,Y) pairs are collinear.

IFAIL= 3
     On entry (X(i),Y(i)) = (X(j),Y(j)) for some i/=j.

7. Accuracy

On successful exit, the computational errors should be negligible
in most situations but the user should always check the computed
surface for acceptability, by drawing contours for instance. The
surface always interpolates the input data exactly.

8. Further Comments

The time taken for a call of E01SAF is approximately proportional
```

```
        to the number of data points, m. The routine is more efficient
        if, before entry, the values in X, Y, F are arranged so that the
        X array is in ascending order.

        9. Example

        This program reads in a set of 30 data points and calls E01SAF to
        construct an interpolating surface. It then calls E01SBF to
        evaluate the interpolant at a sample of points on a rectangular
        grid.

        Note that this example is not typical of a realistic problem: the
        number of data points would normally be larger, and the
        interpolant would need to be evaluated on a finer grid to obtain
        an accurate plot, say.

        The example program is not reproduced here. The source code for
        all example programs is distributed with the NAG Foundation
        Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Evaluate 2D interpolant function from E01SAF

— nage.ht —

```
\begin{page}{manpageXXe01sbf}{NAG Documentation: e01sbf}
\beginscroll
\begin{verbatim}



     E01SBF(3NAG)      Foundation Library (12/10/92)      E01SBF(3NAG)



        E01 -- Interpolation                                     E01SBF
                E01SBF -- NAG Foundation Library Routine Document

        Note: Before using this routine, please read the Users' Note for
        your implementation to check implementation-dependent details.
        The symbol (*) after a NAG routine name denotes a routine that is
        not included in the Foundation Library.
```

1. Purpose

E01SBF evaluates at a given point the two-dimensional interpolant function computed by E01SAF.

2. Specification

```
       SUBROUTINE E01SBF (M, X, Y, F, TRIANG, GRADS, PX, PY, PF,
      1                   IFAIL)
       INTEGER          M, TRIANG(7*M), IFAIL
       DOUBLE PRECISION X(M), Y(M), F(M), GRADS(2,M), PX, PY, PF
```

3. Description

This routine takes as input the parameters defining the interpolant F(x,y) of a set of scattered data points (x_r,y_r,f_r), for r=1,2,...,m, as computed by E01SAF, and evaluates the interpolant at the point (px,py).

If (px,py) is equal to (x_r,y_r) for some value of r, the returned value will be equal to f_r.

If (px,py) is not equal to (x_r,y_r) for any r, the derivatives in GRADS will be used to compute the interpolant. A triangle is sought which contains the point (px,py), and the vertices of the triangle along with the partial derivatives and f_r values at the vertices are used to compute the value F(px,py). If the point (px,py) lies outside the triangulation defined by the input parameters, the returned value is obtained by extrapolation. In this case, the interpolating function F is extended linearly beyond the triangulation boundary. The method is described in more detail in Renka and Cline [2] and the code is derived from Renka [1].

E01SBF must only be called after a call to E01SAF.

4. References

[1] Renka R L (1984) Algorithm 624: Triangulation and Interpolation of Arbitrarily Distributed Points in the Plane. ACM Trans. Math. Softw. 10 440--442.

[2] Renka R L and Cline A K (1984) A Triangle-based C^1

```
     Interpolation Method. Rocky Mountain J. Math. 14 223--237.

5. Parameters

 1:  M -- INTEGER                                          Input

 2:  X(M) -- DOUBLE PRECISION array                        Input

 3:  Y(M) -- DOUBLE PRECISION array                        Input

 4:  F(M) -- DOUBLE PRECISION array                        Input

 5:  TRIANG(7*M) -- INTEGER array                          Input

 6:  GRADS(2,M) -- DOUBLE PRECISION array                  Input
     On entry: M, X, Y, F, TRIANG and GRADS must be unchanged
     from the previous call of E01SAF.

 7:  PX -- DOUBLE PRECISION                                Input

 8:  PY -- DOUBLE PRECISION                                Input
     On entry: the point (px,py) at which the interpolant is to
     be evaluated.

 9:  PF -- DOUBLE PRECISION                               Output
     On exit: the value of the interpolant evaluated at the
     point (px,py).

10:  IFAIL -- INTEGER                               Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

IFAIL= 1
     On entry M < 3.

IFAIL= 2
     On entry the triangulation information held in the array
     TRIANG does not specify a valid triangulation of the data
     points. TRIANG may have been corrupted since the call to
```

```
          E01SAF.

      IFAIL= 3
          The evaluation point (PX,PY) lies outside the nodal
          triangulation, and the value returned in PF is computed by
          extrapolation.

      7. Accuracy

      Computational errors should be negligible in most practical
      situations.

      8. Further Comments

      The time taken for a call of E01SBF is approximately proportional
      to the number of data points, m.

      The results returned by this routine are particularly suitable
      for applications such as graph plotting, producing a smooth
      surface from a number of scattered points.

      9. Example

      See the example for E01SAF.

      The example program is not reproduced here. The source code for
      all example programs is distributed with the NAG Foundation
      Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Generate 2D surface interpolating a scattered data points

— nage.ht —

```
\begin{page}{manpageXXe01sef}{NAG Documentation: e01sef}
\beginscroll
\begin{verbatim}



     E01SEF(3NAG)      Foundation Library (12/10/92)      E01SEF(3NAG)
```

```
E01 -- Interpolation                                       E01SEF
        E01SEF -- NAG Foundation Library Routine Document

Note: Before using this routine, please read the Users' Note for
your implementation to check implementation-dependent details.
The symbol (*) after a NAG routine name denotes a routine that is
not included in the Foundation Library.

1. Purpose

E01SEF generates a two-dimensional surface interpolating a set of
scattered data points, using a modified Shepard method.

2. Specification

       SUBROUTINE E01SEF (M, X, Y, F, RNW, RNQ, NW, NQ, FNODES,
      1                   MINNQ, WRK, IFAIL)
       INTEGER          M, NW, NQ, MINNQ, IFAIL
       DOUBLE PRECISION X(M), Y(M), F(M), RNW, RNQ, FNODES(5*M),
      1                 WRK(6*M)

3. Description

This routine constructs an interpolating surface F(x,y) through a
set of m scattered data points (x ,y ,f ), for r=1,2,...,m, using
                                 r  r  r
a modification of Shepard's method. The surface is continuous and
has continuous first derivatives.

The basic Shepard method, described in [2], interpolates the
input data with the weighted mean

                              m
                              --
                              >  w (x,y)f
                              --  r      r
                              r=1
                   F(x,y)= -------------,
                               m
                               --
                               >  w (x,y)
                               --  r
                               r=1

                        1         2       2       2
           where w (x,y)= --  and  d =(x-x ) +(y-y ) .
                  r        2         r     r       r
                          d
```

r

The basic method is global in that the interpolated value at any point depends on all the data, but this routine uses a modification due to Franke and Nielson described in [1], whereby the method becomes local by adjusting each $w_r(x,y)$ to be zero outside a circle with centre (x_r,y_r) and some radius R_w. Also, to improve the performance of the basic method, each f_r above is replaced by a function $f_r(x,y)$, which is a quadratic fitted by weighted least-squares to data local to (x_r,y_r) and forced to interpolate (x_r,y_r,f_r). In this context, a point (x,y) is defined to be local to another point if it lies within some distance R_q of it. Computation of these quadratics constitutes the main work done by this routine. If there are less than 5 other points within distance R_q from (x_r,y_r), the quadratic is replaced by a linear function. In cases of rank-deficiency, the minimum norm solution is computed.

The user may specify values for R_w and R_q, but it is usually easier to choose instead two integers N_w and N_q, from which the routine will compute R_w and R_q. These integers can be thought of as the average numbers of data points lying within distances R_w and R_q respectively from each node. Default values are provided, and advice on alternatives is given in Section 8.2.

The interpolant $F(x,y)$ generated by this routine can subsequently be evaluated for any point (x,y) in the domain of the data by a call to E01SFF.

4. References

[1] Franke R and Nielson G (1980) Smooth Interpolation of Large Sets of Scattered Data. Internat. J. Num. Methods Engrg. 15 1691--1704.

[2] Shepard D (1968) A Two-dimensional Interpolation Function for Irregularly Spaced Data. Proc. 23rd Nat. Conf. ACM.

Brandon/Systems Press Inc., Princeton. 517--523.

5. Parameters

1: M -- INTEGER Input
On entry: m, the number of data points. Constraint: M >= 3.

2: X(M) -- DOUBLE PRECISION array Input

3: Y(M) -- DOUBLE PRECISION array Input

4: F(M) -- DOUBLE PRECISION array Input
On entry: the co-ordinates of the rth data point, for r=1,2,...,m. The order of the data points is immaterial. Constraint: each of the (X(r),Y(r)) pairs must be unique.

5: RNW -- DOUBLE PRECISION Input/Output

6: RNQ -- DOUBLE PRECISION Input/Output
On entry: suitable values for the radii R_w and R_q, described in Section 3. Constraint: RNQ <= 0 or 0 < RNW <= RNQ. On exit: if RNQ is set less than or equal to zero on entry, then default values for both of them will be computed from the parameters NW and NQ, and RNW and RNQ will contain these values on exit.

7: NW -- INTEGER Input

8: NQ -- INTEGER Input
On entry: if RNQ > 0.0 and RNW > 0.0 then NW and NQ are not referenced by the routine. Otherwise, NW and NQ must specify suitable values for the integers N_w and N_q described in Section 3.

If NQ is less than or equal to zero on entry, then default values for both of them, namely NW = 9 and NQ = 18, will be used. Constraint: NQ <= 0 or 0 < NW <= NQ.

9: FNODES(5*M) -- DOUBLE PRECISION array Output
On exit: the coefficients of the constructed quadratic nodal functions. These are in a form suitable for passing to E01SFF.

10: MINNQ -- INTEGER Output
On exit: the minimum number of data points that lie within radius RNQ of any node, and thus define a nodal function. If MINNQ is very small (say, less than 5), then the interpolant may be unsatisfactory in regions where the data points are

sparse.

11: WRK(6*M) -- DOUBLE PRECISION array Workspace

12: IFAIL -- INTEGER Input/Output

On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL= 1
On entry M < 3.

IFAIL= 2
On entry RNQ > 0 and either RNW > RNQ or RNW <= 0.

IFAIL= 3
On entry NQ > 0 and either NW > NQ or NW <= 0.

IFAIL= 4
On entry (X(i),Y(i)) is equal to (X(j),Y(j)) for some i/=j.

7. Accuracy

On successful exit, the computational errors should be negligible in most situations but the user should always check the computed surface for acceptability, by drawing contours for instance. The surface always interpolates the input data exactly.

8. Further Comments

8.1. Timing

The time taken for a call of E01SEF is approximately proportional to the number of data points, m, provided that N_q is of the same order as its default value (18). However if N_q is increased so that the method becomes more global, the time taken becomes approximately proportional to m^2.

```
                  m N} {{\rm w}}$ and ${\rm N} {{\rm q}}$
8.2. Choice of ${\
Note first that the radii R  and R , described in Section 3, are
                           w      q


                  / N            / N
             D   /   w      D   /   q
computed as  -  /   -- and  -  /   -- respectively, where D is
             2\/    m       2/    m
the maximum distance between any pair of data points.
Default values N =9 and N =18 work quite well when the data
                w        q
points are fairly uniformly distributed. However, for data having
some regions with relatively few points or for small data sets
(m<25), a larger value of N  may be needed. This is to ensure a
                          w
reasonable number of data points within a distance R  of each
                                                    w
node, and to avoid some regions in the data area being left
outside all the discs of radius R  on which the weights w (x,y)
                                 w                       r
are non-zero. Maintaining N  approximately equal to 2N  is
                           q                          w
usually an advantage.

Note however that increasing N  and N  does not improve the
                              w      q
quality of the interpolant in all cases. It does increase the
computational cost and makes the method less local.

9. Example

This program reads in a set of 30 data points and calls E01SEF to
construct an interpolating surface. It then calls E01SFF to
evaluate the interpolant at a sample of points on a rectangular
grid.

Note that this example is not typical of a realistic problem: the
number of data points would normally be larger, and the
interpolant would need to be evaluated on a finer grid to obtain
an accurate plot, say.

The example program is not reproduced here. The source code for
all example programs is distributed with the NAG Foundation
Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

———-

Evaluate 2D interpolating function from E01SEF

— nage.ht —

```
\begin{page}{manpageXXe01sff}{NAG Documentation: e01sff}
\beginscroll
\begin{verbatim}



     E01SFF(3NAG)      Foundation Library (12/10/92)      E01SFF(3NAG)



          E01 -- Interpolation                                     E01SFF
                  E01SFF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          E01SFF evaluates at a given point the two-dimensional
          interpolating function computed by E01SEF.

          2. Specification

                 SUBROUTINE E01SFF (M, X, Y, F, RNW, FNODES, PX, PY, PF,
                1                   IFAIL)
                 INTEGER          M, IFAIL
                 DOUBLE PRECISION X(M), Y(M), F(M), RNW, FNODES(5*M), PX,
                1                 PY, PF

          3. Description

          This routine takes as input the interpolant F(x,y) of a set of
          scattered data points (x ,y ,f ), for r=1,2,...,m, as computed by
                                  r  r  r
          E01SEF, and evaluates the interpolant at the point (px,py).

          If (px,py) is equal to (x ,y ) for some value of r, the returned
                                   r  r
```

value will be equal to f_r.

If (px,py) is not equal to (x_r,y_r) for any r, all points that are within distance RNW of (px,py), along with the corresponding nodal functions given by FNODES, will be used to compute a value of the interpolant.

E01SFF must only be called after a call to E01SEF.

4. References

[1] Franke R and Nielson G (1980) Smooth Interpolation of Large Sets of Scattered Data. Internat. J. Num. Methods Engrg. 15 1691--1704.

[2] Shepard D (1968) A Two-dimensional Interpolation Function for Irregularly Spaced Data. Proc. 23rd Nat. Conf. ACM. Brandon/Systems Press Inc., Princeton. 517--523.

5. Parameters

1: M -- INTEGER Input

2: X(M) -- DOUBLE PRECISION array Input

3: Y(M) -- DOUBLE PRECISION array Input

4: F(M) -- DOUBLE PRECISION array Input

5: RNW -- DOUBLE PRECISION Input

6: FNODES(5*M) -- DOUBLE PRECISION array Input
On entry: M, X, Y, F, RNW and FNODES must be unchanged from the previous call of E01SEF.

7: PX -- DOUBLE PRECISION Input

8: PY -- DOUBLE PRECISION Input
On entry: the point (px,py) at which the interpolant is to be evaluated.

9: PF -- DOUBLE PRECISION Output
On exit: the value of the interpolant evaluated at the point (px,py).

10: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential

```
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

IFAIL= 1
     On entry M < 3.

IFAIL= 2
     The interpolant cannot be evaluated because the evaluation
     point (PX,PY) lies outside the support region of the data
     supplied in X, Y and F. This error exit will occur if
     (PX,PY) lies at a distance greater than or equal to RNW from
     every point given by arrays X and Y.

     The value 0.0 is returned in PF. This value will not provide
     continuity with values obtained at other points (PX,PY),
     i.e., values obtained when IFAIL = 0 on exit.

7. Accuracy

Computational errors should be negligible in most practical
situations.

8. Further Comments

The time taken for a call of E01SFF is approximately proportional
to the number of data points, m.

The results returned by this routine are particularly suitable
for applications such as graph plotting, producing a smooth
surface from a number of scattered points.

9. Example

See the example for E01SEF.

The example program is not reproduced here. The source code for
all example programs is distributed with the NAG Foundation
Library software and should be available on-line.

\end{verbatim}
\endscroll
```

```
\end{page}
```

Curve and Surface Fitting

— nage.ht —

```
\begin{page}{manpageXXe02}{NAG Documentation: e02}
\beginscroll
\begin{verbatim}



     E02(3NAG)         Foundation Library (12/10/92)         E02(3NAG)



          E02 -- Curve and Surface Fitting              Introduction -- E02
                                    Chapter E02
                             Curve and Surface Fitting

          Contents of this Introduction:

          1.     Scope of the Chapter

          2.     Background to the Problems

          2.1.   Preliminary Considerations

          2.1.1. Fitting criteria: norms

          2.1.2. Weighting of data points

          2.2.   Curve Fitting

          2.2.1. Representation of polynomials

          2.2.2. Representation of cubic splines

          2.3.   Surface Fitting

          2.3.1. Bicubic splines: definition and representation

          2.4.   General Linear and Nonlinear Fitting Functions

          2.5.   Constrained Problems
```

1. Scope of the Chapter

The main aim of this chapter is to assist the user in finding a function which approximates a set of data points. Typically the data contain random errors, as of experimental measurement, which need to be smoothed out. To seek an approximation to the data, it is first necessary to specify for the approximating function a mathematical form (a polynomial, for example) which contains a number of unspecified coefficients: the appropriate fitting routine then derives for the coefficients the values which provide the best fit of that particular form. The chapter deals mainly with curve and surface fitting (i.e., fitting with functions of one and of two variables) when a polynomial or a cubic spline is used as the fitting function, since these cover the most common needs. However, fitting with other functions and/or more variables can be undertaken by means of general linear or nonlinear routines (some of which are contained in other chapters) depending on whether the coefficients in the function occur linearly or nonlinearly. Cases where a graph rather than a set of data points is given can be treated simply by first reading a suitable set of points from the graph.

The chapter also contains routines for evaluating, differentiating and integrating polynomial and spline curves and surfaces, once the numerical values of their coefficients have been determined.

2. Background to the Problems

2.1. Preliminary Considerations

In the curve-fitting problems considered in this chapter, we have a dependent variable y and an independent variable x, and we are given a set of data points (x_r, y_r), for r=1,2,...,m. The preliminary matters to be considered in this section will, for simplicity, be discussed in this context of curve-fitting problems. In fact, however, these considerations apply equally well to surface and higher-dimensional problems. Indeed, the discussion presented carries over essentially as it stands if, for these cases, we interpret x as a vector of several independent variables and correspondingly each x_r as a vector containing the rth data value of each independent variable.

We wish, then, to approximate the set of data points as closely as possible with a specified function, f(x) say, which is as smooth as possible -- f(x) may, for example, be a polynomial. The requirements of smoothness and closeness conflict, however, and a balance has to be struck between them. Most often, the smoothness requirement is met simply by limiting the number of coefficients allowed in the fitting function -- for example, by restricting the degree in the case of a polynomial. Given a particular number of coefficients in the function in question, the fitting routines of this chapter determine the values of the coefficients such that the 'distance' of the function from the data points is as small as possible. The necessary balance is struck by the user comparing a selection of such fits having different numbers of coefficients. If the number of coefficients is too low, the approximation to the data will be poor. If the number is too high, the fit will be too close to the data, essentially following the random errors and tending to have unwanted fluctuations between the data points. Between these extremes, there is often a group of fits all similarly close to the data points and then, particularly when least-squares polynomials are used, the choice is clear: it is the fit from this group having the smallest number of coefficients.

The above process can be seen as the user minimizing the smoothness measure (i.e., the number of coefficients) subject to the distance from the data points being acceptably small. Some of the routines, however, do this task themselves. They use a different measure of smoothness (in each case one that is continuous) and minimize it subject to the distance being less than a threshold specified by the user. This is a much more automatic process, requiring only some experimentation with the threshold.

2.1.1. Fitting criteria: norms

A measure of the above 'distance' between the set of data points and the function f(x) is needed. The distance from a single data point (x_r, y_r) to the function can simply be taken as

$$\epsilon_r = y_r - f(x_r), \qquad (1)$$

and is called the residual of the point. (With this definition, the residual is regarded as a function of the coefficients contained in f(x); however, the term is also used to mean the particular value of ϵ_r which corresponds to the fitted values of the coefficients.) However, we need a measure of distance for the set of data points as a whole. Three different measures are used in the different routines (which measure to select, according to circumstances, is discussed later in this sub-section). With ϵ_r defined in (1), these measures, or norms, are

$$\sum_{r=1}^{m} |\epsilon_r|, \qquad (2)$$

$$\sqrt{\sum_{r=1}^{m} (\epsilon)_r^2}, \text{ and} \quad (3)$$

$$\max_r |(\epsilon)_r|, \quad (4)$$

respectively the l_1 norm, the l_2 norm and the l_∞ norm.

Minimization of one or other of these norms usually provides the fitting criterion, the minimization being carried out with respect to the coefficients in the mathematical form used for f(x): with respect to the b_i for example if the mathematical form is the power series in (8) below. The fit which results from minimizing (2) is known as the l_1 fit, or the fit in the l_1 norm: that which results from minimizing (3) is the l_2 fit, the well-known least-squares fit (minimizing (3) is equivalent to minimizing the square of (3), i.e., the sum of squares of residuals, and it is the latter which is used in practice), and that from minimizing (4) is the l_∞, or minimax, fit.

Strictly speaking, implicit in the use of the above norms are the statistical assumptions that the random errors in the y_r are independent of one another and that any errors in the x_r are negligible by comparison. From this point of view, the use of the l_2 norm is appropriate when the random errors in the y_r have a normal distribution, and the l_∞ norm is appropriate when they have a rectangular distribution, as when fitting a table of values rounded to a fixed number of decimal places. The l_1 norm is appropriate when the error distribution has its frequency function proportional to the negative exponential of the modulus of the normalised error -- not a common situation.

However, the user is often indifferent to these statistical considerations, and simply seeks a fit which he can assess by inspection, perhaps visually from a graph of the results. In this event, the l_1 norm is particularly appropriate when the data are thought to contain some 'wild' points (since fitting in this norm tends to be unaffected by the presence of a small number of such points), though of course in simple situations the user may prefer to identify and reject these points. The l_{∞} norm should be used only when the maximum residual is of particular concern, as may be the case for example when the data values have been obtained by accurate computation, as of a mathematical function. Generally, however, a routine based on least-squares should be preferred, as being computationally faster and usually providing more information on which to assess the results. In many problems the three fits will not differ significantly for practical purposes.

Some of the routines based on the l_2 norm do not minimize the norm itself but instead minimize some (intuitively acceptable) measure of smoothness subject to the norm being less than a user-specified threshold. These routines fit with cubic or bicubic splines (see (10) and (14) below) and the smoothing measures relate to the size of the discontinuities in their third derivatives. A much more automatic fitting procedure follows from this approach.

2.1.2. Weighting of data points

The use of the above norms also assumes that the data values y_r are of equal (absolute) accuracy. Some of the routines enable an allowance to be made to take account of differing accuracies. The allowance takes the form of 'weights' applied to the y-values so that those values known to be more accurate have a greater influence on the fit than others. These weights, to be supplied by the user, should be calculated from estimates of the absolute accuracies of the y-values, these estimates being expressed as standard deviations, probable errors or some other measure which has the same dimensions as y. Specifically, for each y_r the corresponding weight w_r should be inversely proportional to the accuracy estimate of y_r. For example, if the percentage accuracy is the same for all y_r, then the absolute accuracy of y_r is proportional to y_r (assuming y_r to be positive, as it usually is in such cases) and so $w_r=K/y_r$, for $r=1,2,...,m$, for an arbitrary positive constant K. (This definition of weight is stressed because often weight is defined as the square of that used here.) The norms (2), (3) and (4) above are then replaced respectively by

$$\sum_{r=1}^{m} |w_r \epsilon_r|, \tag{5}$$

$$\sqrt{\sum_{r=1}^{m} w_r^2 \epsilon_r^2}, \quad \text{and} \tag{6}$$

$$\max_r |w_r \epsilon_r|. \tag{7}$$

Again it is the square of (6) which is used in practice rather than (6) itself.

2.2. Curve Fitting

When, as is commonly the case, the mathematical form of the fitting function is immaterial to the problem, polynomials and cubic splines are to be preferred because their simplicity and ease of handling confer substantial benefits. The cubic spline is the more versatile of the two. It consists of a number of cubic polynomial segments joined end to end with continuity in first and second derivatives at the joins. The third derivative at the joins is in general discontinuous. The x-values of the joins are called knots, or, more precisely, interior knots. Their number determines the number of coefficients in the spline, just as the degree determines the number of coefficients in a polynomial.

2.2.1. Representation of polynomials

Rather than using the power-series form

$$f(x)==b_0+b_1x+b_2x^2+...+b_kx^k \tag{8}$$

to represent a polynomial, the routines in this chapter use the Chebyshev series form

$$f(x) = \frac{1}{2}a_0T_0(x)+a_1T_1(x)+a_2T_2(x)+\dots+a_kT_k(x), \qquad (9)$$

where $T_i(x)$ is the Chebyshev polynomial of the first kind of degree i (see Cox and Hayes [1], page 9), and where the range of x has been normalised to run from -1 to +1. The use of either form leads theoretically to the same fitted polynomial, but in practice results may differ substantially because of the effects of rounding error. The Chebyshev form is to be preferred, since it leads to much better accuracy in general, both in the computation of the coefficients and in the subsequent evaluation of the fitted polynomial at specified points. This form also has other advantages: for example, since the later terms in (9) generally decrease much more rapidly from left to right than do those in (8), the situation is more often encountered where the last terms are negligible and it is obvious that the degree of the polynomial can be reduced (note that on the interval $-1<=x<=1$ for all i, $T_i(x)$ attains the value unity but never exceeds it, so that the coefficient a_i gives directly the maximum value of the term containing it).

2.2.2. Representation of cubic splines

A cubic spline is represented in the form

$$f(x) = c_1N_1(x)+c_2N_2(x)+\dots+c_pN_p(x), \qquad (10)$$

where $N_i(x)$, for i=1,2,...,p, is a normalised cubic B-spline (see Hayes [2]). This form, also, has advantages of computational speed and accuracy over alternative representations.

2.3. Surface Fitting

There are now two independent variables, and we shall denote these by x and y. The dependent variable, which was denoted by y in the curve-fitting case, will now be denoted by f. (This is a rather different notation from that indicated for the general-dimensional problem in the first paragraph of Section 2.1 , but it has some advantages in presentation.)

Again, in the absence of contrary indications in the particular application being considered, polynomials and splines are the approximating functions most commonly used. Only splines are used by the surface-fitting routines in this chapter.

2.3.1. Bicubic splines: definition and representation

The bicubic spline is defined over a rectangle R in the (x,y) plane, the sides of R being parallel to the x- and y-axes. R is divided into rectangular panels, again by lines parallel to the axes. Over each panel the bicubic spline is a bicubic polynomial, that is it takes the form

$$\sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij} x^i y^j . \tag{13}$$

Each of these polynomials joins the polynomials in adjacent panels with continuity up to the second derivative. The constant x-values of the dividing lines parallel to the y-axis form the set of interior knots for the variable x, corresponding precisely to the set of interior knots of a cubic spline. Similarly, the constant y-values of dividing lines parallel to the x-axis form the set of interior knots for the variable y. Instead of representing the bicubic spline in terms of the above set of bicubic polynomials, however, it is represented, for the sake of computational speed and accuracy, in the form

$$f(x,y)= \sum_{i=1}^{p}\sum_{j=1}^{q} c_{ij} M_i(x)N_j(y), \tag{14}$$

where $M_i(x)$, for i=1,2,...,p, and $N_j(y)$, for j=1,2,...,q, are normalised B-splines (see Hayes and Halliday [4] for further details of bicubic splines and Hayes [2] for normalised B-splines).

2.4. General Linear and Nonlinear Fitting Functions

We have indicated earlier that, unless the data-fitting application under consideration specifically requires some other type of fitting function, a polynomial or a spline is usually to be preferred. Special routines for these functions, in one and in two variables, are provided in this chapter. When the application does specify some other fitting function, however, it may be treated by a routine which deals with a general linear function, or by one for a general nonlinear function, depending on whether the coefficients in the given function occur linearly or nonlinearly.

The general linear fitting function can be written in the form

$$f(x)=c_1 \phi_1(x)+c_2 \phi_2(x)+\ldots+c_p \phi_p(x), \qquad (15)$$

where x is a vector of one or more independent variables, and the ϕ_i are any given functions of these variables (though they must be linearly independent of one another if there is to be the possibility of a unique solution to the fitting problem). This is not intended to imply that each ϕ_i is necessarily a function of all the variables: we may have, for example, that each ϕ_i is a function of a different single variable, and even that one of the ϕ_i is a constant. All that is required is that a value of each $\phi_i(x)$ can be computed when a value of each independent variable is given.

When the fitting function f(x) is not linear in its coefficients, no more specific representation is available in general than f(x) itself. However, we shall find it helpful later on to indicate the fact that f(x) contains a number of coefficients (to be determined by the fitting process) by using instead the notation f(x;c), where c denotes the vector of coefficients. An example of a nonlinear fitting function is

$$f(x;c)=c_1+c_2 \exp(-c_4 x)+c_3 \exp(-c_5 x), \qquad (16)$$

which is in one variable and contains five coefficients. Note that here, as elsewhere in this Chapter Introduction, we use the term 'coefficients' to include all the quantities whose values are to be determined by the fitting process, not just those which occur linearly. We may observe that it is only the presence of the coefficients c_4 and c_5 which makes the form (16) nonlinear. If the values of these two coefficients were known beforehand, (16) would instead be a linear function which, in terms of the general linear form (15), has p=3 and

$$\phi_1(x)=1, \quad \phi_2(x)=\exp(-c_4 x), \text{ and } \phi_3(x)=\exp(-c_5 x).$$

We may note also that polynomials and splines, such as (9) and (14), are themselves linear in their coefficients. Thus if, when fitting with these functions, a suitable special routine is not available (as when more than two independent variables are involved or when fitting in the l_1 norm), it is appropriate to use a routine designed for a general linear function.

2.5. Constrained Problems

So far, we have considered only fitting processes in which the values of the coefficients in the fitting function are determined by an unconstrained minimization of a particular norm. Some fitting problems, however, require that further restrictions be placed on the determination of the coefficient values. Sometimes these restrictions are contained explicitly in the formulation of the problem in the form of equalities or inequalities which the coefficients, or some function of them, must satisfy. For example, if the fitting function contains a term Aexp(-kx), it may be required that k>=0. Often, however, the equality or inequality constraints relate to the value of the fitting function or its derivatives at specified values of the independent variable(s), but these too can be expressed in terms of the coefficients of the fitting function, and it is appropriate to do this if a general linear or nonlinear routine is being used. For example, if the fitting function is that given in (10), the requirement that the first derivative of the function at $x=x_0$ be non-negative can be expressed as

$$c_1N_1'(x_0)+c_2N_2'(x_0)+...+c_pN_p'(x_0)>=0, \qquad (17)$$

where the prime denotes differentiation with respect to x and each derivative is evaluated at $x=x_0$. On the other hand, if the requirement had been that the derivative at $x=x_0$ be exactly zero, the inequality sign in (17) would be replaced by an equality.

Routines which provide a facility for minimizing the appropriate norm subject to such constraints are discussed in Section 3.6.

2.6. References

[1] Cox M G and Hayes J G (1973) Curve fitting: a guide and suite of algorithms for the non-specialist user. Report NAC26. National Physical Laboratory.

[2] Hayes J G (1974) Numerical Methods for Curve and Surface Fitting. Bull Inst Math Appl. 10 144--152.

(For definition of normalised B-splines and details of numerical methods.)

[3] Hayes J G (1970) Curve Fitting by Polynomials in One Variable. Numerical Approximation to Functions and Data. (ed J G Hayes) Athlone Press, London.

[4] Hayes J G and Halliday J (1974) The Least-squares Fitting of Cubic Spline Surfaces to General Data Sets. J. Inst. Math. Appl. 14 89--103.

3. Recommendations on Choice and Use of Routines

3.1. General

The choice of a routine to treat a particular fitting problem will depend first of all on the fitting function and the norm to be used. Unless there is good reason to the contrary, the fitting function should be a polynomial or a cubic spline (in the appropriate number of variables) and the norm should be the l_2 norm (leading to the least-squares fit). If some other function is to be used, the choice of routine will depend on whether the function is nonlinear (in which case see Section 3.5.2) or linear in its coefficients (see Section 3.5.1), and, in the latter case, on whether the l_1 or l_2 norm is to be used. The latter section is appropriate for polynomials and splines, too, if the l_1 norm is preferred.

In the case of a polynomial or cubic spline, if there is only one independent variable, the user should choose a spline (Section 3.3) when the curve represented by the data is of complicated form, perhaps with several peaks and troughs. When the curve is of simple form, first try a polynomial (see Section 3.2) of low degree, say up to degree 5 or 6, and then a spline if the polynomial fails to provide a satisfactory fit. (Of course, if third-derivative discontinuities are unacceptable to the user, a polynomial is the only choice.) If the problem is one of surface fitting, one of the spline routines should be used (Section 3.4). If the problem has more than two independent variables, it may be treated by the general linear routine in Section 3.5.1, again using a polynomial in the first instance.

Another factor which affects the choice of routine is the presence of constraints, as previously discussed in Section 2.5. Indeed this factor is likely to be overriding at present, because of the limited number of routines which have the necessary facility. See Section 3.6.

3.1.1. Data considerations

A satisfactory fit cannot be expected by any means if the number and arrangement of the data points do not adequately represent the character of the underlying relationship: sharp changes in behaviour, in particular, such as sharp peaks, should be well covered. Data points should extend over the whole range of interest of the independent variable(s): extrapolation outside the data ranges is most unwise. Then, with polynomials, it is advantageous to have additional points near the ends of the ranges, to counteract the tendency of polynomials to develop fluctuations in these regions. When, with polynomial curves, the user can precisely choose the x-values of the data, the special points defined in Section 3.2.2 should be selected. With splines the choice is less critical as long as the character of the relationship is adequately represented. All fits should be tested graphically before accepting them as satisfactory.

For this purpose it should be noted that it is not sufficient to plot the values of the fitted function only at the data values of the independent variable(s); at the least, its values at a similar number of intermediate points should also be plotted, as unwanted fluctuations may otherwise go undetected. Such fluctuations are the less likely to occur the lower the number of coefficients chosen in the fitting function. No firm guide can be given, but as a rough rule, at least initially, the number of coefficients should not exceed half the number of data points (points with equal or nearly equal values of the independent variable, or both independent variables in surface fitting, counting as a single point for this purpose). However, the situation may be such, particularly with a small number of data points, that a satisfactorily close fit to the data cannot be achieved without unwanted fluctuations occurring. In such cases, it is often possible to improve the situation by a transformation of one or more of the variables, as discussed in the next paragraph: otherwise it will be necessary to provide extra data points. Further advice on curve fitting is given in Cox and Hayes [1] and, for polynomials only, in Hayes [3] of Section 2.7. Much of the advice applies also to surface fitting; see also the Routine Documents.

3.1.2. Transformation of variables

Before starting the fitting, consideration should be given to the choice of a good form in which to deal with each of the variables: often it will be satisfactory to use the variables as they stand, but sometimes the use of the logarithm, square root, or some other function of a variable will lead to a better-behaved relationship. This question is customarily taken into account in preparing graphs and tables of a relationship and the same considerations apply when curve or surface fitting. The practical context will often give a guide. In general, it is best to avoid having to deal with a relationship whose behaviour in one region is radically different from that in another. A steep rise at the left-hand end of a curve, for example, can often best be treated by curve fitting in terms of log(x+c) with some suitable value of the constant c. A case when such a transformation gave substantial benefit is discussed in Hayes [3] page 60. According to the features exhibited in any particular case, transformation of either dependent variable or independent variable(s) or both may be beneficial. When there is a choice it is usually better to transform the independent variable(s): if the dependent variable is transformed, the weights attached to the data points must be adjusted. Thus (denoting the dependent variable by y, as in the notation for curves) if the y_r to be fitted have been obtained by a transformation y=g(Y) from original data values Y_r, with weights W_r, for r=1,2,...,m, we must take

$$w_r = W_r/(dy/dY), \qquad (18)$$

where the derivative is evaluated at Y_r. Strictly, the transformation of Y and the adjustment of weights are valid only when the data errors in the Y_r are small compared with the range spanned by the Y_r, but this is usually the case.

3.2. Polynomial Curves

3.2.1. Least-squares polynomials: arbitrary data points

E02ADF fits to arbitrary data points, with arbitrary weights, polynomials of all degrees up to a maximum degree k, which is at choice. If the user is seeking only a low degree polynomial, up to degree 5 or 6 say, k=10 is an appropriate value, providing there are about 20 data points or more. To assist in deciding the degree of polynomial which satisfactorily fits the data, the routine provides the root-mean-square-residual s_i for all degrees i=1,2,...,k. In a satisfactory case, these s_i will decrease steadily as i increases and then settle down to a fairly constant value, as shown in the example

i	s_i
0	3.5215
1	0.7708
2	0.1861
3	0.0820
4	0.0554
5	0.0251
6	0.0264
7	0.0280
8	0.0277
9	0.0297
10	0.0271

If the s_i values settle down in this way, it indicates that the closest polynomial approximation justified by the data has been achieved. The degree which first gives the approximately constant value of s_i (degree 5 in the example) is the appropriate degree to select. (Users who are prepared to accept a fit higher than sixth degree, should simply find a high enough value of k to enable the type of behaviour indicated by the example to be detected: thus they should seek values of k for which at least 4 or 5 consecutive values of s_i are approximately the same.) If the degree were allowed to go high enough, s_i would, in most cases, eventually start to decrease again, indicating that the data points are being fitted too closely and that undesirable fluctuations are developing between the points. In some cases, particularly with a small number of data points, this final decrease is not distinguishable from the initial decrease in s_i. In such cases, users may seek an acceptable fit by examining the graphs of several of the polynomials obtained. Failing this, they may (a) seek a transformation of variables which improves the behaviour, (b) try fitting a spline, or (c) provide more data points. If data can be provided simply by drawing an approximating curve by hand and reading points from it, use the points discussed in Section 3.2.2.

3.2.2. Least-squares polynomials: selected data points

When users are at liberty to choose the x-values of data points, such as when the points are taken from a graph, it is most advantageous when fitting with polynomials to use the values x_r =cos((pi)r/n), for r=0,1,...,n for some value of n, a suitable value for which is discussed at the end of this section. Note that these x_r relate to the variable x after it has been normalised so that its range of interest is -1 to +1. E02ADF may then be used as in Section 3.2.1 to seek a satisfactory fit.

3.3. Cubic Spline Curves

3.3.1. Least-squares cubic splines

E02BAF fits to arbitrary data points, with arbitrary weights, a cubic spline with interior knots specified by the user. The choice of these knots so as to give an acceptable fit must largely be a matter of trial and error, though with a little experience a satisfactory choice can often be made after one or two trials. It is usually best to start with a small number of knots (too many will result in unwanted fluctuations in the fit, or even in there being no unique solution) and, examining the fit graphically at each stage, to add a few knots at a time at places where the fit is particularly poor. Moving the existing knots towards these places will also often improve the fit. In regions where the behaviour of the curve underlying the data is changing rapidly, closer knots will be needed than elsewhere. Otherwise, positioning is not usually very critical and equally-spaced knots are often satisfactory. See also the next section, however.

A useful feature of the routine is that it can be used in applications which require the continuity to be less than the normal continuity of the cubic spline. For example, the fit may be required to have a discontinuous slope at some point in the range. This can be achieved by placing three coincident knots at the given point. Similarly a discontinuity in the second derivative at a point can be achieved by placing two knots there. Analogy with these discontinuous cases can provide guidance in more usual cases: for example, just as three coincident knots can produce a discontinuity in slope, so three close knots can produce a rapid change in slope. The closer the knots are, the more rapid can the change be.

Figure 1

Please see figure in printed Reference Manual

An example set of data is given in Figure 1. It is a rather tricky set, because of the scarcity of data on the right, but it will serve to illustrate some of the above points and to show some of the dangers to be avoided. Three interior knots (indicated by the vertical lines at the top of the diagram) are chosen as a start. We see that the resulting curve is not steep enough in the middle and fluctuates at both ends, severely on the right. The spline is unable to cope with the shape and more knots are needed.

In Figure 2, three knots have been added in the centre, where the data shows a rapid change in behaviour, and one further out at each end, where the fit is poor. The fit is still poor, so a further knot is added in this region and, in Figure 3, disaster ensues in rather spectacular fashion.

Figure 2
Please see figure in printed Reference Manual

Figure 3
Please see figure in printed Reference Manual

The reason is that, at the right-hand end, the fits in Figure 1 and Figure 2 have been interpreted as poor simply because of the fluctuations about the curve underlying the data (or what it is naturally assumed to be). But the fitting process knows only about the data and nothing else about the underlying curve, so it is important to consider only closeness to the data when deciding goodness of fit.

Thus, in Figure 1, the curve fits the last two data points quite well compared with the fit elsewhere, so no knot should have been added in this region. In Figure 2, the curve goes exactly through the last two points, so a further knot is certainly not needed here.

Figure 4
Please see figure in printed Reference Manual

Figure 4 shows what can be achieved without the extra knot on each of the flat regions. Remembering that within each knot interval the spline is a cubic polynomial, there is really no need to have more than one knot interval covering each flat region.

What we have, in fact, in Figure 2 and Figure 3 is a case of too many knots (so too many coefficients in the spline equation) for the number of data points. The warning in the second paragraph of Section 2.1 was that the fit will then be too close to the data, tending to have unwanted fluctuations between the data points. The warning applies locally for splines, in the sense that, in localities where there are plenty of data points, there can be a lot of knots, as long as there are few knots where there are few points, especially near the ends of the interval. In the present example, with so few data points on the right, just the one extra knot in Figure 2 is too many! The signs are clearly present, with the last two points fitted exactly (at least to the graphical accuracy and actually much closer than that) and fluctuations within the last two knot-intervals (cf. Figure 1, where only the final point is fitted exactly and one of the wobbles spans several data points).

The situation in Figure 3 is different. The fit, if computed exactly, would still pass through the last two data points, with even more violent fluctuations. However, the problem has become so ill-conditioned that all accuracy has been lost. Indeed, if the last interior knot were moved a tiny amount to the right, there would be no unique solution and an error message would have been caused. Near-singularity is, sadly, not picked up by the routine, but can be spotted readily in a graph, as Figure 3. B-spline coefficients becoming large, with alternating signs, is another indication. However, it is better to avoid such situations, firstly by providing, whenever possible, data adequately covering the range of interest, and secondly by placing knots only where there is a reasonable amount of data.

The example here could, in fact, have utilised from the start the observation made in the second paragraph of this section, that three close knots can produce a rapid change in slope. The example has two such rapid changes and so requires two sets of three close knots (in fact, the two sets can be so close that one knot can serve in both sets, so only five knots prove sufficient in Figure 4). It should be noted, however, that the rapid turn occurs within the range spanned by the three knots. This is the reason that the six knots in Figure 2 are not satisfactory as they do not quite span the two turns.

Some more examples to illustrate the choice of knots are given in Cox and Hayes [1].

3.3.2. Automatic fitting with cubic splines

E02BEF also fits cubic splines to arbitrary data points with arbitrary weights but itself chooses the number and positions of the knots. The user has to supply only a threshold for the sum of squares of residuals. The routine first builds up a knot set by a series of trial fits in the l norm. Then, with the knot set

2

decided, the final spline is computed to minimize a certain smoothing measure subject to satisfaction of the chosen threshold. Thus it is easier to use than E02BAF (see previous section), requiring only some experimentation with this threshold. It should therefore be first choice unless the user has a preference for the ordinary least-squares fit or, for example, wishes to experiment with knot positions, trying to keep their number down (E02BEF aims only to be reasonably frugal with knots).

3.4. Spline Surfaces

3.4.1. Least-squares bicubic splines

E02DAF fits to arbitrary data points, with arbitrary weights, a bicubic spline with its two sets of interior knots specified by the user. For choosing these knots, the advice given for cubic splines, in Section 3.3.1 above, applies here too. (See also the next section, however.) If changes in the behaviour of the surface underlying the data are more marked in the direction of one variable than of the other, more knots will be needed for the former variable than the latter. Note also that, in the surface case, the reduction in continuity caused by coincident knots will extend across the whole spline surface: for example, if three knots associated with the variable x are chosen to coincide at a value L, the spline surface will have a discontinuous slope across the whole extent of the line x=L.

With some sets of data and some choices of knots, the least-squares bicubic spline will not be unique. This will not occur, with a reasonable choice of knots, if the rectangle R is well covered with data points: here R is defined as the smallest rectangle in the (x,y) plane, with sides parallel to the axes, which contains all the data points. Where the least-squares solution is not unique, the minimal least-squares solution is computed, namely that least-squares solution which has the smallest value of the sum of squares of the B-spline coefficients c_{ij} (see the end of Section 2.3.2 above). This choice of least-squares solution tends to minimize the risk of unwanted fluctuations in the fit. The fit will not be reliable, however, in regions where there are few or no data points.

3.4.2. Automatic fitting with bicubic splines

E02DDF also fits bicubic splines to arbitrary data points with arbitrary weights but chooses the knot sets itself. The user has to supply only a threshold for the sum of squares of residuals. Just like the automatic curve E02BEF (Section 3.3.2), E02DDF then builds up the knot sets and finally fits a spline minimizing a smoothing measure subject to satisfaction of the threshold. Again, this easier to use routine is normally to be preferred, at least in the first instance.

E02DCF is a very similar routine to E02DDF but deals with data points of equal weight which lie on a rectangular mesh in the (x,y) plane. This kind of data allows a very much faster computation and so is to be preferred when applicable. Substantial departures from equal weighting can be ignored if the user is not concerned with statistical questions, though the quality of the fit will suffer if this is taken too far. In such cases, the user should revert to E02DDF.

3.5. General Linear and Nonlinear Fitting Functions

3.5.1. General linear functions

For the general linear function (15), routines are available for fitting in the l_1 and l_2 norms. The least-squares routines (which are to be preferred unless there is good reason to use another norm -- see Section 2.1.1) are in Chapter F04. The l_1 routine is E02GAF.

All the above routines are essentially linear algebra routines, and in considering their use we need to view the fitting process in a slightly different way from hitherto. Taking y to be the dependent variable and x the vector of independent variables, we have, as for equation (1) but with each x_r now a vector,

$$\epsilon_r = y_r - f(x_r) \quad r=1,2,\ldots,m.$$

Substituting for f(x) the general linear form (15), we can write this as

$$c_1 \phi_1(x_r) + c_2 \phi_2(x_r) + \ldots + c_p \phi_p(x_r) = y_r - \epsilon_r, \quad r=1,2,\ldots,m \qquad (19)$$

Thus we have a system of linear equations in the coefficients c_j. Usually, in writing these equations, the ϵ_r are omitted and simply taken as implied. The system of equations is then described as an overdetermined system (since we must have m>=p if there is to be the possibility of a unique solution to our fitting problem), and the fitting process of computing the c_j to minimize one or other of the norms (2), (3) and (4) can be described, in relation to the system of equations, as solving the overdetermined system in that particular norm. In matrix notation, the system can be written as

$$(\Phi)c=y, \qquad (20)$$

where (Phi) is the m by p matrix whose element in row r and column j is $(\phi)_j(x_r)$, for r=1,2,...,m; j=1,2,...,p. The vectors c and y respectively contain the coefficients c_j and the data values y_r.

The routines, however, use the standard notation of linear algebra, the overdetermined system of equations being denoted by

$$Ax=b \qquad (21)$$

The correspondence between this notation and that which we have used for the data-fitting problem (equation (20)) is therefore given by

$$A==(\Phi), \quad x==c \quad b==y \qquad (22)$$

Note that the norms used by these routines are the unweighted norms (2) and (3). If the user wishes to apply weights to the data points, that is to use the norms (5) or (6), the equivalences (22) should be replaced by

$$A==D(\Phi), \quad x==c \quad b==Dy$$

where D is a diagonal matrix with w_r as the rth diagonal element. Here w_r, for r=1,2,...,m, is the weight of the rth data point as defined in Section 2.1.2.

3.5.2. Nonlinear functions

Routines for fitting with a nonlinear function in the l_2 norm are provided in Chapter E04, and that chapter's Introduction should be consulted for the appropriate choice of routine. Again, however, the notation adopted is different from that we have used for data fitting. In the latter, we denote the fitting function by $f(x;c)$, where x is the vector of independent variables and c is the vector of coefficients, whose values are to be determined. The squared l_2 norm, to be minimized with respect to the elements of c, is then

$$\sum_{r=1}^{m} w_r^2 [y_r - f(x_r;c)]^2 \qquad (23)$$

where y_r is the rth data value of the dependent variable, x_r is the vector containing the rth values of the independent variables, and w_r is the corresponding weight as defined in Section 2.1.2.

On the other hand, in the nonlinear least-squares routines of Chapter E04, the function to be minimized is denoted by

$$\sum_{i=1}^{m} f_i^2(x), \qquad (24)$$

the minimization being carried out with respect to the elements of the vector x. The correspondence between the two notations is given by

$x \equiv c$ and

$f_i(x) \equiv w_r[y_r - f(x_r;c)]$, $i=r=1,2,\ldots,m$.

Note especially that the vector x of variables of the nonlinear least-squares routines is the vector c of coefficients of the data-fitting problem, and in particular that, if the selected routine requires derivatives of the $f_i(x)$ to be provided, these are derivatives of $w_r[y_r - f(x_r;c)]$ with respect to the coefficients of the data-fitting problem.

3.6. Constraints

At present, there are only a limited number of routines which fit subject to constraints. Chapter E04 contains a routine, E04UCF, which can be used for fitting with a nonlinear function in the l_2 norm subject to equality or inequality constraints. This routine, unlike those in that chapter suited to the unconstrained case, is not designed specifically for minimizing functions which are sums of squares, and so the function (23) has to be treated as a general nonlinear function. The E04 Chapter Introduction should be consulted.

The remaining constraint routine relates to fitting with polynomials in the l_2 norm. E02AGF deals with polynomial curves and allows precise values of the fitting function and (if required) all its derivatives up to a given order to be prescribed at one or more values of the independent variable.

3.7. Evaluation, Differentiation and Integration

Routines are available to evaluate, differentiate and integrate polynomials in Chebyshev-series form and cubic or bicubic splines in B-spline form. These polynomials and splines may have been produced by the various fitting routines or, in the case of polynomials, from prior calls of the differentiation and integration routines themselves.

E02AEF and E02AKF evaluate polynomial curves: the latter has a longer parameter list but does not require the user to normalise the values of the independent variable and can accept coefficients which are not stored in contiguous locations. E02BBF evaluates cubic spline curves, and E02DEF and E02DFF bicubic spline surfaces.

Differentiation and integration of polynomial curves are carried out by E02AHF and E02AJF respectively. The results are provided in Chebyshev-series form and so repeated differentiation and integration are catered for. Values of the derivative or integral can then be computed using the appropriate evaluation routine.

For splines the differentiation and integration routines provided are of a different nature from those for polynomials. E02BCF provides values of a cubic spline curve and its first three derivatives (the rest, of course, are zero) at a given value of x spline over its whole range. These routines can also be applied to surfaces of the form (14). For example, if, for each value of j in turn, the coefficients c_{ij}, for i=1,2,...,p are supplied to E02BCF with $x=x_0$ and on each occasion we select from the output the value of the second derivative, d_j say, and if the whole set of d_j are then supplied to the same routine with $x=y_0$, the output will contain all the values at (x_0,y_0) of

$$\frac{\partial^2 f}{\partial x^2} \quad \text{and} \quad \frac{\partial^{r+2} f}{\partial x^2 \partial y^r}, \quad r=1,2,3.$$

Equally, if after each of the first p calls of E02BCF we had selected the function value (E02BBF would also provide this) instead of the second derivative and we had supplied these values to E02BDF, the result obtained would have been the value of

$$\int_A^B f(x_0,y)dy,$$

where A and B are the end-points of the y interval over which the spline was defined.

3.8. Index

```
         with constraints                                  E02AGF
Least-squares surface fit with bicubic splines            E02DAF
l  fit with general linear function,                       E02GAF
 1
Sorting,
     2-D data into panels                                  E02ZAF


E02 -- Curve and Surface Fitting                  Contents -- E02
Chapter E02

Curve and Surface Fitting

E02ADF  Least-squares curve fit, by polynomials, arbitrary data
        points

E02AEF  Evaluation of fitted polynomial in one variable from
        Chebyshev series form (simplified parameter list)

E02AGF  Least-squares polynomial fit, values and derivatives may
        be constrained, arbitrary data points,

E02AHF  Derivative of fitted polynomial in Chebyshev series form

E02AJF  Integral of fitted polynomial in Chebyshev series form

E02AKF  Evaluation of fitted polynomial in one variable, from
        Chebyshev series form

E02BAF  Least-squares curve cubic spline fit (including
        interpolation)

E02BBF  Evaluation of fitted cubic spline, function only

E02BCF  Evaluation of fitted cubic spline, function and
        derivatives

E02BDF  Evaluation of fitted cubic spline, definite integral

E02BEF  Least-squares cubic spline curve fit, automatic knot
        placement

E02DAF  Least-squares surface fit, bicubic splines

E02DCF  Least-squares surface fit by bicubic splines with
        automatic knot placement, data on rectangular grid

E02DDF  Least-squares surface fit by bicubic splines with
        automatic knot placement, scattered data
```

```
         E02DEF  Evaluation of a fitted bicubic spline at a vector of
                 points

         E02DFF  Evaluation of a fitted bicubic spline at a mesh of points

         E02GAF  L -approximation by general linear function
                  1

         E02ZAF  Sort 2-D data into panels for fitting bicubic splines

\end{verbatim}
\endscroll
\end{page}
```

Least-squares polynomial approximations

— nage.ht —

```
\begin{page}{manpageXXe02adf}{NAG Documentation: e02adf}
\beginscroll
\begin{verbatim}



     E02ADF(3NAG)      Foundation Library (12/10/92)      E02ADF(3NAG)



          E02 -- Curve and Surface Fitting                           E02ADF
                  E02ADF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          E02ADF computes weighted least-squares polynomial approximations
          to an arbitrary set of data points.

          2. Specification

                 SUBROUTINE E02ADF (M, KPLUS1, NROWS, X, Y, W, WORK1,
                1                   WORK2, A, S, IFAIL)
```

```
      INTEGER          M, KPLUS1, NROWS, IFAIL
      DOUBLE PRECISION X(M), Y(M), W(M), WORK1(3*M), WORK2
     1                 (2*KPLUS1), A(NROWS,KPLUS1), S(KPLUS1)
```

3. Description

This routine determines least-squares polynomial approximations of degrees $0,1,...,k$ to the set of data points (x_r,y_r) with weights w_r, for $r=1,2,...,m$.

The approximation of degree i has the property that it minimizes σ_i the sum of squares of the weighted residuals ϵ_r, where

$$\epsilon_r=w_r(y_r-f_r)$$

and f_r is the value of the polynomial of degree i at the rth data point.

Each polynomial is represented in Chebyshev-series form with normalised argument x. This argument lies in the range -1 to +1 and is related to the original variable x by the linear transformation

$$x=\frac{(2x-x_{max}-x_{min})}{(x_{max}-x_{min})}.$$

Here x_{max} and x_{min} are respectively the largest and smallest values of x_r. The polynomial approximation of degree i is represented as

$$\frac{1}{2}a_{i+1,1}T_0(x)+a_{i+1,2}T_1(x)+a_{i+1,3}T_2(x)+...+a_{i+1,i+1}T_i(x),$$

where $T_i(x)$ is the Chebyshev polynomial of the first kind of

j

degree j with argument (x).

For i=0,1,...,k, the routine produces the values of $a_{i+1,j+1}$, for j=0,1,...,i, together with the value of the root mean square

residual $s_i = \sqrt{\frac{\sigma_i}{m-i-1}}$. In the case m=i+1 the routine sets the value of s_i to zero.

The method employed is due to Forsythe [4] and is based upon the generation of a set of polynomials orthogonal with respect to summation over the normalised data set. The extensions due to Clenshaw [1] to represent these polynomials as well as the approximating polynomials in their Chebyshev-series forms are incorporated. The modifications suggested by Reinsch and Gentleman (see [5]) to the method originally employed by Clenshaw for evaluating the orthogonal polynomials from their Chebyshev-series representations are used to give greater numerical stability.

For further details of the algorithm and its use see Cox [2] and [3].

Subsequent evaluation of the Chebyshev-series representations of the polynomial approximations should be carried out using E02AEF.

4. References

[1] Clenshaw C W (1960) Curve Fitting with a Digital Computer. Comput. J. 2 170--173.

[2] Cox M G (1974) A Data-fitting Package for the Non-specialist User. Software for Numerical Mathematics. (ed D J Evans) Academic Press.

[3] Cox M G and Hayes J G (1973) Curve fitting: a guide and suite of algorithms for the non-specialist user. Report NAC26. National Physical Laboratory.

[4] Forsythe G E (1957) Generation and use of orthogonal polynomials for data fitting with a digital computer. J. Soc. Indust. Appl. Math. 5 74--88.

[5] Gentlemen W M (1969) An Error Analysis of Goertzel's (Watt's) Method for Computing Fourier Coefficients. Comput. J. 12 160--165.

[6] Hayes J G (1970) Curve Fitting by Polynomials in One Variable. Numerical Approximation to Functions and Data. (ed J G Hayes) Athlone Press, London.

5. Parameters

1: M -- INTEGER Input
On entry: the number m of data points. Constraint: M >= MDIST >= 2, where MDIST is the number of distinct x values in the data.

2: KPLUS1 -- INTEGER Input
On entry: k+1, where k is the maximum degree required. Constraint: 0 < KPLUS1 <= MDIST, where MDIST is the number of distinct x values in the data.

3: NROWS -- INTEGER Input
On entry:
the first dimension of the array A as declared in the (sub)program from which E02ADF is called.
Constraint: NROWS >= KPLUS1.

4: X(M) -- DOUBLE PRECISION array Input
On entry: the values x_r of the independent variable, for r=1,2,...,m. Constraint: the values must be supplied in non-decreasing order with X(M) > X(1).

5: Y(M) -- DOUBLE PRECISION array Input
On entry: the values y_r of the dependent variable, for r=1,2,...,m.

6: W(M) -- DOUBLE PRECISION array Input
On entry: the set of weights, w_r, for r=1,2,...,m. For advice on the choice of weights, see Section 2.1.2 of the Chapter Introduction. Constraint: W(r) > 0.0, for r=1,2,...m.

7: WORK1(3*M) -- DOUBLE PRECISION array Workspace

8: WORK2(2*KPLUS1) -- DOUBLE PRECISION array Workspace

9: A(NROWS,KPLUS1) -- DOUBLE PRECISION array Output

On exit: the coefficients of $T_j(x)$ in the approximating polynomial of degree i. A(i+1,j+1) contains the coefficient $a_{i+1,j+1}$, for i=0,1,...,k; j=0,1,...,i.

10: S(KPLUS1) -- DOUBLE PRECISION array Output
On exit: S(i+1) contains the root mean square residual s_i, for i=0,1,...,k, as described in Section 3. For the interpretation of the values of the s_i and their use in selecting an appropriate degree, see Section 3.1 of the Chapter Introduction.

11: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1
The weights are not all strictly positive.

IFAIL= 2
The values of X(r), for r=1,2,...,M are not in non-decreasing order.

IFAIL= 3
All X(r) have the same value: thus the normalisation of X is not possible.

IFAIL= 4
On entry KPLUS1 < 1 (so the maximum degree required is negative)

or KPLUS1 > MDIST, where MDIST is the number of distinct x values in the data (so there cannot be a unique solution for degree k=KPLUS1-1).

IFAIL= 5
NROWS < KPLUS1.

```
          7. Accuracy

          No error analysis for the method has been published. Practical
          experience with the method, however, is generally extremely
          satisfactory.

          8. Further Comments

          The time taken by the routine is approximately proportional to
          m(k+1)(k+11).

          The approximating polynomials may exhibit undesirable
          oscillations (particularly near the ends of the range) if the
          maximum degree k exceeds a critical value which depends on the
          number of data points m and their relative positions. As a rough
          guide, for equally-spaced data, this critical value is about


          2*\/m. For further details see Hayes [6] page 60.

          9. Example

          Determine weighted least-squares polynomial approximations of
          degrees 0, 1, 2 and 3 to a set of 11 prescribed data points. For
          the approximation of degree 3, tabulate the data and the
          corresponding values of the approximating polynomial, together
          with the residual errors, and also the values of the
          approximating polynomial at points half-way between each pair of
          adjacent data points.

          The example program supplied is written in a general form that
          will enable polynomial approximations of degrees 0,1,...,k to be
          obtained to m data points, with arbitrary positive weights, and
          the approximation of degree k to be tabulated. E02AEF is used to
          evaluate the approximating polynomial. The program is self-
          starting in that any number of data sets can be supplied.

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Evaluate polynomial from Chebyshev-series representation

— nage.ht —

```
\begin{page}{manpageXXe02aef}{NAG Documentation: e02aef}
\beginscroll
\begin{verbatim}



     E02AEF(3NAG)      Foundation Library (12/10/92)      E02AEF(3NAG)



          E02 -- Curve and Surface Fitting                           E02AEF
                  E02AEF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          E02AEF evaluates a polynomial from its Chebyshev-series
          representation.

          2. Specification

                 SUBROUTINE E02AEF (NPLUS1, A, XCAP, P, IFAIL)
                 INTEGER          NPLUS1, IFAIL
                 DOUBLE PRECISION A(NPLUS1), XCAP, P

          3. Description

          This routine evaluates the polynomial

                        1
                        -a T (x)+a T (x)+a T (x)+...+a   T (x)
                        2 1 0     2 1     3 2         n+1 n



          for any value of x satisfying -1<=x<=1. Here T (x) denotes the
                                                        j
          Chebyshev polynomial of the first kind of degree j with argument


          x. The value of n is prescribed by the user.
```

In practice, the variable x will usually have been obtained from an original variable x, where $x_{min} \le x \le x_{max}$ and

$$x = \frac{((x - x_{min}) - (x_{max} - x))}{(x_{max} - x_{min})}$$

Note that this form of the transformation should be used computationally rather than the mathematical equivalent

$$x = \frac{(2x - x_{min} - x_{max})}{(x_{max} - x_{min})}$$

since the former guarantees that the computed value of x differs from its true value by at most 4(epsilon), where (epsilon) is the machine precision, whereas the latter has no such guarantee.

The method employed is based upon the three-term recurrence relation due to Clenshaw [1], with modifications to give greater numerical stability due to Reinsch and Gentleman (see [4]).

For further details of the algorithm and its use see Cox [2] and [3].

4. References

[1] Clenshaw C W (1955) A Note on the Summation of Chebyshev Series. Math. Tables Aids Comput. 9 118--120.

[2] Cox M G (1974) A Data-fitting Package for the Non-specialist User. Software for Numerical Mathematics. (ed D J Evans) Academic Press.

[3] Cox M G and Hayes J G (1973) Curve fitting: a guide and suite of algorithms for the non-specialist user. Report NAC26. National Physical Laboratory.

[4] Gentlemen W M (1969) An Error Analysis of Goertzel's (Watt's) Method for Computing Fourier Coefficients. Comput.

J. 12 160--165.

5. Parameters

1: NPLUS1 -- INTEGER Input
On entry: the number n+1 of terms in the series (i.e., one greater than the degree of the polynomial). Constraint: NPLUS1 >= 1.

2: A(NPLUS1) -- DOUBLE PRECISION array Input
On entry: A(i) must be set to the value of the ith coefficient in the series, for i=1,2,...,n+1.

3: XCAP -- DOUBLE PRECISION Input

On entry: x, the argument at which the polynomial is to be evaluated. It should lie in the range -1 to +1, but a value just outside this range is permitted (see Section 6) to allow for possible rounding errors committed in the

transformation from x to x discussed in Section 3. Provided the recommended form of the transformation is used, a successful exit is thus assured whenever the value of x lies in the range x_{min} to x_{max}.

4: P -- DOUBLE PRECISION Output
On exit: the value of the polynomial.

5: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1
ABS(XCAP) > 1.0 + 4(epsilon), where (epsilon) is the machine precision. In this case the value of P is set arbitrarily to zero.

IFAIL= 2
On entry NPLUS1 < 1.

```
7. Accuracy

The rounding errors committed are such that the computed value of
the polynomial is exact for a slightly perturbed set of
coefficients a +(delta)a . The ratio of the sum of the absolute
              i         i
values of the (delta)a  to the sum of the absolute values of the
                      i
a  is less than a small multiple of (n+1) times machine
 i
precision.

8. Further Comments

The time taken by the routine is approximately proportional to
n+1.

It is expected that a common use of E02AEF will be the evaluation
of the polynomial approximations produced by E02ADF and E02AFF(*)

9. Example



Evaluate at 11 equally-spaced points in the interval -1<=x<=1 the
polynomial of degree 4 with Chebyshev coefficients, 2.0, 0.5, 0.
25, 0.125, 0.0625.

The example program is written in a general form that will enable
a polynomial of degree n in its Chebyshev-series form to be


evaluated at m equally-spaced points in the interval -1<=x<=1.
The program is self-starting in that any number of data sets can
be supplied.

The example program is not reproduced here. The source code for
all example programs is distributed with the NAG Foundation
Library software and should be available on-line.
```

```
\end{verbatim}
\endscroll
\end{page}
```

Constrained weighted least-squares polynomial

— nage.ht —

```
\begin{page}{manpageXXe02agf}{NAG Documentation: e02agf}
\beginscroll
\begin{verbatim}



     E02AGF(3NAG)      Foundation Library (12/10/92)      E02AGF(3NAG)



          E02 -- Curve and Surface Fitting                          E02AGF
                  E02AGF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          E02AGF computes constrained weighted least-squares polynomial
          approximations in Chebyshev-series form to an arbitrary set of
          data points. The values of the approximations and any number of
          their derivatives can be specified at selected points.

          2. Specification

                 SUBROUTINE E02AGF (M, KPLUS1, NROWS, XMIN, XMAX, X, Y, W,
                1                   MF, XF, YF, LYF, IP, A, S, NP1, WRK,
                2                   LWRK, IWRK, LIWRK, IFAIL)
                 INTEGER          M, KPLUS1, NROWS, MF, LYF, IP(MF), NP1,
                1                 LWRK, IWRK(LIWRK), LIWRK, IFAIL
                 DOUBLE PRECISION XMIN, XMAX, X(M), Y(M), W(M), XF(MF), YF
                1                 (LYF), A(NROWS,KPLUS1), S(KPLUS1), WRK
                2                 (LWRK)

          3. Description

          This routine determines least-squares polynomial approximations
          of degrees up to k to the set of data points (x ,y ) with weights
                                                         r  r
          w , for r=1,2,...,m. The value of k, the maximum degree required,
           r
          is prescribed by the user. At each of the values XF , for r =
                                                             r
```

1,2,...,MF, of the independent variable x, the approximations and their derivatives up to order p_r are constrained to have one of the user-specified values YF_s, for s=1,2,...,n, where $n=MF+\sum_{r=1}^{MF} p_r$

The approximation of degree i has the property that, subject to the imposed contraints, it minimizes $(Sigma)_i$, the sum of the squares of the weighted residuals $(epsilon)_r$ for r=1,2,...,m where

$$(epsilon)_r = w_r (y_r - f_i (x_r))$$

and $f_i (x_r)$ is the value of the polynomial approximation of degree i at the rth data point.

Each polynomial is represented in Chebyshev-series form with

normalised argument x. This argument lies in the range -1 to +1 and is related to the original variable x by the linear transformation

$$x = \frac{2x-(x_{max}+x_{min})}{(x_{max}-x_{min})}$$

where x_{min} and x_{max}, specified by the user, are respectively the lower and upper end-points of the interval of x over which the polynomials are to be defined.

The polynomial approximation of degree i can be written as

$$\frac{1}{2}a_{i,0}+a_{i,1}T_1(x)+...+a_{ij}T_j(x)+...+a_{ii}T_i(x)$$

where $T (x)$ is the Chebyshev polynomial of the first kind of j

degree j with argument x. For i=n,n+1,...,k, the routine produces the values of the coefficients a_{ij}, for j=0,1,...,i, together with the value of the root mean square residual, S_i, defined as

$$\sqrt{\frac{\sum_{i}}{(m'+n-i-1)}}$$

, where m' is the number of data points with non-zero weight.

Values of the approximations may subsequently be computed using E02AEF or E02AKF.

First E02AGF determines a polynomial (mu)(x), of degree n-1, which satisfies the given constraints, and a polynomial (nu)(x), of degree n, which has value (or derivative) zero wherever a constrained value (or derivative) is specified. It then fits y_r-(mu)(x_r), for r=1,2,...,m with polynomials of the required

degree in x each with factor (nu)(x). Finally the coefficients of

(mu)(x) are added to the coefficients of these fits to give the coefficients of the constrained polynomial approximations to the data points (x_r,y_r), for r=1,2,...,m. The method employed is given in Hayes [3]: it is an extension of Forsythe's orthogonal polynomials method [2] as modified by Clenshaw [1].

4. References

[1] Clenshaw C W (1960) Curve Fitting with a Digital Computer. Comput. J. 2 170--173.

[2] Forsythe G E (1957) Generation and use of orthogonal polynomials for data fitting with a digital computer. J. Soc. Indust. Appl. Math. 5 74--88.

[3] Hayes J G (1970) Curve Fitting by Polynomials in One Variable. Numerical Approximation to Functions and Data. (ed J G Hayes) Athlone Press, London.

5. Parameters

1: M -- INTEGER Input
On entry: the number m of data points to be fitted.
Constraint: M >= 1.

2: KPLUS1 -- INTEGER Input
On entry: k+1, where k is the maximum degree required.
Constraint: n+1<=KPLUS1<=m''+n, where n is the total number of constraints and m'' is the number of data points with non-zero weights and distinct abscissae which do not coincide with any of the XF(r).

3: NROWS -- INTEGER Input
On entry:
the first dimension of the array A as declared in the (sub)program from which E02AGF is called.
Constraint: NROWS >= KPLUS1.

4: XMIN -- DOUBLE PRECISION Input

5: XMAX -- DOUBLE PRECISION Input
On entry: the lower and upper end-points, respectively, of the interval $[x_{min},x_{max}]$. Unless there are specific reasons to the contrary, it is recommended that XMIN and XMAX be set respectively to the lowest and highest value among the x_r and XF(r). This avoids the danger of extrapolation provided there is a constraint point or data point with non-zero weight at each end-point. Constraint: XMAX > XMIN.

6: X(M) -- DOUBLE PRECISION array Input
On entry: the value x_r of the independent variable at the r th data point, for r=1,2,...,m. Constraint: the X(r) must be in non-decreasing order and satisfy XMIN <= X(r) <= XMAX.

7: Y(M) -- DOUBLE PRECISION array Input
On entry: Y(r) must contain y_r, the value of the dependent variable at the rth data point, for r=1,2,...,m.

8: W(M) -- DOUBLE PRECISION array Input
On entry: the weights w_r to be applied to the data points

x_r, for r=1,2...,m. For advice on the choice of weights see the Chapter Introduction. Negative weights are treated as positive. A zero weight causes the corresponding data point to be ignored. Zero weight should be given to any data point whose x and y values both coincide with those of a constraint (otherwise the denominators involved in the root-mean-square residuals s_i will be slightly in error).

9: MF -- INTEGER Input
On entry: the number of values of the independent variable at which a constraint is specified. Constraint: MF >= 1.

10: XF(MF) -- DOUBLE PRECISION array Input
On entry: the rth value of the independent variable at which a constraint is specified, for r = 1,2,...,MF. Constraint: these values need not be ordered but must be distinct and satisfy XMIN <= XF(r) <= XMAX.

11: YF(LYF) -- DOUBLE PRECISION array Input
On entry: the values which the approximating polynomials and their derivatives are required to take at the points specified in XF. For each value of XF(r), YF contains in successive elements the required value of the approximation, its first derivative, second derivative,..., p_r th derivative, for r = 1,2,...,MF. Thus the value which the kth derivative of each approximation (k=0 referring to the approximation itself) is required to take at the point XF(r) must be contained in YF(s), where

$$s=r+k+p_1+p_2+...+p_{r-1},$$

for k=0,1,...,p_r and r = 1,2,...,MF. The derivatives are with respect to the user's variable x.

12: LYF -- INTEGER Input
On entry:
the dimension of the array YF as declared in the (sub)program from which E02AGF is called.
Constraint: LYF>=n, where $n=MF+p_1+p_2+...+p_{MF}$.

13: IP(MF) -- INTEGER array Input
On entry: IP(r) must contain p_r, the order of the highest-order derivative specified at XF(r), for r = 1,2,...,MF. p_r=0 implies that the value of the approximation at XF(r) is

specified, but not that of any derivative. Constraint: IP(r) >= 0, for r=1,2,...,MF.

14: A(NROWS,KPLUS1) -- DOUBLE PRECISION array Output
On exit: A(i+1,j+1) contains the coefficient a_{ij} in the approximating polynomial of degree i, for i=n,n+1,...,k; j=0,1,...,i.

15: S(KPLUS1) -- DOUBLE PRECISION array Output
On exit: S(i+1) contains s_i, for i=n,n+1,...,k, the root-mean-square residual corresponding to the approximating polynomial of degree i. In the case where the number of data points with non-zero weight is equal to k+1-n, s_i is indeterminate: the routine sets it to zero. For the interpretation of the values of s_i and their use in selecting an appropriate degree, see Section 3.1 of the Chapter Introduction.

16: NP1 -- INTEGER Output
On exit: n+1, where n is the total number of constraint conditions imposed: $n=MF+p_1+p_2+...+p_{MF}$.

17: WRK(LWRK) -- DOUBLE PRECISION array Output
On exit: WRK contains weighted residuals of the highest degree of fit determined (k). The residual at x is in element 2(n+1)+3(m+k+1)+r, for r=1,2,...,m. The rest of the array is used as workspace.

18: LWRK -- INTEGER Input
On entry:
the dimension of the array WRK as declared in the (sub)program from which E02AGF is called.
Constraint: LWRK>=max(4*M+3*KPLUS1, 8*n+5*IPMAX+MF+10)+2*n+2 , where IPMAX = max(IP(R)).

19: IWRK(LIWRK) -- INTEGER array Workspace

20: LIWRK -- INTEGER Input
On entry:
the dimension of the array IWRK as declared in the (sub)program from which E02AGF is called.
Constraint: LIWRK>=2*MF+2.

21: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not

familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1
On entry M < 1,

or KPLUS1 < n + 1,

or NROWS < KPLUS1,

or MF < 1,

or LYF < n,

or LWRK is too small (see Section 5),

or LIWRK<2*MF+2.
(Here n is the total number of constraint conditions.)

IFAIL= 2
IP(r) < 0 for some r = 1,2,...,MF.

IFAIL= 3
XMIN >= XMAX, or XF(r) is not in the interval XMIN to XMAX for some r = 1,2,...,MF, or the XF(r) are not distinct.

IFAIL= 4
X(r) is not in the interval XMIN to XMAX for some r=1,2,...,M.

IFAIL= 5
X(r) < X(r-1) for some r=2,3,...,M.

IFAIL= 6
KPLUS1>m''+n, where m'' is the number of data points with non-zero weight and distinct abscissae which do not coincide with any XF(r). Thus there is no unique solution.

IFAIL= 7
The polynomials (mu)(x) and/or (nu)(x) cannot be determined. The problem supplied is too ill-conditioned. This may occur when the constraint points are very close together, or large in number, or when an attempt is made to constrain high-

order derivatives.

7. Accuracy

No complete error analysis exists for either the interpolating algorithm or the approximating algorithm. However, considerable experience with the approximating algorithm shows that it is generally extremely satisfactory. Also the moderate number of constraints, of low order, which are typical of data fitting applications, are unlikely to cause difficulty with the interpolating routine.

8. Further Comments

The time taken by the routine to form the interpolating polynomial is approximately proportional to n^3, and that to form the approximating polynomials is very approximately proportional to m(k+1)(k+1-n).

To carry out a least-squares polynomial fit without constraints, use E02ADF. To carry out polynomial interpolation only, use E01AEF(*).

9. Example

The example program reads data in the following order, using the notation of the parameter list above:

MF

IP(i), XF(i), Y-value and derivative values (if any) at XF(i), for i= 1,2,...,MF

M

X(i), Y(i), W(i), for i=1,2,...,M

k, XMIN, XMAX

The output is:

the root-mean-square residual for each degree from n to k;

the Chebyshev coefficients for the fit of degree k;

the data points, and the fitted values and residuals for the fit of degree k.

The program is written in a generalized form which will read any

```
         number of data sets.

         The data set supplied specifies 5 data points in the interval [0.
         0,4.0] with unit weights, to which are to be fitted polynomials,
         p, of degrees up to 4, subject to the 3 constraints:

              p(0.0)=1.0,  p'(0.0)=-2.0,  p(4.0)=9.0.

         The example program is not reproduced here. The source code for
         all example programs is distributed with the NAG Foundation
         Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

———

Coefficients of polynomial derivative

— nage.ht —

```
\begin{page}{manpageXXe02ahf}{NAG Documentation: e02ahf}
\beginscroll
\begin{verbatim}



     E02AHF(3NAG)      Foundation Library (12/10/92)      E02AHF(3NAG)



         E02 -- Curve and Surface Fitting                           E02AHF
                 E02AHF -- NAG Foundation Library Routine Document

         Note: Before using this routine, please read the Users' Note for
         your implementation to check implementation-dependent details.
         The symbol (*) after a NAG routine name denotes a routine that is
         not included in the Foundation Library.

         1. Purpose

         E02AHF determines the coefficients in the Chebyshev-series
         representation of the derivative of a polynomial given in
         Chebyshev-series form.

         2. Specification
```

```
      SUBROUTINE E02AHF (NP1, XMIN, XMAX, A, IA1, LA, PATM1,
     1                   ADIF, IADIF1, LADIF, IFAIL)
      INTEGER          NP1, IA1, LA, IADIF1, LADIF, IFAIL
      DOUBLE PRECISION XMIN, XMAX, A(LA), PATM1, ADIF(LADIF)
```

3. Description

This routine forms the polynomial which is the derivative of a given polynomial. Both the original polynomial and its derivative are represented in Chebyshev-series form. Given the coefficients a_i, for i=0,1,...,n, of a polynomial p(x) of degree n, where

$$p(x)=\frac{1}{2}a_0+a_1T_1(x)+...+a_nT_n(x)$$

the routine returns the coefficients a_i, for i=0,1,...,n-1, of the polynomial q(x) of degree n-1, where

$$q(x)=\frac{dp(x)}{dx}=\frac{1}{2}a_0+a_1T_1(x)+...+a_{n-1}T_{n-1}(x).$$

Here $T_j(x)$ denotes the Chebyshev polynomial of the first kind of degree j with argument x. It is assumed that the normalised variable x in the interval [-1,+1] was obtained from the user's original variable x in the interval $[x_{min},x_{max}]$ by the linear transformation

$$x=\frac{2x-(x_{max}+x_{min})}{x_{max}-x_{min}}$$

and that the user requires the derivative to be with respect to the variable x. If the derivative with respect to x is required, set $x_{max}=1$ and $x_{min}=-1$.

Values of the derivative can subsequently be computed, from the coefficients obtained, by using E02AKF.

The method employed is that of [1] modified to obtain the

derivative with respect to x. Initially setting $a_{n+1}=a_n=0$, the routine forms successively

$$a_{i-1}=a_{i+1}+\frac{2}{x_{max}-x_{min}}2ia_i, \quad i=n,n-1,...,1.$$

4. References

[1] Unknown (1961) Chebyshev-series. Modern Computing Methods, Chapter 8. NPL Notes on Applied Science (2nd Edition). 16 HMSO.

5. Parameters

1: NP1 -- INTEGER Input
On entry: n+1, where n is the degree of the given polynomial p(x). Thus NP1 is the number of coefficients in this polynomial. Constraint: NP1 >= 1.

2: XMIN -- DOUBLE PRECISION Input

3: XMAX -- DOUBLE PRECISION Input
On entry: the lower and upper end-points respectively of the interval $[x_{min},x_{max}]$. The Chebyshev-series

representation is in terms of the normalised variable x, where

$$x=\frac{2x-(x_{max}+x_{min})}{x_{max}-x_{min}}.$$

Constraint: XMAX > XMIN.

```
4:  A(LA) -- DOUBLE PRECISION array                          Input
    On entry: the Chebyshev coefficients of the polynomial p(x).
    Specifically, element 1 + i*IA1 of A must contain the
    coefficient a , for i=0,1,...,n. Only these n+1 elements
                 i
    will be accessed.

    Unchanged on exit, but see ADIF, below.

5:  IA1 -- INTEGER                                           Input
    On entry: the index increment of A. Most frequently the
    Chebyshev coefficients are stored in adjacent elements of A,
    and IA1 must be set to 1. However, if, for example, they are
    stored in A(1),A(4),A(7),..., then the value of IA1 must be
    3. See also Section 8. Constraint: IA1 >= 1.

6:  LA -- INTEGER                                            Input
    On entry:
    the dimension of the array A as declared in the (sub)program
    from which E02AHF is called.
    Constraint: LA>=1+(NP1-1)*IA1.

7:  PATM1 -- DOUBLE PRECISION                               Output
    On exit: the value of p(x   ). If this value is passed to
                              min
    the integration routine E02AJF with the coefficients of q(x)
    , then the original polynomial p(x) is recovered, including
    its constant coefficient.

8:  ADIF(LADIF) -- DOUBLE PRECISION array                   Output
    On exit: the Chebyshev coefficients of the derived
    polynomial q(x). (The differentiation is with respect to the
    variable x). Specifically, element 1+i*IADIF1 of ADIF


    contains the coefficient a , i=0,1,...n-1. Additionally
                              i
    element 1+n*IADIF1 is set to zero. A call of the routine may
    have the array name ADIF the same as A, provided that note
    is taken of the order in which elements are overwritten,
    when choosing the starting elements and increments IA1 and
    IADIF1: i.e., the coefficients a ,a ,...,a    must be intact
                                    0  1      i-1


    after coefficient a  is stored. In particular, it is
                       i
    possible to overwrite the a  completely by having IA1 =
                               i
    IADIF1, and the actual arrays for A and ADIF identical.
```

```
 9:  IADIF1 -- INTEGER                                       Input
     On entry: the index increment of ADIF. Most frequently the
     Chebyshev coefficients are required in adjacent elements of
     ADIF, and IADIF1 must be set to 1. However, if, for example,
     they are to be stored in ADIF(1),ADIF(4),ADIF(7),..., then
     the value of IADIF1 must be 3. See Section 8. Constraint:
     IADIF1 >= 1.

10:  LADIF -- INTEGER                                        Input
     On entry:
     the dimension of the array ADIF as declared in the
     (sub)program from which E02AHF is called.
     Constraint: LADIF>=1+(NP1-1)*IADIF1.

11:  IFAIL -- INTEGER                                 Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).
```

6. Error Indicators and Warnings

Errors detected by the routine:

```
IFAIL= 1
     On entry NP1 < 1,

     or       XMAX <= XMIN,

     or       IA1 < 1,

     or       LA<=(NP1-1)*IA1,

     or       IADIF1 < 1,

     or       LADIF<=(NP1-1)*IADIF1.
```

7. Accuracy

There is always a loss of precision in numerical differentiation, in this case associated with the multiplication by 2i in the formula quoted in Section 3.

8. Further Comments

The time taken by the routine is approximately proportional to n+1.

```
         The increments IA1, IADIF1 are included as parameters to give a
         degree of flexibility which, for example, allows a polynomial in
         two variables to be differentiated with respect to either
         variable without rearranging the coefficients.

         9. Example

         Suppose a polynomial has been computed in Chebyshev-series form
         to fit data over the interval [-0.5,2.5]. The example program
         evaluates the 1st and 2nd derivatives of this polynomial at 4
         equally spaced points over the interval. (For the purposes of
         this example, XMIN, XMAX and the Chebyshev coefficients are
         simply supplied in DATA statements. Normally a program would
         first read in or generate data and compute the fitted
         polynomial.)

         The example program is not reproduced here. The source code for
         all example programs is distributed with the NAG Foundation
         Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Find coefficients of indefinite integral of polynomial

— nage.ht —

```
\begin{page}{manpageXXe02ajf}{NAG Documentation: e02ajf}
\beginscroll
\begin{verbatim}



     E02AJF(3NAG)      Foundation Library (12/10/92)      E02AJF(3NAG)



         E02 -- Curve and Surface Fitting                          E02AJF
                 E02AJF -- NAG Foundation Library Routine Document

         Note: Before using this routine, please read the Users' Note for
         your implementation to check implementation-dependent details.
         The symbol (*) after a NAG routine name denotes a routine that is
```

not included in the Foundation Library.

1. Purpose

E02AJF determines the coefficients in the Chebyshev-series representation of the indefinite integral of a polynomial given in Chebyshev-series form.

2. Specification

```
       SUBROUTINE E02AJF (NP1, XMIN, XMAX, A, IA1, LA, QATM1,
      1                   AINT, IAINT1, LAINT, IFAIL)
       INTEGER          NP1, IA1, LA, IAINT1, LAINT, IFAIL
       DOUBLE PRECISION XMIN, XMAX, A(LA), QATM1, AINT(LAINT)
```

3. Description

This routine forms the polynomial which is the indefinite integral of a given polynomial. Both the original polynomial and its integral are represented in Chebyshev-series form. If supplied with the coefficients a_i, for i=0,1,...,n, of a polynomial p(x) of degree n, where

$$p(x)=\frac{1}{2}a_0+a_1T_1(x)+...+a_nT_n(x),$$

the routine returns the coefficients a'_i, for i=0,1,...,n+1, of the polynomial q(x) of degree n+1, where

$$q(x)=\frac{1}{2}a'_0+a'_1T_1(x)+...+a'_{n+1}T_{n+1}(x),$$

and

$$q(x)=\int p(x)dx.$$

Here $T_j(x)$ denotes the Chebyshev polynomial of the first kind of degree j with argument x. It is assumed that the normalised

variable x in the interval [-1,+1] was obtained from the user's original variable x in the interval $[x_{min}, x_{max}]$ by the linear transformation

$$\bar{x} = \frac{2x-(x_{max}+x_{min})}{x_{max}-x_{min}}$$

and that the user requires the integral to be with respect to the variable x. If the integral with respect to $\bar{x}$ is required, set $x_{max}=1$ and $x_{min}=-1$.

Values of the integral can subsequently be computed, from the coefficients obtained, by using E02AKF.

The method employed is that of Chebyshev-series [1] modified for integrating with respect to x. Initially taking $a_{n+1}=a_{n+2}=0$, the routine forms successively

$$a'_i = \frac{a_{i-1}-a_{i+1}}{2i} * \frac{x_{max}-x_{min}}{2}, \quad i=n+1,n,...,1.$$

The constant coefficient a'_0 is chosen so that q(x) is equal to a specified value, QATM1, at the lower end-point of the interval on which it is defined, i.e., $\bar{x}=-1$, which corresponds to $x=x_{min}$.

4. References

[1] Unknown (1961) Chebyshev-series. Modern Computing Methods, Chapter 8. NPL Notes on Applied Science (2nd Edition). 16 HMSO.

5. Parameters

1: NP1 -- INTEGER Input
On entry: n+1, where n is the degree of the given polynomial p(x). Thus NP1 is the number of coefficients in this polynomial. Constraint: NP1 >= 1.

2: XMIN -- DOUBLE PRECISION Input

3: XMAX -- DOUBLE PRECISION Input

On entry: the lower and upper end-points respectively of the interval $[x_{min},x_{max}]$. The Chebyshev-series

representation is in terms of the normalised variable x, where

$$x=\frac{2x-(x_{max}+x_{min})}{x_{max}-x_{min}}.$$

Constraint: XMAX > XMIN.

4: A(LA) -- DOUBLE PRECISION array Input

On entry: the Chebyshev coefficients of the polynomial p(x) . Specifically, element 1+i*IA1 of A must contain the coefficient a_i, for i=0,1,...,n. Only these n+1 elements will be accessed.

Unchanged on exit, but see AINT, below.

5: IA1 -- INTEGER Input

On entry: the index increment of A. Most frequently the Chebyshev coefficients are stored in adjacent elements of A, and IA1 must be set to 1. However, if for example, they are stored in A(1),A(4),A(7),..., then the value of IA1 must be 3. See also Section 8. Constraint: IA1 >= 1.

6: LA -- INTEGER Input

On entry:
the dimension of the array A as declared in the (sub)program from which E02AJF is called.
Constraint: LA>=1+(NP1-1)*IA1.

7: QATM1 -- DOUBLE PRECISION Input

On entry: the value that the integrated polynomial is required to have at the lower end-point of its interval of

definition, i.e., at x=-1 which corresponds to $x=x_{min}$. Thus, QATM1 is a constant of integration and will normally be set to zero by the user.

8: AINT(LAINT) -- DOUBLE PRECISION array Output
On exit: the Chebyshev coefficients of the integral q(x). (The integration is with respect to the variable x, and the constant coefficient is chosen so that $q(x_{min})$ equals QATM1) Specifically, element 1+i*IAINT1 of AINT contains the coefficient a'_i, for i=0,1,...,n+1. A call of the routine may have the array name AINT the same as A, provided that note is taken of the order in which elements are overwritten when choosing starting elements and increments IA1 and IAINT1: i.e., the coefficients, $a_0,a_1,...,a_{i-2}$ must be intact after coefficient a'_i is stored. In particular it is possible to overwrite the a_i entirely by having IA1 = IAINT1, and the actual array for A and AINT identical.

9: IAINT1 -- INTEGER Input
On entry: the index increment of AINT. Most frequently the Chebyshev coefficients are required in adjacent elements of AINT, and IAINT1 must be set to 1. However, if, for example, they are to be stored in AINT(1),AINT(4),AINT(7),..., then the value of IAINT1 must be 3. See also Section 8.
Constraint: IAINT1 >= 1.

10: LAINT -- INTEGER Input
On entry:
the dimension of the array AINT as declared in the (sub)program from which E02AJF is called.
Constraint: LAINT>=1+NP1*IAINT1.

11: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1
On entry NP1 < 1,

or XMAX <= XMIN,

```
     or       IA1 < 1,

     or       LA<=(NP1-1)*IA1,

     or       IAINT1 < 1,

     or       LAINT<=NP1*IAINT1.

7. Accuracy

In general there is a gain in precision in numerical integration,
in this case associated with the division by 2i in the formula
quoted in Section 3.

8. Further Comments

The time taken by the routine is approximately proportional to
n+1.

The increments IA1, IAINT1 are included as parameters to give a
degree of flexibility which, for example, allows a polynomial in
two variables to be integrated with respect to either variable
without rearranging the coefficients.

9. Example

Suppose a polynomial has been computed in Chebyshev-series form
to fit data over the interval [-0.5,2.5]. The example program
evaluates the integral of the polynomial from 0.0 to 2.0. (For
the purpose of this example, XMIN, XMAX and the Chebyshev
coefficients are simply supplied in DATA statements. Normally a
program would read in or generate data and compute the fitted
polynomial).

The example program is not reproduced here. The source code for
all example programs is distributed with the NAG Foundation
Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

———-

Evaluate polynomial in Chebyshev-series representation

— **nage.ht** —

```
\begin{page}{manpageXXe02akf}{NAG Documentation: e02akf}
\beginscroll
\begin{verbatim}



     E02AKF(3NAG)      Foundation Library (12/10/92)      E02AKF(3NAG)



          E02 -- Curve and Surface Fitting                           E02AKF
                  E02AKF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          E02AKF evaluates a polynomial from its Chebyshev-series
          representation, allowing an arbitrary index increment for
          accessing the array of coefficients.

          2. Specification

                 SUBROUTINE E02AKF (NP1, XMIN, XMAX, A, IA1, LA, X, RESULT,
                1                   IFAIL)
                 INTEGER          NP1, IA1, LA, IFAIL
                 DOUBLE PRECISION XMIN, XMAX, A(LA), X, RESULT

          3. Description

          If supplied with the coefficients a , for i=0,1,...,n, of a
                                             i


          polynomial p(x) of degree n, where

                                 1
                           p(x)= -a +a T (x)+...+a T (x),
                                 2 0  1 1         n n



          this routine returns the value of p(x) at a user-specified value


          of the variable x. Here T (x) denotes the Chebyshev polynomial of
                                   j
```

the first kind of degree j with argument x. It is assumed that

the independent variable x in the interval [-1,+1] was obtained from the user's original variable x in the interval $[x_{min},x_{max}]$ by the linear transformation

$$x=\frac{2x-(x_{max}+x_{min})}{x_{max}-x_{min}}.$$

The coefficients a_i may be supplied in the array A, with any increment between the indices of array elements which contain successive coefficients. This enables the routine to be used in surface fitting and other applications, in which the array might have two or more dimensions.

The method employed is based upon the three-term recurrence relation due to Clenshaw [1], with modifications due to Reinsch and Gentleman (see [4]). For further details of the algorithm and its use see Cox [2] and Cox and Hayes [3].

4. References

[1] Clenshaw C W (1955) A Note on the Summation of Chebyshev-series. Math. Tables Aids Comput. 9 118--120.

[2] Cox M G (1973) A data-fitting package for the non-specialist user. Report NAC40. National Physical Laboratory.

[3] Cox M G and Hayes J G (1973) Curve fitting: a guide and suite of algorithms for the non-specialist user. Report NAC26. National Physical Laboratory.

[4] Gentlemen W M (1969) An Error Analysis of Goertzel's (Watt's) Method for Computing Fourier Coefficients. Comput. J. 12 160--165.

5. Parameters

1: NP1 -- INTEGER Input

On entry: n+1, where n is the degree of the given polynomial p(x). Constraint: NP1 >= 1.

2: XMIN -- DOUBLE PRECISION Input

3: XMAX -- DOUBLE PRECISION Input
On entry: the lower and upper end-points respectively of the interval $[x_{min},x_{max}]$. The Chebyshev-series representation is in terms of the normalised variable x, where

$$\bar{x}=\frac{2x-(x_{max}+x_{min})}{x_{max}-x_{min}}.$$

Constraint: XMIN < XMAX.

4: A(LA) -- DOUBLE PRECISION array Input
On entry: the Chebyshev coefficients of the polynomial p(x). Specifically, element 1+i*IA1 must contain the coefficient a_i, for i=0,1,...,n. Only these n+1 elements will be accessed.

5: IA1 -- INTEGER Input
On entry: the index increment of A. Most frequently, the Chebyshev coefficients are stored in adjacent elements of A, and IA1 must be set to 1. However, if, for example, they are stored in A(1),A(4),A(7),..., then the value of IA1 must be 3. Constraint: IA1 >= 1.

6: LA -- INTEGER Input
On entry:
the dimension of the array A as declared in the (sub)program from which E02AKF is called.
Constraint: LA>=(NP1-1)*IA1+1.

7: X -- DOUBLE PRECISION Input
On entry: the argument x at which the polynomial is to be evaluated. Constraint: XMIN <= X <= XMAX.

8: RESULT -- DOUBLE PRECISION Output

On exit: the value of the polynomial p(x).

9: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not

familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1
On entry NP1 < 1,

or IA1 < 1,

or LA<=(NP1-1)*IA1,

or XMIN >= XMAX.

IFAIL= 2
X does not satisfy the restriction XMIN <= X <= XMAX.

7. Accuracy

The rounding errors are such that the computed value of the polynomial is exact for a slightly perturbed set of coefficients a_i+(delta)a_i. The ratio of the sum of the absolute values of the (delta)a_i to the sum of the absolute values of the a_i is less than a small multiple of (n+1)*machine precision.

8. Further Comments

The time taken by the routine is approximately proportional to n+1.

9. Example

Suppose a polynomial has been computed in Chebyshev-series form to fit data over the interval [-0.5,2.5]. The example program evaluates the polynomial at 4 equally spaced points over the interval. (For the purposes of this example, XMIN, XMAX and the Chebyshev coefficients are supplied in DATA statements. Normally a program would first read in or generate data and compute the fitted polynomial.)

The example program is not reproduced here. The source code for all example programs is distributed with the NAG Foundation Library software and should be available on-line.

```
\end{verbatim}
\endscroll
\end{page}
```

———-

Weighted least-squares aprrox to data points

— nage.ht —

```
\begin{page}{manpageXXe02baf}{NAG Documentation: e02baf}
\beginscroll
\begin{verbatim}



     E02BAF(3NAG)      Foundation Library (12/10/92)      E02BAF(3NAG)



          E02 -- Curve and Surface Fitting                           E02BAF
                  E02BAF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          E02BAF computes a weighted least-squares approximation to an
          arbitrary set of data points by a cubic spline with knots
          prescribed by the user. Cubic spline interpolation can also be
          carried out.

          2. Specification

                 SUBROUTINE E02BAF (M, NCAP7, X, Y, W, LAMDA, WORK1, WORK2,
                1                   C, SS, IFAIL)
                 INTEGER          M, NCAP7, IFAIL
                 DOUBLE PRECISION X(M), Y(M), W(M), LAMDA(NCAP7), WORK1(M),
                1                 WORK2(4*NCAP7), C(NCAP7), SS

          3. Description

          This routine determines a least-squares cubic spline
```

approximation $s(x)$ to the set of data points (x_r,y_r) with weights w_r, for $r=1,2,\ldots,m$. The value of NCAP7 = n+7, where n is the number of intervals of the spline (one greater than the number of interior knots), and the values of the knots $\lambda_5,\lambda_6,\ldots,\lambda_{n+3}$, interior to the data interval, are prescribed by the user.

$s(x)$ has the property that it minimizes θ, the sum of squares of the weighted residuals ϵ_r, for $r=1,2,\ldots,m$, where

$$\epsilon_r=w_r(y_r-s(x_r)).$$

The routine produces this minimizing value of θ and the coefficients $c_1,c_2,\ldots,c_q$, where $q=n+3$, in the B-spline representation

$$s(x)=\sum_{i=1}^{q} c_i N_i(x).$$

Here $N_i(x)$ denotes the normalised B-spline of degree 3 defined upon the knots $\lambda_i,\lambda_{i+1},\ldots,\lambda_{i+4}$.

In order to define the full set of B-splines required, eight additional knots $\lambda_1,\lambda_2,\lambda_3,\lambda_4$ and $\lambda_{n+4},\lambda_{n+5},\lambda_{n+6},\lambda_{n+7}$ are inserted automatically by the routine. The first four of these are set equal to the smallest x_r and the last four to the largest x_r.

The representation of $s(x)$ in terms of B-splines is the most compact form possible in that only n+3 coefficients, in addition to the n+7 knots, fully define s(x).

The method employed involves forming and then computing the least-squares solution of a set of m linear equations in the coefficients c_i (i=1,2,...,n+3). The equations are formed using a recurrence relation for B-splines that is unconditionally stable (Cox [1], de Boor [5]), even for multiple (coincident) knots. The least-squares solution is also obtained in a stable manner by using orthogonal transformations, viz. a variant of Givens rotations (Gentleman [6] and [7]). This requires only one equation to be stored at a time. Full advantage is taken of the structure of the equations, there being at most four non-zero values of $N_i(x)$ for any value of x and hence at most four coefficients in each equation.

For further details of the algorithm and its use see Cox [2], [3] and [4].

Subsequent evaluation of s(x) from its B-spline representation may be carried out using E02BBF. If derivatives of s(x) are also required, E02BCF may be used. E02BDF can be used to compute the definite integral of s(x).

4. References

[1] Cox M G (1972) The Numerical Evaluation of B-splines. J. Inst. Math. Appl. 10 134--149.

[2] Cox M G (1974) A Data-fitting Package for the Non-specialist User. Software for Numerical Mathematics. (ed D J Evans) Academic Press.

[3] Cox M G (1975) Numerical methods for the interpolation and approximation of data by spline functions. PhD Thesis. City University, London.

[4] Cox M G and Hayes J G (1973) Curve fitting: a guide and suite of algorithms for the non-specialist user. Report NAC26. National Physical Laboratory.

[5] De Boor C (1972) On Calculating with B-splines. J. Approx. Theory. 6 50--62.

[6] Gentleman W M (1974) Algorithm AS 75. Basic Procedures for

Large Sparse or Weighted Linear Least-squares Problems. Appl. Statist. 23 448--454.

[7] Gentleman W M (1973) Least-squares Computations by Givens Transformations without Square Roots. J. Inst. Math. Applic. 12 329--336.

[8] Schoenberg I J and Whitney A (1953) On Polya Frequency Functions III. Trans. Amer. Math. Soc. 74 246--259.

5. Parameters

1: M -- INTEGER Input

On entry: the number m of data points. Constraint: M >= MDIST >= 4, where MDIST is the number of distinct x values in the data.

2: NCAP7 -- INTEGER Input

On entry: n+7, where n is the number of intervals of the spline (which is one greater than the number of interior knots, i.e., the knots strictly within the range x_1 to x_m) over which the spline is defined. Constraint: 8 <= NCAP7 <= MDIST + 4, where MDIST is the number of distinct x values in the data.

3: X(M) -- DOUBLE PRECISION array Input

On entry: the values x_r of the independent variable (abscissa), for r=1,2,...,m. Constraint: $x_1 <= x_2 <= ... <= x_m$.

4: Y(M) -- DOUBLE PRECISION array Input

On entry: the values y_r of the dependent variable (ordinate), for r=1,2,...,m.

5: W(M) -- DOUBLE PRECISION array Input

On entry: the values w_r of the weights, for r=1,2,...,m. For advice on the choice of weights, see the Chapter Introduction. Constraint: W(r) > 0, for r=1,2,...,m.

6: LAMDA(NCAP7) -- DOUBLE PRECISION array Input/Output

On entry: LAMDA(i) must be set to the (i-4)th (interior) knot, $(lambda)_i$, for i=5,6,...,n+3. Constraint: X(1) < LAMDA

(5) <= LAMDA(6) <=... <= LAMDA(NCAP7-4) < X(M). On exit: the input values are unchanged, and LAMDA(i), for i = 1, 2, 3, 4, NCAP7-3, NCAP7-2, NCAP7-1, NCAP7 contains the additional (exterior) knots introduced by the routine. For advice on the choice of knots, see Section 3.3 of the Chapter Introduction.

7: WORK1(M) -- DOUBLE PRECISION array Workspace

8: WORK2(4*NCAP7) -- DOUBLE PRECISION array Workspace

9: C(NCAP7) -- DOUBLE PRECISION array Output
On exit: the coefficient c_i of the B-spline $N_i(x)$, for i=1,2,...,n+3. The remaining elements of the array are not used.

10: SS -- DOUBLE PRECISION Output
On exit: the residual sum of squares, (theta).

11: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1
The knots fail to satisfy the condition

X(1) < LAMDA(5) <= LAMDA(6) <=... <= LAMDA(NCAP7-4) < X(M).
Thus the knots are not in correct order or are not interior to the data interval.

IFAIL= 2
The weights are not all strictly positive.

IFAIL= 3
The values of X(r), for r = 1,2,...,M are not in non-decreasing order.

IFAIL= 4
NCAP7 < 8 (so the number of interior knots is negative) or

NCAP7 > MDIST + 4, where MDIST is the number of distinct x values in the data (so there cannot be a unique solution).

IFAIL= 5

The conditions specified by Schoenberg and Whitney [8] fail to hold for at least one subset of the distinct data abscissae. That is, there is no subset of NCAP7-4 strictly increasing values, X(R(1)),X(R(2)),...,X(R(NCAP7-4)), among the abscissae such that

X(R(1)) < LAMDA(1) < X(R(5)),

X(R(2)) < LAMDA(2) < X(R(6)),

...

X(R(NCAP7-8)) < LAMDA(NCAP7-8) < X(R(NCAP7-4)).

This means that there is no unique solution: there are regions containing too many knots compared with the number of data points.

7. Accuracy

The rounding errors committed are such that the computed coefficients are exact for a slightly perturbed set of ordinates y_r+(delta)y_r. The ratio of the root-mean-square value for the (delta)y_r to the root-mean-square value of the y_r can be expected to be less than a small multiple of (kappa)*m*machine precision, where (kappa) is a condition number for the problem. Values of (kappa) for 20-30 practical data sets all proved to lie between 4.5 and 7.8 (see Cox [3]). (Note that for these data sets, replacing the coincident end knots at the end-points x_1 and x_m used in the routine by various choices of non-coincident exterior knots gave values of (kappa) between 16 and 180. Again see Cox [3] for further details.) In general we would not expect (kappa) to be large unless the choice of knots results in near-violation of the Schoenberg-Whitney conditions.

A cubic spline which adequately fits the data and is free from spurious oscillations is more likely to be obtained if the knots are chosen to be grouped more closely in regions where the function (underlying the data) or its derivatives change more rapidly than elsewhere.

8. Further Comments

The time taken by the routine is approximately C*(2m+n+7) seconds, where C is a machine-dependent constant.

Multiple knots are permitted as long as their multiplicity does not exceed 4, i.e., the complete set of knots must satisfy

$(\lambda)_i < (\lambda)_{i+4}$, for i=1,2,...,n+3, (cf. Section 6). At a knot of multiplicity one (the usual case), s(x) and its first two derivatives are continuous. At a knot of multiplicity two, s(x) and its first derivative are continuous. At a knot of multiplicity three, s(x) is continuous, and at a knot of multiplicity four, s(x) is generally discontinous.

The routine can be used efficiently for cubic spline

interpolation, i.e.,if m=n+3. The abscissae must then of course satisfy $x_1 < x_2 < ... < x_m$. Recommended values for the knots in this

case are $(\lambda)_i = x_{i-2}$, for i=5,6,...,n+3.

9. Example

Determine a weighted least-squares cubic spline approximation with five intervals (four interior knots) to a set of 14 given data points. Tabulate the data and the corresponding values of the approximating spline, together with the residual errors, and also the values of the approximating spline at points half-way between each pair of adjacent data points.

The example program is written in a general form that will enable

a cubic spline approximation with n intervals (n-1 interior knots) to be obtained to m data points, with arbitrary positive weights, and the approximation to be tabulated. Note that E02BBF is used to evaluate the approximating spline. The program is self-starting in that any number of data sets can be supplied.

The example program is not reproduced here. The source code for all example programs is distributed with the NAG Foundation Library software and should be available on-line.

```
\end{verbatim}
\endscroll
```

```
\end{page}
```

———-

Evaluates a cubic spline from its B-spline representation

— nage.ht —

```
\begin{page}{manpageXXe02bbf}{NAG Documentation: e02bbf}
\beginscroll
\begin{verbatim}



     E02BBF(3NAG)      Foundation Library (12/10/92)      E02BBF(3NAG)



          E02 -- Curve and Surface Fitting                           E02BBF
                  E02BBF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          E02BBF evaluates a cubic spline from its B-spline representation.

          2. Specification

                 SUBROUTINE E02BBF (NCAP7, LAMDA, C, X, S, IFAIL)
                 INTEGER          NCAP7, IFAIL
                 DOUBLE PRECISION LAMDA(NCAP7), C(NCAP7), X, S

          3. Description

          This routine evaluates the cubic spline s(x) at a prescribed
          argument x from its augmented knot set (lambda) , for
                                                         i


          i=1,2,...,n+7, (see E02BAF) and from the coefficients c , for
                                                                 i
          i=1,2,...,q in its B-spline representation
```

$$s(x)=\sum_{i=1}^{q} c_i N_i(x)$$

Here q=n+3, where n is the number of intervals of the spline, and $N_i(x)$ denotes the normalised B-spline of degree 3 defined upon the knots $\lambda_i, \lambda_{i+1}, \ldots, \lambda_{i+4}$. The prescribed argument x must satisfy $\lambda_4 \le x \le \lambda_{n+4}$.

It is assumed that $\lambda_j \ge \lambda_{j-1}$, for j=2,3,...,n+7, and $\lambda_{n+4} > \lambda_4$.

The method employed is that of evaluation by taking convex combinations due to de Boor [4]. For further details of the algorithm and its use see Cox [1] and [3].

It is expected that a common use of E02BBF will be the evaluation of the cubic spline approximations produced by E02BAF. A generalization of E02BBF which also forms the derivative of s(x) is E02BCF. E02BCF takes about 50% longer than E02BBF.

4. References

[1] Cox M G (1972) The Numerical Evaluation of B-splines. J. Inst. Math. Appl. 10 134--149.

[2] Cox M G (1978) The Numerical Evaluation of a Spline from its B-spline Representation. J. Inst. Math. Appl. 21 135--143.

[3] Cox M G and Hayes J G (1973) Curve fitting: a guide and suite of algorithms for the non-specialist user. Report NAC26. National Physical Laboratory.

[4] De Boor C (1972) On Calculating with B-splines. J. Approx. Theory. 6 50--62.

5. Parameters

1: NCAP7 -- INTEGER Input

On entry: n+7, where n is the number of intervals (one greater than the number of interior knots, i.e., the knots strictly within the range $(\lambda)_4$ to $(\lambda)_{n+4}$) over which the spline is defined. Constraint: NCAP7 >= 8.

2: LAMDA(NCAP7) -- DOUBLE PRECISION array Input

On entry: LAMDA(j) must be set to the value of the jth member of the complete set of knots, $(\lambda)_j$ for j=1,2,...,n+7. Constraint: the LAMDA(j) must be in non-decreasing order with LAMDA(NCAP7-3) > LAMDA(4).

3: C(NCAP7) -- DOUBLE PRECISION array Input

On entry: the coefficient c_i of the B-spline $N_i(x)$, for i=1,2,...,n+3. The remaining elements of the array are not used.

4: X -- DOUBLE PRECISION Input

On entry: the argument x at which the cubic spline is to be evaluated. Constraint: LAMDA(4) <= X <= LAMDA(NCAP7-3).

5: S -- DOUBLE PRECISION Output

On exit: the value of the spline, s(x).

6: IFAIL -- INTEGER Input/Output

On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1

The argument X does not satisfy LAMDA(4) <= X <= LAMDA(NCAP7-3).

In this case the value of S is set arbitrarily to zero.

IFAIL= 2
NCAP7 < 8, i.e., the number of interior knots is negative.

7. Accuracy

The computed value of s(x) has negligible error in most practical situations. Specifically, this value has an absolute error bounded in modulus by $18*c_{max}$ * machine precision, where c_{max} is the largest in modulus of c_j, c_{j+1}, c_{j+2} and c_{j+3}, and j is an integer such that $(lambda)_{j+3}$ <=x<=$(lambda)_{j+4}$. If c_j, c_{j+1}, c_{j+2} and c_{j+3} are all of the same sign, then the computed value of s(x) has a relative error not exceeding 20*machine precision in modulus. For further details see Cox [2].

8. Further Comments

The time taken by the routine is approximately C*(1+0.1*log(n+7)) seconds, where C is a machine-dependent constant.

Note: the routine does not test all the conditions on the knots given in the description of LAMDA in Section 5, since to do this would result in a computation time approximately linear in n+7 instead of log(n+7). All the conditions are tested in E02BAF, however.

9. Example

Evaluate at 9 equally-spaced points in the interval 1.0<=x<=9.0 the cubic spline with (augmented) knots 1.0, 1.0, 1.0, 1.0, 3.0, 6.0, 8.0, 9.0, 9.0, 9.0, 9.0 and normalised cubic B-spline coefficients 1.0, 2.0, 4.0, 7.0, 6.0, 4.0, 3.0.

The example program is written in a general form that will enable

a cubic spline with n intervals, in its normalised cubic B-spline form, to be evaluated at m equally-spaced points in the interval

LAMDA(4) <= x <= LAMDA(n+4). The program is self-starting in that any number of data sets may be supplied.

```
        The example program is not reproduced here. The source code for
        all example programs is distributed with the NAG Foundation
        Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

———-

Evaluate cubic spline and 3 derivatives from B-spline

— **nage.ht** —

```
\begin{page}{manpageXXe02bcf}{NAG Documentation: e02bcf}
\beginscroll
\begin{verbatim}



     E02BCF(3NAG)      Foundation Library (12/10/92)      E02BCF(3NAG)



          E02 -- Curve and Surface Fitting                          E02BCF
                  E02BCF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          E02BCF evaluates a cubic spline and its first three derivatives
          from its B-spline representation.

          2. Specification

                 SUBROUTINE E02BCF (NCAP7, LAMDA, C, X, LEFT, S, IFAIL)
                 INTEGER          NCAP7, LEFT, IFAIL
                 DOUBLE PRECISION LAMDA(NCAP7), C(NCAP7), X, S(4)

          3. Description

          This routine evaluates the cubic spline s(x) and its first three
          derivatives at a prescribed argument x. It is assumed that s(x)
```

is represented in terms of its B-spline coefficients c_i, for $i=1,2,...,n+3$ and (augmented) ordered knot set λ_i, for $i=1,2,...,n+7$, (see E02BAF), i.e.,

$$s(x)=\sum_{i=1}^{q} c_i N_i(x)$$

Here $q=n+3$, n is the number of intervals of the spline and $N_i(x)$ denotes the normalised B-spline of degree 3 (order 4) defined upon the knots $\lambda_i,\lambda_{i+1},...,\lambda_{i+4}$. The prescribed argument x must satisfy

$$\lambda_4 \le x \le \lambda_{n+4}$$

At a simple knot λ_i (i.e., one satisfying $\lambda_{i-1} < \lambda_i < \lambda_{i+1}$), the third derivative of the spline is in general discontinuous. At a multiple knot (i.e., two or more knots with the same value), lower derivatives, and even the spline itself, may be discontinuous. Specifically, at a point $x=u$ where (exactly) r knots coincide (such a point is termed a knot of multiplicity r), the values of the derivatives of order $4-j$, for $j=1,2,...,r$, are in general discontinuous. (Here $1\le r\le 4$; $r>4$ is not meaningful.) The user must specify whether the value at such a point is required to be the left- or right-hand derivative.

The method employed is based upon:

(i) carrying out a binary search for the knot interval containing the argument x (see Cox [3]),

(ii) evaluating the non-zero B-splines of orders 1,2,3 and 4 by recurrence (see Cox [2] and [3]),

(iii) computing all derivatives of the B-splines of order 4 by applying a second recurrence to these computed B-spline values (see de Boor [1]),

(iv) multiplying the 4th-order B-spline values and their derivative by the appropriate B-spline coefficients, and summing, to yield the values of s(x) and its derivatives.

E02BCF can be used to compute the values and derivatives of cubic spline fits and interpolants produced by E02BAF.

If only values and not derivatives are required, E02BBF may be used instead of E02BCF, which takes about 50% longer than E02BBF.

4. References

[1] De Boor C (1972) On Calculating with B-splines. J. Approx. Theory. 6 50--62.

[2] Cox M G (1972) The Numerical Evaluation of B-splines. J. Inst. Math. Appl. 10 134--149.

[3] Cox M G (1978) The Numerical Evaluation of a Spline from its B-spline Representation. J. Inst. Math. Appl. 21 135--143.

5. Parameters

1: NCAP7 -- INTEGER Input

On entry: n+7, where n is the number of intervals of the spline (which is one greater than the number of interior knots, i.e., the knots strictly within the range $(\lambda)_4$ to $(\lambda)_{n+4}$ over which the spline is defined).
Constraint: NCAP7 >= 8.

2: LAMDA(NCAP7) -- DOUBLE PRECISION array Input
On entry: LAMDA(j) must be set to the value of the jth member of the complete set of knots, $(\lambda)_j$, for j=1,2,...,n+7. Constraint: the LAMDA(j) must be in non-decreasing order with

LAMDA(NCAP7-3) > LAMDA(4).

3: C(NCAP7) -- DOUBLE PRECISION array Input

On entry: the coefficient c_i of the B-spline $N_i(x)$, for i=1,2,...,n+3. The remaining elements of the array are not used.

4: X -- DOUBLE PRECISION Input
On entry: the argument x at which the cubic spline and its derivatives are to be evaluated. Constraint: LAMDA(4) <= X <= LAMDA(NCAP7-3).

5: LEFT -- INTEGER Input
On entry: specifies whether left- or right-hand values of the spline and its derivatives are to be computed (see Section 3). Left- or right-hand values are formed according to whether LEFT is equal or not equal to 1. If x does not coincide with a knot, the value of LEFT is immaterial. If x = LAMDA(4), right-hand values are computed, and if x = LAMDA (NCAP7-3), left-hand values are formed, regardless of the value of LEFT.

6: S(4) -- DOUBLE PRECISION array Output
On exit: S(j) contains the value of the (j-1)th derivative of the spline at the argument x, for j = 1,2,3,4. Note that S(1) contains the value of the spline.

7: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1
NCAP7 < 8, i.e., the number of intervals is not positive.

IFAIL= 2
Either LAMDA(4) >= LAMDA(NCAP7-3), i.e., the range over which s(x) is defined is null or negative in length, or X is an invalid argument, i.e., X < LAMDA(4) or X > LAMDA(NCAP7-3).

7. Accuracy

The computed value of s(x) has negligible error in most practical situations. Specifically, this value has an absolute error bounded in modulus by $18*c_{max}$ * machine precision, where c_{max} is the largest in modulus of c_j, c_{j+1}, c_{j+2} and c_{j+3}, and j is an integer such that $(lambda)_{j+3} <= x <= (lambda)_{j+4}$. If c_j, c_{j+1}, c_{j+2} and c_{j+3} are all of the same sign, then the computed value of s(x) has relative error bounded by 18*machine precision. For full details see Cox [3].

No complete error analysis is available for the computation of the derivatives of s(x). However, for most practical purposes the absolute errors in the computed derivatives should be small.

8. Further Comments

The time taken by this routine is approximately linear in

log(n+7).

Note: the routine does not test all the conditions on the knots given in the description of LAMDA in Section 5, since to do this

would result in a computation time approximately linear in n+7

instead of log(n+7). All the conditions are tested in E02BAF, however.

9. Example

Compute, at the 7 arguments x = 0, 1, 2, 3, 4, 5, 6, the left- and right-hand values and first 3 derivatives of the cubic spline defined over the interval 0<=x<=6 having the 6 interior knots x = 1, 3, 3, 3, 4, 4, the 8 additional knots 0, 0, 0, 0, 6, 6, 6, 6, and the 10 B-spline coefficients 10, 12, 13, 15, 22, 26, 24, 18, 14, 12.

The input data items (using the notation of Section 5) comprise the following values in the order indicated:

```
n          m

LAMDA(j),   for j= 1,2,...,NCAP7
```

```
    C(j),       for j= 1,2,...,NCAP7-4

    x(i),       for i=1,2,...,m

The example program is written in a general form that will enable
the values and derivatives of a cubic spline having an arbitrary
number of knots to be evaluated at a set of arbitrary points. Any
number of data sets may be supplied. The only changes required to
the program relate to the dimensions of the arrays LAMDA and C.

The example program is not reproduced here. The source code for
all example programs is distributed with the NAG Foundation
Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Definite integral of cubic spline from B-spline

— nage.ht —

```
\begin{page}{manpageXXe02bdf}{NAG Documentation: e02bdf}
\beginscroll
\begin{verbatim}



     E02BDF(3NAG)      Foundation Library (12/10/92)      E02BDF(3NAG)



          E02 -- Curve and Surface Fitting                        E02BDF
                  E02BDF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          E02BDF computes the definite integral of a cubic spline from its
          B-spline representation.
```

2. Specification

```
SUBROUTINE E02BDF (NCAP7, LAMDA, C, DEFINT, IFAIL)
INTEGER          NCAP7, IFAIL
DOUBLE PRECISION LAMDA(NCAP7), C(NCAP7), DEFINT
```

3. Description

This routine computes the definite integral of the cubic spline $s(x)$ between the limits $x=a$ and $x=b$, where a and b are respectively the lower and upper limits of the range over which $s(x)$ is defined. It is assumed that $s(x)$ is represented in terms of its B-spline coefficients c_i, for $i=1,2,\ldots,n+3$ and (augmented) ordered knot set $(\lambda)_i$, for $i=1,2,\ldots,n+7$, with $(\lambda)_i=a$, for $i = 1,2,3,4$ and $(\lambda)_i=b$, for $i=n+4,n+5,n+6,n+7$, (see E02BAF), i.e.,

$$s(x)=\sum_{i=1}^{q} c_i N_i(x).$$

Here $q=n+3$, n is the number of intervals of the spline and $N_i(x)$ denotes the normalised B-spline of degree 3 (order 4) defined upon the knots $(\lambda)_i,(\lambda)_{i+1},\ldots,(\lambda)_{i+4}$.

The method employed uses the formula given in Section 3 of Cox [1].

E02BDF can be used to determine the definite integrals of cubic spline fits and interpolants produced by E02BAF.

4. References

[1] Cox M G (1975) An Algorithm for Spline Interpolation. J.

Inst. Math. Appl. 15 95--108.

5. Parameters

1: NCAP7 -- INTEGER Input

On entry: n+7, where n is the number of intervals of the spline (which is one greater than the number of interior knots, i.e., the knots strictly within the range a to b) over which the spline is defined. Constraint: NCAP7 >= 8.

2: LAMDA(NCAP7) -- DOUBLE PRECISION array Input

On entry: LAMDA(j) must be set to the value of the jth member of the complete set of knots, $(\lambda)_j$ for j=1,2,...,n+7. Constraint: the LAMDA(j) must be in non-decreasing order with LAMDA(NCAP7-3) > LAMDA(4) and satisfy

LAMDA(1)=LAMDA(2)=LAMDA(3)=LAMDA(4)

and

LAMDA(NCAP7-3)=LAMDA(NCAP7-2)=LAMDA(NCAP7-1)=LAMDA(NCAP7).

3: C(NCAP7) -- DOUBLE PRECISION array Input

On entry: the coefficient c_i of the B-spline $N_i(x)$, for i=1,2,...,n+3. The remaining elements of the array are not used.

4: DEFINT -- DOUBLE PRECISION Output

On exit: the value of the definite integral of s(x) between the limits x=a and x=b, where $a=(\lambda)_4$ and $b=(\lambda)_{n+4}$.

5: IFAIL -- INTEGER Input/Output

On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL= 1
NCAP7 < 8, i.e., the number of intervals is not positive.

IFAIL= 2
At least one of the following restrictions on the knots is violated:

LAMDA(NCAP7-3) > LAMDA(4),

LAMDA(j) >= LAMDA(j-1),

for j = 2,3,...,NCAP7, with equality in the cases j=2,3,4,NCAP7-2,NCAP7-1, and NCAP7.

7. Accuracy

The rounding errors are such that the computed value of the integral is exact for a slightly perturbed set of B-spline coefficients c_i differing in a relative sense from those supplied by no more than 2.2*(n+3)*machine precision.

8. Further Comments

The time taken by the routine is approximately proportional to

n+7.

9. Example

Determine the definite integral over the interval 0<=x<=6 of a cubic spline having 6 interior knots at the positions (lambda)=1, 3, 3, 3, 4, 4, the 8 additional knots 0, 0, 0, 0, 6, 6, 6, 6, and the 10 B-spline coefficients 10, 12, 13, 15, 22, 26, 24, 18, 14, 12.

The input data items (using the notation of Section 5) comprise the following values in the order indicated:

n

LAMDA(j) for j = 1,2,...,NCAP7
,

C(j),   for j = 1,2,...,NCAP7-3

```
         The example program is written in a general form that will enable
         the definite integral of a cubic spline having an arbitrary
         number of knots to be computed. Any number of data sets may be
         supplied. The only changes required to the program relate to the
         dimensions of the arrays LAMDA and C.

         The example program is not reproduced here. The source code for
         all example programs is distributed with the NAG Foundation
         Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

———-

Cubic spline approximation to an arbitrary set points

— nage.ht —

```
\begin{page}{manpageXXe02bef}{NAG Documentation: e02bef}
\beginscroll
\begin{verbatim}



     E02BEF(3NAG)      Foundation Library (12/10/92)      E02BEF(3NAG)



         E02 -- Curve and Surface Fitting                          E02BEF
                 E02BEF -- NAG Foundation Library Routine Document

         Note: Before using this routine, please read the Users' Note for
         your implementation to check implementation-dependent details.
         The symbol (*) after a NAG routine name denotes a routine that is
         not included in the Foundation Library.

         1. Purpose

         E02BEF computes a cubic spline approximation to an arbitrary set
         of data points. The knots of the spline are located
         automatically, but a single parameter must be specified to
         control the trade-off between closeness of fit and smoothness of
         fit.

         2. Specification
```

```
      SUBROUTINE E02BEF (START, M, X, Y, W, S, NEST, N, LAMDA,
     1                   C, FP, WRK, LWRK, IWRK, IFAIL)
      INTEGER          M, NEST, N, LWRK, IWRK(NEST), IFAIL
      DOUBLE PRECISION X(M), Y(M), W(M), S, LAMDA(NEST), C(NEST),
     1                 FP, WRK(LWRK)
      CHARACTER*1      START
```

3. Description

This routine determines a smooth cubic spline approximation $s(x)$ to the set of data points (x_r,y_r), with weights w_r, for $r=1,2,...,m$.

The spline is given in the B-spline representation

$$s(x)=\sum_{i=1}^{n-4} c_i N_i(x) \qquad (1)$$

where $N_i(x)$ denotes the normalised cubic B-spline defined upon the knots $\lambda_i,\lambda_{i+1},...,\lambda_{i+4}$.

The total number n of these knots and their values $\lambda_1,...,\lambda_n$ are chosen automatically by the routine. The knots $\lambda_5,...,\lambda_{n-4}$ are the interior knots; they divide the approximation interval $[x_1,x_m]$ into $n-7$ sub-intervals. The coefficients $c_1,c_2,...,c_{n-4}$ are then determined as the solution of the following constrained minimization problem:

minimize

$$\eta=\sum_{i=5}^{n-4} \delta_i^2 \qquad (2)$$

subject to the constraint

m

```
                    --          2
         (theta)= >  (epsilon) <=S                   (3)
                    --          r
                    r=1

where: (delta)    stands for the discontinuity jump in the third
              i   order derivative of s(x) at the interior knot
            (lambda) ,
                        i

       (epsilon)  denotes the weighted residual w (y -s(x )),
               r                                r  r    r

and    S          is a non-negative number to be specified by
                  the user.
```

The quantity (eta) can be seen as a measure of the (lack of) smoothness of s(x), while closeness of fit is measured through (theta). By means of the parameter S, 'the smoothing factor', the user will then control the balance between these two (usually conflicting) properties. If S is too large, the spline will be too smooth and signal will be lost (underfit); if S is too small, the spline will pick up too much noise (overfit). In the extreme cases the routine will return an interpolating spline ((theta)=0) if S is set to zero, and the weighted least-squares cubic polynomial ((eta)=0) if S is set very large. Experimenting with S values between these two extremes should result in a good compromise. (See Section 8.2 for advice on choice of S.)

The method employed is outlined in Section 8.3 and fully described in Dierckx [1], [2] and [3]. It involves an adaptive strategy for locating the knots of the cubic spline (depending on the function underlying the data and on the value of S), and an iterative method for solving the constrained minimization problem once the knots have been determined.

Values of the computed spline, or of its derivatives or definite integral, can subsequently be computed by calling E02BBF, E02BCF or E02BDF, as described in Section 8.4.

4. References

[1] Dierckx P (1975) An Algorithm for Smoothing, Differentiating and Integration of Experimental Data Using Spline Functions. J. Comput. Appl. Math. 1 165--184.

[2] Dierckx P (1982) A Fast Algorithm for Smoothing Data on a Rectangular Grid while using Spline Functions. SIAM J. Numer. Anal. 19 1286--1304.

[3] Dierckx P (1981) An Improved Algorithm for Curve Fitting with Spline Functions. Report TW54. Department of Computer Science, Katholieke Universiteit Leuven.

[4] Reinsch C H (1967) Smoothing by Spline Functions. Num. Math. 10 177--183.

5. Parameters

1: START -- CHARACTER*1 Input

On entry: START must be set to 'C' or 'W'.

If START = 'C' (Cold start), the routine will build up the knot set starting with no interior knots. No values need be assigned to the parameters N, LAMDA, WRK or IWRK.

If START = 'W' (Warm start), the routine will restart the knot-placing strategy using the knots found in a previous call of the routine. In this case, the parameters N, LAMDA, WRK, and IWRK must be unchanged from that previous call. This warm start can save much time in searching for a satisfactory value of S. Constraint: START = 'C' or 'W'.

2: M -- INTEGER Input

On entry: m, the number of data points. Constraint: M >= 4.

3: X(M) -- DOUBLE PRECISION array Input

On entry: the values x_r of the independent variable (abscissa) x, for r=1,2,...,m. Constraint: $x_1 < x_2 < ... < x_m$

4: Y(M) -- DOUBLE PRECISION array Input

On entry: the values y_r of the dependent variable (ordinate) y, for r=1,2,...,m.

5: W(M) -- DOUBLE PRECISION array Input

On entry: the values w_r of the weights, for r=1,2,...,m. For advice on the choice of weights, see the Chapter Introduction, Section 2.1.2. Constraint: W(r) > 0, for r=1,2,...,m.

6: S -- DOUBLE PRECISION Input

On entry: the smoothing factor, S.

If S=0.0, the routine returns an interpolating spline.

If S is smaller than machine precision, it is assumed equal to zero.

For advice on the choice of S, see Section 3 and Section 8.2
Constraint: S >= 0.0.

7: NEST -- INTEGER Input
On entry: an over-estimate for the number, n, of knots required. Constraint: NEST >= 8. In most practical situations, NEST = M/2 is sufficient. NEST never needs to be larger than M + 4, the number of knots needed for interpolation (S = 0.0).

8: N -- INTEGER Input/Output
On entry: if the warm start option is used, the value of N must be left unchanged from the previous call. On exit: the total number, n, of knots of the computed spline.

9: LAMDA(NEST) -- DOUBLE PRECISION array Input/Output
On entry: if the warm start option is used, the values LAMDA(1), LAMDA(2),...,LAMDA(N) must be left unchanged from the previous call. On exit: the knots of the spline i.e., the positions of the interior knots LAMDA(5), LAMDA(6),... ,LAMDA(N-4) as well as the positions of the additional knots LAMDA(1) = LAMDA(2) = LAMDA(3) = LAMDA(4) = x_1 and

LAMDA(N-3) = LAMDA(N-2) = LAMDA(N-1) = LAMDA(N) = x_m needed for the B-spline representation.

10: C(NEST) -- DOUBLE PRECISION array Output
On exit: the coefficient c_i of the B-spline $N_i(x)$ in the spline approximation s(x), for i=1,2,...,n-4.

11: FP -- DOUBLE PRECISION Output
On exit: the sum of the squared weighted residuals, (theta), of the computed spline approximation. If FP = 0.0, this is an interpolating spline. FP should equal S within a relative tolerance of 0.001 unless n=8 when the spline has no interior knots and so is simply a cubic polynomial. For knots to be inserted, S must be set to a value below the value of FP produced in this case.

12: WRK(LWRK) -- DOUBLE PRECISION array Workspace
On entry: if the warm start option is used, the values WRK (1),...,WRK(n) must be left unchanged from the previous call.

13: LWRK -- INTEGER Input

```
     On entry:
     the dimension of the array WRK as declared in the
     (sub)program from which E02BEF is called.
     Constraint: LWRK>=4*M+16*NEST+41.

14:  IWRK(NEST) -- INTEGER array                        Workspace
     On entry: if the warm start option is used, the values IWRK
     (1), ..., IWRK(n) must be left unchanged from the previous
     call.

     This array is used as workspace.

15:  IFAIL -- INTEGER                                Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

IFAIL= 1
     On entry START /= 'C' or 'W',

     or       M < 4,

     or       S < 0.0,

     or       S = 0.0 and NEST < M + 4,

     or       NEST < 8,

     or       LWRK<4*M+16*NEST+41.

IFAIL= 2
     The weights are not all strictly positive.

IFAIL= 3
     The values of X(r), for r=1,2,...,M, are not in strictly
     increasing order.

IFAIL= 4
     The number of knots required is greater than NEST. Try
     increasing NEST and, if necessary, supplying larger arrays
```

for the parameters LAMDA, C, WRK and IWRK. However, if NEST is already large, say NEST > M/2, then this error exit may indicate that S is too small.

IFAIL= 5

The iterative process used to compute the coefficients of the approximating spline has failed to converge. This error exit may occur if S has been set very small. If the error persists with increased S, consult NAG.

If IFAIL = 4 or 5, a spline approximation is returned, but it fails to satisfy the fitting criterion (see (2) and (3) in Section 3) - perhaps by only a small amount, however.

7. Accuracy

On successful exit, the approximation returned is such that its weighted sum of squared residuals FP is equal to the smoothing factor S, up to a specified relative tolerance of 0.001 - except that if n=8, FP may be significantly less than S: in this case the computed spline is simply a weighted least-squares polynomial approximation of degree 3, i.e., a spline with no interior knots.

8. Further Comments

8.1. Timing

The time taken for a call of E02BEF depends on the complexity of the shape of the data, the value of the smoothing factor S, and the number of data points. If E02BEF is to be called for different values of S, much time can be saved by setting START =

8.2. Choice of S

If the weights have been correctly chosen (see Section 2.1.2 of the Chapter Introduction), the standard deviation of $w_r y_r$ would be the same for all r, equal to (sigma), say. In this case, choosing the smoothing factor S in the range $(\sigma)^2 (m\pm\sqrt{2m})$, as suggested by Reinsch [4], is likely to give a good start in the search for a satisfactory value. Otherwise, experimenting with different values of S will be required from the start, taking account of the remarks in Section 3.

In that case, in view of computation time and memory requirements, it is recommended to start with a very large value for S and so determine the least-squares cubic polynomial; the value returned for FP, call it FP_0, gives an upper bound for S.

Then progressively decrease the value of S to obtain closer fits - say by a factor of 10 in the beginning, i.e., $S=FP_0/10$, $S=FP_0/100$, and so on, and more carefully as the approximation shows more details.

The number of knots of the spline returned, and their location, generally depend on the value of S and on the behaviour of the function underlying the data. However, if E02BEF is called with START = 'W', the knots returned may also depend on the smoothing factors of the previous calls. Therefore if, after a number of trials with different values of S and START = 'W', a fit can finally be accepted as satisfactory, it may be worthwhile to call E02BEF once more with the selected value for S but now using START = 'C'. Often, E02BEF then returns an approximation with the same quality of fit but with fewer knots, which is therefore better if data reduction is also important.

8.3. Outline of Method Used

If S=0, the requisite number of knots is known in advance, i.e., n=m+4; the interior knots are located immediately as $(\lambda)_i = x_{i-2}$, for i=5,6,...,n-4. The corresponding least-squares spline (see E02BAF) is then an interpolating spline and therefore a solution of the problem.

If S>0, a suitable knot set is built up in stages (starting with no interior knots in the case of a cold start but with the knot set found in a previous call if a warm start is chosen). At each stage, a spline is fitted to the data by least-squares (see E02BAF) and (theta), the weighted sum of squares of residuals, is computed. If (theta)>S, new knots are added to the knot set to reduce (theta) at the next stage. The new knots are located in intervals where the fit is particularly poor, their number depending on the value of S and on the progress made so far in reducing (theta). Sooner or later, we find that (theta)<=S and at that point the knot set is accepted. The routine then goes on to compute the (unique) spline which has this knot set and which satisfies the full fitting criterion specified by (2) and (3). The theoretical solution has (theta)=S. The routine computes the spline by an iterative scheme which is ended when (theta)=S within a relative tolerance of 0.001. The main part of each iteration consists of a linear least-squares computation of special form, done in a similarly stable and efficient manner as in E02BAF.

An exception occurs when the routine finds at the start that, even with no interior knots (n=8), the least-squares spline already has its weighted sum of squares of residuals <=S. In this case, since this spline (which is simply a cubic polynomial) also has an optimal value for the smoothness measure (eta), namely zero, it is returned at once as the (trivial) solution. It will usually mean that S has been chosen too large.

For further details of the algorithm and its use, see Dierckx [3]

8.4. Evaluation of Computed Spline

The value of the computed spline at a given value X may be obtained in the double precision variable S by the call:

```
CALL E02BBF(N,LAMDA,C,X,S,IFAIL)
```

where N, LAMDA and C are the output parameters of E02BEF.

The values of the spline and its first three derivatives at a given value X may be obtained in the double precision array SDIF of dimension at least 4 by the call:

```
CALL E02BCF(N,LAMDA,C,X,LEFT,SDIF,IFAIL)
```

where if LEFT = 1, left-hand derivatives are computed and if LEFT /= 1, right-hand derivatives are calculated. The value of LEFT is only relevant if X is an interior knot.

The value of the definite integral of the spline over the interval X(1) to X(M) can be obtained in the double precision variable SINT by the call:

```
CALL E02BDF(N,LAMDA,C,SINT,IFAIL)
```

9. Example

This example program reads in a set of data values, followed by a set of values of S. For each value of S it calls E02BEF to compute a spline approximation, and prints the values of the knots and the B-spline coefficients c_i.

The program includes code to evaluate the computed splines, by calls to E02BBF, at the points x_r and at points mid-way between them. These values are not printed out, however; instead the results are illustrated by plots of the computed splines, together with the data points (indicated by *) and the positions of the knots (indicated by vertical lines): the effect of decreasing S can be clearly seen. (The plots were obtained by

```
        calling NAG Graphical Supplement routine J06FAF(*).)


                Please see figures in printed Reference Manual

        The example program is not reproduced here. The source code for
        all example programs is distributed with the NAG Foundation
        Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Minimal, weighted least-squares bicubic spline fit

— nage.ht —

```
\begin{page}{manpageXXe02daf}{NAG Documentation: e02daf}
\beginscroll
\begin{verbatim}



     E02DAF(3NAG)      Foundation Library (12/10/92)      E02DAF(3NAG)



          E02 -- Curve and Surface Fitting                           E02DAF
                  E02DAF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          E02DAF forms a minimal, weighted least-squares bicubic spline
          surface fit with prescribed knots to a given set of data points.

          2. Specification

                 SUBROUTINE E02DAF (M, PX, PY, X, Y, F, W, LAMDA, MU,
                1                   POINT, NPOINT, DL, C, NC, WS, NWS, EPS,
                2                   SIGMA, RANK, IFAIL)
```

```
 INTEGER          M, PX, PY, POINT(NPOINT), NPOINT, NC, NWS,
1                 RANK, IFAIL
 DOUBLE PRECISION X(M), Y(M), F(M), W(M), LAMDA(PX), MU(PY),
1                 DL(NC), C(NC), WS(NWS), EPS, SIGMA
```

3. Description

This routine determines a bicubic spline fit s(x,y) to the set of data points (x_r, y_r, f_r) with weights w_r, for r=1,2,...,m. The two sets of internal knots of the spline, {(lambda)} and {(mu)}, associated with the variables x and y respectively, are prescribed by the user. These knots can be thought of as dividing the data region of the (x,y) plane into panels (see diagram in Section 5). A bicubic spline consists of a separate bicubic polynomial in each panel, the polynomials joining together with continuity up to the second derivative across the panel boundaries.

s(x,y) has the property that (Sigma), the sum of squares of its weighted residuals $(rho)_r$, for r=1,2,...,m, where

$$(rho)_r = w_r (s(x_r, y_r) - f_r), \qquad (1)$$

is as small as possible for a bicubic spline with the given knot sets. The routine produces this minimized value of (Sigma) and the coefficients c_{ij} in the B-spline representation of s(x,y) - see Section 8. E02DEF and E02DFF are available to compute values of the fitted spline from the coefficients c_{ij}.

The least-squares criterion is not always sufficient to determine the bicubic spline uniquely: there may be a whole family of splines which have the same minimum sum of squares. In these cases, the routine selects from this family the spline for which the sum of squares of the coefficients c_{ij} is smallest: in other words, the minimal least-squares solution. This choice, although arbitrary, reduces the risk of unwanted fluctuations in the spline fit. The method employed involves forming a system of m linear equations in the coefficients c_{ij} and then computing its least-squares solution, which will be the minimal least-squares solution when appropriate. The basis of the method is described in Hayes and Halliday [4]. The matrix of the equation is formed using a recurrence relation for B-splines which is numerically stable (see Cox [1] and de Boor [2] - the former contains the more elementary derivation but, unlike [2], does not cover the case of coincident knots). The least-squares solution is also obtained in a stable manner by using orthogonal transformations, viz. a variant of Givens rotation (see Gentleman [3]). This requires only one row of the matrix to be stored at a time. Advantage is taken of the stepped-band structure which the matrix possesses when the data points are suitably ordered, there being at most sixteen non-zero elements in any row because of the definition of B-splines. First the matrix is reduced to upper triangular form and then the diagonal elements of this triangle are examined in turn. When an element is encountered whose square, divided by the mean squared weight, is less than a threshold (epsilon), it is replaced by zero and the rest of the elements in its row are reduced to zero by rotations with the remaining rows. The rank of the system is taken to be the number of non-zero diagonal elements in the final triangle, and the non-zero rows of this triangle are used to compute the minimal least-squares solution. If all the diagonal elements are non-zero, the rank is equal to the number of coefficients c_{ij} and the solution obtained is the ordinary least-squares solution, which is unique in this case.

4. References

[1] Cox M G (1972) The Numerical Evaluation of B-splines. J. Inst. Math. Appl. 10 134--149.

[2] De Boor C (1972) On Calculating with B-splines. J. Approx. Theory. 6 50--62.

[3] Gentleman W M (1973) Least-squares Computations by Givens Transformations without Square Roots. J. Inst. Math. Applic. 12 329--336.

[4] Hayes J G and Halliday J (1974) The Least-squares Fitting of Cubic Spline Surfaces to General Data Sets. J. Inst. Math. Appl. 14 89--103.

5. Parameters

1: M -- INTEGER Input
On entry: the number of data points, m. Constraint: M > 1.

2: PX -- INTEGER Input

3: PY -- INTEGER Input
On entry: the total number of knots (lambda) and (mu) associated with the variables x and y, respectively. Constraint: PX >= 8 and PY >= 8.

(They are such that PX-8 and PY-8 are the corresponding numbers of interior knots.) The running time and storage required by the routine are both minimized if the axes are labelled so that PY is the smaller of PX and PY.

4: X(M) -- DOUBLE PRECISION array Input

5: Y(M) -- DOUBLE PRECISION array Input

6: F(M) -- DOUBLE PRECISION array Input
On entry: the co-ordinates of the data point (x_r, y_r, f_r), for r=1,2,...,m. The order of the data points is immaterial, but see the array POINT, below.

7: W(M) -- DOUBLE PRECISION array Input
On entry: the weight w_r of the rth data point. It is important to note the definition of weight implied by the equation (1) in Section 3, since it is also common usage to define weight as the square of this weight. In this routine, each w_r should be chosen inversely proportional to the (absolute) accuracy of the corresponding f_r, as expressed, for example, by the standard deviation or probable error of the f_r. When the f_r are all of the same accuracy, all the w_r may be set equal to 1.0.

8: LAMDA(PX) -- DOUBLE PRECISION array Input/Output
On entry: LAMDA(i+4) must contain the ith interior knot $(lambda)_{i+4}$ associated with the variable x, for i=1,2,...,PX-8. The knots must be in non-decreasing order and lie strictly within the range covered by the data values of x. A knot is a value of x at which the spline is allowed to be discontinuous in the third derivative with respect to x, though continuous up to the second derivative. This degree of continuity can be reduced, if the user requires, by the use of coincident knots, provided that no more than four knots are chosen to coincide at any point. Two, or three, coincident knots allow loss of continuity in, respectively, the second and first derivative with respect to x at the value of x at which they coincide. Four coincident knots split the spline surface into two independent parts. For choice of knots see Section 8. On exit: the interior knots LAMDA(5) to LAMDA(PX-4) are unchanged, and the segments LAMDA(1:4) and LAMDA(PX-3:PX)

contain additional (exterior) knots introduced by the routine in order to define the full set of B-splines required. The four knots in the first segment are all set equal to the lowest data value of x and the other four additional knots are all set equal to the highest value: there is experimental evidence that coincident end-knots are best for numerical accuracy. The complete array must be left undisturbed if E02DEF or E02DFF is to be used subsequently.

9: MU(PY) -- DOUBLE PRECISION array Input
On entry: MU(i+4) must contain the ith interior knot $(mu)_{i+4}$ associated with the variable y, i=1,2,...,PY-8. The same remarks apply to MU as to LAMDA above, with Y replacing X, and y replacing x.

10: POINT(NPOINT) -- INTEGER array Input
On entry: indexing information usually provided by E02ZAF which enables the data points to be accessed in the order which produces the advantageous matrix structure mentioned in Section 3. This order is such that, if the (x,y) plane is thought of as being divided into rectangular panels by the two sets of knots, all data in a panel occur before data in succeeding panels, where the panels are numbered from bottom to top and then left to right with the usual arrangement of axes, as indicated in the diagram.

Please see figure in printed Reference Manual

A data point lying exactly on one or more panel sides is considered to be in the highest numbered panel adjacent to the point. E02ZAF should be called to obtain the array POINT, unless it is provided by other means.

11: NPOINT -- INTEGER Input
On entry:
the dimension of the array POINT as declared in the (sub)program from which E02DAF is called.
Constraint: NPOINT >= M + (PX-7)*(PY-7).

12: DL(NC) -- DOUBLE PRECISION array Output
On exit: DL gives the squares of the diagonal elements of the reduced triangular matrix, divided by the mean squared weight. It includes those elements, less than (epsilon), which are treated as zero (see Section 3).

13: C(NC) -- DOUBLE PRECISION array Output
On exit: C gives the coefficients of the fit. C((PY-4)*(i-1)+j) is the coefficient c_{ij} of Section 3 and Section 8 for

```
     i=1,2,...,PX-4 and j=1,2,...,PY-4. These coefficients are
     used by E02DEF or E02DFF to calculate values of the fitted
     function.

14:  NC -- INTEGER                                         Input
     On entry: the value (PX-4)*(PY-4).

15:  WS(NWS) -- DOUBLE PRECISION array                 Workspace

16:  NWS -- INTEGER                                        Input
     On entry:
     the dimension of the array WS as declared in the
     (sub)program from which E02DAF is called.
     Constraint: NWS>=(2*NC+1)*(3*PY-6)-2.

17:  EPS -- DOUBLE PRECISION                               Input
     On entry: a threshold (epsilon) for determining the
     effective rank of the system of linear equations. The rank
     is determined as the number of elements of the array DL (see
     below) which are non-zero. An element of DL is regarded as
     zero if it is less than (epsilon). Machine precision is a
     suitable value for (epsilon) in most practical applications
     which have only 2 or 3 decimals accurate in data. If some
     coefficients of the fit prove to be very large compared with
     the data ordinates, this suggests that (epsilon) should be
     increased so as to decrease the rank. The array DL will give
     a guide to appropriate values of (epsilon) to achieve this,
     as well as to the choice of (epsilon) in other cases where
     some experimentation may be needed to determine a value
     which leads to a satisfactory fit.

18:  SIGMA -- DOUBLE PRECISION                            Output
     On exit: (Sigma), the weighted sum of squares of residuals.
     This is not computed from the individual residuals but from
     the right-hand sides of the orthogonally-transformed linear
     equations. For further details see Hayes and Halliday [4]
     page 97. The two methods of computation are theoretically
     equivalent, but the results may differ because of rounding
     error.

19:  RANK -- INTEGER                                      Output
     On exit: the rank of the system as determined by the value
     of the threshold (epsilon). When RANK = NC, the least-
     squares solution is unique: in other cases the minimal
     least-squares solution is computed.

20:  IFAIL -- INTEGER                               Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.
```

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1
At least one set of knots is not in non-decreasing order, or an interior knot is outside the range of the data values.

IFAIL= 2
More than four knots coincide at a single point, possibly because all data points have the same value of x (or y) or because an interior knot coincides with an extreme data value.

IFAIL= 3
Array POINT does not indicate the data points in panel order. Call E02ZAF to obtain a correct array.

IFAIL= 4
On entry M <= 1,

or PX < 8,

or PY < 8,

or NC /= (PX-4)*(PY-4),

or NWS is too small,

or NPOINT is too small.

IFAIL= 5
All the weights w_r are zero or rank determined as zero.

7. Accuracy

The computation of the B-splines and reduction of the observation matrix to triangular form are both numerically stable.

8. Further Comments

The time taken by this routine is approximately proportional to the number of data points, m, and to $(3*(PY-4)+4)^2$.

The B-spline representation of the bicubic spline is

$$s(x,y)=\sum_{ij} c_{ij} M_i(x) N_j(y)$$

summed over i=1,2,...,PX-4 and over j=1,2,...,PY-4. Here $M_i(x)$ and $N_j(y)$ denote normalised cubic B-splines,the former defined on the knots $\lambda_i,\lambda_{i+1},...,\lambda_{i+4}$ and the latter on the knots $\mu_j,\mu_{j+1},...,\mu_{j+4}$. For further details, see Hayes and Halliday [4] for bicubic splines and de Boor [2] for normalised B-splines.

The choice of the interior knots, which help to determine the spline's shape, must largely be a matter of trial and error. It is usually best to start with a small number of knots and, examining the fit at each stage, add a few knots at a time at places where the fit is particularly poor. In intervals of x or y where the surface represented by the data changes rapidly, in function value or derivatives, more knots will be needed than elsewhere. In some cases guidance can be obtained by analogy with the case of coincident knots: for example, just as three coincident knots can produce a discontinuity in slope, three close knots can produce rapid change in slope. Of course, such rapid changes in behaviour must be adequately represented by the data points, as indeed must the behaviour of the surface generally, if a satisfactory fit is to be achieved. When there is no rapid change in behaviour, equally-spaced knots will often suffice.

In all cases the fit should be examined graphically before it is accepted as satisfactory.

The fit obtained is not defined outside the rectangle

$$\lambda_4 \le x \le \lambda_{PX-3} , \quad \mu_4 \le y \le \mu_{PY-3}$$

The reason for taking the extreme data values of x and y for these four knots is that, as is usual in data fitting, the fit cannot be expected to give satisfactory values outside the data region. If, nevertheless, the user requires values over a larger rectangle, this can be achieved by augmenting the data with two artificial data points (a,c,0) and (b,d,0) with zero weight, where $a\le x\le b$, $c\le y\le d$ defines the enlarged rectangle. In the case when the data are adequate to make the least-squares solution unique (RANK = NC), this enlargement will not affect the fit over the original rectangle, except for possibly enlarged rounding errors, and will simply continue the bicubic polynomials in the panels bordering the rectangle out to the new boundaries: in other cases the fit will be affected. Even using the original rectangle there may be regions within it, particularly at its corners, which lie outside the data region and where, therefore, the fit will be unreliable. For example, if there is no data point in panel 1 of the diagram in Section 5, the least-squares criterion leaves the spline indeterminate in this panel: the minimal spline determined by the subroutine in this case passes through the value zero at the point ((lambda)$_4$,(mu)$_4$).

9. Example

This example program reads a value for (epsilon), and a set of data points, weights and knot positions. If there are more y knots than x knots, it interchanges the x and y axes. It calls E02ZAF to sort the data points into panel order, E02DAF to fit a bicubic spline to them, and E02DEF to evaluate the spline at the data points.

Finally it prints:

the weighted sum of squares of residuals computed from the linear equations;

the rank determined by E02DAF;

data points, fitted values and residuals in panel order;

the weighted sum of squares of the residuals;

the coefficients of the spline fit.

The program is written to handle any number of data sets.

Note: the data supplied in this example is not typical of a realistic problem: the number of data points would normally be much larger (in which case the array dimensions and the value of NWS in the program would have to be increased); and the value of (epsilon) would normally be much smaller on most machines (see Section 5; the relatively large value of 10^{-6} has been chosen in order to illustrate a minimal least-squares solution when RANK < NC; in this example NC = 24).

The example program is not reproduced here. The source code for

```
         all example programs is distributed with the NAG Foundation
         Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

———-

Bicubic spline approximation to a set of data values

— nage.ht —

```
\begin{page}{manpageXXe02dcf}{NAG Documentation: e02dcf}
\beginscroll
\begin{verbatim}



     E02DCF(3NAG)      Foundation Library (12/10/92)      E02DCF(3NAG)



          E02 -- Curve and Surface Fitting                        E02DCF
                  E02DCF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          E02DCF computes a bicubic spline approximation to a set of data
          values, given on a rectangular grid in the x-y plane. The knots
          of the spline are located automatically, but a single parameter
          must be specified to control the trade-off between closeness of
          fit and smoothness of fit.

          2. Specification

                 SUBROUTINE E02DCF (START, MX, X, MY, Y, F, S, NXEST,
                1                   NYEST, NX, LAMDA, NY, MU, C, FP, WRK,
                2                   LWRK, IWRK, LIWRK, IFAIL)
                 INTEGER          MX, MY, NXEST, NYEST, NX, NY, LWRK, IWRK
                1                 (LIWRK), LIWRK, IFAIL
                 DOUBLE PRECISION X(MX), Y(MY), F(MX*MY), S, LAMDA(NXEST),
```

```
     1                  MU(NYEST), C((NXEST-4)*(NYEST-4)), FP, WRK
     2                  (LWRK)
      CHARACTER*1       START
```

3. Description

This routine determines a smooth bicubic spline approximation $s(x,y)$ to the set of data points $(x_q,y_r,f_{q,r})$, for $q=1,2,...,m_x$ and $r=1,2,...,m_y$.

The spline is given in the B-spline representation

$$s(x,y)=\sum_{i=1}^{n_x-4}\sum_{j=1}^{n_y-4}c_{ij}M_i(x)N_j(y), \qquad (1)$$

where $M_i(x)$ and $N_j(y)$ denote normalised cubic B-splines, the former defined on the knots λ_i to λ_{i+4} and the latter on the knots μ_j to μ_{j+4}. For further details, see Hayes and Halliday [4] for bicubic splines and de Boor [1] for normalised B-splines.

The total numbers n_x and n_y of these knots and their values $\lambda_1,...,\lambda_{n_x}$ and $\mu_1,...,\mu_{n_y}$ are chosen automatically by the routine. The knots $\lambda_5,..., \lambda_{n_x-4}$ and $\mu_5,...,\mu_{n_y-4}$ are the interior knots; they divide the approximation domain $[x_1,x_{m_x}]*[y_1,y_{m_y}]$ into $(n_x-7)*(n_y-7)$ subpanels $[\lambda_i,\lambda_{i+1}]*[\mu_j,\mu_{j+1}]$, for $i=4,5,...,n_x-4$, $j=4,5,...,n_y-4$. Then, much as in the curve case (see E02BEF), the coefficients c_{ij} are determined as the solution of the following constrained minimization problem:

minimize

$$\eta, \qquad (2)$$

subject to the constraint

$$\theta = \sum_{q=1}^{m_x} \sum_{r=1}^{m_y} \epsilon_{q,r}^2 \le S, \qquad (3)$$

where η is a measure of the (lack of) smoothness of $s(x,y)$. Its value depends on the discontinuity jumps in $s(x,y)$ across the boundaries of the subpanels. It is zero only when there are no discontinuities and is positive otherwise, increasing with the size of the jumps (see Dierckx [2] for details).

$\epsilon_{q,r}$ denotes the residual $f_{q,r}-s(x_q,y_r)$,

and S is a non-negative number to be specified by the user.

By means of the parameter S, 'the smoothing factor', the user will then control the balance between smoothness and closeness of fit, as measured by the sum of squares of residuals in (3). If S is too large, the spline will be too smooth and signal will be lost (underfit); if S is too small, the spline will pick up too much noise (overfit). In the extreme cases the routine will return an interpolating spline ($\theta=0$) if S is set to zero, and the least-squares bicubic polynomial ($\eta=0$) if S is set very large. Experimenting with S-values between these two extremes should result in a good compromise. (See Section 8.3 for advice on choice of S.)

The method employed is outlined in Section 8.5 and fully described in Dierckx [2] and [3]. It involves an adaptive strategy for locating the knots of the bicubic spline (depending on the function underlying the data and on the value of S), and an iterative method for solving the constrained minimization problem once the knots have been determined.

Values of the computed spline can subsequently be computed by calling E02DEF or E02DFF as described in Section 8.6.

4. References

[1] De Boor C (1972) On Calculating with B-splines. J. Approx. Theory. 6 50--62.

[2] Dierckx P (1982) A Fast Algorithm for Smoothing Data on a Rectangular Grid while using Spline Functions. SIAM J. Numer. Anal. 19 1286--1304.

[3] Dierckx P (1981) An Improved Algorithm for Curve Fitting with Spline Functions. Report TW54. Department of Computer Science, Katholieke Universiteit Leuven.

[4] Hayes J G and Halliday J (1974) The Least-squares Fitting of Cubic Spline Surfaces to General Data Sets. J. Inst. Math. Appl. 14 89--103.

[5] Reinsch C H (1967) Smoothing by Spline Functions. Num. Math. 10 177--183.

5. Parameters

1: START -- CHARACTER*1 Input

On entry: START must be set to 'C' or 'W'.

If START = 'C' (Cold start), the routine will build up the knot set starting with no interior knots. No values need be assigned to the parameters NX, NY, LAMDA, MU, WRK or IWRK.

If START = 'W' (Warm start), the routine will restart the knot-placing strategy using the knots found in a previous call of the routine. In this case, the parameters NX, NY, LAMDA, MU, WRK and IWRK must be unchanged from that previous call. This warm start can save much time in searching for a satisfactory value of S. Constraint: START = 'C' or 'W'.

2: MX -- INTEGER Input

On entry: m_x, the number of grid points along the x axis.

Constraint: MX >= 4.

3: X(MX) -- DOUBLE PRECISION array Input

On entry: X(q) must be set to x_q, the x co-ordinate of the qth grid point along the x axis, for $q=1,2,...,m_x$.

Constraint: $x_1<x_2<...<x_{m_x}$.

4: MY -- INTEGER Input

On entry: m_y, the number of grid points along the y axis.

Constraint: $MY >= 4$.

5: Y(MY) -- DOUBLE PRECISION array Input

On entry: Y(r) must be set to y_r, the y co-ordinate of the rth grid point along the y axis, for $r=1,2,...,m_y$.

Constraint: $y_1<y_2<...<y_{m_y}$.

6: F(MX*MY) -- DOUBLE PRECISION array Input

On entry: $F(m_y*(q-1)+r)$ must contain the data value $f_{q,r}$, for $q=1,2,...,m_x$ and $r=1,2,...,m_y$.

7: S -- DOUBLE PRECISION Input

On entry: the smoothing factor, S.

If S=0.0, the routine returns an interpolating spline.

If S is smaller than machine precision, it is assumed equal to zero.

For advice on the choice of S, see Section 3 and Section 8.3
Constraint: S >= 0.0.

8: NXEST -- INTEGER Input

9: NYEST -- INTEGER Input

On entry: an upper bound for the number of knots n_x and n_y required in the x- and y-directions respectively.

In most practical situations, NXEST $=m_x/2$ and NYEST $m_y/2$ is sufficient. NXEST and NYEST never need to be larger than m_x+4 and m_y+4 respectively, the numbers of knots needed for interpolation (S=0.0). See also Section 8.4. Constraint: NXEST >= 8 and NYEST >= 8.

10: NX -- INTEGER Input/Output

On entry: if the warm start option is used, the value of NX must be left unchanged from the previous call. On exit: the total number of knots, n_x, of the computed spline with respect to the x variable.

11: LAMDA(NXEST) -- DOUBLE PRECISION array Input/Output
On entry: if the warm start option is used, the values LAMDA(1), LAMDA(2),...,LAMDA(NX) must be left unchanged from the previous call. On exit: LAMDA contains the complete set of knots $\mathrm{(lambda)}_i$ associated with the x variable, i.e., the interior knots LAMDA(5), LAMDA(6), ..., LAMDA(NX-4) as well as the additional knots LAMDA(1) = LAMDA(2) = LAMDA(3) = LAMDA(4) = X(1) and LAMDA(NX-3) = LAMDA(NX-2) = LAMDA(NX-1) = LAMDA(NX) = X(MX) needed for the B-spline representation.

12: NY -- INTEGER Input/Output
On entry: if the warm start option is used, the value of NY must be left unchanged from the previous call. On exit: the total number of knots, n_y, of the computed spline with respect to the y variable.

13: MU(NYEST) -- DOUBLE PRECISION array Input/Output
On entry: if the warm start option is used, the values MU (1), MU(2),...,MU(NY) must be left unchanged from the previous call. On exit: MU contains the complete set of knots $\mathrm{(mu)}_i$ associated with the y variable, i.e., the interior knots MU(5), MU(6),...,MU(NY-4) as well as the additional knots MU(1) = MU(2) = MU(3) = MU(4) = Y(1) and MU (NY-3) = MU(NY-2) = MU(NY-1) = MU(NY) = Y(MY) needed for the B-spline representation.

14: C((NXEST-4)*(NYEST-4)) -- DOUBLE PRECISION array Output
On exit: the coefficients of the spline approximation. C($(n_y-4)*(i-1)+j$) is the coefficient c_{ij} defined in Section 3.

15: FP -- DOUBLE PRECISION Output
On exit: the sum of squared residuals, (theta), of the computed spline approximation. If FP = 0.0, this is an interpolating spline. FP should equal S within a relative tolerance of 0.001 unless NX = NY = 8, when the spline has no interior knots and so is simply a bicubic polynomial. For knots to be inserted, S must be set to a value below the value of FP produced in this case.

16: WRK(LWRK) -- DOUBLE PRECISION array Workspace
On entry: if the warm start option is used, the values WRK (1),...,WRK(4) must be left unchanged from the previous call.

This array is used as workspace.

```
17:  LWRK -- INTEGER                                       Input
     On entry:
     the dimension of the array WRK as declared in the
     (sub)program from which E02DCF is called.
     Constraint:
          LWRK>=4*(MX+MY)+11*(NXEST+NYEST)+NXEST*MY

                +max(MY,NXEST)+54.

18:  IWRK(LIWRK) -- INTEGER array                      Workspace
     On entry: if the warm start option is used, the values IWRK
     (1), ..., IWRK(3) must be left unchanged from the previous
     call.

     This array is used as workspace.

19:  LIWRK -- INTEGER                                      Input
     On entry:
     the dimension of the array IWRK as declared in the
     (sub)program from which E02DCF is called.
     Constraint: LIWRK >= 3 + MX + MY + NXEST + NYEST.

20:  IFAIL -- INTEGER                               Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

IFAIL= 1
     On entry START /= 'C' or 'W',

     or       MX < 4,

     or       MY < 4,

     or       S < 0.0,

     or       S = 0.0 and NXEST < MX + 4,

     or       S = 0.0 and NYEST < MY + 4,
```

```
     or       NXEST < 8,

     or       NYEST < 8,

     or       LWRK < 4*(MX+MY)+11*(NXEST+NYEST)+NXEST*MY+
                     +max(MY,NXEST)+54

     or       LIWRK < 3 + MX + MY + NXEST + NYEST.
```

IFAIL= 2
The values of X(q), for q = 1,2,...,MX, are not in strictly increasing order.

IFAIL= 3
The values of Y(r), for r = 1,2,...,MY, are not in strictly increasing order.

IFAIL= 4
The number of knots required is greater than allowed by NXEST and NYEST. Try increasing NXEST and/or NYEST and, if necessary, supplying larger arrays for the parameters LAMDA, MU, C, WRK and IWRK. However, if NXEST and NYEST are already large, say NXEST > MX/2 and NYEST > MY/2, then this error exit may indicate that S is too small.

IFAIL= 5
The iterative process used to compute the coefficients of the approximating spline has failed to converge. This error exit may occur if S has been set very small. If the error persists with increased S, consult NAG.

If IFAIL = 4 or 5, a spline approximation is returned, but it fails to satisfy the fitting criterion (see (2) and (3) in Section 3) -- perhaps by only a small amount, however.

7. Accuracy

On successful exit, the approximation returned is such that its sum of squared residuals FP is equal to the smoothing factor S, up to a specified relative tolerance of 0.001 - except that if $n_x=8$ and $n_y=8$, FP may be significantly less than S: in this case the computed spline is simply the least-squares bicubic polynomial approximation of degree 3, i.e., a spline with no interior knots.

8. Further Comments

8.1. Timing

The time taken for a call of E02DCF depends on the complexity of the shape of the data, the value of the smoothing factor S, and the number of data points. If E02DCF is to be called for different values of S, much time can be saved by setting START =

8.2. Weighting of Data Points

E02DCF does not allow individual weighting of the data values. If these were determined to widely differing accuracies, it may be better to use E02DDF. The computation time would be very much longer, however.

8.3. Choice of S

If the standard deviation of $f_{q,r}$ is the same for all q and r (the case for which this routine is designed - see Section 8.2.) and known to be equal, at least approximately, to (sigma), say, then following Reinsch [5] and choosing the smoothing factor S in the range $(sigma)^2$ (m+-\/2m), where $m=m_x m_y$, is likely to give a good start in the search for a satisfactory value. If the standard deviations vary, the sum of their squares over all the data points could be used. Otherwise experimenting with different values of S will be required from the start, taking account of the remarks in Section 3.

In that case, in view of computation time and memory requirements, it is recommended to start with a very large value for S and so determine the least-squares bicubic polynomial; the value returned for FP, call it FP_0, gives an upper bound for S. Then progressively decrease the value of S to obtain closer fits - say by a factor of 10 in the beginning, i.e., $S=FP_0/10$, $S=FP_0/100$, and so on, and more carefully as the approximation shows more details.

The number of knots of the spline returned, and their location, generally depend on the value of S and on the behaviour of the function underlying the data. However, if E02DCF is called with START = 'W', the knots returned may also depend on the smoothing factors of the previous calls. Therefore if, after a number of trials with different values of S and START = 'W', a fit can finally be accepted as satisfactory, it may be worthwhile to call E02DCF once more with the selected value for S but now using START = 'C'. Often, E02DCF then returns an approximation with the same quality of fit but with fewer knots, which is therefore better if data reduction is also important.

8.4. Choice of NXEST and NYEST

The number of knots may also depend on the upper bounds NXEST and NYEST. Indeed, if at a certain stage in E02DCF the number of knots in one direction (say n_x) has reached the value of its upper bound (NXEST), then from that moment on all subsequent knots are added in the other (y) direction. Therefore the user has the option of limiting the number of knots the routine locates in any direction. For example, by setting NXEST = 8 (the lowest allowable value for NXEST), the user can indicate that he wants an approximation which is a simple cubic polynomial in the variable x.

8.5. Outline of Method Used

If S=0, the requisite number of knots is known in advance, i.e., $n_x=m_x+4$ and $n_y=m_y+4$; the interior knots are located immediately as $(\lambda)_i = x_{i-2}$ and $(\mu)_j = y_{j-2}$, for $i=5,6,\ldots,n_x-4$ and $j=5,6,\ldots,n_y-4$. The corresponding least-squares spline is then an interpolating spline and therefore a solution of the problem.

If S>0, suitable knot sets are built up in stages (starting with no interior knots in the case of a cold start but with the knot set found in a previous call if a warm start is chosen). At each stage, a bicubic spline is fitted to the data by least-squares, and (theta), the sum of squares of residuals, is computed. If (theta)>S, new knots are added to one knot set or the other so as to reduce (theta) at the next stage. The new knots are located in intervals where the fit is particularly poor, their number depending on the value of S and on the progress made so far in reducing (theta). Sooner or later, we find that (theta)<=S and at that point the knot sets are accepted. The routine then goes on to compute the (unique) spline which has these knot sets and which satisfies the full fitting criterion specified by (2) and (3). The theoretical solution has (theta)=S. The routine computes the spline by an iterative scheme which is ended when (theta)=S within a relative tolerance of 0.001. The main part of each iteration consists of a linear least-squares computation of special form, done in a similarly stable and efficient manner as in E02BAF for least-squares curve fitting.

An exception occurs when the routine finds at the start that, even with no interior knots ($n_x=n_y=8$), the least-squares spline

```
                              x  y
already has its sum of residuals <=S. In this case, since this
spline (which is simply a bicubic polynomial) also has an optimal
value for the smoothness measure (eta), namely zero, it is
returned at once as the (trivial) solution. It will usually mean
that S has been chosen too large.

For further details of the algorithm and its use see Dierckx [2].

8.6. Evaluation of Computed Spline

The values of the computed spline at the points (TX(r),TY(r)),
for r = 1,2,...,N, may be obtained in the double precision array
FF, of length at least N, by the following code:

       IFAIL = 0
       CALL E02DEF(N,NX,NY,TX,TY,LAMDA,MU,C,FF,WRK,IWRK,IFAIL)

where NX, NY, LAMDA, MU and C are the output parameters of E02DCF
, WRK is a double precision workspace array of length at least
NY-4, and IWRK is an integer workspace array of length at least
NY-4.

To evaluate the computed spline on a KX by KY rectangular grid of
points in the x-y plane, which is defined by the x co-ordinates
stored in TX(q), for q=1,2,...,KX, and the y co-ordinates stored
in TY(r), for r=1,2,...,KY, returning the results in the double
precision array FG which is of length at least KX*KY, the
following call may be used:

      IFAIL = 0
      CALL E02DFF(KX,KY,NX,NY,TX,TY,LAMDA,MU,C,FG,WRK,LWRK,
     *            IWRK,LIWRK,IFAIL)

where NX, NY, LAMDA, MU and C are the output parameters of E02DCF
, WRK is a double precision workspace array of length at least
LWRK = min(NWRK1,NWRK2), NWRK1 = KX*4+NX, NWRK2 = KY*4+NY, and
IWRK is an integer workspace array of length at least LIWRK = KY
+ NY - 4 if NWRK1 >= NWRK2, or KX + NX - 4 otherwise. The result
of the spline evaluated at grid point (q,r) is returned in
element (KY*(q-1)+r) of the array FG.

9. Example

This example program reads in values of MX, MY, x , for q = 1,2,.
                                                 q
            r
ordinates f    defined at the grid points (x ,y ). It then calls
           q,r                              q  r
E02DCF to compute a bicubic spline approximation for one
```

```
         specified value of S, and prints the values of the computed knots
         and B-spline coefficients. Finally it evaluates the spline at a
         small sample of points on a rectangular grid.

         The example program is not reproduced here. The source code for
         all example programs is distributed with the NAG Foundation
         Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Bicubic spline approximation to a set of scattered data

— nage.ht —

```
\begin{page}{manpageXXe02ddf}{NAG Documentation: e02ddf}
\beginscroll
\begin{verbatim}



     E02DDF(3NAG)      Foundation Library (12/10/92)      E02DDF(3NAG)



         E02 -- Curve and Surface Fitting                          E02DDF
                 E02DDF -- NAG Foundation Library Routine Document

         Note: Before using this routine, please read the Users' Note for
         your implementation to check implementation-dependent details.
         The symbol (*) after a NAG routine name denotes a routine that is
         not included in the Foundation Library.

         1. Purpose

         E02DDF computes a bicubic spline approximation to a set of
         scattered data. The knots of the spline are located
         automatically, but a single parameter must be specified to
         control the trade-off between closeness of fit and smoothness of
         fit.

         2. Specification

                SUBROUTINE E02DDF (START, M, X, Y, F, W, S, NXEST, NYEST,
```

```
     1                   NX, LAMDA, NY, MU, C, FP, RANK, WRK,
     2                   LWRK, IWRK, LIWRK, IFAIL)
      INTEGER          M, NXEST, NYEST, NX, NY, RANK, LWRK, IWRK
     1                 (LIWRK), LIWRK, IFAIL
      DOUBLE PRECISION X(M), Y(M), F(M), W(M), S, LAMDA(NXEST),
     1                 MU(NYEST), C((NXEST-4)*(NYEST-4)), FP, WRK
     2                 (LWRK)
      CHARACTER*1      START
```

3. Description

This routine determines a smooth bicubic spline approximation $s(x,y)$ to the set of data points (x_r,y_r,f_r) with weights w_r, for $r=1,2,...,m$.

The approximation domain is considered to be the rectangle $[x_{min},x_{max}]*[y_{min},y_{max}]$, where x_{min} (y_{min}) and x_{max} (y_{max}) denote the lowest and highest data values of x (y).

The spline is given in the B-spline representation

$$s(x,y)=\sum_{i=1}^{n_x-4}\sum_{j=1}^{n_y-4} c_{ij} M_i(x)N_j(y), \qquad (1)$$

where $M_i(x)$ and $N_j(y)$ denote normalised cubic B-splines, the former defined on the knots $(\lambda)_i$ to $(\lambda)_{i+4}$ and the latter on the knots $(\mu)_j$ to $(\mu)_{j+4}$. For further details, see Hayes and Halliday [4] for bicubic splines and de Boor [1] for normalised B-splines.

The total numbers n_x and n_y of these knots and their values $(\lambda)_1,...,(\lambda)_{n_x}$ and $(\mu)_1,...,(\mu)_{n_y}$ are chosen automatically by the routine. The knots $(\lambda)_5,...,$ $(\lambda)_{n_x-4}$ and $(\mu)_5,..., (\mu)_{n_y-4}$ are the interior knots; they divide the approximation domain $[x_{min},x_{max}]*[y_{min},y_{max}]$ into (

$(n_x-7)*(n_y-7)$ subpanels $[(\lambda)_i^{min},(\lambda)_{i+1}^{max}]*[(\mu)_j^{min},(\mu)_{j+1}^{max}]$, for $i=4,5,\ldots,n_x-4$; $j=4,5,\ldots,n_y-4$. Then, much as in the curve case (see E02BEF), the coefficients c_{ij} are determined as the solution of the following constrained minimization problem:

minimize

$$(\eta), \tag{2}$$

subject to the constraint

$$(\theta)=\sum_{r=1}^{m} (\epsilon)_r^2 \le S \tag{3}$$

where: (η) is a measure of the (lack of) smoothness of $s(x,y)$. Its value depends on the discontinuity jumps in $s(x,y)$ across the boundaries of the subpanels. It is zero only when there are no discontinuities and is positive otherwise, increasing with the size of the jumps (see Dierckx [2] for details).

$(\epsilon)_r$ denotes the weighted residual $w_r(f_r-s(x_r,y_r))$,

and S is a non-negative number to be specified by the user.

By means of the parameter S, 'the smoothing factor', the user will then control the balance between smoothness and closeness of fit, as measured by the sum of squares of residuals in (3). If S is too large, the spline will be too smooth and signal will be lost (underfit); if S is too small, the spline will pick up too much noise (overfit). In the extreme cases the method would return an interpolating spline ($(\theta)=0$) if S were set to zero, and returns the least-squares bicubic polynomial ($(\eta)=0$) if S is set very large. Experimenting with S-values between these two extremes should result in a good compromise. (See Section 8.2 for advice on choice of S.) Note however, that this routine, unlike E02BEF and E02DCF, does not allow S to be set exactly to zero: to compute an interpolant to scattered data, E01SAF or E01SEF should be used.

The method employed is outlined in Section 8.5 and fully described in Dierckx [2] and [3]. It involves an adaptive strategy for locating the knots of the bicubic spline (depending on the function underlying the data and on the value of S), and an iterative method for solving the constrained minimization problem once the knots have been determined.

Values of the computed spline can subsequently be computed by calling E02DEF or E02DFF as described in Section 8.6.

4. References

[1] De Boor C (1972) On Calculating with B-splines. J. Approx. Theory. 6 50--62.

[2] Dierckx P (1981) An Algorithm for Surface Fitting with Spline Functions. IMA J. Num. Anal. 1 267--283.

[3] Dierckx P (1981) An Improved Algorithm for Curve Fitting with Spline Functions. Report TW54. Department of Computer Science, Katholieke Universiteit Leuven.

[4] Hayes J G and Halliday J (1974) The Least-squares Fitting of Cubic Spline Surfaces to General Data Sets. J. Inst. Math. Appl. 14 89--103.

[5] Peters G and Wilkinson J H (1970) The Least-squares Problem and Pseudo-inverses. Comput. J. 13 309--316.

[6] Reinsch C H (1967) Smoothing by Spline Functions. Num. Math. 10 177--183.

5. Parameters

1: START -- CHARACTER*1 Input
On entry: START must be set to 'C' or 'W'.

If START = 'C' (Cold start), the routine will build up the knot set starting with no interior knots. No values need be assigned to the parameters NX, NY, LAMDA, MU or WRK.

If START = 'W' (Warm start), the routine will restart the knot-placing strategy using the knots found in a previous call of the routine. In this case, the parameters NX, NY, LAMDA, MU and WRK must be unchanged from that previous call. This warm start can save much time in searching for a satisfactory value of S. Constraint: START = 'C' or 'W'.

2: M -- INTEGER Input
On entry: m, the number of data points.

The number of data points with non-zero weight (see W below) must be at least 16.

3: X(M) -- DOUBLE PRECISION array Input

4: Y(M) -- DOUBLE PRECISION array Input

5: F(M) -- DOUBLE PRECISION array Input
On entry: X(r), Y(r), F(r) must be set to the co-ordinates of (x_r, y_r, f_r), the rth data point, for r=1,2,...,m. The order of the data points is immaterial.

6: W(M) -- DOUBLE PRECISION array Input
On entry: W(r) must be set to w_r, the rth value in the set of weights, for r=1,2,...,m. Zero weights are permitted and the corresponding points are ignored, except when determining x_{min}, x_{max}, y_{min} and y_{max} (see Section 8.4). For advice on the choice of weights, see Section 2.1.2 of the Chapter Introduction. Constraint: the number of data points with non-zero weight must be at least 16.

7: S -- DOUBLE PRECISION Input
On entry: the smoothing factor, S.

For advice on the choice of S, see Section 3 and Section 8.2 . Constraint: S > 0.0.

8: NXEST -- INTEGER Input

9: NYEST -- INTEGER Input
On entry: an upper bound for the number of knots n_x and n_y required in the x- and y-directions respectively.

In most practical situations, NXEST = NYEST = $4+\sqrt{m/2}$ is sufficient. See also Section 8.3. Constraint: NXEST >= 8 and NYEST >= 8.

10: NX -- INTEGER Input/Output
On entry: if the warm start option is used, the value of NX must be left unchanged from the previous call. On exit: the total number of knots, n_x, of the computed spline with respect to the x variable.

11: LAMDA(NXEST) -- DOUBLE PRECISION array Input/Output
On entry: if the warm start option is used, the values LAMDA

(1), LAMDA(2),...,LAMDA(NX) must be left unchanged from the previous call. On exit: LAMDA contains the complete set of knots $(lambda)_i$ associated with the x variable, i.e., the interior knots LAMDA(5), LAMDA(6),...,LAMDA(NX-4) as well as the additional knots LAMDA(1) = LAMDA(2) = LAMDA(3) = LAMDA (4) = x_{min} and LAMDA(NX-3) = LAMDA(NX-2) = LAMDA(NX-1) = LAMDA(NX) = x_{max} needed for the B-spline representation (where x_{min} and x_{max} are as described in Section 3).

12: NY -- INTEGER Input/Output
On entry: if the warm start option is used, the value of NY must be left unchanged from the previous call. On exit: the total number of knots, n_y, of the computed spline with respect to the y variable.

13: MU(NYEST) -- DOUBLE PRECISION array Input/Output
On entry: if the warm start option is used, the values MU(1) MU(2),...,MU(NY) must be left unchanged from the previous call. On exit: MU contains the complete set of knots $(mu)_i$ associated with the y variable, i.e., the interior knots MU (5), MU(6),...,MU(NY-4) as well as the additional knots MU (1) = MU(2) = MU(3) = MU(4) = y_{min} and MU(NY-3) = MU(NY-2) = MU(NY-1) = MU(NY) = y_{max} needed for the B-spline representation (where y_{min} and y_{max} are as described in Section 3).

14: C((NXEST-4)*(NYEST-4)) -- DOUBLE PRECISION array Output
On exit: the coefficients of the spline approximation. C($(n_y-4)*(i-1)+j$) is the coefficient c_{ij} defined in Section 3.

15: FP -- DOUBLE PRECISION Output
On exit: the weighted sum of squared residuals, (theta), of the computed spline approximation. FP should equal S within a relative tolerance of 0.001 unless NX = NY = 8, when the spline has no interior knots and so is simply a bicubic polynomial. For knots to be inserted, S must be set to a value below the value of FP produced in this case.

16: RANK -- INTEGER Output
On exit: RANK gives the rank of the system of equations used

to compute the final spline (as determined by a suitable machine-dependent threshold). When RANK = (NX-4)*(NY-4), the solution is unique; otherwise the system is rank-deficient and the minimum-norm solution is computed. The latter case may be caused by too small a value of S.

17: WRK(LWRK) -- DOUBLE PRECISION array Workspace
On entry: if the warm start option is used, the value of WRK (1) must be left unchanged from the previous call.

This array is used as workspace.

18: LWRK -- INTEGER Input
On entry:
the dimension of the array WRK as declared in the (sub)program from which E02DDF is called.
Constraint: LWRK >= (7*u*v+25*w)*(w+1)+2*(u+v+4*M)+23*w+56,

where

u=NXEST-4, v=NYEST-4, and w=max(u,v).

For some problems, the routine may need to compute the minimal least-squares solution of a rank-deficient system of linear equations (see Section 3). The amount of workspace required to solve such problems will be larger than specified by the value given above, which must be increased by an amount, LWRK2 say. An upper bound for LWRK2 is given by 4*u*v*w+2*u*v+4*w, where u, v and w are as above. However, if there are enough data points, scattered uniformly over the approximation domain, and if the smoothing factor S is not too small, there is a good chance that this extra workspace is not needed. A lot of memory might therefore be saved by assuming LWRK2 = 0.

19: IWRK(LIWRK) -- INTEGER array Workspace

20: LIWRK -- INTEGER Input
On entry:
the dimension of the array IWRK as declared in the (sub)program from which E02DDF is called.
Constraint: LIWRK>=M+2*(NXEST-7)*(NYEST-7).

21: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see

```
     Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

IFAIL= 1
     On entry START /= 'C' or 'W',

     or       the number of data points with non-zero weight <
              16,

     or       S <= 0.0,

     or       NXEST < 8,

     or       NYEST < 8,

     or       LWRK < (7*u*v+25*w)*(w+1)+2*(u+v+4*M)+23*w+56,
              where u = NXEST - 4, v = NYEST - 4 and w=max(u,v),

     or       LIWRK <M+2*(NXEST-7)*(NYEST-7).

IFAIL= 2
     On entry either all the X(r), for r = 1,2,...,M, are equal,
     or all the Y(r), for r = 1,2,...,M, are equal.

IFAIL= 3
     The number of knots required is greater than allowed by
     NXEST and NYEST. Try increasing NXEST and/or NYEST and, if
     necessary, supplying larger arrays for the parameters LAMDA,
     MU, C, WRK and IWRK. However, if NXEST and NYEST are already


     large, say NXEST, NYEST > 4 + \/M/2, then this error exit
     may indicate that S is too small.

IFAIL= 4
     No more knots can be added because the number of B-spline
     coefficients (NX-4)*(NY-4) already exceeds the number of
     data points M. This error exit may occur if either of S or M
     is too small.

IFAIL= 5
     No more knots can be added because the additional knot would
     (quasi) coincide with an old one. This error exit may occur
     if too large a weight has been given to an inaccurate data
```

point, or if S is too small.

IFAIL= 6

The iterative process used to compute the coefficients of the approximating spline has failed to converge. This error exit may occur if S has been set very small. If the error persists with increased S, consult NAG.

IFAIL= 7

LWRK is too small; the routine needs to compute the minimal least-squares solution of a rank-deficient system of linear equations, but there is not enough workspace. There is no approximation returned but, having saved the information contained in NX, LAMDA, NY, MU and WRK, and having adjusted the value of LWRK and the dimension of array WRK accordingly, the user can continue at the point the program was left by calling E02DDF with START = 'W'. Note that the requested value for LWRK is only large enough for the current phase of the algorithm. If the routine is restarted with LWRK set to the minimum value requested, a larger request may be made at a later stage of the computation. See Section 5 for the upper bound on LWRK. On soft failure, the minimum requested value for LWRK is returned in IWRK(1) and the safe value for LWRK is returned in IWRK(2).

If IFAIL = 3,4,5 or 6, a spline approximation is returned, but it fails to satisfy the fitting criterion (see (2) and (3) in Section 3 -- perhaps only by a small amount, however.

7. Accuracy

On successful exit, the approximation returned is such that its weighted sum of squared residuals FP is equal to the smoothing factor S, up to a specified relative tolerance of 0.001 - except that if $n_x=8$ and $n_y=8$, FP may be significantly less than S: in this case the computed spline is simply the least-squares bicubic polynomial approximation of degree 3, i.e., a spline with no interior knots.

8. Further Comments

8.1. Timing

The time taken for a call of E02DDF depends on the complexity of the shape of the data, the value of the smoothing factor S, and the number of data points. If E02DDF is to be called for different values of S, much time can be saved by setting START = It should be noted that choosing S very small considerably increases computation time.

8.2. Choice of S

If the weights have been correctly chosen (see Section 2.1.2 of the Chapter Introduction), the standard deviation of $w_r f_r$ would be the same for all r, equal to (sigma), say. In this case, choosing the smoothing factor S in the range $(\sigma)^2(m \pm \sqrt{2m})$, as suggested by Reinsch [6], is likely to give a good start in the search for a satisfactory value. Otherwise, experimenting with different values of S will be required from the start.

In that case, in view of computation time and memory requirements, it is recommended to start with a very large value for S and so determine the least-squares bicubic polynomial; the value returned for FP, call it FP_0, gives an upper bound for S. Then progressively decrease the value of S to obtain closer fits - say by a factor of 10 in the beginning, i.e., $S=FP_0/10$, $S=FP_0/100$, and so on, and more carefully as the approximation shows more details.

To choose S very small is strongly discouraged. This considerably increases computation time and memory requirements. It may also cause rank-deficiency (as indicated by the parameter RANK) and endanger numerical stability.

The number of knots of the spline returned, and their location, generally depend on the value of S and on the behaviour of the function underlying the data. However, if E02DDF is called with START = 'W', the knots returned may also depend on the smoothing factors of the previous calls. Therefore if, after a number of trials with different values of S and START = 'W', a fit can finally be accepted as satisfactory, it may be worthwhile to call E02DDF once more with the selected value for S but now using START = 'C'. Often, E02DDF then returns an approximation with the same quality of fit but with fewer knots, which is therefore better if data reduction is also important.

8.3. Choice of NXEST and NYEST

The number of knots may also depend on the upper bounds NXEST and NYEST. Indeed, if at a certain stage in E02DDF the number of knots in one direction (say n_x) has reached the value of its upper bound (NXEST), then from that moment on all subsequent knots are added in the other (y) direction. This may indicate that the value of NXEST is too small. On the other hand, it gives the user the option of limiting the number of knots the routine locates in any direction. For example, by setting NXEST = 8 (the lowest allowable value for NXEST), the user can indicate that he wants an approximation which is a simple cubic polynomial in the variable x.

8.4. Restriction of the approximation domain

The fit obtained is not defined outside the rectangle $[(\lambda)_4,(\lambda)_{n_x-3}]*[(\mu)_4,(\mu)_{n_y-3}]$. The reason for taking the extreme data values of x and y for these four knots is that, as is usual in data fitting, the fit cannot be expected to give satisfactory values outside the data region. If, nevertheless, the user requires values over a larger rectangle, this can be achieved by augmenting the data with two artificial data points (a,c,0) and (b,d,0) with zero weight, where [a,b]*[c,d] denotes the enlarged rectangle.

8.5. Outline of method used

First suitable knot sets are built up in stages (starting with no interior knots in the case of a cold start but with the knot set found in a previous call if a warm start is chosen). At each stage, a bicubic spline is fitted to the data by least-squares and (theta), the sum of squares of residuals, is computed. If (theta)>S, a new knot is added to one knot set or the other so as to reduce (theta) at the next stage. The new knot is located in an interval where the fit is particularly poor. Sooner or later, we find that (theta)<=S and at that point the knot sets are accepted. The routine then goes on to compute a spline which has these knot sets and which satisfies the full fitting criterion specified by (2) and (3). The theoretical solution has (theta)=S. The routine computes the spline by an iterative scheme which is ended when (theta)=S within a relative tolerance of 0.001. The main part of each iteration consists of a linear least-squares computation of special form, done in a similarly stable and efficient manner as in E02DAF. As there also, the minimal least-squares solution is computed wherever the linear system is found to be rank-deficient.

An exception occurs when the routine finds at the start that, even with no interior knots (N = 8), the least-squares spline already has its sum of squares of residuals <=S. In this case, since this spline (which is simply a bicubic polynomial) also has an optimal value for the smoothness measure (eta), namely zero, it is returned at once as the (trivial) solution. It will usually mean that S has been chosen too large.

For further details of the algorithm and its use see Dierckx [2].

8.6. Evaluation of computed spline

The values of the computed spline at the points (TX(r),TY(r)), for r = 1,2,...,N, may be obtained in the double precision array FF, of length at least N, by the following code:

```
      IFAIL = 0
      CALL E02DEF(N,NX,NY,TX,TY,LAMDA,MU,C,FF,WRK,IWRK,IFAIL)
```

where NX, NY, LAMDA, MU and C are the output parameters of E02DDF , WRK is a double precision workspace array of length at least NY-4, and IWRK is an integer workspace array of length at least NY-4.

To evaluate the computed spline on a KX by KY rectangular grid of points in the x-y plane, which is defined by the x co-ordinates stored in TX(q), for q=1,2,...,KX, and the y co-ordinates stored in TY(r), for r=1,2,...,KY, returning the results in the double precision array FG which is of length at least KX*KY, the following call may be used:

```
      IFAIL = 0
      CALL E02DFF(KX,KY,NX,NY,TX,TY,LAMDA,MU,C,FG,WRK,LWRK,
     *            IWRK,LIWRK,IFAIL)
```

where NX, NY, LAMDA, MU and C are the output parameters of E02DDF , WRK is a double precision workspace array of length at least LWRK = min(NWRK1,NWRK2), NWRK1 = KX*4+NX, NWRK2 = KY*4+NY, and IWRK is an integer workspace array of length at least LIWRK = KY + NY - 4 if NWRK1 >= NWRK2, or KX + NX - 4 otherwise. The result of the spline evaluated at grid point (q,r) is returned in element (KY*(q-1)+r) of the array FG.

9. Example

This example program reads in a value of M, followed by a set of M data points (x_r, y_r, f_r) and their weights w_r. It then calls E02DDF to compute a bicubic spline approximation for one specified value of S, and prints the values of the computed knots and B-spline coefficients. Finally it evaluates the spline at a small sample of points on a rectangular grid.

```
        The example program is not reproduced here. The source code for
        all example programs is distributed with the NAG Foundation
        Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Calculates values of a bicubic spline from B-spline

— nage.ht —

```
\begin{page}{manpageXXe02def}{NAG Documentation: e02def}
\beginscroll
\begin{verbatim}



     E02DEF(3NAG)      Foundation Library (12/10/92)      E02DEF(3NAG)



        E02 -- Curve and Surface Fitting                           E02DEF
                E02DEF -- NAG Foundation Library Routine Document

        Note: Before using this routine, please read the Users' Note for
        your implementation to check implementation-dependent details.
        The symbol (*) after a NAG routine name denotes a routine that is
        not included in the Foundation Library.

        1. Purpose

        E02DEF calculates values of a bicubic spline from its B-spline
        representation.

        2. Specification

               SUBROUTINE E02DEF (M, PX, PY, X, Y, LAMDA, MU, C, FF, WRK,
              1                   IWRK, IFAIL)
               INTEGER          M, PX, PY, IWRK(PY-4), IFAIL
               DOUBLE PRECISION X(M), Y(M), LAMDA(PX), MU(PY), C((PX-4)*
              1                 (PY-4)), FF(M), WRK(PY-4)

        3. Description
```

This routine calculates values of the bicubic spline $s(x,y)$ at prescribed points (x_r,y_r), for $r=1,2,...,m$, from its augmented knot sets {(lambda)} and {(mu)} and from the coefficients c_{ij}, for $i=1,2,...,PX-4$; $j=1,2,...,PY-4$, in its B-spline representation

$$s(x,y)=\sum_{ij} c_{ij} M_i(x)N_j(y).$$

Here $M_i(x)$ and $N_j(y)$ denote normalised cubic B-splines, the former defined on the knots $(\lambda)_i$ to $(\lambda)_{i+4}$ and the latter on the knots $(\mu)_j$ to $(\mu)_{j+4}$.

This routine may be used to calculate values of a bicubic spline given in the form produced by E01DAF, E02DAF, E02DCF and E02DDF. It is derived from the routine B2VRE in Anthony et al [1].

4. References

[1] Anthony G T, Cox M G and Hayes J G (1982) DASL - Data Approximation Subroutine Library. National Physical Laboratory.

[2] Cox M G (1978) The Numerical Evaluation of a Spline from its B-spline Representation. J. Inst. Math. Appl. 21 135--143.

5. Parameters

1: M -- INTEGER Input
On entry: m, the number of points at which values of the spline are required. Constraint: M >= 1.

2: PX -- INTEGER Input

3: PY -- INTEGER Input
On entry: PX and PY must specify the total number of knots associated with the variables x and y respectively. They are such that PX-8 and PY-8 are the corresponding numbers of interior knots. Constraint: PX >= 8 and PY >= 8.

4: X(M) -- DOUBLE PRECISION array Input

5: Y(M) -- DOUBLE PRECISION array Input

On entry: X and Y must contain x_r and y_r, for r=1,2,...,m, respectively. These are the co-ordinates of the points at which values of the spline are required. The order of the points is immaterial. Constraint: X and Y must satisfy

LAMDA(4) <= X(r) <= LAMDA(PX-3)

and

MU(4) <= Y(r) <= MU(PY-3), for r=1,2,...,m.

The spline representation is not valid outside these intervals.

6: LAMDA(PX) -- DOUBLE PRECISION array Input

7: MU(PY) -- DOUBLE PRECISION array Input
On entry: LAMDA and MU must contain the complete sets of knots {(lambda)} and {(mu)} associated with the x and y variables respectively. Constraint: the knots in each set must be in non-decreasing order, with LAMDA(PX-3) > LAMDA(4) and MU(PY-3) > MU(4).

8: C((PX-4)*(PY-4)) -- DOUBLE PRECISION array Input
On entry: C((PY-4)*(i-1)+j) must contain the coefficient c_{ij} described in Section 3, for i=1,2,...,PX-4; j=1,2,...,PY-4.

9: FF(M) -- DOUBLE PRECISION array Output
On exit: FF(r) contains the value of the spline at the point (x_r,y_r), for r=1,2,...,m.

10: WRK(PY-4) -- DOUBLE PRECISION array Workspace

11: IWRK(PY-4) -- INTEGER array Workspace

12: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL= 1
On entry M < 1,

or PY < 8,

or PX < 8.

IFAIL= 2
On entry the knots in array LAMDA, or those in array MU, are not in non-decreasing order, or LAMDA(PX-3) <= LAMDA(4), or MU(PY-3) <= MU(4).

IFAIL= 3
On entry at least one of the prescribed points (x_r,y_r) lies outside the rectangle defined by LAMDA(4), LAMDA(PX-3) and MU(4), MU(PY-3).

7. Accuracy

The method used to evaluate the B-splines is numerically stable, in the sense that each computed value of $s(x_r,y_r)$ can be regarded as the value that would have been obtained in exact arithmetic from slightly perturbed B-spline coefficients. See Cox [2] for details.

8. Further Comments

Computation time is approximately proportional to the number of points, m, at which the evaluation is required.

9. Example

This program reads in knot sets LAMDA(1),..., LAMDA(PX) and MU(1) ,..., MU(PY), and a set of bicubic spline coefficients c_{ij}. Following these are a value for m and the co-ordinates (x_r,y_r), for r=1,2,...,m, at which the spline is to be evaluated.

The example program is not reproduced here. The source code for all example programs is distributed with the NAG Foundation Library software and should be available on-line.

\end{verbatim}

```
\endscroll
\end{page}
```

———-

Calculates values of a bicubic spline from B-spline

— **nage.ht** —

```
\begin{page}{manpageXXe02dff}{NAG Documentation: e02dff}
\beginscroll
\begin{verbatim}


     E02DFF(3NAG)      Foundation Library (12/10/92)      E02DFF(3NAG)



          E02 -- Curve and Surface Fitting                           E02DFF
                  E02DFF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          E02DFF calculates values of a bicubic spline from its B-spline
          representation. The spline is evaluated at all points on a
          rectangular grid.

          2. Specification

                 SUBROUTINE E02DFF (MX, MY, PX, PY, X, Y, LAMDA, MU, C, FF,
                1                   WRK, LWRK, IWRK, LIWRK, IFAIL)
                 INTEGER          MX, MY, PX, PY, LWRK, IWRK(LIWRK), LIWRK,
                1                 IFAIL
                 DOUBLE PRECISION X(MX), Y(MY), LAMDA(PX), MU(PY), C((PX-4)*
                1                 (PY-4)), FF(MX*MY), WRK(LWRK)

          3. Description

          This routine calculates values of the bicubic spline s(x,y) on a
          rectangular grid of points in the x-y plane, from its augmented
          knot sets {(lambda)} and {(mu)} and from the coefficients c   ,
```

for i=1,2,...,PX-4; j=1,2,...,PY-4, in its B-spline representation

$$s(x,y)=\sum_{ij} c_{ij} M_i(x)N_j(y).$$

Here $M_i(x)$ and $N_j(y)$ denote normalised cubic B-splines, the former defined on the knots $(\lambda)_i$ to $(\lambda)_{i+4}$ and the latter on the knots $(\mu)_j$ to $(\mu)_{j+4}$.

The points in the grid are defined by co-ordinates x_q, for $q=1,2,...,m_x$, along the x axis, and co-ordinates y_r, for $r=1,2,...,m_y$ along the y axis.

This routine may be used to calculate values of a bicubic spline given in the form produced by E01DAF, E02DAF, E02DCF and E02DDF. It is derived from the routine B2VRE in Anthony et al [1].

4. References

[1] Anthony G T, Cox M G and Hayes J G (1982) DASL - Data Approximation Subroutine Library. National Physical Laboratory.

[2] Cox M G (1978) The Numerical Evaluation of a Spline from its B-spline Representation. J. Inst. Math. Appl. 21 135--143.

5. Parameters

1: MX -- INTEGER Input

2: MY -- INTEGER Input
On entry: MX and MY must specify m_x and m_y respectively, the number of points along the x and y axis that define the rectangular grid. Constraint: MX >= 1 and MY >= 1.

3: PX -- INTEGER Input

4: PY -- INTEGER Input
On entry: PX and PY must specify the total number of knots

associated with the variables x and y respectively. They are such that PX-8 and PY-8 are the corresponding numbers of interior knots. Constraint: PX >= 8 and PY >= 8.

5: X(MX) -- DOUBLE PRECISION array Input

6: Y(MY) -- DOUBLE PRECISION array Input

On entry: X and Y must contain x_q, for $q=1,2,...,m_x$, and y_r, for $r=1,2,...,m_y$, respectively. These are the x and y co-ordinates that define the rectangular grid of points at which values of the spline are required. Constraint: X and Y must satisfy

LAMDA(4) <= X(q) < X(q+1) <= LAMDA(PX-3), for $q=1,2,...,m_x-1$

and

MU(4) <= Y(r) < Y(r+1) <= MU(PY-3), for $r=1,2,...,m_y-1$.

The spline representation is not valid outside these intervals.

7: LAMDA(PX) -- DOUBLE PRECISION array Input

8: MU(PY) -- DOUBLE PRECISION array Input

On entry: LAMDA and MU must contain the complete sets of knots {(lambda)} and {(mu)} associated with the x and y variables respectively. Constraint: the knots in each set must be in non-decreasing order, with LAMDA(PX-3) > LAMDA(4) and MU(PY-3) > MU(4).

9: C((PX-4)*(PY-4)) -- DOUBLE PRECISION array Input

On entry: C((PY-4)*(i-1)+j) must contain the coefficient c_{ij} described in Section 3, for i=1,2,...,PX-4; j=1,2,...,PY-4.

10: FF(MX*MY) -- DOUBLE PRECISION array Output

On exit: FF(MY*(q-1)+r) contains the value of the spline at the point (x_q,y_r), for $q=1,2,...,m_x$; $r=1,2,...,m_y$.

11: WRK(LWRK) -- DOUBLE PRECISION array Workspace

12: LWRK -- INTEGER Input

On entry:
the dimension of the array WRK as declared in the

(sub)program from which E02DFF is called.
Constraint: LWRK >= min(NWRK1,NWRK2), where NWRK1=4*MX+PX, NWRK2=4*MY+PY.

13: IWRK(LIWRK) -- INTEGER array Workspace

14: LIWRK -- INTEGER Input
On entry:
the dimension of the array IWRK as declared in the (sub)program from which E02DFF is called.
Constraint: LIWRK >= MY + PY - 4 if NWRK1 > NWRK2, or MX + PX - 4 otherwise, where NWRK1 and NWRK2 are as defined in the description of argument LWRK.

15: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL= 1
On entry MX < 1,

or MY < 1,

or PY < 8,

or PX < 8.

IFAIL= 2
On entry LWRK is too small,

or LIWRK is too small.

IFAIL= 3
On entry the knots in array LAMDA, or those in array MU, are not in non-decreasing order, or LAMDA(PX-3) <= LAMDA(4), or MU(PY-3) <= MU(4).

IFAIL= 4
On entry the restriction LAMDA(4) <= X(1) <... < X(MX) <=

```
     LAMDA(PX-3), or the restriction MU(4) <= Y(1) <... < Y(MY)
     <= MU(PY-3), is violated.

7. Accuracy

The method used to evaluate the B-splines is numerically stable,
in the sense that each computed value of s(x ,y ) can be regarded
                                            r  r
as the value that would have been obtained in exact arithmetic
from slightly perturbed B-spline coefficients. See Cox [2] for
details.

8. Further Comments

Computation time is approximately proportional to m m +4(m +m ).
                                                   x y    x  y

9. Example

This program reads in knot sets LAMDA(1),..., LAMDA(PX) and MU(1)
,..., MU(PY), and a set of bicubic spline coefficients c  .
                                                        ij
Following these are values for m  and the x co-ordinates x , for
                                x                         q
q=1,2,...,m , and values for m  and the y co-ordinates y , for
           x                  y                         r
r=1,2,...,m , defining the grid of points on which the spline is
           y
to be evaluated.

The example program is not reproduced here. The source code for
all example programs is distributed with the NAG Foundation
Library software and should be available on-line.
```

```
\end{verbatim}
\endscroll
\end{page}
```

———-

Calculates l_1 solution to over-determined system equations

— nage.ht —

```
\begin{page}{manpageXXe02gaf}{NAG Documentation: e02gaf}
\beginscroll
\begin{verbatim}
```

E02 -- Curve and Surface Fitting E02GAF
E02GAF -- NAG Foundation Library Routine Document

Note: Before using this routine, please read the Users' Note for your implementation to check implementation-dependent details. The symbol (*) after a NAG routine name denotes a routine that is not included in the Foundation Library.

1. Purpose

E02GAF calculates an l_1 solution to an over-determined system of linear equations.

2. Specification

```
       SUBROUTINE E02GAF (M, A, LA, B, NPLUS2, TOLER, X, RESID,
      1                   IRANK, ITER, IWORK, IFAIL)
       INTEGER          M, LA, NPLUS2, IRANK, ITER, IWORK(M),
      1                 IFAIL
       DOUBLE PRECISION A(LA,NPLUS2), B(M), TOLER, X(NPLUS2),
      1                 RESID
```

3. Description

Given a matrix A with m rows and n columns (m>=n) and a vector b with m elements, the routine calculates an l_1 solution to the over-determined system of equations

$$Ax=b.$$

That is to say, it calculates a vector x, with n elements, which minimizes the l_1-norm (the sum of the absolute values) of the residuals

$$r(x)=\sum_{i=1}^{m}|r_i|,$$

where the residuals r_i are given by

$$r_i = b_i - \sum_{j=1}^{n} a_{ij} x_j , \quad i=1,2,\ldots,m.$$

Here a_{ij} is the element in row i and column j of A, b_i is the ith element of b and x_j the jth element of x. The matrix A need not be of full rank.

Typically in applications to data fitting, data consisting of m points with co-ordinates (t_i,y_i) are to be approximated in the l_1-norm by a linear combination of known functions $\phi_j(t)$,

$$\alpha_1 \phi_1(t)+\alpha_2 \phi_2(t)+\ldots+\alpha_n \phi_n(t).$$

This is equivalent to fitting an l_1 solution to the over-determined system of equations

$$\sum_{j=1}^{n} \phi_j(t_i)\alpha_j = y_i , \quad i=1,2,\ldots,m.$$

Thus if, for each value of i and j, the element a_{ij} of the matrix A in the previous paragraph is set equal to the value of $\phi_j(t_i)$ and b_i is set equal to y_i, the solution vector x will contain the required values of the α_j. Note that the independent variable t above can, instead, be a vector of several independent variables (this includes the case where each ϕ_i is a function of a different variable, or set of variables).

The algorithm is a modification of the simplex method of linear programming applied to the primal formulation of the l_1 problem (see Barrodale and Roberts [1] and [2]). The modification allows

```
several neighbouring simplex vertices to be passed through in a
single iteration, providing a substantial improvement in
efficiency.

4. References

[1]   Barrodale I and Roberts F D K (1973) An Improved Algorithm
      for Discrete \\ll  Linear Approximation. SIAM J. Numer.
                        1
      Anal. 10 839--848.

[2]   Barrodale I and Roberts F D K (1974) Solution of an
      Overdetermined System of Equations in the \\ll -norm. Comm.
                                                     1
      ACM. 17, 6 319--320.

5. Parameters

 1:  M -- INTEGER                                          Input
     On entry: the number of equations, m (the number of rows of
     the matrix A). Constraint: M >= n >= 1.

 2:  A(LA,NPLUS2) -- DOUBLE PRECISION array          Input/Output
     On entry: A(i,j) must contain a  , the element in the ith
                                    ij
     row and jth column of the matrix A, for i=1,2,...,m and
     j=1,2,...,n. The remaining elements need not be set. On
     exit: A contains the last simplex tableau generated by the
     simplex method.

 3:  LA -- INTEGER                                         Input
     On entry:
     the first dimension of the array A as declared in the
     (sub)program from which E02GAF is called.
     Constraint: LA >= M + 2.

 4:  B(M) -- DOUBLE PRECISION array                  Input/Output
     On entry: b , the ith element of the vector b, for
                i
     i=1,2,...,m. On exit: the ith residual r  corresponding to
                                            i
     the solution vector x, for i=1,2,...,m.

 5:  NPLUS2 -- INTEGER                                     Input
     On entry: n+2, where n is the number of unknowns (the
     number of columns of the matrix A). Constraint: 3 <= NPLUS2
     <= M + 2.

 6:  TOLER -- DOUBLE PRECISION                             Input
     On entry: a non-negative value. In general TOLER specifies
```

```
     a threshold below which numbers are regarded as zero. The
                                                        2/3
     recommended threshold value is (epsilon)    where (epsilon)
     is the machine precision. The recommended value can be
     computed within the routine by setting TOLER to zero. If
     premature termination occurs a larger value for TOLER may
     result in a valid solution. Suggested value: 0.0.

 7:  X(NPLUS2) -- DOUBLE PRECISION array                   Output
     On exit: X(j) contains the jth element of the solution
     vector x, for j=1,2,...,n. The elements X(n+1) and X(n+2)
     are unused.

 8:  RESID -- DOUBLE PRECISION                             Output
     On exit: the sum of the absolute values of the residuals
     for the solution vector x.

 9:  IRANK -- INTEGER                                      Output
     On exit: the computed rank of the matrix A.

10:  ITER -- INTEGER                                       Output
     On exit: the number of iterations taken by the simplex
     method.

11:  IWORK(M) -- INTEGER array                          Workspace

12:  IFAIL -- INTEGER                                Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1
     An optimal solution has been obtained but this may not be
     unique.

IFAIL= 2
     The calculations have terminated prematurely due to rounding
     errors. Experiment with larger values of TOLER or try
     scaling the columns of the matrix (see Section 8).

IFAIL= 3
     On entry NPLUS2 < 3,
```

or NPLUS2 > M + 2,

or LA < M + 2.

7. Accuracy

Experience suggests that the computational accuracy of the solution x is comparable with the accuracy that could be obtained by applying Gaussian elimination with partial pivoting to the n equations satisfied by this algorithm (i.e., those equations with zero residuals). The accuracy therefore varies with the conditioning of the problem, but has been found generally very satisfactory in practice.

8. Further Comments

The effects of m and n on the time and on the number of iterations in the Simplex Method vary from problem to problem, but typically the number of iterations is a small multiple of n and the total time taken by the routine is approximately proportional to mn^2.

It is recommended that, before the routine is entered, the columns of the matrix A are scaled so that the largest element in each column is of the order of unity. This should improve the conditioning of the matrix, and also enable the parameter TOLER to perform its correct function. The solution x obtained will then, of course, relate to the scaled form of the matrix. Thus if the scaling is such that, for each j=1,2,...,n, the elements of the jth column are multiplied by the constant k_j, the element x_j of the solution vector x must be multiplied by k_j if it is desired to recover the solution corresponding to the original matrix A.

9. Example

Suppose we wish to approximate a set of data by a curve of the form

$$y=Ke^t+Le^{-t}+M$$

where K, L and M are unknown. Given values y_i at 5 points t_i we may form the over-determined set of equations for K, L and M

x -x

```
                          i     i
                        e  K+e   L+M=y ,  i=1,2,...,5.
                                      i

          E02GAF is used to solve these in the l  sense.
                                               1

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Sorts two-dimensional data into rectangular panels

— nage.ht —

```
\begin{page}{manpageXXe02zaf}{NAG Documentation: e02zaf}
\beginscroll
\begin{verbatim}



     E02ZAF(3NAG)      Foundation Library (12/10/92)      E02ZAF(3NAG)



          E02 -- Curve and Surface Fitting                           E02ZAF
                  E02ZAF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          E02ZAF sorts two-dimensional data into rectangular panels.

          2. Specification

                 SUBROUTINE E02ZAF (PX, PY, LAMDA, MU, M, X, Y, POINT,
                1                   NPOINT, ADRES, NADRES, IFAIL)
```

```
       INTEGER          PX, PY, M, POINT(NPOINT), NPOINT, ADRES
      1                 (NADRES), NADRES, IFAIL
       DOUBLE PRECISION LAMDA(PX), MU(PY), X(M), Y(M)
```

3. Description

A set of m data points with rectangular Cartesian co-ordinates x_r, y_r are sorted into panels defined by lines parallel to the y and x axes. The intercepts of these lines on the x and y axes are given in LAMDA(i), for i=5,6,...,PX-4 and MU(j), for j=5,6,...,PY-4, respectively. The subroutine orders the data so that all points in a panel occur before data in succeeding panels, where the panels are numbered from bottom to top and then left to right, with the usual arrangement of axes, as shown in the diagram. Within a panel the points maintain their original order.

Please see figure in printed Reference Manual

A data point lying exactly on one or more panel sides is taken to be in the highest-numbered panel adjacent to the point. The subroutine does not physically rearrange the data, but provides the array POINT which contains a linked list for each panel, pointing to the data in that panel. The total number of panels is (PX-7)*(PY-7).

4. References

None.

5. Parameters

1: PX -- INTEGER Input

2: PY -- INTEGER Input

On entry: PX and PY must specify eight more than the number of intercepts on the x axis and y axis, respectively. Constraint: PX >= 8 and PY >= 8.

3: LAMDA(PX) -- DOUBLE PRECISION array Input
On entry: LAMDA(5) to LAMDA(PX-4) must contain, in non-decreasing order, the intercepts on the x axis of the sides of the panels parallel to the y axis.

4: MU(PY) -- DOUBLE PRECISION array Input
On entry: MU(5) to MU(PY-4) must contain, in non-decreasing order, the intercepts on the y axis of the sides of the panels parallel to the x axis.

5: M -- INTEGER Input
On entry: the number m of data points.

6: X(M) -- DOUBLE PRECISION array Input

7: Y(M) -- DOUBLE PRECISION array Input
On entry: the co-ordinates of the rth data point (x_r, y_r), for r=1,2,...,m.

8: POINT(NPOINT) -- INTEGER array Output
On exit: for i = 1,2,...,NADRES, POINT(m+i) = I1 is the index of the first point in panel i, POINT(I1) = I2 is the index of the second point in panel i and so on.

POINT(IN) = 0 indicates that X(IN),Y(IN) was the last point in the panel.

The co-ordinates of points in panel i can be accessed in turn by means of the following instructions:

```
   IN = M + I
10 IN = POINT(IN)
   IF (IN.EQ. 0) GOTO 20
   XI = X(IN)
   YI = Y(IN)
   .
   .
   .
   GOTO 10
20...
```

9: NPOINT -- INTEGER Input
On entry:
the dimension of the array POINT as declared in the (sub)program from which E02ZAF is called.
Constraint: NPOINT >= M + (PX-7)*(PY-7).

10: ADRES(NADRES) -- INTEGER array Workspace

11: NADRES -- INTEGER Input
On entry: the value (PX-7)*(PY-7), the number of panels into which the (x,y) plane is divided.

12: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see

```
     Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

IFAIL= 1
     The intercepts in the array LAMDA, or in the array MU, are
     not in non-decreasing order.

IFAIL= 2
     On entry PX < 8,

     or       PY < 8,

     or       M <= 0,

     or       NADRES /= (PX-7)*(PY-7),

     or       NPOINT < M + (PX-7)*(PY-7).

7. Accuracy

Not applicable.

8. Further Comments

The time taken by this routine is approximately proportional to
m*log(NADRES).

This subroutine was written to sort two dimensional data in the
manner required by routines E02DAF and E02DBF(*). The first 9
parameters of E02ZAF are the same as the parameters in E02DAF and
E02DBF(*) which have the same name.

9. Example

This example program reads in data points and the intercepts of
the panel sides on the x and y axes; it calls E02ZAF to set up
the index array POINT; and finally it prints the data points in
panel order.

The example program is not reproduced here. The source code for
all example programs is distributed with the NAG Foundation
Library software and should be available on-line.

\end{verbatim}
```

```
\endscroll
\end{page}
```

———-

Minimizing or Maximizing a Function

— nage.ht —

```
\begin{page}{manpageXXe04}{NAG Documentation: e04}
\beginscroll
\begin{verbatim}


     E04(3NAG)         Foundation Library (12/10/92)         E04(3NAG)



          E04 -- Minimizing or Maximizing a Function    Introduction -- E04
                                    Chapter E04
                        Minimizing or Maximizing a Function

          Contents of this Introduction:

          1.     Scope of the Chapter

          2.     Background to the Problems

          2.1.   Types of Optimization Problems

          2.1.1. Unconstrained minimization

          2.1.2. Nonlinear least-squares problems

          2.1.3. Minimization subject to bounds on the variables

          2.1.4. Minimization subject to linear constraints

          2.1.5. Minimization subject to nonlinear constraints

          2.2.   Geometric Representation and Terminology

          2.2.1. Gradient vector

          2.2.2. Hessian matrix
```

1. Scope of the Chapter

An optimization problem involves minimizing a function (called the objective function) of several variables, possibly subject to restrictions on the values of the variables defined by a set of constraint functions. The routines in the NAG Foundation Library are concerned with function minimization only, since the problem of maximizing a given function can be transformed into a minimization problem simply by multiplying the function by -1.

This introduction is only a brief guide to the subject of optimization designed for the casual user. Anyone with a difficult or protracted problem to solve will find it beneficial to consult a more detailed text, such as Gill et al [5] or Fletcher [3].

Readers who are unfamiliar with the mathematics of the subject may find some sections difficult at first reading; if so, they should concentrate on Sections 2.1, 2.2, 2.5, 2.6 and 3.

2. Background to the Problems

2.1. Types of Optimization Problems

Solution of optimization problems by a single, all-purpose, method is cumbersome and inefficient. Optimization problems are therefore classified into particular categories, where each category is defined by the properties of the objective and constraint functions, as illustrated by some examples below.

Properties of Objective Function	Properties of Constraints
Nonlinear	Nonlinear
Sums of squares of nonlinear functions	Sparse linear
Quadratic	Linear
Sums of squares of linear functions	Bounds
Linear	None

For instance, a specific problem category involves the minimization of a nonlinear objective function subject to bounds on the variables. In the following sections we define the particular categories of problems that can be solved by routines contained in this Chapter.

2.1.1. Unconstrained minimization

In unconstrained minimization problems there are no constraints on the variables. The problem can be stated mathematically as follows:

$$\min_{x} F(x)$$

where x is in R^n, that is, $x=(x_1,x_2,\ldots,x_n)^T$.

2.1.2. Nonlinear least-squares problems

Special consideration is given to the problem for which the function to be minimized can be expressed as a sum of squared functions. The least-squares problem can be stated mathematically as follows:

$$\min_{x} \left\{ f^T f = \sum_{i=1}^{m} f_i^2(x) \right\}, \quad x \text{ is in } R^n$$

where the ith element of the m-vector f is the function $f_i(x)$.

2.1.3. Minimization subject to bounds on the variables

These problems differ from the unconstrained problem in that at least one of the variables is subject to a simple restriction on its value, e.g. $x_5 \le 10$, but no constraints of a more general form are present.

The problem can be stated mathematically as follows:

$$\min_{x} F(x), \quad x \text{ is in } R^n$$

subject to $l_i \le x_i \le u_i$, $i=1,2,\ldots,n$.

This format assumes that upper and lower bounds exist on all the variables. By conceptually allowing u_i=infty and l_i=-infty all the variables need not be restricted.

2.1.4. Minimization subject to linear constraints

A general linear constraint is defined as a constraint function that is linear in more than one of the variables, e.g. $3x_1+2x_2>=4$ The various types of linear constraint are reflected in the following mathematical statement of the problem:

$$\underset{x}{\text{minimize}}\ F(x), \quad x \text{ is in } R^n$$

subject to the

equality constraints:	$a_i^T x=b_i$	$i=1,2,\ldots,m_1$;
inequality constraints:	$a_i^T x>=b_i$	$i=m_1+1,m_1+2,\ldots,m_2$;
	$a_i^T x<=b_i$	$i=m_2+1,m_2+2,\ldots,m_3$;
range constraints:	$s_j<=a_i^T x<=t_j$	$i=m_3+1,m_3+2,\ldots,m_4$; $j=1,2,\ldots,m_4-m_3$;
bounds constraints:	$l_i<=x_i<=u_i$	$i=1,2,\ldots,n$

where each a_i is a vector of length n; b_i, s_j and t_j are constant scalars; and any of the categories may be empty.

Although the bounds on x_i could be included in the definition of general linear constraints, we prefer to distinguish between them for reasons of computational efficiency.

If F(x) is a linear function, the linearly-constrained problem is termed a linear programming problem (LP problem); if F(x) is a quadratic function, the problem is termed a quadratic programming problem (QP problem). For further discussion of LP and QP problems, including the dual formulation of such problems, see Dantzig [2].

2.1.5. Minimization subject to nonlinear constraints

A problem is included in this category if at least one constraint function is nonlinear, e.g. $x_1^2+x_3+x_4-2>=0$. The mathematical statement of the problem is identical to that for the linearly-constrained case, except for the addition of the following constraints:

equality constraints:	$c_i(x)=0$	$i=1,2,...,m_5$;
inequality constraints:	$c_i(x)>=0$	$i=m_5+1,m_5+2,...,m_6$;
range constraints:	$v_j<=c_i(x)<=w_j$	$i=m_6+1,m_6+2,...,m_7$, $j=1,2,...,m_7-m_6$

where each c_i is a nonlinear function; v_j and w_j are constant scalars; and any category may be empty. Note that we do not include a separate category for constraints of the form $c_i(x)<=0$, since this is equivalent to $-c_i(x)>=0$.

2.2. Geometric Representation and Terminology

To illustrate the nature of optimization problems it is useful to consider the following example in two dimensions

$$F(x)=e^{x_1}(4x_1^2+2x_2^2+4x_1x_2+2x_2+1).$$

(This function is used as the example function in the documentation for the unconstrained routines.)

Figure 1

Please see figure in printed Reference Manual

Figure 1 is a contour diagram of $F(x)$. The contours labelled $F_0, F_1, \ldots, F_4$ are isovalue contours, or lines along which the function $F(x)$ takes specific constant values. The point x^* is a local unconstrained minimum, that is, the value of $F(x^*)$ is less than at all the neighbouring points. A function may have several such minima. The lowest of the local minima is termed a global minimum. In the problem illustrated in Figure 1, x^* is the only local minimum. The point x is said to be a saddle point because it is a minimum along the line AB, but a maximum along CD.

If we add the constraint $x_1 >= 0$ to the problem of minimizing $F(x)$, the solution remains unaltered. In Figure 1 this constraint is represented by the straight line passing through $x_1 = 0$, and the shading on the line indicates the unacceptable region. The region in R^n satisfying the constraints of an optimization problem is termed the feasible region. A point satisfying the constraints is defined as a feasible point.

If we add the nonlinear constraint $x_1 + x_2 - x_1 x_2 - 1.5 >= 0$, represented by the curved shaded line in Figure 1, then x^* is not a feasible point. The solution of the new constrained problem is $\hat{x}$, the feasible point with the smallest function value.

2.2.1. Gradient vector

The vector of first partial derivatives of $F(x)$ is called the gradient vector, and is denoted by $g(x)$, i.e.,

$$g(x) = \left[\frac{dF(x)}{dx_1}, \frac{dF(x)}{dx_2}, \ldots, \frac{dF(x)}{dx_n} \right]^T .$$

For the function illustrated in Figure 1,

[x]

$$g(x)=\begin{bmatrix} F(x)+e^{x_1}(8x_1+4x_2) \\ e^{x_1}(4x_2+4x_1+2) \end{bmatrix}.$$

The gradient vector is of importance in optimization because it must be zero at an unconstrained minimum of any function with continuous first derivatives.

2.2.2. Hessian matrix

The matrix of second partial derivatives of a function is termed its Hessian matrix. The Hessian matrix of $F(x)$ is denoted by $G(x)$ and its (i,j)th element is given by $\partial^2 F(x)/\partial x_i \partial x_j$. If $F(x)$ has continuous second derivatives, then $G(x)$ must be positive semi-definite at any unconstrained minimum of F.

2.2.3. Jacobian matrix; matrix of constraint normals

In nonlinear least-squares problems, the matrix of first partial derivatives of the vector-valued function $f(x)$ is termed the Jacobian matrix of $f(x)$ and its (i,j)th component is $\partial f_i/\partial x_j$.

The vector of first partial derivatives of the constraint $c_i(x)$ is denoted by

$$a_i(x)=\left[\frac{\partial c_i(x)}{\partial x_1},\ldots,\frac{\partial c_i(x)}{\partial x_n}\right]^T.$$

At a point, $\hat{x}$, the vector $a_i(\hat{x})$ is orthogonal (normal) to the isovalue contour of $c_i(x)$ passing through $\hat{x}$; this relationship is illustrated for a two-dimensional function in Figure 2.

Figure 2
Please see figure in printed Reference Manual

The matrix whose columns are the vectors $\{a_i\}$ is termed the matrix of constraint normals. Note that if $c_i(x)$ is a linear constraint involving $a_i^T x$, then its vector of first partial derivatives is simply the vector a_i.

2.3. Sufficient Conditions for a Solution

All nonlinear functions will be assumed to have continuous second derivatives in the neighbourhood of the solution.

2.3.1. Unconstrained minimization

The following conditions are sufficient for the point x^* to be an unconstrained local minimum of $F(x)$:

(i) $|||g(x^*)|||=0$; and

(ii) $G(x^*)$ is positive-definite,

where $|||g|||$ denotes the Euclidean length of g.

2.3.2. Minimization subject to bounds on the variables

At the solution of a bounds-constrained problem, variables which are not on their bounds are termed free variables. If it is known in advance which variables are on their bounds at the solution, the problem can be solved as an unconstrained problem in just the free variables; thus, the sufficient conditions for a solution are similar to those for the unconstrained case, applied only to the free variables.

Sufficient conditions for a feasible point x^* to be the solution of a bound-constrained problem are as follows:

(i) $|||g(x^*)|||=0$; and

(ii) $G(x^*)$ is positive-definite; and

(iii) $g_j(x^*)<0, x_j=u_j$; $g_j(x^*)>0, x_j=l_j$,

where g(x) is the gradient of F(x) with respect to the free variables, and G(x) is the Hessian matrix of F(x) with respect to the free variables. The extra condition (iii) ensures that F(x) cannot be reduced by moving off one or more of the bounds.

2.3.3. Linearly-constrained minimization

For the sake of simplicity, the following description does not include a specific treatment of bounds or range constraints, since the results for general linear inequality constraints can be applied directly to these cases.

At a solution x^*, of a linearly-constrained problem, the constraints which hold as equalities are called the active or binding constraints. Assume that there are t active constraints at the solution x^*, and let $\hat{A}$ denote the matrix whose columns are the columns of A corresponding to the active constraints, with $\hat{b}$ the vector similarly obtained from b; then

$$\hat{A}^T x^* = \hat{b}.$$

The matrix Z is defined as an n by (n-t) matrix satisfying:

$$\hat{A}^T Z=0; \quad Z^T Z=I.$$

The columns of Z form an orthogonal basis for the set of vectors orthogonal to the columns of $\hat{A}$.

Define

$g_z(x)=Z^T g(x)$, the projected gradient vector of F(x);

$G_z(x)=Z^T G(x)Z$, the projected Hessian matrix of F(x).

z

At the solution of a linearly-constrained problem, the projected gradient vector must be zero, which implies that the gradient vector $g(x^*)$ can be written as a linear combination of the columns of $\hat{A}$, i.e., $g(x^*)=\sum_{i=1}^{t} (\lambda)_i \hat{a}_i=\hat{A}(\lambda)$. The scalar $(\lambda)_i$ is defined as the Lagrange multiplier corresponding to the ith active constraint. A simple interpretation of the ith Lagrange multiplier is that it gives the gradient of F(x) along the ith active constraint normal; a convenient definition of the Lagrange multiplier vector (although not a recommended method for computation) is:

$$(\lambda)=(\hat{A}^T\hat{A})^{-1}\hat{A}^T g(x^*).$$

Sufficient conditions for x^* to be the solution of a linearly-constrained problem are:

(i) x^* is feasible, and $\hat{A}^T x^*=\hat{b}$; and

(ii) $\|g_z(x^*)\|=0$, or equivalently, $g(x^*)=\hat{A}(\lambda)$; and

(iii) $G_z(x^*)$ is positive-definite; and

(iv) $(\lambda)_i>0$ if $(\lambda)_i$ corresponds to a constraint $\hat{a}_i^T x^*>=\hat{b}_i$;

$(\lambda)_i<0$ if $(\lambda)_i$ corresponds to a constraint $\hat{a}_i^T x^*<=\hat{b}_i$.

The sign of $(\lambda)_i$ is immaterial for equality constraints, which by definition are always active.

2.3.4. Nonlinearly-constrained minimization

For nonlinearly-constrained problems, much of the terminology is defined exactly as in the linearly-constrained case. The set of active constraints at x again means the set of constraints that hold as equalities at x, with corresponding definitions of $\hat{c}$ and $\hat{A}$: the vector $\hat{c}(x)$ contains the active constraint functions, and the columns of $\hat{A}(x)$ are the gradient vectors of the active constraints. As before, Z is defined in terms of $\hat{A}(x)$ as a matrix such that:

$$\hat{A}^T Z=0; \quad Z^T Z=I$$

where the dependence on x has been suppressed for compactness.

The projected gradient vector $g_z(x)$ is the vector $Z^T g(x)$. At the solution x^* of a nonlinearly-constrained problem, the projected gradient must be zero, which implies the existence of Lagrange multipliers corresponding to the active constraints, i.e., $g(x^*)=\hat{A}(x^*)(\lambda)$.

The Lagrangian function is given by:

$$L(x,(\lambda))=F(x)-(\lambda)^T\hat{c}(x).$$

We define $g_L(x)$ as the gradient of the Lagrangian function; $G_L(x)$ as its Hessian matrix, and $\hat{G}_L(x)$ as its projected Hessian matrix, i.e., $\hat{G}_L=Z^T G_L Z$.

*

Sufficient conditions for x^* to be a solution of nonlinearly-constrained problem are:

(i) x^* is feasible, and $\hat{c}(x^*)=0$; and

(ii) $\|g_z(x^*)\|=0$, or, equivalently, $g(x^*)=\hat{A}(x^*)\lambda^*$; and

(iii) $\hat{G}_L(x^*)$ is positive-definite; and

(iv) $\lambda_i>0$ if λ_i corresponds to a constraint of the form $\hat{c}_i>=0$; the sign of λ_i is immaterial for an equality constraint.

Note that condition (ii) implies that the projected gradient of the Lagrangian function must also be zero at x^*, since the application of Z^T annihilates the matrix $\hat{A}(x^*)$.

2.4. Background to Optimization Methods

All the algorithms contained in this Chapter generate an iterative sequence $\{x^{(k)}\}$ that converges to the solution x^* in the limit, except for some special problem categories (i.e., linear and quadratic programming). To terminate computation of the sequence, a convergence test is performed to determine whether the current estimate of the solution is an adequate approximation. The convergence tests are discussed in Section 2.6

Most of the methods construct a sequence $\{x^{(k)}\}$ satisfying:

$$x^{(k+1)}=x^{(k)}+\alpha^{(k)}p^{(k)},$$

where the vector $p^{(k)}$ is termed the direction of search, and $\alpha^{(k)}$ is the steplength. The steplength $\alpha^{(k)}$ is chosen so that $F(x^{(k+1)})<F(x^{(k)})$.

2.4.1. Methods for unconstrained optimization

The distinctions among methods arise primarily from the need to use varying levels of information about derivatives of F(x) in defining the search direction. We describe three basic approaches to unconstrained problems, which may be extended to other problem categories. Since a full description of the methods would fill several volumes, the discussion here can do little more than allude to the processes involved, and direct the reader to other sources for a full explanation.

(a) Newton-type Methods (Modified Newton Methods)

Newton-type methods use the Hessian matrix $G(x^{(k)})$, or a finite difference approximation to $G(x^{(k)})$, to define the search direction. The routines in the Library either require a subroutine that computes the elements of $G(x^{(k)})$, or they approximate $G(x^{(k)})$ by finite differences.

Newton-type methods are the most powerful methods available for general problems and will find the minimum of a quadratic function in one iteration. See Sections 4.4. and 4.5.1. of Gill et al [5].

(b) Quasi-Newton Methods

Quasi-Newton methods approximate the Hessian $G(x^{(k)})$ by a matrix $B^{(k)}$ which is modified at each iteration to include information obtained about the curvature of F along the latest search direction. Although not as robust as Newton-type methods, quasi-Newton methods can be more efficient because $G(x^{(k)})$ is not computed, or approximated by finite-differences. Quasi-Newton methods minimize a quadratic function in n iterations. See Section 4.5.2 of Gill et al [5].

(c) Conjugate-Gradient Methods

Unlike Newton-type and quasi-Newton methods, conjugate gradient methods do not require the storage of an n by n matrix and so are ideally suited to solve large problems.

Conjugate-gradient type methods are not usually as reliable or efficient as Newton-type, or quasi-Newton methods. See Section 4.8.3 of Gill et al [5].

2.4.2. Methods for nonlinear least-squares problems

These methods are similar to those for unconstrained optimization, but exploit the special structure of the Hessian matrix to give improved computational efficiency.

Since

$$F(x)=\sum_{i=1}^{m} f_i^2(x)$$

the Hessian matrix G(x) is of the form

$$G(x)=2[J(x)^T J(x)+\sum_{i=1}^{m} f_i(x)G_i(x)],$$

where J(x) is the Jacobian matrix of f(x), and $G_i(x)$ is the Hessian matrix of $f_i(x)$.

In the neighbourhood of the solution, $\|f(x)\|$ is often small compared to $\|J(x)^T J(x)\|$ (for example, when f(x) represents the goodness of fit of a nonlinear model to observed data). In such cases, $2J(x)^T J(x)$ may be an adequate approximation to G(x), thereby avoiding the need to compute or approximate second derivatives of $\{f_i(x)\}$. See Section 4.7 of Gill et al [5].

2.4.3. Methods for handling constraints

Bounds on the variables are dealt with by fixing some of the variables on their bounds and adjusting the remaining free variables to minimize the function. By examining estimates of the Lagrange multipliers it is possible to adjust the set of variables fixed on their bounds so that eventually the bounds active at the solution should be correctly identified. This type of method is called an active set method. One feature of such methods is that, given an initial feasible point, all approximations $x^{(k)}$ are feasible. This approach can be extended to general linear constraints. At a point, x, the set of constraints which hold as equalities being used to predict, or approximate, the set of active constraints is called the working set.

Nonlinear constraints are more difficult to handle. If at all possible, it is usually beneficial to avoid including nonlinear constraints during the formulation of the problem. The methods currently implemented in the Library handle nonlinearly constrained problems either by transforming them into a sequence of bound constraint problems, or by transforming them into a sequence of quadratic programming problems. A feature of almost all methods for nonlinear constraints is that $x^{(k)}$ is not guaranteed to be feasible except in the limit, and this is certainly true of the routines currently in the Library. See Chapter 6, particularly Section 6.4 and Section 6.5 of Gill et al [5].

Anyone interested in a detailed description of methods for optimization should consult the references.

2.5. Scaling

Scaling (in a broadly defined sense) often has a significant influence on the performance of optimization methods. Since convergence tolerances and other criteria are necessarily based on an implicit definition of 'small' and 'large', problems with unusual or unbalanced scaling may cause difficulties for some algorithms. Nonetheless, there are currently no scaling routines in the Library, although the position is under constant review. In light of the present state of the art, it is considered that sensible scaling by the user is likely to be more effective than any automatic routine. The following sections present some general comments on problem scaling.

2.5.1. Transformation of variables

One method of scaling is to transform the variables from their original representation, which may reflect the physical nature of the problem, to variables that have certain desirable properties in terms of optimization. It is generally helpful for the following conditions to be satisfied:

(i) the variables are all of similar magnitude in the region of interest;

(ii) a fixed change in any of the variables results in similar changes in $F(x)$. Ideally, a unit change in any variable produces a unit change in $F(x)$;

(iii) the variables are transformed so as to avoid cancellation error in the evaluation of $F(x)$.

Normally, users should restrict themselves to linear transformations of variables, although occasionally nonlinear transformations are possible. The most common such transformation (and often the most appropriate) is of the form

$$x_{new} = Dx_{old},$$

where D is a diagonal matrix with constant coefficients. Our experience suggests that more use should be made of the transformation

$$x_{new} = Dx_{old} + v,$$

where v is a constant vector.

Consider, for example, a problem in which the variable x_3 represents the position of the peak of a Gaussian curve to be fitted to data for which the extreme values are 150 and 170; therefore x_3 is known to lie in the range 150--170. One possible scaling would be to define a new variable x_3, given by

$$x_3 = \frac{x_3}{170}.$$

A better transformation, however, is given by defining x_3 as

$$x = \frac{x_3 - 160}{}.$$

3 10

Frequently, an improvement in the accuracy of evaluation of F(x) can result if the variables are scaled before the routines to evaluate F(x) are coded. For instance, in the above problem just mentioned of Gaussian curve fitting, x_3 may always occur in terms of the form (x_3-x_m), where x_m is a constant representing the mean peak position.

2.5.2. Scaling the objective function

The objective function has already been mentioned in the discussion of scaling the variables. The solution of a given problem is unaltered if F(x) is multiplied by a positive constant, or if a constant value is added to F(x). It is generally preferable for the objective function to be of the order of unity in the region of interest; thus, if in the original formulation F(x) is always of the order of 10^{+5} (say), then the value of F(x) should be multiplied by 10^{-5} when evaluating the function within the optimization routines. If a constant is added or subtracted in the computation of F(x), usually it should be omitted - i.e., it is better to formulate F(x) as $x_1^2+x_2^2$ rather than as $x_1^2+x_2^2+1000$ or even $x_1^2+x_2^2+1$. The inclusion of such a constant in the calculation of F(x) can result in a loss of significant figures.

2.5.3. Scaling the constraints

The solution of a nonlinearly-constrained problem is unaltered if the ith constraint is multiplied by a positive weight w_i. At the approximation of the solution determined by a Library routine, the active constraints will not be satisfied exactly, but will have 'small' values (for example, $c_1=10^{-8}$, $c_2=10^{-6}$, etc.). In general, this discrepancy will be minimized if the constraints are weighted so that a unit change in x produces a similar change in each constraint.

A second reason for introducing weights is related to the effect of the size of the constraints on the Lagrange multiplier estimates and, consequently, on the active set strategy. Additional discussion is given in Gill et al [5].

2.6. Analysis of Computed Results

2.6.1. Convergence criteria

The convergence criteria inevitably vary from routine to routine, since in some cases more information is available to be checked (for example, is the Hessian matrix positive-definite?), and different checks need to be made for different problem categories (for example, in constrained minimization it is necessary to verify whether a trial solution is feasible). Nonetheless, the underlying principles of the various criteria are the same; in non-mathematical terms, they are:

(i) is the sequence $\{x^{(k)}\}$ converging?

(ii) is the sequence $\{F^{(k)}\}$ converging?

(iii) are the necessary and sufficient conditions for the solution satisfied?

The decision as to whether a sequence is converging is necessarily speculative. The criterion used in the present routines is to assume convergence if the relative change occurring between two successive iterations is less than some prescribed quantity. Criterion (iii) is the most reliable but often the conditions cannot be checked fully because not all the required information may be available.

2.6.2. Checking results

Little a priori guidance can be given as to the quality of the solution found by a nonlinear optimization algorithm, since no guarantees can be given that the methods will always work. Therefore, it is necessary for the user to check the computed solution even if the routine reports success. Frequently a ' solution' may have been found even when the routine does not report a success. The reason for this apparent contradiction is that the routine needs to assess the accuracy of the solution. This assessment is not an exact process and consequently may be unduly pessimistic. Any 'solution' is in general only an approximation to the exact solution, and it is possible that the accuracy specified by the user is too stringent.

Further confirmation can be sought by trying to check whether or not convergence tests are almost satisfied, or whether or not some of the sufficient conditions are nearly satisfied. When it is thought that a routine has returned a non-zero value of IFAIL only because the requirements for 'success' were too stringent it may be worth restarting with increased convergence tolerances.

For nonlinearly-constrained problems, check whether the solution returned is feasible, or nearly feasible; if not, the solution returned is not an adequate solution.

Confidence in a solution may be increased by resolving the problem with a different initial approximation to the solution. See Section 8.3 of Gill et al [5] for further information.

2.6.3. Monitoring progress

Many of the routines in the Chapter have facilities to allow the user to monitor the progress of the minimization process, and users are encouraged to make use of these facilities. Monitoring information can be a great aid in assessing whether or not a satisfactory solution has been obtained, and in indicating difficulties in the minimization problem or in the routine's ability to cope with the problem.

The behaviour of the function, the estimated solution and first derivatives can help in deciding whether a solution is acceptable and what to do in the event of a return with a non-zero value of IFAIL.

2.6.4. Confidence intervals for least-squares solutions

When estimates of the parameters in a nonlinear least-squares problem have been found, it may be necessary to estimate the variances of the parameters and the fitted function. These can be calculated from the Hessian of F(x) at the solution.

In many least-squares problems, the Hessian is adequately approximated at the solution by $G=2J^TJ$ (see Section 2.4.3). The Jacobian, J, or a factorization of J is returned by all the comprehensive least-squares routines and, in addition, a routine is supplied in the Library to estimate variances of the parameters following the use of most of the nonlinear least-squares routines, in the case that $G=2J^TJ$ is an adequate approximation.

Let H be the inverse of G, and S be the sum of squares, both

calculated at the solution x; an unbiased estimate of the variance of the ith parameter x_i is

$$\mathrm{var}\, x_i = \frac{2S}{m-n} H_{ii}$$

and an unbiased estimate of the covariance of x_i and x_j is

$$\mathrm{covar}(x_i, x_j) = \frac{2S}{m-n} H_{ij} .$$

If x^* is the true solution, then the 100(1-(beta)) confidence interval on x is

$$x_i - \sqrt{\mathrm{var}\, x_i} . t_{(1-(\beta)/2, m-n)} < x_i^* < x_i$$

$$+ \sqrt{\mathrm{var}\, x_i} . t_{(1-(\beta)/2, m-n)} , \quad i=1,2,\ldots,n$$

where $t_{(1-(\beta)/2, m-n)}$ is the 100(1-(beta))/2 percentage point of the t-distribution with m-n degrees of freedom.

In the majority of problems, the residuals f_i, for $i=1,2,\ldots,m$, contain the difference between the values of a model function (phi)(z,x) calculated for m different values of the independent variable z, and the corresponding observed values at these points. The minimization process determines the parameters, or

constants x, of the fitted function (phi)(z,x). For any value, z, of the independent variable z, an unbiased estimate of the variance of (phi) is

n   n

$$\mathrm{var}\,(\phi) = \frac{2S}{m-n} \sum_{i=1} \sum_{j=1} \left[\frac{d\phi}{dx_i} \right]_z \left[\frac{d\phi}{dx_j} \right]_z H_{ij} .$$

The $100(1-\beta)$ confidence interval on F at the point z is

$$\phi(z,x) - \sqrt{\mathrm{var}\,(\phi)}\,.\,t_{(\beta/2,m-n)} < \phi(z,x^*)$$

$$< \phi(z,x) + \sqrt{\mathrm{var}\,(\phi)}\,.\,t_{(\beta/2,m-n)} .$$

For further details on the analysis of least-squares solutions see Bard [1] and Wolberg [7].

2.7. References

[1] Bard Y (1974) Nonlinear Parameter Estimation. Academic Press.

[2] Dantzig G B (1963) Linear Programming and Extensions. Princeton University Press.

[3] Fletcher R (1987) Practical Methods of Optimization. Wiley (2nd Edition).

[4] Gill P E and Murray W (eds) (1974) Numerical Methods for Constrained Optimization. Academic Press.

[5] Gill P E, Murray W and Wright M H (1981) Practical Optimization. Academic Press.

[6] Murray W (ed) (1972) Numerical Methods for Unconstrained Optimization. Academic Press.

[7] Wolberg J R (1967) Prediction Analysis. Van Nostrand.

3. Recommendations on Choice and Use of Routines

The choice of routine depends on several factors: the type of problem (unconstrained, etc.); the level of derivative information available (function values only, etc.); the experience of the user (there are easy-to-use versions of some routines); whether or not storage is a problem; and whether computational time has a high priority.

3.1. Choice of Routine

Routines are provided to solve the following types of problem:

Nonlinear Programming	E04UCF
Quadratic Programming	E04NAF
Linear Programming	E04MBF
Nonlinear Function (using 1st derivatives)	E04DGF
Nonlinear Function, unconstrained or simple bounds (using function values only)	E04JAF
Nonlinear least-squares (using function values only)	E04FDF
Nonlinear least-squares (using function values and 1st derivatives)	E04GCF

E04UCF can be used to solve unconstrained, bound-constrained and linearly-constrained problems.

E04NAF can be used as a comprehensive linear programming solver; however, in most cases the easy-to-use routine E04MBFwill be adequate.

E04MBF can be used to obtain a feasible point for a set of linear constraints.

E04DGF can be used to solve large scale unconstrained problems.

The routines can be used to solve problems in a single variable.

3.2. Service Routines

One of the most common errors in use of optimization routines is that the user's subroutines incorrectly evaluate the relevant partial derivatives. Because exact gradient information normally enhances efficiency in all areas of optimization, the user should be encouraged to provide analytical derivatives whenever possible. However, mistakes in the computation of derivatives can result in serious and obscure run-time errors, as well as complaints that the Library routines are incorrect.

E04UCF incorporates a check on the gradients being supplied and users are encouraged to utilize this option; E04GCF also incorporates a call to a derivative checker.

E04YCF estimates selected elements of the variance-covariance matrix for the computed regression parameters following the use of a nonlinear least-squares routine.

3.3. Function Evaluations at Infeasible Points

Users must not assume that the routines for constrained problems will require the objective function to be evaluated only at points which satisfy the constraints, i.e., feasible points. In the first place some of the easy-to-use routines call a service routine which will evaluate the objective function at the user-supplied initial point, and at neighbouring points (to check user-supplied derivatives or to estimate intervals for finite differencing). Apart from this, all routines will ensure that any evaluations of the objective function occur at points which approximately satisfy any simple bounds or linear constraints. Satisfaction of such constraints is only approximate because:

(a) routines which have a parameter FEATOL may allow such constraints to be violated by a margin specified by FEATOL;

(b) routines which estimate derivatives by finite differences may require function evaluations at points which just violate such constraints even though the current iteration just satisfies them.

There is no attempt to ensure that the current iteration satisfies any nonlinear constraints. Users who wish to prevent their objective function being evaluated outside some known region (where it may be undefined or not practically computable), may try to confine the iteration within this region by imposing suitable simple bounds or linear constraints (but beware as this may create new local minima where these constraints are active).

Note also that some routines allow the user-supplied routine to return a parameter (MODE) with a negative value to force an immediate clean exit from the minimization when the objective function cannot be evaluated.

3.4. Related Problems

Apart from the standard types of optimization problem, there are other related problems which can be solved by routines in this or other chapters of the Library.

E04MBF can be used to find a feasible point for a set of linear constraints and simple bounds.

Two routines in Chapter F04 solve linear least-squares problems,

i.e., minimize $\sum^{m} r(x)^{2}$ where $r(x)=b-\sum^{n} a\, x$.

```
                --  i           i     i  --  ij j
                i=1                      j=1
```

E02GAF solves an overdetermined system of linear equations in the

```
                          m
                          --
l  norm, i.e., minimizes  >  |r (x)|, with r  as above.
 1                        --   i            i
                          i=1
```

E04 -- Minimizing or Maximizing a Function   Contents -- E04
Chapter E04

Minimizing or Maximizing a Function

```
\end{verbatim}
\endscroll
\end{page}
```

———-

Minimizes a nonlinear function of several variable

— nage.ht —

```
\begin{page}{manpageXXe04dgf}{NAG Documentation: e04dgf}
\beginscroll
\begin{verbatim}



   E04DGF(3NAG)                 E04DGF                 E04DGF(3NAG)



          E04 -- Minimizing or Maximizing a Function                 E04DGF
                  E04DGF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          Note for users via the Axiom system: the interface to this routine
          has been enhanced for use with Axiom and is slightly different to
          that offered in the standard version of the Foundation Library.  In
          particular, the optional parameters of the NAG routine are now
          included in the parameter list.  These are described in section
          5.1.2, below.

          1. Purpose

          E04DGF minimizes an unconstrained nonlinear function of several
          variables using a pre-conditioned, limited memory quasi-Newton
          conjugate gradient method. First derivatives are required. The
          routine is intended for use on large scale problems.

          2. Specification

                 SUBROUTINE E04DGF(N,OBJFUN,ITER,OBJF,OBJGRD,X,IWORK,WORK,IUSER,
```

```
     1                   USER,ES,FU,IT,LIN,LIST,MA,OP,PR,STA,STO,
     2                   VE,IFAIL)
      INTEGER            N, ITER, IWORK(N+1), IUSER(*),
     1                   IT, PR, STA, STO, VE, IFAIL
      DOUBLE PRECISION OBJF, OBJGRD(N), X(N), WORK(13*N), USER(*)
     1                   ES, FU, LIN, OP, MA
      LOGICAL            LIST
      EXTERNAL           OBJFUN
```

3. Description

E04DGF uses a pre-conditioned conjugate gradient method and is based upon algorithm PLMA as described in Gill and Murray [1] and Gill et al [2] Section 4.8.3.

The algorithm proceeds as follows:

Let x_0 be a given starting point and let k denote the current iteration, starting with k=0. The iteration requires g_k, the gradient vector evaluated at x_k, the kth estimate of the minimum. At each iteration a vector p_k (known as the direction of search) is computed and the new estimate x_{k+1} is given by $x_k+(\alpha)_k p_k$ where $(\alpha)_k$ (the step length) minimizes the function $F(x_k+(\alpha)_k p_k)$ with respect to the scalar $(\alpha)_k$. A choice of initial step $(\alpha)_0$ is taken as

$$(\alpha)_0=\min\{1,2|F_k-F_{est}|/g_k^T g_k\}$$

where F_{est} is a user-supplied estimate of the function value at the solution. If F_{est} is not specified, the software always chooses the unit step length for $(\alpha)_0$. Subsequent step length estimates are computed using cubic interpolation with safeguards.

A quasi-Newton method can be used to compute the search direction p_k by updating the inverse of the approximate Hessian (H_k) and computing

$$p_{k+1} = -H_{k+1} g_{k+1} \qquad (1)$$

The updating formula for the approximate inverse is given by

$$H_{k+1} = H_k - \frac{1}{y_k^T s_k}\left(H_k y_k s_k^T + s_k y_k^T H_k\right) + \frac{1}{y_k^T s_k}\left(1 + \frac{y_k^T H_k y_k}{y_k^T s_k}\right) s_k s_k^T \qquad (2)$$

where $y_k = g_{k-1} - g_k$ and $s_k = x_{k+1} - x_k = (\alpha)_k p_k$.

The method used by E04DGF to obtain the search direction is based upon computing p_{k+1} as $-H_{k+1} g_{k+1}$ where H_{k+1} is a matrix obtained by updating the identity matrix with a limited number of quasi-Newton corrections. The storage of an n by n matrix is avoided by storing only the vectors that define the rank two corrections - hence the term limited-memory quasi-Newton method. The precise method depends upon the number of updating vectors stored. For example, the direction obtained with the 'one-step' limited memory update is given by (1) using (2) with H_k equal to the identity matrix, viz.

$$p_{k+1} = -g_{k+1} + \frac{1}{y_k^T s_k}\left(s_k^T g_{k+1} y_k + y_k^T g_{k+1} s_k\right) - \frac{s_k^T g_{k+1}}{y_k^T s_k}\left(1 + \frac{y_k^T y_k}{y_k^T s_k}\right) s_k$$

E04DGF uses a two-step method described in detail in Gill and Murray [1] in which restarts and pre-conditioning are incorporated. Using a limited-memory quasi-Newton formula, such as the one above, guarantees p_{k+1} to be a descent direction if all the inner products y_k^T are positive for all vectors y_k and s_k used in the updating formula.

The termination criterion of E04DGF is as follows:

Let (tau) specify a parameter that indicates the number of

F
correct figures desired in F_k (τ_F is equivalent to Optimality Tolerance in the optional parameter list, see Section 5.1). If the following three conditions are satisfied

(i) $F_{k-1} - F_k < \tau_F (1+|F_k|)$

(ii) $||x_{k-1} - x_k|| < \sqrt{\tau_F} (1+||x_k||)$

(iii) $||g_k|| \le \sqrt[3]{\tau_F} (1+|F_k|)$ or $||g_k|| < \epsilon_A$,
where ϵ_A is the absolute error associated with computing the objective function

then the algorithm is considered to have converged. For a full discussion on termination criteria see Gill et al [2] Chapter 8.

4. References

[1] Gill P E and Murray W (1979) Conjugate-gradient Methods for Large-scale Nonlinear Optimization. Technical Report SOL 79-15. Department of Operations Research, Stanford University.

[2] Gill P E, Murray W and Wright M H (1981) Practical Optimization. Academic Press.

5. Parameters

1: N -- INTEGER Input
On entry: the number n of variables. Constraint: N >= 1.

2: OBJFUN -- SUBROUTINE, supplied by the user. External Procedure
OBJFUN must calculate the objective function F(x) and its gradient for a specified n element vector x.

Its specification is:

```
       SUBROUTINE OBJFUN (MODE, N, X, OBJF, OBJGRD,
      1                   NSTATE, IUSER, USER)
       INTEGER          MODE, N, NSTATE, IUSER(*)
       DOUBLE PRECISION X(N), OBJF, OBJGRD(N), USER(*)
```

1: MODE -- INTEGER Input/Output

```
        MODE is a flag that the user may set within OBJFUN to
        indicate a failure in the evaluation of the objective
        function. On entry: MODE is always non-negative. On
        exit: if MODE is negative the execution of E04DGF is
        terminated with IFAIL set to MODE.

    2:  N -- INTEGER                                      Input
        On entry: the number n of variables.

    3:  X(N) -- DOUBLE PRECISION array                    Input
        On entry: the point x at which the objective function
        is required.

    4:  OBJF -- DOUBLE PRECISION                         Output
        On exit: the value of the objective function F at the
        current point x.

    5:  OBJGRD(N) -- DOUBLE PRECISION array              Output
                                                     ddF
        On exit: OBJGRD(i) must contain the value of  ---- at
                                                     ddx
                                                        i
        the point x, for i=1,2,...,n.

    6:  NSTATE -- INTEGER                                 Input
        On entry: NSTATE will be 1 on the first call of OBJFUN
        by E04DGF, and is 0 for all subsequent calls. Thus, if
        the user wishes, NSTATE may be tested within OBJFUN in
        order to perform certain calculations once only. For
        example the user may read data or initialise COMMON
        blocks when NSTATE = 1.

    7:  IUSER(*) -- INTEGER array                User Workspace

    8:  USER(*) -- DOUBLE PRECISION array        User Workspace
        OBJFUN is called from E04DGF with the parameters IUSER
        and USER as supplied to E04DGF. The user is free to use
        arrays IUSER and USER to supply information to OBJFUN
        as an alternative to using COMMON.
    OBJFUN must be declared as EXTERNAL in the (sub)program
    from which E04DGF is called. Parameters denoted as
    Input must not be changed by this procedure.

 3: ITER -- INTEGER                                      Output
    On exit: the number of iterations performed.

 4: OBJF -- DOUBLE PRECISION                             Output
    On exit: the value of the objective function F(x) at the
    final iterate.
```

```
 5:  OBJGRD(N) -- DOUBLE PRECISION array                   Output
     On exit: the objective gradient at the final iterate.

 6:  X(N) -- DOUBLE PRECISION array                  Input/Output
     On entry: an initial estimate of the solution. On exit: the
     final estimate of the solution.

 7:  IWORK(N+1) -- INTEGER array                        Workspace

 8:  WORK(13*N) -- DOUBLE PRECISION array               Workspace

 9:  IUSER(*) -- INTEGER array                     User Workspace
     Note: the dimension of the array IUSER must be at least 1.
     This array is not used by E04DGF, but is passed directly to
     routine OBJFUN and may be used to supply information to
     OBJFUN.

10:  USER(*) -- DOUBLE PRECISION array             User Workspace
     Note: the dimension of the array USER must be at least 1.
     This array is not used by E04DGF, but is passed directly to
     routine OBJFUN and may be used to supply information to
     OBJFUN.

11:  IFAIL -- INTEGER                                Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. Users who are
     unfamiliar with this parameter should refer to the Essential
     Introduction for details.

     On exit: IFAIL = 0 unless the routine detects an error or
     gives a warning (see Section 6).

     For this routine, because the values of output parameters
     may be useful even if IFAIL /=0 on exit, users are
     recommended to set IFAIL to -1 before entry. It is then
     essential to test the value of IFAIL on exit.
```

5.1. Optional Input Parameters

Several optional parameters in E04DGF define choices in the behaviour of the routine. In order to reduce the number of formal parameters of E04DGF these optional parameters have associated default values (see Section 5.1.3) that are appropriate for most problems. Therefore the user need only specify those optional parameters whose values are to be different from their default values.

The remainder of this section can be skipped by users who wish to use the default values for all optional parameters. A complete list of optional parameters and their default values is given in

Section 5.1.3.

5.1.1. Specification of the Optional Parameters

Optional parameters may be specified by calling one, or both, of E04DJF and E04DKF prior to a call to E04DGF.

E04DJF reads options from an external options file, with Begin and End as the first and last lines respectively and each intermediate line defining a single optional parameter. For example,

```
Begin
  Print Level = 1
End
```

The call

```
CALL E04DJF(IOPTNS, INFORM)
```

can then be used to read the file on unit IOPTNS. INFORM will be zero on successful exit. E04DJF should be consulted for a full description of this method of supplying optional parameters.

E04DKF can be called to supply options directly, one call being necessary for each optional parameter.

For example,

```
CALL E04DKF(‘Print level = 1’)
```

E04DKF should be consulted for a full description of this method of supplying optional parameters.

All optional parameters not specified by the user are set to their default values. Optional parameters specified by the user are unaltered by E04DGF (unless they define invalid values) and so remain in effect for subsequent calls to E04DGF, unless altered by the user.

5.1.2. Description of the Optional Parameters

The following list (in alphabetical order) gives the valid options. For each option, we give the keyword, any essential optional qualifiers, the default value, and the definition. The minimum valid abbreviation of each keyword is underlined. If no characters of an optional qualifier are underlined, the qualifier may be omitted. The letter a denotes a phrase (character string) that qualifies an option. The letters i and r denote INTEGER and real values required with certain options. The number (epsilon) is a generic notation for machine precision, and (epsilon)$_R$ denotes the relative precision of the objective function (the optional parameter Function Precision; see below).

Defaults

This special keyword may be used to reset the default values following a call to E04DGF.

Estimated Optimal Function Value r

(Axiom parameter ES)

This value of r specifies the user-supplied guess of the optimum objective function value. This value is used by E04DGF to calculate an initial step length (see Section 3). If the value of r is not specified by the user (the default), then this has the effect of setting the initial step length to unity. It should be noted that for badly scaled functions a unit step along the steepest descent direction will often compute the function at very large values of x.

Function Precision r Default = (epsilon)$^{0.9}$

(Axiom parameter FU)

The parameter defines (epsilon)$_R$, which is intended to be a measure of the accuracy with which the problem function F can be computed. The value of (epsilon)$_R$ should reflect the relative precision of 1+|F(x)|; i.e. (epsilon)$_R$ acts as a relative precision when |F| is large, and as an absolute precision when |F| is small. For example, if F(x) is typically of order 1000 and the first six significant digits are known to be correct, an appropriate value for (epsilon)$_R$ would be 1.0E-6. In contrast, if F(x) is typically of order 10^{-4} and the first six significant digits are known to be correct, an appropriate value for (epsilon)$_R$ would be 1.0E-10. The choice of (epsilon)$_R$ can be quite complicated for badly scaled problems; see Chapter 8 of Gill and Murray [2], for a discussion of scaling techniques. The default value is appropriate for most simple functions that are computed with full accuracy. However when the accuracy of the computed function values is known to be significantly worse than full precision, the value of $(epsilon)_R$ should be large enough so that E04DGF will not attempt to distinguish between function values that differ by less than the error inherent in the calculation. If 0<=r<(epsilon), where (epsilon) is the machine precision then the default value is used.

Iteration Limit i Default = max(50,5n)

Iters

Itns

(Axiom parameter IT)

The value i (i>=0) specifies the maximum number of iterations allowed before termination. If i<0 the default value is used. See Section 8 for further information.

Linesearch Tolerance r Default = 0.9

(Axiom parameter LIN)

The value r (0<=r<1) controls the accuracy with which the step (alpha) taken during each iteration approximates a minimum of the function along the search direction (the smaller the value of r, the more accurate the linesearch). The default value r=0.9 requests an inaccurate search, and is appropriate for most problems. A more accurate search may be appropriate when it is desirable to reduce the number of iterations - for example, if the objective function is cheap to evaluate.

List   Default = List
Nolist

(Axiom parameter LIST)

Normally each optional parameter specification is printed as it is supplied. Nolist may be used to suppress the printing and List may be used to restore printing.

Maximum Step Length r Default = 10^{10}

(Axiom parameter MA)

The value r (r>0) defines the maximum allowable step length for the line search. If r<=0 the default value is used.

Optimality Tolerance r Default = $(epsilon)_R^{0.8}$

(Axiom parameter OP)

The parameter r ($(epsilon)_R$ <=r<1) specifies the accuracy to which the user wishes the final iterate to approximate a solution of the problem. Broadly speaking, r indicates the number of correct figures desired in the objective function at the solution. For example, if r is 10^{-6} and E04DGF terminates successfully, the final value of F should have approximately six correct figures. E04DGF will terminate successfully if the iterative sequence of x -values is judged to have converged and the final point satisfies the termination criteria (see Section 3, where $(tau)_F$ represents Optimality Tolerance).

Print Level i Default = 10

(Axiom parameter PR)

The value i controls the amount of printout produced by E04DGF. The following levels of printing are available.

i	Output.
0	No output.
1	The final solution.
5	One line of output for each iteration.
10	The final solution and one line of output for each iteration.

Start Objective Check at Variable i Default = 1

(Axiom parameter STA)

Stop Objective Check at Variable i Default = n

(Axiom parameter STO)

These keywords take effect only if Verify Level > 0 (see below). They may be used to control the verification of gradient elements computed by subroutine OBJFUN. For example if the first 30 variables appear linearly in the objective, so that the

```
corresponding gradient elements are constant, then it is
reasonable to specify Start Objective Check at Variable  31.

Verify Level  i Default = 0

Verify                         No

Verify Level                   -1

Verify Level                   0

Verify

Verify                         Yes

Verify Objective Gradients

Verify Gradients

Verify Level                   1

   (Axiom parameter VE)

These keywords refer to finite-difference checks on the gradient
elements computed by the user-provided subroutine OBJFUN. It is
possible to set Verify Level in several ways, as indicated above.
For example, the gradients will be verified if Verify, Verify
Yes, Verify Gradients, Verify Objective Gradients or Verify Level
= 1 is specified.

If i<0 then no checking will be performed. If i>0 then the
gradients will be verified at the user-supplied point. If i=0
only a 'cheap' test will be performed, requiring one call to
OBJFUN. If i=1, a more reliable (but more expensive) check will
be made on individual gradient components, within the ranges
specified by the Start and Stop keywords as described above. A
result of the form OK or BAD? is printed by E04DGF to indicate
whether or not each component appears to be correct.

5.1.3. Optional parameter checklist and default values

For easy reference, the following sample list shows all valid
keywords and their default values. The default options Function
Precision and Optimality Tolerance depend upon (epsilon), the
machine precision.

Optional Parameters          Default Values

Estimated Optimal Function
Value
```

Function precision	$(epsilon)^{0.9}$
Iterations	max(50,5n)
Linesearch Tolerance	0.9
Maximum Step Length	10^{10}
List/Nolist	List
Optimality Tolerance	$(epsilon)^{0.8}$
Print Level	10
Start Objective Check at Variable	1
Stop Objective Check at Variable	n
Verify Level	0

5.2. Description of Printed Output

The level of printed output from E04DGF is controlled by the user (see the description of Print Level in Section 5.1).

When Print Level >= 5, the following line of output is produced at each iteration.

Itn	is the iteration count.
Step	is the step (alpha) taken along the computed search direction. On reasonably well-behaved problems, the unit step will be taken as the solution is approached.
Nfun	is the cumulated number of evaluations of the objective function needed for the linesearch. Evaluations needed for the estimation of the gradients by finite differences are not included. Nfun is printed as a guide to the amount of work required for the linesearch. E04DGF will perform at most 16 function evaluations per iteration.
Objective	is the value of the objective function.

```
Norm G         is the Euclidean norm of the gradient of the
               objective function.

Norm X         is the Euclidean norm of x.

Norm (X(k-1)-X(k)) is the Euclidean norm of x   -x .
                                             k-1  k

When Print Level = 1 or Print Level >= 10 then the solution at
the end of execution of E04DGF is printed out.

The following describes the printout for each variable:

Variable       gives the name (VARBL) and index j (j = 1 to n) of
               the variable

Value          is the value of the variable at the final iterate

Gradient Value is the value of the gradient of the objective
               function with respect to the jth variable at the
               final iterate

6. Error Indicators and Warnings

Errors or warnings specified by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

On exit from E04DGF, IFAIL should be tested. If Print Level > 0
then a short description of IFAIL is printed.

Errors and diagnostics indicated by IFAIL from E04DGF are as
follows:

IFAIL< 0
     A negative value of IFAIL indicates an exit from E04DGF
     because the user set MODE negative in routine OBJFUN. The
     value of IFAIL will be the same as the user's setting of
     MODE.

IFAIL= 1
     Not used by this routine.

IFAIL= 2
     Not used by this routine.

IFAIL= 3
     The maximum number of iterations has been performed. If the
```

```
     algorithm appears to be making progress the iterations value
     may be too small (see Section 5.1.2) so the user should
     increase iterations and rerun E04DGF. If the algorithm seems
     to be 'bogged down',the user should check for incorrect
     gradients or ill-conditioning as described below under IFAIL
     = 6.

IFAIL= 4
     The computed upper bound on the step length taken during the
     linesearch was too small. A rerun with an increased value of
     the Maximum Step Length ((rho) say) may be successful unless
             10
     (rho)>=10   (the default value), in which case the current
     point cannot be improved upon.

IFAIL= 5
     Not used by this routine.

IFAIL= 6
     A sufficient decrease in the function value could not be
     attained during the final linesearch. If the subroutine
     OBJFUN computes the function and gradients correctly, then
     this may occur because an overly stringent accuracy has been
     requested, i.e., Optimality Tolerance is too small or if the
     minimum lies close to a step length of zero. In this case
     the user should apply the four tests described in Section 3
     to determine whether or not the final solution is acceptable
     (the user will need to set Print Level >= 5). For a
     discussion of attainable accuracy see Gill and Murray [2].

     If many iterations have occurred in which essentially no
     progress has been made or E04DGF has failed to move from the
     initial point, subroutine OBJFUN may be incorrect. The user
     should refer to the comments below under IFAIL = 7 and check
     the gradients using the Verify parameter. Unfortunately,
     there may be small errors in the objective gradients that
     cannot be detected by the verification process. Finite-
     difference approximations to first derivatives are
     catastrophically affected by even small inaccuracies.

IFAIL= 7
     Large errors were found in the derivatives of the objective
     function. This value of IFAIL will occur if the verification
     process indicated that at least one gradient component had
     no correct figures. The user should refer to the printed
     output to determine which elements are suspected to be in
     error.

     As a first step, the user should check that the code for the
     objective values is correct - for example, by computing the
```

function at a point where the correct value is known. However, care should be taken that the chosen point fully tests the evaluation of the function. It is remarkable how often the values x=0 or x=1 are used to test function evaluation procedures, and how often the special properties of these numbers make the test meaningless.

Special care should be used in this test if computation of the objective function involves subsidiary data communicated in COMMON storage. Although the first evaluation of the function may be correct, subsequent calculations may be in error because some of the subsidiary data has accidentally been overwritten.

Errors in programming the function may be quite subtle in that the function value is 'almost' correct. For example, the function may not be accurate to full precision because of the inaccurate calculation of a subsidiary quantity, or the limited accuracy of data upon which the function depends. A common error on machines where numerical calculations are usually performed in double precision is to include even one single-precision constant in the calculation of the function; since some compilers do not convert such constants to double precision, half the correct figures may be lost by such a seemingly trivial error.

IFAIL= 8

The gradient (g) at the starting point is too small. The value $g^T g$ is less than $(\epsilon)_m |F(x_o)|$, where $(\epsilon)_m$ is the machine precision.

The problem should be rerun at a different starting point.

IFAIL= 9

On entry N < 1.

7. Accuracy

On successful exit the accuracy of the solution will be as defined by the optional parameter Optimality Tolerance.

8. Further Comments

Problems whose Hessian matrices at the solution contain sets of clustered eigenvalues are likely to be minimized in significantly fewer than n iterations. Problems without this property may require anything between n and 5n iterations, with approximately 2n iterations being a common figure for moderately difficult

```
        problems.

        9. Example

        To find a minimum of the function

                              x
                               1   2   2
                          F=e  (4x +2x +4x x +2x +1).
                                   1   2   1 2   2

        The example program is not reproduced here. The source code for
        all example programs is distributed with the NAG Foundation
        Library software and should be available on-line.
\end{verbatim}
\endscroll
\end{page}
```

Supply optional parameters to E04DGF from file

— nage.ht —

```
\begin{page}{manpageXXe04djf}{NAG Documentation: e04djf}
\beginscroll
\begin{verbatim}



     E04DJF(3NAG)      Foundation Library (12/10/92)      E04DJF(3NAG)



          E04 -- Minimizing or Maximizing a Function                 E04DJF
                  E04DJF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          To supply optional parameters to E04DGF from an external file.

          2. Specification
```

```
SUBROUTINE E04DJF (IOPTNS, INFORM)
INTEGER          IOPTNS, INFORM
```

3. Description

E04DJF may be used to supply values for optional parameters to E04DGF. E04DJF reads an external file and each line of the file defines a single optional parameter. It is only necessary to supply values for those parameters whose values are to be different from their default values.

Each optional parameter is defined by a single character string of up to 72 characters, consisting of one or more items. The items associated with a given option must be separated by spaces, or equal signs (=). Alphabetic characters may be upper or lower case. The string

```
Print level = 1
```

is an example of a string used to set an optional parameter. For each option the string contains one or more of the following items:

(a) A mandatory keyword.

(b) A phrase that qualifies the keyword.

(c) A number that specifies an INTEGER or real value. Such numbers may be up to 16 contiguous characters in Fortran 77's I, F, E or D formats, terminated by a space if this is not the last item on the line.

Blank strings and comments are ignored. A comment begins with an asterisk (*) and all subsequent characters in the string are regarded as part of the comment.

The file containing the options must start with begin and must finish with end An example of a valid options file is:

```
Begin * Example options file
  Print level = 10
End
```

Normally each line of the file is printed as it is read, on the current advisory message unit (see X04ABF), but printing may be suppressed using the keyword nolist. To suppress printing of begin, nolist must be the first option supplied as in the file:

```
Begin
  Nolist
  Print level = 10
End
```

Printing will automatically be turned on again after a call to E04DGF and may be turned on again at any time by the user by using the keyword list.

Optional parameter settings are preserved following a call to E04DGF, and so the keyword defaults is provided to allow the user to reset all the optional parameters to their default values prior to a subsequent call to E04DGF.

A complete list of optional parameters, their abbreviations, synonyms and default values is given in Section 5.1 of the routine document for E04DGF.

4. References

None.

5. Parameters

```
 1:  IOPTNS -- INTEGER                                      Input
     On entry: IOPTNS must be the unit number of the options
     file. Constraint: 0 <= IOPTNS <= 99.

 2:  INFORM -- INTEGER                                     Output
     On exit: INFORM will be zero if an options file with the
     correct structure has been read. Otherwise INFORM will be
     positive. Positive values of INFORM indicate that an options
     file may not have been successfully read as follows:
     INFORM = 1
           IOPTNS is not in the range [0,99].

     INFORM = 2
           begin was found, but end-of-file was found before end
           was found.

     INFORM = 3
           end-of-file was found before begin was found.
```

6. Error Indicators and Warnings

If a line is not recognised as a valid option, then a warning message is output on the current advisory message unit (see X04ABF).

7. Accuracy

```
         Not applicable.

         8. Further Comments

         E04DKF may also be used to supply optional parameters to E04DGF.

         9. Example

         See the example for E04DGF.

\end{verbatim}
\endscroll
\end{page}
```

———-

Supply individual optional params to E04DGF

— nage.ht —

```
\begin{page}{manpageXXe04dkf}{NAG Documentation: e04dkf}
\beginscroll
\begin{verbatim}



     E04DKF(3NAG)      Foundation Library (12/10/92)      E04DKF(3NAG)



          E04 -- Minimizing or Maximizing a Function                E04DKF
                  E04DKF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          To supply individual optional parameters to E04DGF.

          2. Specification

                 SUBROUTINE E04DKF (STRING)
                 CHARACTER*(*)    STRING
```

3. Description

E04DKF may be used to supply values for optional parameters to E04DGF. It is only necessary to call E04DKF for those parameters whose values are to be different from their default values. One call to E04DKF sets one parameter value.

Each optional parameter is defined by a single character string of up to 72 characters, consisting of one or more items. The items associated with a given option must be separated by spaces, or equal signs (=). Alphabetic characters may be upper or lower case. The string

```
Print Level = 1
```

is an example of a string used to set an optional parameter. For each option the string contains one or more of the following items:

(a) A mandatory keyword.

(b) A phrase that qualifies the keyword.

(c) A number that specifies an INTEGER or real value. Such numbers may be up to 16 contiguous characters in Fortran 77's I, F, E or D formats, terminated by a space if this is not the last item on the line.

Blank strings and comments are ignored. A comment begins with an asterisk (*) and all subsequent characters in the string are regarded as part of the comment.

Normally, each user-specified option is printed as it is defined, on the current advisory message unit (see X04ABF), but this printing may be suppressed using the keyword nolist Thus the statement

```
CALL E04DKF ('Nolist')
```

suppresses printing of this and subsequent options. Printing will automatically be turned on again after a call to E04DGF, and may be turned on again at any time by the user, by using the keyword list.

Optional parameter settings are preserved following a call to E04DGF, and so the keyword defaults is provided to allow the user to reset all the optional parameters to their default values by the statement,

```
     CALL E04DKF ('Defaults')

prior to a subsequent call to E04DGF.

A complete list of optional parameters, their abbreviations,
synonyms and default values is given in Section 5.1 of the
routine document for E04DGF.

4. References

None.

5. Parameters

 1:  STRING -- CHARACTER*(*)                               Input
     On entry: STRING must be a single valid option string. See
     Section 3 above, and Section 5.1 of the routine document for
     E04DGF.

6. Error Indicators and Warnings

If the parameter STRING is not recognised as a valid option
string, then a warning message is output on the current advisory
message unit (see X04ABF).

7. Accuracy

Not applicable.

8. Further Comments

E04DJF may also be used to supply optional parameters to E04DGF.

9. Example

See the example for E04DGF.

\end{verbatim}
\endscroll
\end{page}
```

———-

Finding an unconstrained minimum of a sum of squares

— nage.ht —

```
\begin{page}{manpageXXe04fdf}{NAG Documentation: e04fdf}
\beginscroll
\begin{verbatim}


     E04FDF(3NAG)      Foundation Library (12/10/92)      E04FDF(3NAG)



          E04 -- Minimizing or Maximizing a Function                 E04FDF
                  E04FDF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          E04FDF is an easy-to-use algorithm for finding an unconstrained
          minimum of a sum of squares of m nonlinear functions in n
          variables (m>=n). No derivatives are required.

          It is intended for functions which are continuous and which have
          continuous first and second derivatives (although it will usually
          work even if the derivatives have occasional discontinuities).

          2. Specification

                 SUBROUTINE E04FDF (M, N, X, FSUMSQ, IW, LIW, W, LW, IFAIL)
                 INTEGER          M, N, IW(LIW), LIW, LW, IFAIL
                 DOUBLE PRECISION X(N), FSUMSQ, W(LW)

          3. Description

          This routine is essentially identical to the subroutine LSNDN1 in
          the National Physical Laboratory Algorithms Library. It is
          applicable to problems of the form

                                          m
                                          --        2
                           Minimize F(x)= >  [f (x)]
                                          --   i
                                          i=1

                                T
          where x=(x ,x ,...,x )  and m>=n. (The functions f (x) are often
                    1  2      n                             i
          referred to as 'residuals'.) The user must supply a subroutine
```

```
LSFUN1 to evaluate functions f (x) at any point x.
                              i

From a starting point supplied by the user, a sequence of points
is generated which is intended to converge to a local minimum of
the sum of squares. These points are generated using estimates of
the curvature of F(x).

4. References

[1]   Gill P E and Murray W (1978) Algorithms for the Solution of
      the Nonlinear Least-squares Problem. SIAM J. Numer. Anal. 15
      977--992.

5. Parameters

 1:  M -- INTEGER                                            Input

 2:  N -- INTEGER                                            Input
     On entry: the number m of residuals f (x), and the number n
                                          i
     of variables, x . Constraint: 1 <= N <= M.
                    j

 3:  X(N) -- DOUBLE PRECISION array                   Input/Output
     On entry: X(j) must be set to a guess at the jth component
     of the position of the minimum, for j=1,2,...,n. On exit:
     the lowest point found during the calculations. Thus, if
     IFAIL = 0 on exit, X(j) is the jth component of the position
     of the minimum.

 4:  FSUMSQ -- DOUBLE PRECISION                             Output
     On exit: the value of the sum of squares, F(x),
     corresponding to the final point stored in X.

 5:  IW(LIW) -- INTEGER array                            Workspace

 6:  LIW -- INTEGER                                          Input
     On entry: the length of IW as declared in the (sub)program
     from which E04FDF has been called. Constraint: LIW >= 1.

 7:  W(LW) -- DOUBLE PRECISION array                     Workspace

 8:  LW -- INTEGER                                           Input
     On entry: the length of W as declared in the (sub)program
     from which E04FDF is called. Constraints:
          LW >= N*(7 + N + 2*M + (N-1)/2) + 3*M, if N > 1,

          LW >= 9 + 5*>M, if N = 1.
```

9: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. Users who are unfamiliar with this parameter should refer to the Essential Introduction for details.

On exit: IFAIL = 0 unless the routine detects an error or gives a warning (see Section 6).

For this routine, because the values of output parameters may be useful even if IFAIL /=0 on exit, users are recommended to set IFAIL to -1 before entry. It is then essential to test the value of IFAIL on exit.

5.1. Optional Parameters

LSFUN1 -- SUBROUTINE, supplied by the user. External Procedure
This routine must be supplied by the user to calculate the vector of values $f_i(x)$ at any point x. Since the routine is not a parameter to E04FDF, it must be called LSFUN1. It should be tested separately before being used in conjunction with E04FDF (see the Chapter Introduction).

Its specification is:

```
SUBROUTINE LSFUN1 (M, N, XC, FVECC)
INTEGER          M, N
DOUBLE PRECISION XC(N), FVECC(M)
```

1: M -- INTEGER Input

2: N -- INTEGER Input
On entry: the numbers m and n of residuals and variables, respectively.

3: XC(N) -- DOUBLE PRECISION array Input
On entry: the point x at which the values of the f_i are required.

4: FVECC(M) -- DOUBLE PRECISION array Output
On exit: FVECC(i) must contain the value of f_i at the point x, for i=1,2,...,m.

LSFUN1 must be declared as EXTERNAL in the (sub)program from which E04FDF is called. Parameters denoted as Input must not be changed by this procedure.

6. Error Indicators and Warnings

```
Errors or warnings specified by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

IFAIL= 1
     On entry N < 1,

     or       M < N,

     or       LIW < 1,

     or       LW < N*(7 + N + 2*M + (N-1)/2) + 3*M, when N > 1,

     or       LW < 9 + 5*>M, when N = 1.

IFAIL= 2
     There have been 400*n calls of LSFUN1, yet the algorithm
     does not seem to have converged. This may be due to an
     awkward function or to a poor starting point, so it is worth
     restarting E04FDF from the final point held in X.

IFAIL= 3
     The final point does not satisfy the conditions for
     acceptance as a minimum, but no lower point could be found.

IFAIL= 4
     An auxiliary routine has been unable to complete a singular
     value decomposition in a reasonable number of sub-
     iterations.

IFAIL= 5

IFAIL= 6

IFAIL= 7

IFAIL= 8
     There is some doubt about whether the point x found by
     E04FDF is a minimum of F(x). The degree of confidence in the
     result decreases as IFAIL increases. Thus when IFAIL = 5, it
     is probable that the final x gives a good estimate of the
     position of a minimum, but when IFAIL = 8 it is very
     unlikely that the routine has found a minimum.

If the user is not satisfied with the result (e.g. because IFAIL
lies between 3 and 8), it is worth restarting the calculations
from a different starting point (not the point at which the
failure occurred) in order to avoid the region which caused the
```

failure. Repeated failure may indicate some defect in the formulation of the problem.

7. Accuracy

If the problem is reasonably well scaled and a successful exit is made, then, for a computer with a mantissa of t decimals, one would expect to get about t/2-1 decimals accuracy in the components of x and between t-1 (if F(x) is of order 1 at the minimum) and 2t-2 (if F(x) is close to zero at the minimum) decimals accuracy in F(x).

8. Further Comments

The number of iterations required depends on the number of variables, the number of residuals and their behaviour, and the distance of the starting point from the solution. The number of multiplications performed per iteration of E04FDF varies, but for $m>>n$ is approximately $n*m^2+O(n^3)$. In addition, each iteration makes at least n+1 calls of LSFUN1. So, unless the residuals can be evaluated very quickly, the run time will be dominated by the time spent in LSFUN1.

Ideally, the problem should be scaled so that the minimum value of the sum of squares is in the range (0,1), and so that at points a unit distance away from the solution the sum of squares is approximately a unit value greater than at the minimum. It is unlikely that the user will be able to follow these recommendations very closely, but it is worth trying (by guesswork), as sensible scaling will reduce the difficulty of the minimization problem, so that E04FDF will take less computer time.

When the sum of squares represents the goodness of fit of a nonlinear model to observed data, elements of the variance-covariance matrix of the estimated regression coefficients can be computed by a subsequent call to E04YCF, using information returned in segments of the workspace array W. See E04YCF for further details.

9. Example

To find least-squares estimates of x_1, x_2 and x_3 in the model

$$y=x_1 + \frac{t_1}{x_2 t_2 + x_3 t_3}$$

```
                              2 2  3 3

     using the 15 sets of data given in the following table.

                             y    t   t    t
                                   1   2    3
                           0.14  1.0 15.0 1.0
                           0.18  2.0 14.0 2.0
                           0.22  3.0 13.0 3.0
                           0.25  4.0 12.0 4.0
                           0.29  5.0 11.0 5.0
                           0.32  6.0 10.0 6.0
                           0.35  7.0  9.0 7.0
                           0.39  8.0  8.0 8.0
                           0.37  9.0  7.0 7.0
                           0.58 10.0  6.0 6.0
                           0.73 11.0  5.0 5.0
                           0.96 12.0  4.0 4.0
                           1.34 13.0  3.0 3.0
                           2.10 14.0  2.0 2.0
                           4.39 15.0  1.0 1.0

     The program uses (0.5, 1.0, 1.5) as the initial guess at the
     position of the minimum.

     The example program is not reproduced here. The source code for
     all example programs is distributed with the NAG Foundation
     Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

———-

Finding an unconstrained minimum of a sum of squares

— nage.ht —

```
\begin{page}{manpageXXe04gcf}{NAG Documentation: e04gcf}
\beginscroll
\begin{verbatim}



     E04GCF(3NAG)      Foundation Library (12/10/92)      E04GCF(3NAG)
```

E04 -- Minimizing or Maximizing a Function E04GCF
E04GCF -- NAG Foundation Library Routine Document

Note: Before using this routine, please read the Users' Note for your implementation to check implementation-dependent details. The symbol (*) after a NAG routine name denotes a routine that is not included in the Foundation Library.

1. Purpose

E04GCF is an easy-to-use quasi-Newton algorithm for finding an unconstrained minimum of a sum of squares of m nonlinear functions in n variables (m>=n). First derivatives are required.

It is intended for functions which are continuous and which have continuous first and second derivatives (although it will usually work even if the derivatives have occasional discontinuities).

2. Specification

```
      SUBROUTINE E04GCF (M, N, X, FSUMSQ, IW, LIW, W, LW, IFAIL)
      INTEGER          M, N, IW(LIW), LIW, LW, IFAIL
      DOUBLE PRECISION X(N), FSUMSQ, W(LW)
```

3. Description

This routine is essentially identical to the subroutine LSFDQ2 in the National Physical Laboratory Algorithms Library. It is applicable to problems of the form

$$\text{Minimize } F(x)=\sum_{i=1}^{m}[f_i(x)]^2$$

where $x=(x_1,x_2,\ldots,x_n)^T$ and $m>=n$. (The functions $f_i(x)$ are often referred to as 'residuals'.) The user must supply a subroutine LSFUN2 to evaluate the residuals and their first derivatives at any point x.

Before attempting to minimize the sum of squares, the algorithm checks LSFUN2 for consistency. Then, from a starting point supplied by the user, a sequence of points is generated which is intended to converge to a local minimum of the sum of squares. These points are generated using estimates of the curvature of

F(x).

4. References

[1] Gill P E and Murray W (1978) Algorithms for the Solution of the Nonlinear Least-squares Problem. SIAM J. Numer. Anal. 15 977--992.

5. Parameters

1: M -- INTEGER Input

2: N -- INTEGER Input
On entry: the number m of residuals $f_i(x)$, and the number n of variables, x_j. Constraint: 1 <= N <= M.

3: X(N) -- DOUBLE PRECISION array Input/Output
On entry: X(j) must be set to a guess at the jth component of the position of the minimum, for j=1,2,...,n. The routine checks the first derivatives calculated by LSFUN2 at the starting point, and so is more likely to detect an error in the user's routine if the initial X(j) are non-zero and mutually distinct. On exit: the lowest point found during the calculations. Thus, if IFAIL = 0 on exit, X(j) is the j th component of the position of the minimum.

4: FSUMSQ -- DOUBLE PRECISION Output
On exit: the value of the sum of squares, F(x), corresponding to the final point stored in X.

5: IW(LIW) -- INTEGER array Workspace

6: LIW -- INTEGER Input
On entry: the length of IW as declared in the (sub)program from which E04GCF is called. Constraint: LIW >= 1.

7: W(LW) -- DOUBLE PRECISION array Workspace

8: LW -- INTEGER Input
On entry: the length of W as declared in the (sub)program from which E04GCF is called. Constraints:

LW >= 2*N*(4 + N + M) + 3*M, if N > 1,

LW >= 11 + 5*M, if N = 1.

9: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. Users who are unfamiliar with this parameter should refer to the Essential

Introduction for details.

On exit: IFAIL = 0 unless the routine detects an error or gives a warning (see Section 6).

For this routine, because the values of output parameters may be useful even if IFAIL /=0 on exit, users are recommended to set IFAIL to -1 before entry. It is then essential to test the value of IFAIL on exit.

5.1. Optional Parameters

LSFUN2 -- SUBROUTINE, supplied by the user.
External Procedure

This routine must be supplied by the user to calculate the vector of values $f_i(x)$ and the Jacobian matrix of first derivatives $\frac{\partial f_i}{\partial x_j}$ at any point x. Since the routine is not a parameter to E04GCF, it must be called LSFUN2. It should be tested separately before being used in conjunction with E04GCF (see the Chapter Introduction).

Its specification is:

```
       SUBROUTINE LSFUN2 (M, N, XC, FVECC, FJACC, LJC)
       INTEGER          M, N, LJC
       DOUBLE PRECISION XC(N), FVECC(M), FJACC(LJC,N)
```

Important: The dimension declaration for FJACC must contain the variable LJC, not an integer constant.

1: M -- INTEGER Input

2: N -- INTEGER Input
On entry: the numbers m and n of residuals and variables, respectively.

3: XC(N) -- DOUBLE PRECISION array Input
On entry: the point x at which the values of the f_i and the $\frac{\partial f_i}{\partial x_j}$ are required.

4: FVECC(M) -- DOUBLE PRECISION array Output

On exit: FVECC(i) must contain the value of f_i at the point x, for i=1,2,...,m.

5: FJACC(LJC,N) -- DOUBLE PRECISION array Output

On exit: FJACC(i,j) must contain the value of $\frac{ddf_i}{ddx_j}$ at the point x, for i=1,2,...,m; j=1,2,...,n.

6: LJC -- INTEGER Input

On entry: the first dimension of the array FJACC.

LSFUN2 must be declared as EXTERNAL in the (sub)program from which E04GCF is called. Parameters denoted as Input must not be changed by this procedure.

6. Error Indicators and Warnings

Errors or warnings specified by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL= 1

On entry N < 1,

or M < N,

or LIW < 1,

or LW < 2*N*(4 + N + M) + 3*M, when N > 1,

or LW < 9 + 5*>M, when N = 1.

IFAIL= 2

There have been 50*n calls of LSFUN2, yet the algorithm does not seem to have converged. This may be due to an awkward function or to a poor starting point, so it is worth restarting E04GCF from the final point held in X.

IFAIL= 3

The final point does not satisfy the conditions for acceptance as a minimum, but no lower point could be found.

IFAIL= 4

An auxiliary routine has been unable to complete a singular value decomposition in a reasonable number of sub-iterations.

IFAIL= 5

IFAIL= 6

IFAIL= 7

IFAIL= 8

There is some doubt about whether the point X found by E04GCF is a minimum of F(x). The degree of confidence in the result decreases as IFAIL increases. Thus, when IFAIL = 5, it is probable that the final x gives a good estimate of the position of a minimum, but when IFAIL = 8 it is very unlikely that the routine has found a minimum.

IFAIL= 9

It is very likely that the user has made an error in forming the derivatives $\frac{\partial f_i}{\partial x_j}$ in LSFUN2.

If the user is not satisfied with the result (e.g. because IFAIL lies between 3 and 8), it is worth restarting the calculations from a different starting point (not the point at which the failure occurred) in order to avoid the region which caused the failure. Repeated failure may indicate some defect in the formulation of the problem.

7. Accuracy

If the problem is reasonably well scaled and a successful exit is made then, for a computer with a mantissa of t decimals, one would expect to get t/2-1 decimals accuracy in the components of x and between t-1 (if F(x) is of order 1 at the minimum) and 2t-2 (if F(x) is close to zero at the minimum) decimals accuracy in F(x).

8. Further Comments

The number of iterations required depends on the number of variables, the number of residuals and their behaviour, and the distance of the starting point from the solution. The number of multiplications performed per iteration of E04GCF varies, but for m>>n is approximately $n*m^2+O(n^3)$. In addition, each iteration makes at least one call of LSFUN2. So, unless the residuals and their derivatives can be evaluated very quickly, the run time will be dominated by the time spent in LSFUN2.

Ideally the problem should be scaled so that the minimum value of the sum of squares is in the range (0,1) and so that at points a unit distance away from the solution the sum of squares is approximately a unit value greater than at the minimum. It is unlikely that the user will be able to follow these recommendations very closely, but it is worth trying (by guesswork), as sensible scaling will reduce the difficulty of the minimization problem, so that E04GCF will take less computer time.

When the sum of squares represents the goodness of fit of a nonlinear model to observed data, elements of the variance-covariance matrix of the estimated regression coefficients can be computed by a subsequent call to E04YCF, using information returned in segments of the workspace array W. See E04YCF for further details.

9. Example

To find the least-squares estimates of x_1, x_2 and x_3 in the model

$$y=x_1+\frac{t_1}{x_2 t_2+x_3 t_3}$$

using the 15 sets of data given in the following table.

y	t_1	t_2	t_3
0.14	1.0	15.0	1.0
0.18	2.0	14.0	2.0
0.22	3.0	13.0	3.0
0.25	4.0	12.0	4.0
0.29	5.0	11.0	5.0
0.32	6.0	10.0	6.0
0.35	7.0	9.0	7.0
0.39	8.0	8.0	8.0
0.37	9.0	7.0	7.0
0.58	10.0	6.0	6.0
0.73	11.0	5.0	5.0
0.96	12.0	4.0	4.0
1.34	13.0	3.0	3.0
2.10	14.0	2.0	2.0
4.39	15.0	1.0	1.0

The program uses (0.5, 1.0, 1.5) as the initial guess at the

```
         position of the minimum.

         The example program is not reproduced here. The source code for
         all example programs is distributed with the NAG Foundation
         Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Finding a minimum of a function

— nage.ht —

```
\begin{page}{manpageXXe04jaf}{NAG Documentation: e04jaf}
\beginscroll
\begin{verbatim}



     E04JAF(3NAG)      Foundation Library (12/10/92)      E04JAF(3NAG)



         E04 -- Minimizing or Maximizing a Function                E04JAF
                 E04JAF -- NAG Foundation Library Routine Document

         Note: Before using this routine, please read the Users' Note for
         your implementation to check implementation-dependent details.
         The symbol (*) after a NAG routine name denotes a routine that is
         not included in the Foundation Library.

         1. Purpose

         E04JAF is an easy-to-use quasi-Newton algorithm for finding a
         minimum of a function F(x ,x ,...,x ), subject to fixed upper and
                                  1  2      n
         lower bounds of the independent variables x ,x ,...,x , using
                                                    1  2      n
         function values only.

         It is intended for functions which are continuous and which have
         continuous first and second derivatives (although it will usually
         work even if the derivatives have occasional discontinuities).
```

2. Specification

```
       SUBROUTINE E04JAF (N, IBOUND, BL, BU, X, F, IW, LIW, W,
      1                   LW, IFAIL)
       INTEGER          N, IBOUND, IW(LIW), LIW, LW, IFAIL
       DOUBLE PRECISION BL(N), BU(N), X(N), F, W(LW)
```

3. Description

This routine is applicable to problems of the form:

$$\text{Minimize } F(x_1,x_2,\ldots,x_n) \text{ subject to } l_j \le x_j \le u_j,\ j=1,2,\ldots,n$$

when derivatives of F(x) are unavailable.

Special provision is made for problems which actually have no bounds on the x_j, problems which have only non-negativity bounds and problems in which $l_1=l_2=\ldots=l_n$ and $u_1=u_2=\ldots=u_n$. The user must supply a subroutine FUNCT1 to calculate the value of F(x) at any point x.

From a starting point supplied by the user there is generated, on the basis of estimates of the gradient and the curvature of F(x), a sequence of feasible points which is intended to converge to a local minimum of the constrained function. An attempt is made to verify that the final point is a minimum.

4. References

[1] Gill P E and Murray W (1976) Minimization subject to bounds on the variables. Report NAC 72. National Physical Laboratory.

5. Parameters

1: N -- INTEGER Input
On entry: the number n of independent variables.
Constraint: N >= 1.

2: IBOUND -- INTEGER Input
On entry: indicates whether the facility for dealing with bounds of special forms is to be used.

It must be set to one of the following values:
IBOUND = 0
if the user will be supplying all the l_j and u_j

individually.

IBOUND = 1
if there are no bounds on any x_j.

IBOUND = 2
if all the bounds are of the form $0<=x_j$.

IBOUND = 3
if $l_1=l_2=...=l_n$ and $u_1=u_2=...=u_n$.

3: BL(N) -- DOUBLE PRECISION array Input/Output
On entry: the lower bounds l_j.

If IBOUND is set to 0, the user must set BL(j) to l_j, for j=1,2,...,n. (If a lower bound is not specified for a particular x_j, the corresponding BL(j) should be set to -10^6.)

If IBOUND is set to 3, the user must set BL(1) to l_1; E04JAF will then set the remaining elements of BL equal to BL(1).
On exit: the lower bounds actually used by E04JAF.

4: BU(N) -- DOUBLE PRECISION array Input/Output
On entry: the upper bounds u_j.

If IBOUND is set to 0, the user must set BU(j) to u_j, for j=1,2,...,n. (If an upper bound is not specified for a particular x_j, the corresponding BU(j) should be set to 10^6.)

If IBOUND is set to 3, the user must set BU(1) to u_1; E04JAF will then set the remaining elements of BU equal to BU(1).
On exit: the upper bounds actually used by E04JAF.

5: X(N) -- DOUBLE PRECISION array Input/Output
On entry: X(j) must be set to an estimate of the jth component of the position of the minimum, for j=1,2,...,n.
On exit: the lowest point found during the calculations.

```
     Thus, if IFAIL = 0 on exit, X(j) is the jth component of the
     position of the minimum.

  6: F -- DOUBLE PRECISION                                 Output
     On exit: the value of F(x) corresponding to the final point
     stored in X.

  7: IW(LIW) -- INTEGER array                           Workspace

  8: LIW -- INTEGER                                         Input
     On entry: the length of IW as declared in the (sub)program
     from which E04JAF is called. Constraint: LIW >= N + 2.

  9: W(LW) -- DOUBLE PRECISION array                    Workspace

 10: LW -- INTEGER                                          Input
     On entry: the length of W as declared in the (sub)program
     from which E04JAF is called. Constraint: LW>=max(N*(N-
     1)/2+12*N,13).

 11: IFAIL -- INTEGER                                Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. Users who are
     unfamiliar with this parameter should refer to the Essential
     Introduction for details.

     On exit: IFAIL = 0 unless the routine detects an error or
     gives a warning (see Section 6).

     For this routine, because the values of output parameters
     may be useful even if IFAIL /=0 on exit, users are
     recommended to set IFAIL to -1 before entry. It is then
     essential to test the value of IFAIL on exit. To suppress
     the output of an error message when soft failure occurs, set
     IFAIL to 1.

 5.1. Optional Parameters

     FUNCT1 -- SUBROUTINE, supplied by the user.
                                           External Procedure
     This routine must be supplied by the user to calculate the
     value of the function F(x) at any point x. Since this
     routine is not a parameter to E04JAF, it must be called
     FUNCT1. It should be tested separately before being used in
     conjunction with E04JAF (see the Chapter Introduction).

     Its specification is:

            SUBROUTINE FUNCT1 (N, XC, FC)
            INTEGER          N
            DOUBLE PRECISION XC(N), FC
```

```
 1:  N -- INTEGER                                        Input
     On entry: the number n of variables.

 2:  XC(N) -- DOUBLE PRECISION array                     Input
     On entry: the point x at which the function value is
     required.

 3:  FC -- DOUBLE PRECISION                             Output
     On exit: the value of the function F at the current
     point x.
    FUNCT1 must be declared as EXTERNAL in the (sub)program
    from which E04JAF is called. Parameters denoted as
    Input must not be changed by this procedure.
```

6. Error Indicators and Warnings

Errors or warnings specified by the routine:

```
IFAIL= 1
     On entry N < 1,

     or       IBOUND < 0,

     or       IBOUND > 3,

     or       IBOUND = 0 and BL(j) > BU(j) for some j,

     or       IBOUND = 3 and BL(1) > BU(1),

     or       LIW < N + 2,

     or       LW<max(13,12*N+N*(N-1)/2).

IFAIL= 2
     There have been 400*n function evaluations, yet the
     algorithm does not seem to be converging. The calculations
     can be restarted from the final point held in X. The error
     may also indicate that F(x) has no minimum.

IFAIL= 3
     The conditions for a minimum have not all been met but a
     lower point could not be found and the algorithm has failed.

IFAIL= 4
     An overflow has occurred during the computation. This is an
     unlikely failure, but if it occurs the user should restart
     at the latest point given in X.

IFAIL= 5
```

IFAIL= 6

IFAIL= 7

IFAIL= 8

There is some doubt about whether the point x found by E04JAF is a minimum. The degree of confidence in the result decreases as IFAIL increases. Thus, when IFAIL = 5 it is probable that the final x gives a good estimate of the position of a minimum, but when IFAIL = 8 it is very unlikely that the routine has found a minimum.

IFAIL= 9

In the search for a minimum, the modulus of one of the variables has become very large ($\sim 10^6$). This indicates that there is a mistake in FUNCT1, that the user's problem has no finite solution, or that the problem needs rescaling (see Section 8).

If the user is dissatisfied with the result (e.g. because IFAIL = 5, 6, 7 or 8), it is worth restarting the calculations from a different starting point (not the point at which the failure occurred) in order to avoid the region which caused the failure. If persistent trouble occurs and the gradient can be calculated, it may be advisable to change to a routine which uses gradients (see the Chapter Introduction).

7. Accuracy

When a successful exit is made then, for a computer with a mantissa of t decimals, one would expect to get about t/2-1 decimals accuracy in x and about t-1 decimals accuracy in F, provided the problem is reasonably well scaled.

8. Further Comments

The number of iterations required depends on the number of variables, the behaviour of F(x) and the distance of the starting point from the solution. The number of operations performed in an iteration of E04JAF is roughly proportional to n^2. In addition, each iteration makes at least m+1 calls of FUNCT1, where m is the number of variables not fixed on bounds. So, unless F(x) can be evaluated very quickly, the run time will be dominated by the time spent in FUNCT1.

Ideally the problem should be scaled so that at the solution the value of F(x) and the corresponding values of x ,x ,...,x are

```
                                                 1  2      n
     each in the range (-1,+1), and so that at points a unit distance
     away from the solution, F is approximately a unit value greater
     than at the minimum. It is unlikely that the user will be able to
     follow these recommendations very closely, but it is worth trying
     (by guesswork), as sensible scaling will reduce the difficulty of
     the minimization problem, so that E04JAF will take less computer
     time.

     9. Example

     To minimize

                      2         2         4          4
           F=(x +10x ) +5(x -x ) +(x -2x ) +10(x -x )
               1    2      3  4     2   3       1  4

     subject to

                              1<=x <=3
                                  1

                              -2<=x <=0
                                   2

                              1<=x <=3,
                                  4

     starting from the initial guess (3, - 1, 0, 1).

     The example program is not reproduced here. The source code for
     all example programs is distributed with the NAG Foundation
     Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Solving linear programming problems

— nage.ht —

```
\begin{page}{manpageXXe04mbf}{NAG Documentation: e04mbf}
\beginscroll
\begin{verbatim}
```

E04 -- Minimizing or Maximizing a Function E04MBF
E04MBF -- NAG Foundation Library Routine Document

Note: Before using this routine, please read the Users' Note for your implementation to check implementation-dependent details. The symbol (*) after a NAG routine name denotes a routine that is not included in the Foundation Library.

1. Purpose

E04MBF is an easy-to-use routine for solving linear programming problems, or for finding a feasible point for such problems. It is not intended for large sparse problems.

2. Specification

```
       SUBROUTINE E04MBF (ITMAX, MSGLVL, N, NCLIN, NCTOTL, NROWA,
      1                   A, BL, BU, CVEC, LINOBJ, X, ISTATE,
      2                   OBJLP, CLAMDA, IWORK, LIWORK, WORK,
      3                   LWORK, IFAIL)
       INTEGER          ITMAX, MSGLVL, N, NCLIN, NCTOTL, NROWA,
      1                 ISTATE(NCTOTL), IWORK(LIWORK), LIWORK,
      2                 LWORK, IFAIL
       DOUBLE PRECISION A(NROWA,N), BL(NCTOTL), BU(NCTOTL), CVEC
      1                 (N), X(N), OBJLP, CLAMDA(NCTOTL), WORK
      2                 (LWORK)
       LOGICAL          LINOBJ
```

3. Description

E04MBF solves linear programming (LP) problems of the form

$$\underset{x \text{ is in } R^n}{\text{Minimize}} \quad c^T x \quad \text{subject to} \quad l \le \begin{pmatrix} x \\ Ax \end{pmatrix} \le u \qquad \text{(LP)}$$

where c is an n element vector and A is an m by n matrix i.e., there are n variables and m general linear constraints. m may be zero in which case the LP problem is subject only to bounds on the variables. Notice that upper and lower bounds are specified for all the variables and constraints. This form allows full generality in specifying other types of constraints. For example the ith constraint may be specified as equality by setting $l_i=u_i$.

If certain bounds are not present the associated elements of l or u can be set to special values that will be treated as -infty or +infty.

The routine allows the linear objective function to be omitted in which case a feasible point for the set of constraints is sought.

The user must supply an initial estimate of the solution.

Users who wish to exercise additional control and users with problems whose solution would benefit from additional flexibility should consider using the comprehensive routine E04NAF.

4. References

[1] Gill P E, Murray W and Wright M H (1981) Practical Optimization. Academic Press.

[2] Gill P E, Murray W, Saunders M A and Wright M H (1983) User's Guide for SOL/QPSOL. Report SOL 83-7. Department of Operations Research, Stanford University.

5. Parameters

1: ITMAX -- INTEGER Input
On entry: an upper bound on the number of iterations to be taken. If ITMAX is not positive, then the value 50 is used in place of ITMAX.

2: MSGLVL -- INTEGER Input
On entry: indicates whether or not printout is required at the final solution. When printing occurs the output is on the advisory message channel (see X04ABF). A description of the printed output is given in Section 5.1. The level of printing is determined as follows:

MSGLVL < 0
No printing.

MSGLVL = 0
Printing only if an input parameter is incorrect, or if the problem is so ill-conditioned that subsequent overflow is likely. This setting is strongly recommended in preference to MSGLVL < 0.

MSGLVL = 1
Printing at the solution.

MSGLVL > 1

```
        Values greater than 1 should normally be used only at
        the direction of NAG; such values may generate large
        amounts of printed output.

 3:  N -- INTEGER                                        Input
     On entry: the number n of variables. Constraint: N >= 1.

 4:  NCLIN -- INTEGER                                    Input
     On entry: the number of general linear constraints in the
     problem. Constraint: NCLIN >= 0.

 5:  NCTOTL -- INTEGER                                   Input
     On entry: the value (N+NCLIN).

 6:  NROWA -- INTEGER                                    Input
     On entry:
     the first dimension of the array A as declared in the
     (sub)program from which E04MBF is called.
     Constraint: NROWA >= max(1,NCLIN).

 7:  A(NROWA,N) -- DOUBLE PRECISION array                Input
     On entry: the leading NCLIN by n part of A must contain the
     NCLIN general constraints, with the coefficients of the ith
     constraint in the ith row of A. If NCLIN = 0, then A is not
     referenced.

 8:  BL(NCTOTL) -- DOUBLE PRECISION array                Input
     On entry: the first n elements of BL must contain the lower
     bounds on the n variables, and when NCLIN > 0, the next
     NCLIN elements of BL must contain the lower bounds on the
     NCLIN general linear constraints. To specify a non-existent
     lower bound (l =-infty), set BL(j)<=-1.0E+20.
                   j

 9:  BU(NCTOTL) -- DOUBLE PRECISION array                Input
     On entry: the first n elements of BU must contain the upper
     bounds on the n variables, and when NCLIN > 0, the next
     NCLIN elements of BU must contain the upper bounds on the
     NCLIN general linear constraints. To specify a non-existent
     upper bound (u =+infty), set BU(j)>=1.0E+20. Constraint:
                   j
     BL(j)<=BU(j), for j=1,2,...,NCTOTL.

10:  CVEC(N) -- DOUBLE PRECISION array                   Input
     On entry: with LINOBJ = .TRUE., CVEC must contain the
     coefficients of the objective function. If LINOBJ = .FALSE.,
     then CVEC is not referenced.

11:  LINOBJ -- LOGICAL                                   Input
     On entry: indicates whether or not a linear objective
```

```
     function is present. If LINOBJ = .TRUE., then the full LP
     problem is solved, but if LINOBJ = .FALSE., only a feasible
     point is found and the array CVEC is not referenced.

12:  X(N) -- DOUBLE PRECISION array                  Input/Output
     On entry: an estimate of the solution, or of a feasible
     point. Even when LINOBJ = .TRUE. it is not necessary for the
     point supplied in X to be feasible. In the absence of better
     information all elements of X may be set to zero. On exit:
     the solution to the LP problem when LINOBJ = .TRUE., or a
     feasible point when LINOBJ = .FALSE..

     When no feasible point exists (see IFAIL = 1 in Section 6)
     then X contains the point for which the sum of the
     infeasibilities is a minimum. On return with IFAIL = 2, 3 or
     4, X contains the point at which E04MBF terminated.

13:  ISTATE(NCTOTL) -- INTEGER array                       Output
     On exit: with IFAIL < 5, ISTATE indicates the status of
     every constraint at the final point. The first n elements of
     ISTATE refer to the upper and lower bounds on the variables
     and when NCLIN > 0 the next NCLIN elements refer to the
     general constraints.

     Their meaning is:
     ISTATE(j) Meaning

     -2        The constraint violates its lower bound. This
               value cannot occur for any element of ISTATE when
               a feasible point has been found.

     -1        The constraint violates its upper bound. This
               value cannot occur for any element of ISTATE when
               a feasible point has been found.

     0         The constraint is not in the working set (is not
               active) at the final point. Usually this means
               that the constraint lies strictly between its
               bounds.

     1         This inequality constraint is in the working set
               (is active) at its lower bound.

     2         This inequality constraint is in the working set
               (is active) at its upper bound.

     3         This constraint is included in the working set (is
               active) as an equality. This value can only occur
               when BL(j) = BU(j).
```

14: OBJLP -- DOUBLE PRECISION Output
On exit: when LINOBJ = .TRUE., then on successful exit, OBJLP contains the value of the objective function at the solution, and on exit with IFAIL = 2, 3 or 4, OBJLP contains the value of the objective function at the point returned in X.

When LINOBJ = .FALSE., then on successful exit OBJLP will be zero and on return with IFAIL = 1, OBJLP contains the minimum sum of the infeasibilities corresponding to the point returned in X.

15: CLAMDA(NCTOTL) -- DOUBLE PRECISION array Output
On exit: when LINOBJ = .TRUE., then on successful exit, or on exit with IFAIL = 2, 3, or 4, CLAMDA contains the Lagrange multipliers (reduced costs) for each constraint with respect to the working set. The first n components of CLAMDA contain the multipliers for the bound constraints on the variables and the remaining NCLIN components contain the multipliers for the general linear constraints.

If ISTATE(j) = 0 so that the jth constraint is not in the working set then CLAMDA(j) is zero. If X is optimal and ISTATE(j) = 1, then CLAMDA(j) should be non-negative, and if ISTATE(j) = 2, then CLAMDA(j) should be non-positive.

When LINOBJ = .FALSE., all NCTOTL elements of CLAMDA are returned as zero.

16: IWORK(LIWORK) -- INTEGER array Workspace

17: LIWORK -- INTEGER Input
On entry: the length of the array IWORK as declared in the (sub)program from which E04MBF is called. Constraint: LIWORK>=2*N.

18: WORK(LWORK) -- DOUBLE PRECISION array Workspace

19: LWORK -- INTEGER Input
On entry: the length of the array WORK as declared in the (sub)program from which E04MBF is called. Constraints:
when N <= NCLIN then

$$LWORK>=2*N^2+6*N+4*NCLIN+NROWA;$$

when 0 <= NCLIN < N then

$$LWORK>=2*(NCLIN+1)^2+4*NCLIN+6*N+NROWA.$$

20: IFAIL -- INTEGER Input/Output

On entry: IFAIL must be set to 0, -1 or 1. Users who are unfamiliar with this parameter should refer to the Essential Introduction for details.

On exit: IFAIL = 0 unless the routine detects an error or gives a warning (see Section 6).

For this routine, because the values of output parameters may be useful even if IFAIL /=0 on exit, users are recommended to set IFAIL to -1 before entry. It is then essential to test the value of IFAIL on exit. To suppress the output of an error message when soft failure occurs, set IFAIL to 1.

5.1. Description of the Printed Output

When MSGLVL = 1, then E04MBF will produce output on the advisory message channel (see X04ABF), giving information on the final point. The following describes the printout associated with each variable.

Output	Meaning
VARBL	The name (V) and index j, for j=1,2,...,n, of the variable.
STATE	The state of the variable. (FR if neither bound is in the working set, EQ for a fixed variable, LL if on its lower bound, UL if on its upper bound and TB if held on a temporary bound.) If the value of the variable lies outside the upper or lower bound then STATE will be ++ or -- respectively.
VALUE	The value of the variable at the final iteration.
LOWER BOUND	The lower bound specified for the variable.
UPPER BOUND	The upper bound specified for the variable.
LAGR MULT	The value of the Lagrange multiplier for the associated bound.
RESIDUAL	The difference between the value of the variable and the nearer of its bounds.

For each of the general constraints the printout is as above with refers to the jth element of Ax, except that VARBL is replaced by:

LNCON	The name (L) and index j, for j = 1,2,...,NCLIN of

```
             the constraint.

6. Error Indicators and Warnings

Errors or warnings specified by the routine:

Note: when MSGLVL=1 a short description of the error is printed.

IFAIL= 1
     No feasible point could be found. Moving violated
     constraints so that they are satisfied at the point returned
     in X gives the minimum moves necessary to make the LP
     problem feasible.

IFAIL= 2
     The solution to the LP problem is unbounded.

IFAIL= 3
     A total of 50 changes were made to the working set without
     altering x. Cycling is probably occurring. The user should
     consider using E04NAF with MSGLVL >= 5 to monitor constraint
     additions and deletions in order to determine whether or not
     cycling is taking place.

IFAIL= 4
     The limit on the number of iterations has been reached.
     Increase ITMAX or consider using E04NAF to monitor progress.

IFAIL= 5
     An input parameter is invalid. Unless MSGLVL < 0 a message
     will be printed.

IFAILOverflow
     If the printed output before the overflow occurred contains
     a warning about serious ill-conditioning in the working set
     when adding the jth constraint, then either the user should
     try using E04NAF and experiment with the magnitude of FEATOL
     (j) in that routine, or the offending linearly dependent
     constraint (with index j) should be removed from the
     problem.

7. Accuracy

The routine implements a numerically stable active set strategy
and returns solutions that are as accurate as the condition of
the LP problem warrants on the machine.

8. Further Comments

The time taken by each iteration is approximately proportional to
```

$\min(n^2, NCLIN^2)$.

Sensible scaling of the problem is likely to reduce the number of iterations required and make the problem less sensitive to perturbations in the data, thus improving the condition of the LP problem. In the absence of better information it is usually sensible to make the Euclidean lengths of each constraint of comparable magnitude. See Gill et al [1] for further information and advice.

Note that the routine allows constraints to be violated by an absolute tolerance equal to the machine precision (see X02AJF(*))

9. Example

To minimize the function

$$-0.02x_1 -0.2x_2 -0.2x_3 -0.2x_4 -0.2x_5 +0.04x_6 +0.04x_7$$

subject to the bounds

$$-0.01 \le x_1 \le 0.01$$
$$-0.1 \le x_2 \le 0.15,$$
$$-0.01 \le x_3 \le 0.03,$$
$$-0.04 \le x_4 \le 0.02,$$
$$-0.1 \le x_5 \le 0.05,$$
$$-0.01 \le x_6$$
$$-0.01 \le x_7$$

and the general constraints

$$x_1 +x_2 +x_3 +x_4 +x_5 +x_6 +x_7 =-0.13$$

$$0.15x_1 +0.04x_2 +0.02x_3 +0.04x_4 +0.02x_5 +0.01x_6 +0.03x_7 \le -0.0049$$

$$0.03x_1 +0.05x_2 +0.08x_3 +0.02x_4 +0.06x_5 +0.01x_6 \le -0.0064$$

$$0.02x +0.04x +0.01x +0.02x +0.02x \le -0.0037$$

```
                 1      2      3      4      5

                  0.02x +0.03x +0.01x <=-0.0012
                       1      2      5

       -0.0992<=0.70x +0.75x +0.80x +0.75x +0.80x +0.97x
                     1      2      3      4      5      6

 -0.003<=0.02x +0.06x +0.08x +0.12x +0.02x +0.01x +0.97x <=0.002
              1      2      3      4      5      6      7

The initial point, which is infeasible, is

                                                         T
        x =(-0.01, -0.03, 0.0, -0.01, -0.1, 0.02, 0.01) .
         0

The example program is not reproduced here. The source code for
all example programs is distributed with the NAG Foundation
Library software and should be available on-line.
```

```
\end{verbatim}
\endscroll
\end{page}
```

Solving linear or quadratic problems

— nage.ht —

```
\begin{page}{manpageXXe04naf}{NAG Documentation: e04naf}
\beginscroll
\begin{verbatim}



     E04NAF(3NAG)      Foundation Library (12/10/92)      E04NAF(3NAG)



          E04 -- Minimizing or Maximizing a Function                E04NAF
                  E04NAF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
```

not included in the Foundation Library.

1. Purpose

E04NAF is a comprehensive routine for solving quadratic programming (QP) or linear programming (LP) problems. It is not intended for large sparse problems.

2. Specification

```
       SUBROUTINE E04NAF (ITMAX, MSGLVL, N, NCLIN, NCTOTL, NROWA,
      1                   NROWH, NCOLH, BIGBND, A, BL, BU, CVEC,
      2                   FEATOL, HESS, QPHESS, COLD, LP, ORTHOG,
      3                   X, ISTATE, ITER, OBJ, CLAMDA, IWORK,
      4                   LIWORK, WORK, LWORK, IFAIL)
       INTEGER          ITMAX, MSGLVL, N, NCLIN, NCTOTL, NROWA,
      1                 NROWH, NCOLH, ISTATE(NCTOTL), ITER, IWORK
      2                 (LIWORK), LIWORK, LWORK, IFAIL
       DOUBLE PRECISION BIGBND, A(NROWA,N), BL(NCTOTL),
      1                 BU(NCTOTL), CVEC(N), FEATOL(NCTOTL), HESS
      2                 (NROWH,NCOLH), X(N), OBJ, CLAMDA(NCTOTL),
      3                 WORK(LWORK)
       LOGICAL          COLD, LP, ORTHOG
       EXTERNAL         QPHESS
```

3. Description

E04NAF is essentially identical to the subroutine SOL/QPSOL described in Gill et al [1].

E04NAF is designed to solve the quadratic programming (QP) problem - the minimization of a quadratic function subject to a set of linear constraints on the variables. The problem is assumed to be stated in the following form:

$$\text{Minimize } c^T x + \frac{1}{2} x^T H x \quad \text{subject to } l \le \begin{pmatrix} x \\ Ax \end{pmatrix} \le u, \qquad (1)$$

where c is a constant n-vector and H is a constant n by n symmetric matrix; note that H is the Hessian matrix (matrix of second partial derivatives) of the quadratic objective function. The matrix A is m by n, where m may be zero; A is treated as a dense matrix.

The constraints involving A will be called the general constraints. Note that upper and lower bounds are specified for all the variables and for all the general constraints. The form of (1) allows full generality in specifying other types of constraints. In particular, an equality constraint is specified by setting $l_i=u_i$. If certain bounds are not present, the associated elements of l or u can be set to special values that will be treated as -infty or +infty.

The user must supply an initial estimate of the solution to (1), and a subroutine that computes the product Hx for any given vector x. If H is positive-definite or positive semi-definite, E04NAF will obtain a global minimum; otherwise, the solution obtained will be a local minimum (which may or may not be a global minimum). If H is defined as the zero matrix, E04NAF will solve the resulting linear programming (LP) problem; however, this can be accomplished more efficiently by setting a logical variable in the call of the routine (see the parameter LP in Section 5).

E04NAF allows the user to provide the indices of the constraints that are believed to be exactly satisfied at the solution. This facility, known as a warm start, can lead to significant savings in computational effort when solving a sequence of related problems.

The method has two distinct phases. In the first (the LP phase), an iterative procedure is carried out to determine a feasible point. In this context, feasibility is defined by a user-provided array FEATOL; the jth constraint is considered satisfied if its violation does not exceed FEATOL(j). The second phase (the QP phase) generates a sequence of feasible iterates in order to minimize the quadratic objective function. In both phases, a subset of the constraints - called the working set - is used to define the search direction at each iteration; typically, the working set includes constraints that are satisfied to within the corresponding tolerances in the FEATOL array.

We now briefly describe a typical iteration in the QP phase. Let x_k denote the estimate of the solution at the kth iteration; the next iterate is defined by

$$x_{k+1}=x_k+(\alpha)_k p_k$$

where p_k is an n-dimensional search direction and $(\alpha)_k$ is a scalar step length. Assume that the working (active) set contains t_k linearly independent constraints, and let C_k denote the matrix of coefficients of the bounds and general constraints in the current working set.

Let Z_k denote a matrix whose columns form a basis for the null space of C_k, so that $C_k Z_k=0$. (Note that Z_k has n_z columns, where $n_z=n-t_k$.) The vector $Z_k^T(c+Hx_k)$ is called the projected gradient at x_k. If the projected gradient is zero at x_k (i.e., x_k is a constrained stationary point in the subspace defined by Z_k), Lagrange multipliers $(\lambda)_k$ are defined as the solution of the compatible overdetermined system

$$C_k^T(\lambda)_k=c+Hx_k \qquad (2)$$

The Lagrange multiplier (lambda) corresponding to an inequality constraint in the working set is said to be optimal if (lambda)<=0 when the associated constraint is at its upper bound, or if (lambda)>=0 when the associated constraint is at its lower bound. If a multiplier is non-optimal, the objective function can be reduced by deleting the corresponding constraint (with index JDEL, see Section 5.1) from the working set.

If the projected gradient at x_k is non-zero, the search direction p_k is defined as

$$p_k=Z_k p_z \qquad (3)$$

where p_z is an n_z-vector. In effect, the constraints in the working set are treated as equalities, by constraining p_k to lie within the subspace of vectors orthogonal to the rows of C_k. This definition ensures that $C_k p_k=0$, and hence the values of the constraints in the working set are not altered by any move along p_k.

The vector p_z is obtained by solving the equations

$$Z_k^T H Z_k p_z = -Z_k^T (c+Hx_k) \qquad (4)$$

(The matrix $Z_k^T H Z_k$ is called the projected Hessian matrix.) If the projected Hessian is positive-definite, the vector defined by (3) and (4) is the step to the minimum of the quadratic function in the subspace defined by Z_k.

If the projected Hessian is positive-definite and x_k+p_k is feasible, $(alpha)_k$ will be taken as unity. In this case, the projected gradient at x_{k+1} will be zero (see NORM ZTG in Section 5.1), and Lagrange multipliers can be computed (see Gill et al [2]). Otherwise, $(alpha)_k$ is set to the step to the 'nearest' constraint (with index JADD, see Section 5.1), which is added to the working set at the next iteration.

The matrix Z_k is obtained from the TQ factorization of C_k, in which C_k is represented as

$$C_k Q=(0\ T_k) \qquad (5)$$

where T_k is reverse-triangular. It follows from (5) that Z_k may be taken as the first n_z columns of Q. If the projected Hessian is positive-definite, (3) is solved using the Cholesky factorization

$$Z_k^T H Z_k = R_k^T R_k$$

where R_k is upper triangular. These factorizations are updated as constraints enter or leave the working set (see Gill et al [2] for further details).

An important feature of E04NAF is the treatment of indefiniteness in the projected Hessian. If the projected Hessian is positive-definite, it may become indefinite only when a constraint is deleted from the working set. In this case, a temporary modification (of magnitude HESS MOD, see Section 5.1) is added to the last diagonal element of the Cholesky factor. Once a modification has occurred, no further constraints are deleted from the working set until enough constraints have been added so that the projected Hessian is again positive-definite. If equation (1) has a finite solution, a move along the direction obtained by solving (4) with the modified Cholesky factor must encounter a constraint that is not already in the working set.

In order to resolve indefiniteness in this way, we must ensure that the projected Hessian is positive-definite at the first iterate in the QP phase. Given the n_z by n_z projected Hessian, a step-wise Cholesky factorization is performed with symmetric interchanges (and corresponding rearrangement of the columns of Z), terminating if the next step would cause the matrix to become indefinite. This determines the largest possible positive-definite principal sub-matrix of the (permuted) projected Hessian. If n_R steps of the Cholesky factorization have been successfully completed, the relevant projected Hessian is an n_R by n_R positive-definite matrix $Z_R^T HZ_R$, where Z_R comprises the first n_R columns of Z. The quadratic function will subsequently be minimized within subspaces of reduced dimension until the full projected Hessian is positive-definite.

If a linear program is being solved and there are fewer general constraints than variables, the method moves from one vertex to another while minimizing the objective function. When necessary, an initial vertex is defined by temporarily fixing some of the variables at their initial values.

Several strategies are used to control ill-conditioning in the working set. One such strategy is associated with the FEATOL array. Allowing the jth constraint to be violated by as much as FEATOL(j) often provides a choice of constraints that could be added to the working set. When a choice exists, the decision is based on the conditioning of the working set. Negative steps are occasionally permitted, since x_k may violate the constraint to be added.

4. References

[1] Gill P E, Murray W, Saunders M A and Wright M H (1983) User's Guide for SOL/QPSOL. Report SOL 83-7. Department of Operations Research, Stanford University.

[2] Gill P E, Murray W, Saunders M A and Wright M H (1982) The design and implementation of a quadratic programming algorithm. Report SOL 82-7. Department of Operations Research, Stanford University.

[3] Gill P E, Murray W and Wright M H (1981) Practical Optimization. Academic Press.

5. Parameters

1: ITMAX -- INTEGER Input
On entry: an upper bound on the number of iterations to be taken during the LP phase or the QP phase. If ITMAX is not positive, then the value 50 is used in place of ITMAX.

2: MSGLVL -- INTEGER Input
On entry: MSGLVL must indicate the amount of intermediate output desired (see Section 5.1 for a description of the printed output). All output is written to the current advisory message unit (see X04ABF). For MSGLVL >= 10, each level includes the printout for all lower levels.

Value	Definition
<0	No printing.
0	Printing only if an input parameter is incorrect, or if the working set is so ill-conditioned that subsequent overflow is likely. This setting is strongly recommended in preference to MSGLVL < 0.
1	The final solution only.
5	One brief line of output for each constraint addition or deletion (no printout of the final solution).
>=10	The final solution and one brief line of output for each constraint addition or deletion.
>=15	At each iteration, X, ISTATE, and the indices of the free variables (i.e.,the variables not currently held on a bound).
>=20	At each iteration, the Lagrange multiplier estimates and the general constraint values.

```
     >=30    At each iteration, the diagonal elements of the
             matrix T associated with the TQ factorization of the
             working set, and the diagonal elements of the
             Cholesky factor R of the projected Hessian.

     >=80    Debug printout.

     99      The arrays CVEC and HESS.

  3: N -- INTEGER                                         Input
     On entry: the number, n, of variables. Constraint: N >= 1.

  4: NCLIN -- INTEGER                                     Input
     On entry: the number of general linear constraints in the
     problem. Constraint: NCLIN >= 0.

  5: NCTOTL -- INTEGER                                    Input
     On entry: the value (N+NCLIN).

  6: NROWA -- INTEGER                                     Input
     On entry:
     the first dimension of the array A as declared in the
     (sub)program from which E04NAF is called.
     Constraint: NROWA >= max(1,NCLIN).

  7: NROWH -- INTEGER                                     Input
     On entry: the first dimension of the array HESS as declared
     in the (sub)program from which E04NAF is called.
     Constraint: NROWH >= 1.

  8: NCOLH -- INTEGER                                     Input
     On entry: the column dimension of the array HESS as declared
     in the (sub)program from which E04NAF is called.
     Constraint: NCOLH >= 1.

  9: BIGBND -- DOUBLE PRECISION                           Input
     On entry: BIGBND must denote an 'infinite' component of l
     and u. Any upper bound greater than or equal to BIGBND will
     be regarded as plus infinity, and a lower bound less than or
     equal to -BIGBND will be regarded as minus infinity.
     Constraint: BIGBND > 0.0.

 10: A(NROWA,N) -- DOUBLE PRECISION array                 Input
     On entry: the leading NCLIN by n part of A must contain the
     NCLIN general constraints, with the ith constraint in the i
     th row of A. If NCLIN = 0, then A is not referenced.

 11: BL(NCTOTL) -- DOUBLE PRECISION array                 Input
     On entry: the lower bounds for all the constraints, in the
     following order. The first n elements of BL must contain the
```

```
     lower bounds on the variables. If NCLIN > 0, the next NCLIN
     elements of BL must contain the lower bounds for the general
     linear constraints. To specify a non-existent lower bound
     (i.e., l =-infty), the value used must satisfy BL(j)<=-
             j
     BIGBND To specify the jth constraint as an equality, the
     user must set BL(j) = BU(j). Constraint: BL(j) <= BU(j),
     j=1,2,...,NCTOTL.

12:  BU(NCTOTL) -- DOUBLE PRECISION array                   Input
     On entry: the upper bounds for all the constraints, in the
     following order. The first n elements of BU must contain the
     upper bounds on the variables. If NCLIN > 0, the next NCLIN
     elements of BU must contain the upper bounds for the general
     linear constraints. To specify a non-existent upper bound
     (i.e., u =+infty), the value used must satisfy BU(j) >=
             j
     BIGBND. To specify the jth constraint as an equality, the
     user must set BU(j) = BL(j). Constraint: BU(j) >= BL(j),
     j=1,2,...,NCTOTL.

13:  CVEC(N) -- DOUBLE PRECISION array                      Input
     On entry: the coefficients of the linear term of the
     objective function (the vector c in equation  (1)).

14:  FEATOL(NCTOTL) -- DOUBLE PRECISION array               Input
     On entry: a set of positive tolerances that define the
     maximum permissible absolute violation in each constraint in
     order for a point to be considered feasible, i.e., if the
     violation in constraint j is less than FEATOL(j), the point
     is considered to be feasible with respect to the jth
     constraint. The ordering of the elements of FEATOL is the
     same as that described above for BL.

     The elements of FEATOL should not be too small and a warning
     message will be printed on the current advisory message
     channel if any element of FEATOL is less than the machine
     precision (see X02AJF(*)). As the elements of FEATOL
     increase, the algorithm is less likely to encounter
     difficulties with ill-conditioning and degeneracy. However,
     larger values of FEATOL(j) mean that constraint j could be
     violated by a significant amount. It is recommended that
     FEATOL(j) be set to a value equal to the largest acceptable
     violation for constraint j. For example, if the data
     defining the constraints are of order unity and are correct
     to about 6 decimal digits, it would be appropriate to choose
                    -6
     FEATOL(j) as 10   for all relevant j. Often the square root
     of the machine precision is a reasonable choice if the
     constraint is well scaled.
```

```
15:  HESS(NROWH,NCOLH) -- DOUBLE PRECISION array           Input
     On entry: HESS may be used to store the Hessian matrix H of
     equation (1) if desired. HESS is accessed only by the
     subroutine QPHESS and is not accessed if LP = .TRUE.. Refer
     to the specification of QPHESS (below) for further details
     of how HESS may be used to pass data to QPHESS.

16:  QPHESS -- SUBROUTINE, supplied by the user.
                                             External Procedure
     QPHESS must define the product of the Hessian matrix H and a
     vector x. The elements of H need not be defined explicitly.
     QPHESS is not accessed if LP is set to .TRUE. and in this
     case QPHESS may be the dummy routine E04NAN. (E04NAN is
     included in the NAG Foundation Library and so need not be
     supplied by the user. Its name may be implementation-
     dependent: see the Users' Note for your implementation for
     details.)

     Its specification is:

            SUBROUTINE QPHESS (N, NROWH, NCOLH, JTHCOL,
           1                   HESS, X, HX)
            INTEGER          N, NROWH, NCOLH, JTHCOL
            DOUBLE PRECISION HESS(NROWH,NCOLH), X(N), HX(N)

      1:  N -- INTEGER                                     Input
          On entry: the number n of variables.

      2:  NROWH -- INTEGER                                 Input
          On entry: the row dimension of the array HESS.

      3:  NCOLH -- INTEGER                                 Input
          On entry: the column dimension of the array HESS.

      4:  JTHCOL -- INTEGER                                Input
          The input parameter JTHCOL is included to allow
          flexibility for the user in the special situation when
          x is the jth co-ordinate vector (i.e.,the jth column of
          the identity matrix). This may be of interest because
          the product Hx is then the jth column of H, which can
          sometimes be computed very efficiently. The user may
          code QPHESS to take advantage of this case. On entry:
          if JTHCOL = j, where j>0, HX must contain column JTHCOL
          of H, and hence special code may be included in QPHESS
          to test JTHCOL if desired. However, special code is not
          necessary, since the vector x always contains column
          JTHCOL of the identity matrix whenever QPHESS is called
          with JTHCOL > 0.
```

```
 5:  HESS(NROWH,NCOLH) -- DOUBLE PRECISION array       Input
     On entry: the Hessian matrix H.

     In some cases, it may be desirable to use a one-
     dimensional array to transmit data or workspace to
     QPHESS; HESS should then be declared with dimension
     (NROWH) in the (sub)program from which E04NAF is called
     and the parameter NCOLH must be 1.

     In other situations, it may be desirable to compute Hx
     without accessing HESS - for example, if H is sparse or
     has special structure. (This is illustrated in the
     subroutine QPHES1 in the example program in Section 9.)
     The parameters HESS, NROWH and NCOLH may then refer to
     any convenient array.

     When MSGLVL = 99, the (possibly undefined) contents of
     HESS will be printed, except if NROWH and NCOLH are
     both 1. Also printed are the results of calling QPHESS
     with JTHCOL = 1,2,...,n.

 6:  X(N) -- DOUBLE PRECISION array                    Input
     On entry: the vector x.

 7:  HX(N) -- DOUBLE PRECISION array                  Output
     On exit: HX must contain the product Hx.
QPHESS must be declared as EXTERNAL in the (sub)program
from which E04NAF is called. Parameters denoted as
Input must not be changed by this procedure.
```

```
17: COLD -- LOGICAL                                      Input
    On entry: COLD must indicate whether the user has specified
    an initial estimate of the active set of constraints. If
    COLD is set to .TRUE., the initial working set is determined
    by E04NAF. If COLD is set to .FALSE. (a 'warm start'), the
    user must define the ISTATE array which gives the status of
    each constraint with respect to the working set. E04NAF will
    override the user's specification of ISTATE if necessary, so
    that a poor choice of working set will not cause a fatal
    error.

    The warm start option is particularly useful when E04NAF is
    called repeatedly to solve related problems.

18: LP -- LOGICAL                                        Input
    On entry: if LP = .FALSE., E04NAF will solve the specified
    quadratic programming problem. If LP = .TRUE., E04NAF will
    treat H as zero and solve the resulting linear programming
    problem; in this case, the parameters HESS and QPHESS will
    not be referenced.
```

19: ORTHOG -- LOGICAL Input

On entry: ORTHOG must indicate whether orthogonal transformations are to be used in computing and updating the TQ factorization of the working set

$$A_s Q=(0\ T),$$

where A_s is a sub-matrix of A and T is reverse-triangular.

If ORTHOG = .TRUE., the TQ factorization is computed using Householder reflections and plane rotations, and the matrix Q is orthogonal. If ORTHOG = .FALSE., stabilized elementary transformations are used to maintain the factorization, and Q is not orthogonal. A rule of thumb in making the choice is that orthogonal transformations require more work, but provide greater numerical stability. Thus, we recommend setting ORTHOG to .TRUE. if the problem is reasonably small or the active set is ill-conditioned. Otherwise, setting ORTHOG to .FALSE. will often lead to a reduction in solution time with negligible loss of reliability.

20: X(N) -- DOUBLE PRECISION array Input/Output

On entry: an estimate of the solution. In the absence of better information all elements of X may be set to zero. On exit: from E04NAF, X contains the best estimate of the solution.

21: ISTATE(NCTOTL) -- INTEGER array Input/Output

On entry: with COLD as .FALSE., ISTATE must indicate the status of every constraint with respect to the working set. The ordering of ISTATE is as follows; the first n elements of ISTATE refer to the upper and lower bounds on the variables and elements n+1 through n + NCLIN refer to the upper and lower bounds on Ax. The significance of each possible value of ISTATE(j) is as follows:

ISTATE(j)	Meaning
-2	The constraint violates its lower bound by more than FEATOL(j). This value of ISTATE cannot occur after a feasible point has been found.
-1	The constraint violates its upper bound by more than FEATOL(j). This value of ISTATE cannot occur after a feasible point has been found.
0	The constraint is not in the working set. Usually, this means that the constraint lies strictly between its bounds.
1	This inequality constraint is included in the

```
          working set at its lower bound. The value of the
          constraint is within FEATOL(j) of its lower bound.

2         This inequality constraint is included in the
          working set at its upper bound. The value of the
          constraint is within FEATOL(j) of its upper bound.

3         The constraint is included in the working set as
          an equality. This value of ISTATE can occur only
          when BL(j) = BU(j). The corresponding constraint
          is within FEATOL(j) of its required value.
If COLD = .TRUE., ISTATE need not be set by the user.
However, when COLD = .FALSE., every element of ISTATE must
be set to one of the values given above to define a
suggested initial working set (which will be changed by
E04NAF if necessary). The most likely values are:
ISTATE(j) Meaning

0         The corresponding constraint should not be in the
          initial working set.

1         The constraint should be in the initial working
          set at its lower bound.

2         The constraint should be in the initial working
          set at its upper bound.

3         The constraint should be in the initial working
          set as an equality. This value must not be
          specified unless BL(j) = BU(j). The values 1, 2 or
          3 all have the same effect when BL(j) = BU(j).
Note that if E04NAF has been called previously with the same
values of N and NCLIN, ISTATE already contains satisfactory
values. On exit: when E04NAF exits with IFAIL set to 0, 1 or
3, the values in the array ISTATE indicate the status of the
constraints in the active set at the solution. Otherwise,
ISTATE indicates the composition of the working set at the
final iterate.

22: ITER -- INTEGER                                       Output
    On exit: the number of iterations performed in either the LP
    phase or the QP phase, whichever was last entered.

    Note that ITER is reset to zero after the LP phase.

23: OBJ -- DOUBLE PRECISION                               Output
    On exit: the value of the quadratic objective function at x
    if x is feasible (IFAIL <= 5), or the sum of infeasibilities
    at x otherwise (6 <= IFAIL <= 8).
```

24: CLAMDA(NCTOTL) -- DOUBLE PRECISION array Output
On exit: the values of the Lagrange multiplier for each constraint with respect to the current working set. The ordering of CLAMDA is as follows; the first n components contain the multipliers for the bound constraints on the variables, and the remaining components contain the multipliers for the general linear constraints. If ISTATE(j) = 0 (i.e.,constraint j is not in the working set), CLAMDA(j) is zero. If x is optimal and ISTATE(j) = 1, CLAMDA(j) should be non-negative; if ISTATE(j) = 2, CLAMDA(j) should be non-positive.

25: IWORK(LIWORK) -- INTEGER array Workspace

26: LIWORK -- INTEGER Input
On entry:
the dimension of the array IWORK as declared in the (sub)program from which E04NAF is called.
Constraint: LIWORK>=2*N.

27: WORK(LWORK) -- DOUBLE PRECISION array Workspace

28: LWORK -- INTEGER Input
On entry:
the dimension of the array WORK as declared in the (sub)program from which E04NAF is called.

```
Constrai if LP = .FALSE. or NCLIN >= N then
nts:                2
         LWORK>=2*N +4*N*NCLIN+NROWA.

         if LP = .TRUE. and NCLIN < N then
                                  2
         LWORK>=2*(NCLIN+1) +4*N+2*NCLIN+NROWA.
```

If MSGLVL > 0, the amount of workspace provided and the amount of workspace required are output on the current advisory message unit (as defined by X04ABF). As an alternative to computing LWORK from the formula given above, the user may prefer to obtain an appropriate value from the output of a preliminary run with a positive value of MSGLVL and LWORK set to 1 (E04NAF will then terminate with IFAIL = 9).

29: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. Users who are unfamiliar with this parameter should refer to the Essential Introduction for details.

On exit: IFAIL = 0 unless the routine detects an error or gives a warning (see Section 6).

For this routine, because the values of output parameters may be useful even if IFAIL /=0 on exit, users are recommended to set IFAIL to -1 before entry. It is then essential to test the value of IFAIL on exit. To suppress the output of an error message when soft failure occurs, set IFAIL to 1.

IFAIL contains zero on exit if x is a strong local minimum. i.e., the projected gradient is neglible, the Lagrange multipliers are optimal, and the projected Hessian is positive-definite. In some cases, a zero value of IFAIL means that x is a global minimum (e.g. when the Hessian matrix is positive-definite).

5.1. Description of the Printed Output

When MSGLVL >= 5, a line of output is produced for every change in the working set (thus, several lines may be printed during a single iteration).

To aid interpretation of the printed results, we mention the convention for numbering the constraints: indices 1 through to n refer to the bounds on the variables, and when NCLIN > 0 indices n+1 through to n + NCLIN refer to the general constraints. When the status of a constraint changes, the index of the constraint is printed, along with the designation L (lower bound), U (upper bound) or E (equality).

In the LP phase, the printout includes the following:

ITN is the iteration count.

JDEL is the index of the constraint deleted from the working set. If JDEL is zero, no constraint was deleted.

JADD is the index of the constraint added to the working set. If JADD is zero, no constraint was added.

STEP is the step taken along the computed search direction.

COND T is a lower bound on the condition number of the matrix of predicted active constraints.

NUMINF is the number of violated constraints (infeasibilities).

SUMINF is a weighted sum of the magnitudes of the constraint violations.

LPOBJ — is the value of the linear objective function $c^T x$. It is printed only if LP = .TRUE..

During the QP phase, the printout includes the following:

ITN — is the iteration count (reset to zero after the LP phase).

JDEL — is the index of the constraint deleted from the working set. If JDEL is zero, no constraint was deleted.

JADD — is the index of the constraint added to the working set. If JADD is zero, no constraint was added.

STEP — is the step $(alpha)_k$ taken along the direction of search (if STEP is 1.0, the current point is a minimum in the subspace defined by the current working set).

NHESS — is the number of calls to subroutine QPHESS.

OBJECTIVE — is the value of the quadratic objective function.

NCOLZ — is the number of columns of Z (see Section 3). In general, it is the dimension of the subspace in which the quadratic is currently being minimized.

NORM GFREE — is the Euclidean norm of the gradient of the objective function with respect to the free variables, i.e. variables not currently held at a bound (NORM GFREE is not printed if ORTHOG = . FALSE.). In some cases, the objective function and gradient are updated rather than recomputed. If so, this entry will be -- to indicate that the gradient with respect to the free variables has not been computed.

NORM QTG — is a weighted norm of the gradient of the objective function with respect to the free variables (NORM QTG is not printed if ORTHOG = . TRUE.). In some cases, the objective function and gradient are updated rather than recomputed. If so, this entry will be -- to indicate that the gradient with respect to the free variables has not been computed.

NORM ZTG is the Euclidean norm of the projected gradient (see Section 3).

COND T is a lower bound on the condition number of the matrix of constraints in the working set.

COND ZHZ is a lower bound on the condition number of the projected Hessian matrix.

HESS MOD is the correction added to the diagonal of the projected Hessian to ensure that a satisfactory Cholesky factorization exists (see Section 3). When the projected Hessian is sufficiently positive-definite, HESS MOD will be zero.

When MSGLVL = 1 or MSGLVL >= 10, the summary printout at the end of execution of E04NAF includes a listing of the status of every constraint. Note that default names are assigned to all variables and constraints.

The following describes the printout for each variable.

VARBL is the name (V) and index j, j=1,2,...,n, of the variable.

STATE gives the state of the variable (FR if neither bound is in the working set, EQ if a fixed variable, LL if on its lower bound, UL if on its upper bound, TB if held on a temporary bound). If VALUE lies outside the upper or lower bounds by more than FEATOL(j), STATE will be ++ or -- respectively.

VALUE is the value of the variable at the final iteration.

LOWER BOUND is the lower bound specified for the variable.

UPPER BOUND is the upper bound specified for the variable.

LAGR MULT is the value of the Lagrange multiplier for the associated bound constraint. This will be zero if STATE is FR. If x is optimal and STATE is LL, the multiplier should be non-negative; if STATE is UL, the multiplier should be non-positive.

RESIDUAL is the difference between the variable and the nearer of its bounds BL(j) and BU(j).

For each of the general constraints the printout is as above with refers to the jth element of Ax, except that VARBL is replaced by

LNCON The name (L) and index j, j=1,2,...,NCLIN, of the constraint.

6. Error Indicators and Warnings

Errors or warnings specified by the routine:

IFAIL= 1
x is a weak local minimum (the projected gradient is negligible, the Lagrange multipliers are optimal, but the projected Hessian is only semi-definite). This means that the solution is not unique.

IFAIL= 2
The solution appears to be unbounded, i.e., the quadratic function is unbounded below in the feasible region. This value of IFAIL occurs when a step of infinity would have to be taken in order to continue the algorithm.

IFAIL= 3
x appears to be a local minimum, but optimality cannot be verified because some of the Lagrange multipliers are very small in magnitude.

E04NAF has probably found a solution. However, the presence of very small Lagrange multipliers means that the predicted active set may be incorrect, or that x may be only a constrained stationary point rather than a local minimum. The method in E04NAF is not guaranteed to find the correct active set when there are very small multipliers. E04NAF attempts to delete constraints with zero multipliers, but this does not necessarily resolve the issue. The determination of the correct active set is a combinatorial problem that may require an extremely large amount of time. The occurrence of small multipliers often (but not always) indicates that there are redundant constraints.

IFAIL= 4
The iterates of the QP phase could be cycling, since a total of 50 changes were made to the working set without altering x.

This value will occur if 50 iterations are performed in the QP phase without changing x. The user should check the printed output for a repeated pattern of constraint deletions and additions. If a sequence of constraint changes is being repeated, the iterates are probably cycling. (E04NAF does not contain a method that is guaranteed to avoid cycling, which would be combinatorial in nature.) Cycling may occur in two circumstances: at a constrained stationary point where there are some small or zero Lagrange multipliers (see the discussion of IFAIL = 3); or at a point (usually a vertex) where the constraints that are satisfied exactly are nearly linearly dependent. In the latter case, the user has the option of identifying the offending dependent constraints and removing them from the problem, or restarting the run with larger values of FEATOL for nearly dependent constraints. If E04NAF terminates with IFAIL = 4, but no suspicious pattern of constraint changes can be observed, it may be worthwhile to restart with the final x (with or without the warm start option).

IFAIL= 5

The limit of ITMAX iterations was reached in the QP phase before normal termination occurred.

The value of ITMAX may be too small. If the method appears to be making progress (e.g. the objective function is being satisfactorily reduced), increase ITMAX and rerun E04NAF (possibly using the warm start facility to specify the initial working set). If ITMAX is already large, but some of the constraints could be nearly linearly dependent, check the output for a repeated pattern of constraints entering and leaving the working set. (Near-dependencies are often indicated by wide variations in size in the diagonal elements of the T matrix, which will be printed if MSGLVL >= 30.) In this case, the algorithm could be cycling (see the comments for IFAIL = 4).

IFAIL= 6

The LP phase terminated without finding a feasible point, and hence it is not possible to satisfy all the constraints to within the tolerances specified by the FEATOL array. In this case, the final iterate will reveal values for which there will be a feasible point (e.g. a feasible point will exist if the feasibility tolerance for each violated constraint exceeds its RESIDUAL at the final point). The modified problem (with altered values in FEATOL) may then be solved using a warm start.

The user should check that there are no constraint redundancies. If the data for the jth constraint are accurate only to the absolute precision (delta), the user should ensure that the value of FEATOL(j) is greater than (delta). For example, if all elements of A are of order unity and are accurate only to three decimal places, every component of FEATOL should be at least 10^{-3}.

IFAIL= 7
The iterates may be cycling during the LP phase; see the comments above under IFAIL = 4.

IFAIL= 8
The limit of ITMAX iterations was reached during the LP phase. See comments above under IFAIL = 5.

IFAIL= 9
An input parameter is invalid.

Overflow
If the printed output before the overflow error contains a warning about serious ill-conditioning in the working set when adding the jth constraint, it may be possible to avoid the difficulty by increasing the magnitude of FEATOL(j) and rerunning the program. If the message recurs even after this change, the offending linearly dependent constraint (with index j) must be removed from the problem. If a warning message did not precede the fatal overflow, the user should contact NAG.

7. Accuracy

The routine implements a numerically stable active set strategy and returns solutions that are as accurate as the condition of the QP problem warrants on the machine.

8. Further Comments

The number of iterations depends upon factors such as the number of variables and the distances of the starting point from the solution. The number of operations performed per iteration is roughly proportional to $(NFREE)^2$, where NFREE (NFREE<=n) is the number of variables fixed on their upper or lower bounds.

Sensible scaling of the problem is likely to reduce the number of iterations required and make the problem less sensitive to perturbations in the data, thus improving the condition of the QP problem. See the Chapter Introduction and Gill et al [1] for further information and advice.

9. Example

To minimize the function $c^T x + \frac{1}{2} x^T Hx$, where

```
                                        2

                                                                  T
                    c=[-0.02,-0.2,-0.2,-0.2,-0.2,0.04,0.04]

                                 [2 0 0 0 0  0  0]
                                 [0 2 0 0 0  0  0]
                                 [0 0 2 2 0  0  0]
                               H=[0 0 2 2 0  0  0]
                                 [0 0 0 0 2  0  0]
                                 [0 0 0 0 0 -2 -2]
                                 [0 0 0 0 0 -2 -2]

          subject to the bounds

\end{verbatim}
\endscroll
\end{page}
```

Minimize an arbitrary smooth constrainted function

— nage.ht —

```
\begin{page}{manpageXXe04ucf}{NAG Documentation: e04ucf}
\beginscroll
\begin{verbatim}



   E04UCF(3NAG)                 E04UCF                  E04UCF(3NAG)



        E04 -- Minimizing or Maximizing a Function                E04UCF
                E04UCF -- NAG Foundation Library Routine Document

        Note: Before using this routine, please read the Users' Note for
        your implementation to check implementation-dependent details.
        The symbol (*) after a NAG routine name denotes a routine that is
        not included in the Foundation Library.

        Note for users via the Axiom system: the interface to this routine
        has been enhanced for use with Axiom and is slightly different to
        that offered in the standard version of the Foundation Library.  In
```

particular, the optional parameters of the NAG routine are now included in the parameter list. These are described in section 5.1.2, below.

1. Purpose

E04UCF is designed to minimize an arbitrary smooth function subject to constraints, which may include simple bounds on the variables, linear constraints and smooth nonlinear constraints. (E04UCF may be used for unconstrained, bound-constrained and linearly constrained optimization.) The user must provide subroutines that define the objective and constraint functions and as many of their first partial derivatives as possible. Unspecified derivatives are approximated by finite differences. All matrices are treated as dense, and hence E04UCF is not intended for large sparse problems.

E04UCF uses a sequential quadratic programming (SQP) algorithm in which the search direction is the solution of a quadratic programming (QP) problem. The algorithm treats bounds, linear constraints and nonlinear constraints separately.

2. Specification

```
       SUBROUTINE E04UCF (N, NCLIN, NCNLN, NROWA, NROWJ, NROWR,
      1                   A, BL, BU, CONFUN, OBJFUN, ITER,
      2                   ISTATE, C, CJAC, CLAMDA, OBJF, OBJGRD,
      3                   R, X, IWORK, LIWORK, WORK, LWORK,
      4                   IUSER, USER, STA, CRA, DER, FEA, FUN,
      5                   HES, INFB, INFS, LINF, LINT, LIST,
      6                   MAJI, MAJP, MINI, MINP, MON, NONF,
      7                   OPT, STE, STAO, STAC, STOO, STOC, VE,
      8                   IFAIL)
       INTEGER          N, NCLIN, NCNLN, NROWA, NROWJ, NROWR,
      1                 ITER, ISTATE(N+NCLIN+NCNLN), IWORK(LIWORK)
      2                 , LIWORK, LWORK, IUSER(*), DER, MAJI,
      3                 MAJP, MINI, MINP, MON, STAO, STAC, STOO,
      4                 STOC, VE, IFAIL
       DOUBLE PRECISION A(NROWA,*), BL(N+NCLIN+NCNLN), BU
      1                 (N+NCLIN+NCNLN), C(*), CJAC(NROWJ,*),
      2                 CLAMDA(N+NCLIN+NCNLN), OBJF, OBJGRD(N), R
      3                 (NROWR,N), X(N), WORK(LWORK), USER(*),
      4                 CRA, FEA, FUN,  INFB, INFS, LINF, LINT,
      5                 NONF, OPT, STE
       LOGICAL          LIST, STA, HES
       EXTERNAL         CONFUN, OBJFUN
```

3. Description

E04UCF is designed to solve the nonlinear programming problem --

the minimization of a smooth nonlinear function subject to a set of constraints on the variables. The problem is assumed to be stated in the following form:

$$\underset{x \text{ is in } R^n}{\text{Minimize}} \quad F(x) \quad \text{subject to} \quad l \le \left\{ \begin{array}{c} x \\ A_L x \\ c(x) \end{array} \right\} \le u, \qquad (1)$$

where $F(x)$, the objective function, is a nonlinear function, A_L is an n_L by n constant matrix, and $c(x)$ is an n_N element vector of nonlinear constraint functions. (The matrix A_L and the vector $c(x)$ may be empty.) The objective function and the constraint functions are assumed to be smooth, i.e., at least twice-continuously differentiable. (The method of E04UCF will usually solve (1) if there are only isolated discontinuities away from the solution.)

This routine is essentially identical to the subroutine SOL/NPSOL described in Gill et al [8].

Note that upper and lower bounds are specified for all the variables and for all the constraints.

An equality constraint can be specified by setting $l_i=u_i$. If certain bounds are not present, the associated elements of l or u can be set to special values that will be treated as -infty or +infty.

If there are no nonlinear constraints in (1) and F is linear or quadratic then one of E04MBF, E04NAF or E04NCF(*) will generally be more efficient. If the problem is large and sparse the MINOS package (see Murtagh and Saunders [13]) should be used, since E04UCF treats all matrices as dense.

The user must supply an initial estimate of the solution to (1), together with subroutines that define $F(x)$, $c(x)$ and as many first partial derivatives as possible; unspecified derivatives are approximated by finite differences.

The objective function is defined by subroutine OBJFUN, and the nonlinear constraints are defined by subroutine CONFUN. On every call, these subroutines must return appropriate values of the objective and nonlinear constraints. The user should also provide the available partial derivatives. Any unspecified derivatives are approximated by finite differences; see Section 5.1 for a discussion of the optional parameter Derivative Level. Just before either OBJFUN or CONFUN is called, each element of the current gradient array OBJGRD or CJAC is initialised to a special value. On exit, any element that retains the value is estimated by finite differences. Note that if there are nonlinear costraints, then the first call to CONFUN will precede the first call to OBJFUN.

For maximum reliability, it is preferable for the user to provide all partial derivatives (see Chapter 8 of Gill et al [10], for a detailed discussion). If all gradients cannot be provided, it is similarly advisable to provide as many as possible. While developing the subroutines OBJFUN and CONFUN, the optional parameter Verify (see Section 5.1) should be used to check the calculation of any known gradients.

E04UCF implements a sequential quadratic programming (SQP) method. The document for E04NCF(*) should be consulted in conjunction with this document.

In the rest of this section we briefly summarize the main features of the method of E04UCF. Where possible, explicit reference is made to the names of variables that are parameters of subroutines E04UCF or appear in the printed output (see Section 5.2).

At a solution of (1), some of the constraints will be active, i.e., satisfied exactly. An active simple bound constraint implies that the corresponding variable is fixed at its bound, and hence the variables are partitioned into fixed and free variables. Let C denote the m by n matrix of gradients of the active general linear and nonlinear constraints. The number of fixed variables will be denoted by n_{FX}, with n_{FR} ($n_{FR}=n-n_{FX}$) the number of free variables. The subscripts 'FX' and 'FR' on a vector or matrix will denote the vector or matrix composed of the components corresponding to fixed or free variables.

A point x is a first-order Kuhn-Tucker point for (1) (see, e.g., Powell [14]) if the following conditions hold:

(i) x is feasible;

(ii) there exist vectors (xi) and (lambda) (the Lagrange multiplier vectors for the bound and general constraints) such that

$$g=C^T(\lambda)+(\xi), \qquad (2)$$

where g is the gradient of F evaluated at x, and $(\xi)_j=0$ if the jth variable is free.

(iii) The Lagrange multiplier corresponding to an inequality constraint active at its lower bound must be non-negative, and non-positive for an inequality constraint active at its upper bound.

Let Z denote a matrix whose columns form a basis for the set of vectors orthogonal to the rows of C_{FR}; i.e., $C_{FR}Z=0$. An equivalent statement of the condition (2) in terms of Z is

$$Z^T g_{FR}=0.$$

The vector $Z^T g_{FR}$ is termed the projected gradient of F at x. Certain additional conditions must be satisfied in order for a first-order Kuhn-Tucker point to be a solution of (1) (see, e.g., Powell [14]).

The method of E04UCF is a sequential quadratic programming (SQP) method. For an overview of SQP methods, see, for example, Fletcher [5], Gill et al [10] and Powell [15].

The basic structure of E04UCF involves major and minor iterations. The major iterations generate a sequence of iterates $\{x_k\}$ that converge to x^*, a first-order Kuhn-Tucker point of (1).

At a typical major iteration, the new iterate $\bar{x}$ is defined by

$$\bar{x}=x+(\alpha)p \qquad (3)$$

where x is the current iterate, the non-negative scalar (alpha) is the step length, and p is the search direction. (For simplicity, we shall always consider a typical iteration and avoid reference to the index of the iteration.) Also associated with each major iteration are estimates of the Lagrange multipliers and a prediction of the active set.

The search direction p in (3) is the solution of a quadratic programming subproblem of the form

$$\text{Minimize } g^T p+\frac{1}{2}p^T Hp, \quad \text{subject to } \bar{l}\le\begin{Bmatrix} p \\ A p \end{Bmatrix}\le\bar{u}, \qquad (4)$$

$$p \qquad 2 \qquad \left\{\begin{matrix} L \\ A_N p \end{matrix}\right\}$$

where g is the gradient of F at x, the matrix H is a positive-definite quasi-Newton approximation to the Hessian of the Lagrangian function (see Section 8.3), and A_N is the Jacobian matrix of c evaluated at x. (Finite-difference estimates may be used for g and A_N; see the optional parameter Derivative Level in Section 5.1.) Let l in (1) be partitioned into three sections: l_B, l_L and l_N, corresponding to the bound, linear and nonlinear constraints. The vector $\bar{l}$ in (4) is similarly partitioned, and is defined as

$$\bar{l}_B=l_B-x, \quad \bar{l}_L=l_L-A_L x, \quad \text{and } \bar{l}_N=l_N-c,$$

where c is the vector of nonlinear constraints evaluated at x.

The vector $\bar{u}$ is defined in an analogous fashion.

The estimated Lagrange multipliers at each major iteration are the Lagrange multipliers from the subproblem (4) (and similarly for the predicted active set). (The numbers of bounds, general linear and nonlinear constraints in the QP active set are the quantities Bnd, Lin and Nln in the printed output of E04UCF.) In E04UCF, (4) is solved using E04NCF(*). Since solving a quadratic program as an iterative procedure, the minor iterations of E04UCF are the iterations of E04NCF(*). (More details about solving the subproblem are given in Section 8.1.)

Certain matrices associated with the QP subproblem are relevant in the major iterations. Let the subscripts 'FX' and 'FR' refer to the predicted fixed and free variables, and let C denote the m by n matrix of gradients of the general linear and nonlinear constraints in the predicted active set. First, we have available the TQ factorization of C_{FR}:

$$C_{FR} Q_{FR} =(0 \; T), \qquad (5)$$

where T is a nonsingular m by m reverse-triangular matrix (i.e., $t_{ij}=0$ if $i+j<m$), and the non-singular n_{FR} by n_{FR} matrix Q_{FR} is

the product of orthogonal transformations (see Gill et al [6]). Second, we have the upper-triangular Cholesky factor R of the transformed and re-ordered Hessian matrix

$$R^T R = H_Q = Q^T \tilde{H} Q, \qquad (6)$$

where $\tilde{H}$ is the Hessian H with rows and columns permuted so that the free variables are first, and Q is the n by n matrix

$$Q = \begin{pmatrix} Q_{FR} & \\ & I_{FX} \end{pmatrix}, \qquad (7)$$

with I_{FX} the identity matrix of order n_{FX}. If the columns of Q_{FR} are partitioned so that

$$Q_{FR} = (Z\ Y),$$

the n_z ($n_z = n_{FR} - m$) columns of Z form a basis for the null space of C_{FR}. The matrix Z is used to compute the projected gradient $Z^T g_{FR}$ at the current iterate. (The values Nz, Norm Gf and Norm Gz printed by E04UCF give n_z and the norms of g_{FR} and $Z^T g_{FR}$.)

A theoretical characteristic of SQP methods is that the predicted active set from the QP subproblem (4) is identical to the correct active set in a neighbourhood of x^*. In E04UCF, this feature is exploited by using the QP active set from the previous iteration as a prediction of the active set for the next QP subproblem, which leads in practice to optimality of the subproblems in only one iteration as the solution is approached. Separate treatment of bound and linear constraints in E04UCF also saves computation in factorizing C_{FR} and H_Q.

Once p has been computed, the major iteration proceeds by determining a step length (alpha) that produces a 'sufficient decrease' in an augmented Lagrangian merit function (see Section 8.2). Finally, the approximation to the transformed Hessian matrix H_Q is updated using a modified BFGS quasi-Newton update (see Section 8.3) to incorporate new curvature information obtained in the move from x to $\bar{x}$.

On entry to E04UCF, an iterative procedure from E04NCF(*) is executed, starting with the user-provided initial point, to find a point that is feasible with respect to the bounds and linear constraints (using the tolerance specified by Linear Feasibility Tolerance see Section 5.1). If no feasible point exists for the bound and linear constraints, (1) has no solution and E04UCF terminates. Otherwise, the problem functions will thereafter be evaluated only at points that are feasible with respect to the bounds and linear constraints. The only exception involves variables whose bounds differ by an amount comparable to the finite-difference interval (see the discussion of Difference Interval in Section 5.1). In contrast to the bounds and linear constraints, it must be emphasised that the nonlinear constraints will not generally be satisfied until an optimal point is reached.

Facilities are provided to check whether the user-provided gradients appear to be correct (see the optional parameter Verify in Section 5.1). In general, the check is provided at the first point that is feasible with respect to the linear constraints and bounds. However, the user may request that the check be performed at the initial point.

In summary, the method of E04UCF first determines a point that satisfies the bound and linear constraints. Thereafter, each iteration includes:

(a) the solution of a quadratic programming subproblem;

(b) a linesearch with an augmented Lagrangian merit function; and

(c) a quasi-Newton update of the approximate Hessian of the Lagrangian function.

These three procedures are described in more detail in Section 8.

4. References

[1] Dennis J E Jr and More J J (1977) Quasi-Newton Methods, Motivation and Theory. SIAM Review. 19 46--89.

[2] Dennis J E Jr and Schnabel R B (1981) A New Derivation of Symmetric Positive-Definite Secant Updates. Nonlinear

Programming 4. (ed O L Mangasarian, R R Meyer and S M. Robinson) Academic Press. 167--199.

[3] Dennis J E Jr and Schnabel R B (1983) Numerical Methods for Unconstrained Optimixation and Nonlinear Equations. Prentice-Hall.

[4] Dongarra J J, Du Croz J J, Hammarling S and Hanson R J (1985) A Proposal for an Extended set of Fortran Basic Linear Algebra Subprograms. SIGNUM Newsletter. 20 (1) 2--18.

[5] Fletcher R (1981) Practical Methods of Optimization, Vol 2. Constrained Optimization. Wiley.

[6] Gill P E, Murray W, Saunders M A and Wright M H (1984) User's Guide for SOL/QPSOL Version 3.2. Report SOL 84-5. Department of Operations Research, Stanford University.

[7] Gill P E, Murray W, Saunders M A and Wright M H (1984) Procedures for Optimization Problems with a Mixture of Bounds and General Linear Constraints. ACM Trans. Math. Softw. 10 282--298.

[8] Gill P E, Hammarling S, Murray W, Saunders M A and Wright M H (1986) User's Guide for LSSOL (Version 1.0). Report SOL 86-1. Department of Operations Research, Stanford University.

[9] Gill P E, Murray W, Saunders M A and Wright M H (1986) Some Theoretical Properties of an Augmented Lagrangian Merit Function. Report SOL 86-6R. Department of Operations Research, Stanford University.

[10] Gill P E, Murray W and Wright M H (1981) Practical Optimization. Academic Press.

[11] Hock W and Schittkowski K (1981) Test Examples for Nonlinear Programming Codes. Lecture Notes in Economics and Mathematical Systems. 187 Springer-Verlag.

[12] Lawson C L, Hanson R J, Kincaid D R and Krogh F T (1979) Basic Linear Algebra Subprograms for Fortran Usage. ACM Trans. Math. Softw. 5 308--325.

[13] Murtagh B A and Saunders M A (1983) MINOS 5.0 User's Guide. Report SOL 83-20. Department of Operations Research, Stanford University.

[14] Powell M J D (1974) Introduction to Constrained Optimization. Numerical Methods for Constrained

Optimization. (ed P E Gill and W Murray) Academic Press. 1--28.

[15] Powell M J D (1983) Variable Metric Methods in Constrained Optimization. Mathematical Programming: The State of the Art. (ed A Bachem, M Groetschel and B Korte) Springer-Verlag. 288--311.

5. Parameters

1: N -- INTEGER Input
On entry: the number, n, of variables in the problem.
Constraint: N > 0.

2: NCLIN -- INTEGER Input
On entry: the number, n_L, of general linear constraints in the problem. Constraint: NCLIN >= 0.

3: NCNLN -- INTEGER Input
On entry: the number, n_N, of nonlinear constraints in the problem. Constraint: NCNLN >= 0.

4: NROWA -- INTEGER Input
On entry:
the first dimension of the array A as declared in the (sub)program from which E04UCF is called.
Constraint: NROWA >= max(1,NCLIN).

5: NROWJ -- INTEGER Input
On entry:
the first dimension of the array CJAC as declared in the (sub)program from which E04UCF is called.
Constraint: NROWJ >= max(1,NCNLN).

6: NROWR -- INTEGER Input
On entry:
the first dimension of the array R as declared in the (sub)program from which E04UCF is called.
Constraint: NROWR >= N.

7: A(NROWA,*) -- DOUBLE PRECISION array Input
The second dimension of the array A must be >= N for NCLIN > 0. On entry: the ith row of the array A must contain the ith row of the matrix A_L of general linear constraints in (1). That is, the ith row contains the coefficients of the ith general linear constraint, for i = 1,2,...,NCLIN.

If NCLIN = 0 then the array A is not referenced.

8: BL(N+NCLIN+NCNLN) -- DOUBLE PRECISION array Input
On entry: the lower bounds for all the constraints, in the following order. The first n elements of BL must contain the lower bounds on the variables. If NCLIN > 0, the next n_L elements of BL must contain the lower bounds on the general linear constraints. If NCNLN > 0, the next n_N elements of BL must contain the lower bounds for the nonlinear constraints. To specify a non-existent lower bound (i.e., l_j =-infty), the value used must satisfy BL(j)<=-BIGBND, where BIGBND is the value of the optional parameter Infinite Bound Size whose default value is 10^{10} (see Section 5.1). To specify the jth constraint as an equality, the user must set BL(j) = BU(j) = (beta), say, where |(beta)|<BIGBND. Constraint: BL(j) <= BU(j), for j=1,2,...,N+NCLIN+NCNLN.

9: BU(N+NCLIN+NCNLN) -- DOUBLE PRECISION array Input
On entry: the upper bounds for all the constraints in the following order. The first n elements of BU must contain the upper bounds on the variables. If NCLIN > 0, the next n_L elements of BU must contain the upper bounds on the general linear constraints. If NCNLN > 0, the next n_N elements of BU must contain the upper bounds for the nonlinear constraints. To specify a non-existent upper bound (i.e., u_j =+infty), the value used must satisfy BU(j) >= BIGBND, where BIGBND is the value of the optional parameter Infinite Bound Size, whose default value is 10^{10} (see Section 5.1). To specify the jth constraint as an equality, the user must set BU(j) = BL(j) = (beta), say, where |(beta)| < BIGBND. Constraint: BU(j) >= BL(j), for j=1,2,...,N+NCLIN+NCNLN.

10: CONFUN -- SUBROUTINE, supplied by the user. External Procedure
CONFUN must calculate the vector c(x) of nonlinear constraint functions and (optionally) its Jacobian for a specified n element vector x. If there are no nonlinear constraints (NCNLN=0), CONFUN will never be called by E04UCF and CONFUN may be the dummy routine E04UDM. (E04UDM is included in the NAG Foundation Library and so need not be supplied by the user. Its name may be implementation-dependent: see the Users' Note for your implementation for

details.) If there are nonlinear constraints, the first call to CONFUN will occur before the first call to OBJFUN.

Its specification is:

```
 SUBROUTINE CONFUN (MODE, NCNLN, N, NROWJ, NEEDC,
1                   X, C, CJAC, NSTATE, IUSER,
2                   USER)
 INTEGER          MODE, NCNLN, N, NROWJ, NEEDC
1                 (NCNLN), NSTATE, IUSER(*)
 DOUBLE PRECISION X(N), C(NCNLN), CJAC(NROWJ,N),
1                 USER(*)
```

1: MODE -- INTEGER Input/Output
On entry: MODE indicates the values that must be assigned during each call of CONFUN. MODE will always have the value 2 if all elements of the Jacobian are available, i.e., if Derivative Level is either 2 or 3 (see Section 5.1). If some elements of CJAC are unspecified, E04UCF will call CONFUN with MODE = 0, 1, or 2:

If MODE = 2, only the elements of C corresponding to positive values of NEEDC must be set (and similarly for the available components of the rows of CJAC).

If MODE = 1, the available components of the rows of CJAC corresponding to positive values in NEEDC must be set. Other rows of CJAC and the array C will be ignored.

If MODE = 0, the components of C corresponding to positive values in NEEDC must be set. Other components and the array CJAC are ignored. On exit: MODE may be set to a negative value if the user wishes to terminate the solution to the current problem. If MODE is negative on exit from CONFUN then E04UCF will terminate with IFAIL set to MODE.

2: NCNLN -- INTEGER Input
On entry: the number, n_N, of nonlinear constraints.

3: N -- INTEGER Input
On entry: the number, n, of variables.

4: NROWJ -- INTEGER Input
On entry: the first dimension of the array CJAC.

5: NEEDC(NCNLN) -- INTEGER array Input

```
On entry: the indices of the elements of C or CJAC that
must be evaluated by CONFUN. If NEEDC(i)>0 then the ith
element of C and/or the ith row of CJAC (see parameter
MODE above) must be evaluated at x.

6: X(N) -- DOUBLE PRECISION array                   Input
   On entry: the vector x of variables at which the
   constraint functions are to be evaluated.

7: C(NCNLN) -- DOUBLE PRECISION array              Output
   On exit: if NEEDC(i)>0 and MODE = 0 or 2, C(i) must
   contain the value of the ith constraint at x. The
   remaining components of C, corresponding to the non-
   positive elements of NEEDC, are ignored.

8: CJAC(NROWJ,N) -- DOUBLE PRECISION array         Output
   On exit: if NEEDC(i)>0 and MODE = 1 or 2, the ith row
   of CJAC must contain the available components of the
   vector (nabla)c  given by
                  i
                       ( ddc   ddc       ddc )
                       (    i     i         i)T
             (nabla)c =( ----, ----,..., ----) ,
                     i ( ddx   ddx       ddx )
                       (    1     2         n)
          ddc
             i
   where  ---- is the partial derivative of the ith
          ddx
             j
   constraint with respect to the jth variable, evaluated
   at the point x. See also the parameter NSTATE below.
   The remaining rows of CJAC, corresponding to non-
   positive elements of NEEDC, are ignored.

   If all constraint gradients (Jacobian elements) are
   known (i.e., Derivative Level = 2 or 3; see Section 5.1)
   any constant elements may be assigned to CJAC one
   time only at the start of the optimization. An element
   of CJAC that is not subsequently assigned in CONFUN
   will retain its initial value throughout. Constant
   elements may be loaded into CJAC either before the call
   to E04UCF or during the first call to CONFUN (signalled
   by the value NSTATE = 1). The ability to preload
   constants is useful when many Jacobian elements are
   identically zero, in which case CJAC may be initialised
   to zero and non-zero elements may be reset by CONFUN.

   Note that constant non-zero elements do affect the
   values of the constraints. Thus, if CJAC(i,j) is set to
```

a constant value, it need not be reset in subsequent calls to CONFUN, but the value CJAC(i,j)*X(j) must nonetheless be added to C(i).

It must be emphasized that, if Derivative Level < 2, unassigned elements of CJAC are not treated as constant; they are estimated by finite differences, at non-trivial expense. If the user does not supply a value for Difference Interval (see Section 5.1), an interval for each component of x is computed automatically at the start of the optimization. The automatic procedure can usually identify constant elements of CJAC, which are then computed once only by finite differences.

9: NSTATE -- INTEGER Input

On entry: if NSTATE = 1 then E04UCF is calling CONFUN for the first time. This parameter setting allows the user to save computation time if certain data must be read or calculated only once.

10: IUSER(*) -- INTEGER array User Workspace

11: USER(*) -- DOUBLE PRECISION array User Workspace

CONFUN is called from E04UCF with the parameters IUSER and USER as supplied to E04UCF. The user is free to use the arrays IUSER and USER to supply information to CONFUN as an alternative to using COMMON.

CONFUN must be declared as EXTERNAL in the (sub)program from which E04UCF is called. Parameters denoted as Input must not be changed by this procedure.

11: OBJFUN -- SUBROUTINE, supplied by the user. External Procedure

OBJFUN must calculate the objective function F(x) and (optionally) the gradient g(x) for a specified n element vector x.

Its specification is:

```
       SUBROUTINE OBJFUN (MODE, N, X, OBJF, OBJGRD,
      1                   NSTATE, IUSER, USER)
       INTEGER          MODE, N, NSTATE, IUSER(*)
       DOUBLE PRECISION X(N), OBJF, OBJGRD(N), USER(*)
```

1: MODE -- INTEGER Input/Output

On entry: MODE indicates the values that must be assigned during each call of OBJFUN.

MODE will always have the value 2 if all components of

```
    the objective gradient are specified by the user, i.e.,
    if Derivative Level is either 1 or 3. If some gradient
    elements are unspecified, E04UCF will call OBJFUN with
    MODE = 0, 1 or 2.
         If MODE = 2, compute OBJF and the available
         components of OBJGRD.

         If MODE = 1, compute all available components of
         OBJGRD; OBJF is not required.

         If MODE = 0, only OBJF needs to be computed;
         OBJGRD is ignored.
    On exit: MODE may be set to a negative value if the
    user wishes to terminate the solution to the current
    problem. If MODE is negative on exit from OBJFUN, then
    E04UCF will terminate with IFAIL set to MODE.

 2: N -- INTEGER                                      Input
    On entry: the number, n, of variables.

 3: X(N) -- DOUBLE PRECISION array                    Input
    On entry: the vector x of variables at which the
    objective function is to be evaluated.

 4: OBJF -- DOUBLE PRECISION                         Output
    On exit: if MODE = 0 or 2, OBJF must be set to the
    value of the objective function at x.

 5: OBJGRD(N) -- DOUBLE PRECISION array              Output
    On exit: if MODE = 1 or 2, OBJGRD must return the
    available components of the gradient evaluated at x.

 6: NSTATE -- INTEGER                                 Input
    On entry: if NSTATE = 1 then E04UCF is calling OBJFUN
    for the first time. This parameter setting allows the
    user to save computation time if certain data must be
    read or calculated only once.

 7: IUSER(*) -- INTEGER array                User Workspace

 8: USER(*) -- DOUBLE PRECISION array        User Workspace
    OBJFUN is called from E04UCF with the parameters IUSER
    and USER as supplied to E04UCF. The user is free to use
    the arrays IUSER and USER to supply information to
    OBJFUN as an alternative to using COMMON.
OBJFUN must be declared as EXTERNAL in the (sub)program
from which E04UCF is called. Parameters denoted as
Input must not be changed by this procedure.

12: ITER -- INTEGER                                  Output
```

On exit: the number of iterations performed.

13: ISTATE(N+NCLIN+NCNLN) -- INTEGER array Input/Output
On entry: ISTATE need not be initialised if E04UCF is called with (the default) Cold Start option. The ordering of ISTATE is as follows. The first n elements of ISTATE refer to the upper and lower bounds on the variables, elements n+1 through $n+n_L$ refer to the upper and lower bounds on $A_L x$, and elements $n+n_L+1$ through $n+n_L+n_N$ refer to the upper and lower bounds on c(x). When a Warm Start option is chosen, the elements of ISTATE corresponding to the bounds and linear constraints define the initial working set for the procedure that finds a feasible point for the linear constraints and bounds. The active set at the conclusion of this procedure and the elements of ISTATE corresponding to nonlinear constraints then define the initial working set for the first QP subproblem. Possible values for ISTATE(j) are:

ISTATE(j)	Meaning
0	The corresponding constraint is not in the initial QP working set.
1	This inequality constraint should be in the working set at its lower bound.
2	This inequality constraint should be in the working set at its upper bound.
3	This equality constraint should be in the initial working set. This value must not be specified unless BL(j) = BU(j). The values 1,2 or 3 all have the same effect when BL(j) = BU(j).

Note that if E04UCF has been called previously with the same values of N, NCLIN and NCNLN, ISTATE already contains satisfactory values. If necessary, E04UCF will override the user's specification of ISTATE so that a poor choice will not cause the algorithm to fail. On exit: with IFAIL = 0 or 1, the values in the array ISTATE correspond to the active set of the final QP subproblem, and are a prediction of the status of the constraints at the solution of the problem. Otherwise, ISTATE indicates the composition of the QP working set at the final iterate. The significance of each possible value of ISTATE(j) is as follows:

-2 This constraint violates its lower bound by more than the appropriate feasibility tolerance (see the optional parameters LinearFeasibility Tolerance and Nonlinear Feasibility Tolerance in

Section 5.1). This value can occur only when no feasible point can be found for a QP subproblem.

-1 This constraint violates its upper bound by more than the appropriate feasibility tolerance (see the optional parameters Linearear Feasibility Tolerance and Nonlinear Feasibility Tolerance in Section 5.1). This value can occur only when no feasible point can be found for a QP subproblem.

0 The constraint is satisfied to within the feasibility tolerance, but is not in the working set.

1 This inequality constraint is included in the QP working set at its upper bound.

2 This inequality constraint is included in the QP working set at its upper bound.

3 This constraint is included in the QP working set as an equality. This value of ISTATE can occur only when BL(j) = BU(j).

14: C(*) -- DOUBLE PRECISION array Output
Note: the dimension of the array C must be at least max(1,NCNLN).
On exit: if NCNLN > 0, C(i) contains the value of the ith nonlinear constraint function c_i at the final iterate, for i=1,2,...,NCNLN. If NCNLN = 0, then the array C is not referenced.

15: CJAC(NROWJ,*) -- DOUBLE PRECISION array Input/Output
Note: the second dimension of the array CJAC must be at least N for NCNLN >0 and 1 otherwise On entry: in general, CJAC need not be initialised before the call to E04UCF. However, if Derivative Level = 3, the user may optionally set the constant elements of CJAC (see parameter NSTATE in the description of CONFUN). Such constant elements need not be re-assigned on subsequent calls to CONFUN. If NCNLN = 0, then the array CJAC is not referenced. On exit: if NCNLN > 0, CJAC contains the Jacobian matrix of the nonlinear constraint functions at the final iterate, i.e., CJAC(i,j) contains the partial derivative of the ith constraint function with respect to the jth variable, for i=1,2,..., NCNLN; j = 1,2,...,N. (See the discussion of parameter CJAC under CONFUN.)

16: CLAMDA(N+NCLIN+NCNLN) -- DOUBLE PRECISION array Input/Output

On entry: CLAMDA need not be initialised if E04UCF is called with the (default) Cold Start option. With the Warm Start option, CLAMDA must contain a multiplier estimate for each nonlinear constraint with a sign that matches the status of the constraint specified by the ISTATE array (as above). The ordering of CLAMDA is as follows; the first n elements contain the multipliers for the bound constraints on the variables, elements $n+1$ through $n+n_L$ contain the multipliers for the general linear constraints, and elements $n+n_L+1$ through $n+n_L+n_N$ contain the multipliers for the nonlinear constraints. If the jth constraint is defined as 'inactive' by the initial value of the ISTATE array, CLAMDA(j) should be zero; if the jth constraint is an inequality active at its lower bound, CLAMDA(j) should be non-negative; if the j th constraint is an inequality active at its upper bound, CLAMDA(j) should be non-positive. On exit: the values of the QP multipliers from the last QP subproblem. CLAMDA(j) should be non-negative if ISTATE(j) = 1 and non-positive if ISTATE(j) = 2.

17: OBJF -- DOUBLE PRECISION Output
On exit: the value of the objective function, F(x), at the final iterate.

18: OBJGRD(N) -- DOUBLE PRECISION array Output
On exit: the gradient (or its finite-difference approximation) of the objective function at the final iterate.

19: R(NROWR,N) -- DOUBLE PRECISION array Input/Output
On entry: R need not be initialised if E04UCF is called with a Cold Start option (the default), and will be taken as the identity. With a Warm Start R must contain the upper-triangular Cholesky factor R of the initial approximation of the Hessian of the Lagrangian function, with the variables in the natural order. Elements not in the upper-triangular part of R are assumed to be zero and need not be assigned. On exit: if Hessian = No, (the default; see Section 5.1), R contains the upper-triangular Cholesky factor R of $Q^T\tilde{H}Q$, an estimate of the transformed and re-ordered Hessian of the Lagrangian at x (see (6) in Section 3). If Hessian = Yes, R contains the upper-triangular Cholesky factor R of H, the approximate (untransformed) Hessian of the Lagrangian, with the variables in the natural order.

20: X(N) -- DOUBLE PRECISION array Input/Output

```
     On entry: an initial estimate of the solution. On exit: the
     final estimate of the solution.

21:  IWORK(LIWORK) -- INTEGER array                     Workspace

22:  LIWORK -- INTEGER                                      Input
     On entry:
     the dimension of the array IWORK as declared in the
     (sub)program from which E04UCF is called.
     Constraint: LIWORK>=3*N+NCLIN+2*NCNLN.

23:  WORK(LWORK) -- DOUBLE PRECISION array              Workspace

24:  LWORK -- INTEGER                                       Input
     On entry:
     the dimension of the array WORK as declared in the
     (sub)program from which E04UCF is called.
     Constraints:
     if NCLIN = NCNLN = 0 then
           LWORK >=20*N

     if NCNLN = 0 and NCLIN > 0 then
                     2
           LWORK >=2*N +20*N+11*NCLIN

     if NCNLN > 0 and NCLIN >= 0 then
                    2
           LWORK>=2*N +N*NCLIN+20*N*NCNLN+20*N+ 11*NCLIN+21*NCNLN

     If Major Print Level > 0, the required amounts of workspace
     are output on the current advisory message channel (see
     X04ABF). As an alternative to computing LIWORK and LWORK
     from the formulas given above, the user may prefer to obtain
     appropriate values from the output of a preliminary run with
     a positive value of Major Print Level and LIWORK and LWORK
     set to 1. (E04UCF will then terminate with IFAIL = 9.)

25:  IUSER(*) -- INTEGER array                     User Workspace
     Note: the dimension of the array IUSER must be at least 1.
     IUSER is not used by E04UCF, but is passed directly to
     routines CONFUN and OBJFUN and may be used to pass
     information to those routines.

26:  USER(*) -- DOUBLE PRECISION array             User Workspace
     Note: the dimension of the array USER must be at least 1.
     USER is not used by E04UCF, but is passed directly to
     routines CONFUN and OBJFUN and may be used to pass
     information to those routines.

27:  IFAIL -- INTEGER                                Input/Output
```

On entry: IFAIL must be set to 0, -1 or 1. Users who are unfamiliar with this parameter should refer to the Essential Introduction for details.

On exit: IFAIL = 0 unless the routine detects an error or gives a warning (see Section 6).

For this routine, because the values of output parameters may be useful even if IFAIL /=0 on exit, users are recommended to set IFAIL to -1 before entry. It is then essential to test the value of IFAIL on exit.

E04UCF returns with IFAIL = 0 if the iterates have converged to a point x that satisfies the first-order Kuhn-Tucker conditions to the accuracy requested by the optional parameter Optimality Tolerance (see Section 5.1), i.e., the projected gradient and active constraint residuals are negligible at x.

The user should check whether the following four conditions are satisfied:

(i) the final value of Norm Gz is significantly less than that at the starting point;

(ii) during the final major iterations, the values of Step and ItQP are both one;

(iii) the last few values of both Norm Gz and Norm C become small at a fast linear rate;

(iv) Cond Hz is small.

If all these conditions hold, x is almost certainly a local minimum of (1). (See Section 9 for a specific example.)

5.1. Optional Input Parameters

Several optional parameters in E04UCF define choices in the behaviour of the routine. In order to reduce the number of formal parameters of E04UCF these optional parameters have associated default values (see Section 5.1.3) that are appropriate for most problems. Therefore the user need only specify those optional parameters whose values are to be different from their default values.

The remainder of this section can be skipped by users who wish to use the default values for all optional prameters. A complete list of optional parameters and their default values is given in Section 5.1.3

5.1.1. Specification of the optional parameters

```
Optional parameters may be specified by calling one, or both, of
E04UDF and E04UEF prior to a call to E04UCF.

E04UDF reads options from an external options file, with Begin
and End as the first and last lines respectively and each
intermediate line defining a single optional parameter. For
example,

      Begin
        Print Level = 1
      End

The call

      CALL E04UDF (IOPTNS, INFORM)

can then be used to read the file on unit IOPTNS. INFORM will be
zero on successful exit. E04UDF should be consulted for a full
description of this method of supplying optional parameters.

E04UEF can be called directly to supply options, one call being
necessary for each optional parameter. For example,

      CALL E04UEF ('Print level = 1')

E04UEF should be consulted for a full description of this method
of supplying optional parameters.

All optional parameters not specified by the user are set to
their default values. Optional parameters specified by the user
are unaltered by E04UCF (unless they define invalid values) and
so remain in effect for subsequent calls to E04UCF, unless
altered by the user.

5.1.2. Description of the optional parameters

The following list (in alphabetical order) gives the valid
options. For each option, we give the keyword, any essential
optional qualifiers, the default value, and the definition. The
minimum valid abbreviation of each keyword is underlined. If no
characters of an optional qualifier are underlined, the qualifier
may be omitted. The letter a denotes a phrase (character string)
that qualifies an option. The letters i and r denote INTEGER and
DOUBLE PRECISION values required with certain options. The number
(epsilon) is a generic notation for machine precision (see
X02AJF(*) ), and (epsilon)  denotes the relative precision of the
                          R
objective function (the optional parameter Function Precision see
below).
```

Central Difference Interval r Default values are computed

If the algorithm switches to central differences because the forward-difference approximation is not sufficiently accurate, the value of r is used as the difference interval for every component of x. The use of finite-differences is discussed further below under the optional parameter Difference Interval.

Cold Start Default = Cold Start

Warm Start

   (Axiom parameter STA, warm start when .TRUE.)

This option controls the specification of the initial working set in both the procedure for finding a feasible point for the linear constraints and bounds, and in the first QP subproblem thereafter. With a Cold Start, the first working set is chosen by E04UCF based on the values of the variables and constraints at the initial point. Broadly speaking, the initial working set will include equality constraints and bounds or inequality constraints that violate or 'nearly' satisfy their bounds (within Crash Tolerance; see below). With a Warm Start, the user must set the ISTATE array and define CLAMDA and R as discussed in Section 5. ISTATE values associated with bounds and linear constraints determine the initial working set of the procedure to find a feasible point with respect to the bounds and linear constraints. ISTATE values associated with nonlinear constraints determine the initial working set of the first QP subproblem after such a feasible point has been found. E04UCF will override the user's specification of ISTATE if necessary, so that a poor choice of the working set will not cause a fatal error. A warm start will be advantageous if a good estimate of the initial working set is available - for example, when E04UCF is called repeatedly to solve related problems.

Crash Tolerance r Default = 0.01

   (Axiom parameter CRA)

This value is used in conjunction with the optional parameter Cold Start (the default value). When making a cold start, the QP algorithm in E04UCF must select an initial working set. When r>=0 , the initial working set will include (if possible) bounds or general inequality constraints that lie within r of their bounds.

In particular, a constraint of the form $a_j^T x>=l$ will be included

T

in the initial working set if $|a_j x-l| \le r(1+|l|)$. If $r<0$ or $r>1$, the default value is used.

Defaults

This special keyword may be used to reset the default values following a call to E04UCF.

Derivative Level i Default = 3

(Axiom parameter DER)

This parameter indicates which derivatives are provided by the user in subroutines OBJFUN and CONFUN. The possible choices for i are the following.

i	Meaning
3	All objective and constraint gradients are provided by the user.
2	All of the Jacobian is provided, but some components of the objective gradient are not specified by the user.
1	All elements of the objective gradient are known, but some elements of the Jacobian matrix are not specified by the user.
0	Some elements of both the objective gradient and the Jacobian matrix are not specified by the user.

The value i=3 should be used whenever possible, since E04UCF is more reliable and will usually be more efficient when all derivatives are exact.

If i=0 or 2, E04UCF will estimate the unspecified components of the objective gradient, using finite differences. The computation of finite-difference approximations usually increases the total run-time, since a call to OBJFUN is required for each unspecified element. Furthermore, less accuracy can be attained in the solution (see Chapter 8 of Gill et al [10], for a discussion of limiting accuracy).

If i=0 or 1, E04UCF will approximate unspecified elements of the Jacobian. One call to CONFUN is needed for each variable for which partial derivatives are not available. For example, if the Jacobian has the form

(* * * *)

```
(* ? ? *)
(* * ? *)
(* * * *)
```

where '*' indicates an element provided by the user and '?' indicates an unspecified element, E04UCF will call CONFUN twice: once to estimate the missing element in column 2, and again to estimate the two missing elements in column 3. (Since columns 1 and 4 are known, they require no calls to CONFUN.)

At times, central differences are used rather than forward differences, in which case twice as many calls to OBJFUN and CONFUN are needed. (The switch to central differences is not under the user's control.)

Difference Interval r Default values are computed

(Axiom parameter DIF)

This option defines an interval used to estimate gradients by finite differences in the following circumstances:

(a) For verifying the objective and/or constraint gradients (see the description of Verify, below).

(b) For estimating unspecified elements of the objective gradient of the Jacobian matrix.

In general, a derivative with respect to the jth variable is approximated using the interval $(delta)_j$, where $(delta)_j = r(1+|\hat{x}_j|)$ with $\hat{x}$ the first point feasible with respect to the bounds and linear constraints. If the functions are well scaled, the resulting derivative approximation should be accurate to O(r).

See Gill et al [10] for a discussion of the accuracy in finite-difference approximations.

If a difference interval is not specified by the user, a finite-difference interval will be computed automatically for each variable by a procedure that requires up to six calls of CONFUN and OBJFUN for each component. This option is recommended if the function is badly scaled or the user wishes to have E04UCF determine constant elements in the objective and constraint gradients (see the descriptions of CONFUN and OBJFUN in Section 5).

Feasibility Tolerance r Default = \/(epsilon)

(Axiom parameter FEA)

The scalar r defines the maximum acceptable absolute violations in linear and nonlinear constraints at a 'feasible' point; i.e., a constraint is considered satisfied if its violation does not exceed r. If r<(epsilon) or r>=1, the default value is used. Using this keyword sets both optional parameters Linear Feasibility Tolerance and Nonlinear Feasibility Tolerance to r, if (epsilon)<=r<1. (Additional details are given below under the descriptions of these parameters.)

Function Precision r Default = (epsilon)$^{0.9}$

(Axiom parameter FUN)

This parameter defines (epsilon)$_R$, which is intended to be a measure of the accuracy with which the problem functions f and c can be computed. If r<(epsilon) or r>=1, the default value is used. The value of (epsilon)$_R$ should reflect the relative precision of 1+|F(x)|; i.e., (epsilon)$_R$ acts as a relative precision when |F| is large, and as an absolute precision when |F| is small. For example, if F(x) is typically of order 1000 and the first six significant digits are known to be correct, an appropriate value for (epsilon)$_R$ would be 1.0E-6. In contrast, if F(x) is typically of order 10^{-4} and the first six significant digits are known to be correct, an appropriate value for (epsilon)$_R$ would be 1.0E-10. The choice of (epsilon)$_R$ can be quite complicated for badly scaled problems; see Chapter 8 of Gill et al [10] for a discussion of scaling techniques. The default value is appropriate for most simple functions that are computed with full accuracy. However, when the accuracy of the computed function values is known to be significantly worse than full precision, the value of (epsilon)$_R$ should be large enough so that E04UCF will not attempt to distinguish between function values that differ by less than the error inherent in the calculation.

Hessian No Default = No

Hessian Yes

(No Axiom parameter - fixed as Yes)

This option controls the contents of the upper-triangular matrix R (see Section 5). E04UCF works exclusively with the transformed and re-ordered Hessian H_Q (6), and hence extra computation is required to form the Hessian itself. If Hessian = No, R contains the Cholesky factor of the transformed and re-ordered Hessian. If Hessian = Yes the Cholesky factor of the approximate Hessian itself is formed and stored in R. The user should select Hessian = Yes if a warm start will be used for the next call to E04UCF.

Infinite Bound Size r Default = 10^{10}

(Axiom parameter INFB)

If r>0, r defines the 'infinite' bound BIGBND in the definition of the problem constraints. Any upper bound greater than or equal to BIGBND will be regarded as plus infinity (and similarly for a lower bound less than or equal to -BIGBND). If r<=0, the default value is used.

Infinite Step Size r Default = max(BIGBND,10^{10})

(Axiom parameter INFS)

If r>0, r specifies the magnitude of the change in variables that is treated as a step to an unbounded solution. If the change in x during an iteration would exceed the value of Infinite Step Size, the objective function is considered to be unbounded below in the feasible region. If r<=0, the default value is used.

Iteration limit i Default = max(50,3($n+n_L$)+10n_N)

See Major Iteration Limit below.

Linear Feasibility Tolerance r_1 Default = $\sqrt{\epsilon}$

(Axiom parameter LINF)

Nonlinear Feasibility Tolerance r_2 Default = $\sqrt{\epsilon}$ if

(Axiom parameter NONF)

Derivative Level >= 2 and $(epsilon)^{0.33}$ otherwise

The scalars r_1 and r_2 define the maximum acceptable absolute violations in linear and nonlinear constraints at a 'feasible' point; i.e., a linear constraint is considered satisfied if its violation does not exceed r_1, and similarly for a nonlinear constraint and r_2. If r_i <(epsilon) or r_i >=1, the default value is used, for i=1,2.

On entry to E04UCF, an iterative procedure is executed in order to find a point that satisfies the linear constraint and bounds on the variables to within the tolerance r_1. All subsequent iterates will satisfy the linear constraints to within the same tolerance (unless r_1 is comparable to the finite-difference interval).

For nonlinear constraints, the feasibility tolerance r_2 defines the largest constraint violation that is acceptable at an optimal point. Since nonlinear constraints are generally not satisfied until the final iterate, the value of Nonlinear Feasibility Tolerance acts as a partial termination criterion for the iterative sequence generated by E04UCF (see the discussion of Optimality Tolerance).

These tolerances should reflect the precision of the corresponding constraints. For example, if the variables and the coefficients in the linear constraints are of order unity, and the latter are correct to about 6 decimal digits, it would be appropriate to specify r_1 as 10^{-6}.

Linesearch Tolerance r Default = 0.9

(Axiom parameter LINT)

The value r (0 <= r < 1) controls the accuracy with which the step (alpha) taken during each iteration approximates a minimum of the merit function along the search direction (the smaller the value of r, the more accurate the linesearch). The default value r=0.9 requests an inaccurate search, and is appropriate for most problems, particularly those with any nonlinear constraints.

If there are no nonlinear constraints, a more accurate search may be appropriate when it is desirable to reduce the number of major iterations - for example, if the objective function is cheap to evaluate, or if a substantial number of gradients are unspecified.

List Default = List

Nolist

(Axiom parameter LIST)

Normally each optional parameter specification is printed as it is supplied. Nolist may be used to suppress the printing and List may be used to restore printing.

Major Iteration Limit i Default = $max(50,3(n+n_L)+10n_N)$

Iteration Limit

Iters

Itns

(Axiom parameter MAJI)

The value of i specifies the maximum number of major iterations allowed before termination. Setting i=0 and Major Print Level> 0 means that the workspace needed will be computed and printed, but no iterations will be performed.

Major Print level i Default = 10

Print Level

(Axiom parameter MAJP)

The value of i controls the amount of printout produced by the major iterations of E04UCF. (See also Minor Print level below.) The levels of printing are indicated below.

i	Output
0	No output.
1	The final solution only.

```
5        One line for each major iteration (no printout of the
         final solution).

>=10     The final solution and one line of output for each
         iteration.

>=20     At each major iteration, the objective function, the
         Euclidean norm of the nonlinear constraint violations,
         the values of the nonlinear constraints (the vector c),
         the values of the linear constraints (the vector A x),
                                                           L
         and the current values of the variables (the vector x).

>=30     At each major iteration, the diagonal elements of the
         matrix T associated with the TQ factorization (5) of the
         QP working set, and the diagonal elements of R, the
         triangular factor of the transformed and re-ordered
         Hessian (6).

Minor Iteration Limit i Default = max(50,3(n+n +n ))
                                              L  N

   (Axiom parameter MINI)

The value of i specifies the maximum number of iterations for the
optimality phase of each QP subproblem.

Minor Print Level i Default = 0

   (Axiom parameter MINP)

The value of i controls the amount of printout produced by the
minor iterations of E04UCF, i.e., the iterations of the quadratic
programming algorithm. (See also Major Print Level, above.) The
following levels of printing are available.

i        Output

0        No output.

1        The final QP solution.

5        One line of output for each minor iteration (no printout
         of the final QP solution).

>=10     The final QP solution and one brief line of output for
         each minor iteration.

>=20     At each minor iteration, the current estimates of the QP
         multipliers, the current estimate of the QP search
```

direction, the QP constraint values, and the status of each QP constraint.

>=30 At each minor iteration, the diagonal elements of the matrix T associated with the TQ factorization (5) of the QP working set, and the diagonal elements of the Cholesky factor R of the transformed Hessian (6).

Nonlinear Feasibility Tolerance r Default = $\sqrt{(\epsilon)}$

See Linear Feasibility Tolerance, above.

Optimality Tolerance r Default = $(\epsilon)^{0.8}$

(Axiom parameter OPT)

The parameter r ($(\epsilon)_R$ <=r<1) specifies the accuracy to which the user wishes the final iterate to approximate a solution of the problem. Broadly speaking, r indicates the number of correct figures desired in the objective function at the solution. For example, if r is 10^{-6} and E04UCF terminates successfully, the final value of F should have approximately six correct figures. If r<$(\epsilon)_R$ or r>=1 the default value is used.

E04UCF will terminate successfully if the iterative sequence of x -values is judged to have converged and the final point satisfies the first-order Kuhn-Tucker conditions (see Section 3). The sequence of iterates is considered to have converged at x if

$$(\alpha)\, \|p\| \le \sqrt{r}\,(1+\|x\|), \qquad (8a)$$

where p is the search direction and (alpha) the step length from (3). An iterate is considered to satisfy the first-order conditions for a minimum if

$$\|Z^T g_{FR}\| \le \sqrt{r}\,(1+\max(1+|F(x)|,\|g_{FR}\|)) \qquad (8b)$$

and

$$|res_j| \le ftol \text{ for all } j, \qquad (8c)$$

where $Z^T g_{FR}$ is the projected gradient (see Section 3), g_{FR} is the gradient of F(x) with respect to the free variables, res_j is the violation of the jth active nonlinear constraint, and ftol is the Nonlinear Feasibility Tolerance.

Step Limit r Default = 2.0

(Axiom parameter STE)

If r>0, r specifies the maximum change in variables at the first step of the linesearch. In some cases, such as $F(x)=ae^{bx}$ or $F(x)=ax^b$, even a moderate change in the components of x can lead to floating-point overflow. The parameter r is therefore used to encourage evaluation of the problem functions at meaningful points. Given any major iterate x, the first point $\tilde{x}$ at which F and c are evaluated during the linesearch is restricted so that

$$||\tilde{x}-x||_2 <= r(1+||x||_2).$$

The linesearch may go on and evaluate F and c at points further from x if this will result in a lower value of the merit function. In this case, the character L is printed at the end of the optional line of printed output, (see Section 5.2). If L is printed for most of the iterations, r should be set to a larger value.

Wherever possible, upper and lower bounds on x should be used to prevent evaluation of nonlinear functions at wild values. The default value Step Limit = 2.0 should not affect progress on well-behaved functions, but values 0.1 or 0.01 may be helpful when rapidly varying functions are present. If a small value of Step Limit is selected, a good starting point may be required. An important application is to the class of nonlinear least-squares problems. If r<=0, the default value is used.

Start Objective Check At Variable k Default = 1

(Axiom parameter STAO)

Start Constraint Check At Variable k Default = 1

(Axiom parameter STAC)

Stop Objective Check At Variable l Default = n

(Axiom parameter STOO)

Stop Constraint Check At Variable l Default = n

(Axiom parameter STOC)

These keywords take effect only if Verify Level > 0 (see below). They may be used to control the verification of gradient elements computed by subroutines OBJFUN and CONFUN. For example, if the first 30 components of the objective gradient appeared to be correct in an earlier run, so that only component 31 remains questionable, it is reasonable to specify Start Objective Check At Variable 31. If the first 30 variables appear linearly in the objective, so that the corresponding gradient elements are constant, the above choice would also be appropriate.

Verify Level i Default = 0

Verify No

Verify Level - 1

Verify Level 0

Verify Objective Gradients

Verify Level 1

Verify Constraint Gradients

Verify Level 2

Verify

Verify Yes

Verify Gradients

Verify Level 3

(Axiom parameter VE)

These keywords refer to finite-difference checks on the gradient elements computed by the user-provided subroutines OBJFUN and CONFUN. (Unspecified gradient components are not checked.) It is possible to specify Verify Levels 0-3 in several ways, as indicated above. For example, the nonlinear objective gradient

```
(if any) will be verified if either Verify Objective Gradients or
Verify Level 1 is specified. Similarly, the objective and the
constraint gradients will be verified if Verify Yes or Verify
Level 3 or Verify is specified.

If 0<=i<=3, gradients will be verified at the first point that
satisfies the linear constraints and bounds. If i=0, only a '
cheap' test will be performed, requiring one call to OBJFUN and
one call to CONFUN. If 1<=i<=3, a more reliable (but more
expensive) check will be made on individual gradient components,
within the ranges specified by the Start and Stop keywords
described above. A result of the form OK or BAD? is printed by
E04UCF to indicate whether or not each component appears to be
correct.

If 10<=i<=13, the action is the same as for i - 10, except that
it will take place at the user-specified initial value of x.

We suggest that Verify Level 3 be specified whenever a new
function routine is being developed.

5.1.3. Optional parameter checklist and default values

For easy reference, the following list shows all the valid
keywords and their default values. The symbol (epsilon)
represents the machine precision (see X02AJF(*) ).

Optional Parameters      Default Values



Central difference       Computed automatically
interval

Cold/Warm start          Cold start

Crash tolerance          0.01

Defaults

Derivative level         3

Difference interval      Computed automatically

                           ---------
Feasibility tolerance    \/(epsilon)

                                  0.9
Function precision       (epsilon)
```

Hessian	No
Infinite bound size	10^{10}
Infinite step size	10^{10}
Linear feasibility tolerance	$\sqrt{(\epsilon)}$
Linesearch tolerance	0.9
List/Nolist	List
Major iteration limit	$\max(50,3(n+n_L)+10n_N)$
Major print level	10
Minor iteration limit	$\max(50,3(n+n_L+n_N))$
Minor print level	0
Nonlinear feasibility tolerance	$\sqrt{(\epsilon)}$ if Derivative Level >= 2 otherwise $(\epsilon)^{0.33}$
Optimality tolerance	$(\epsilon)_R^{0.8}$
Step limit	2.0
Start objective check	1
Start constraint check	1
Stop objective check	n
Stop constraint check	n
Verify level	0

5.2. Description of Printed Output

The level of printed output from E04UCF is controlled by the user (see the description of Major Print Level and Minor Print Level in Section 5.1). If Minor Print Level > 0, output is obtained from the subroutines that solve the QP subproblem. For a detailed description of this information the reader should refer to E04NCF(*).

When Major Print Level >= 5, the following line of output is produced at every major iteration of E04UCF. In all cases, the values of the quantities printed are those in effect on completion of the given iteration.

Itn   is the iteration count.

ItQP   is the sum of the iterations required by the feasibility and optimality phases of the QP subproblem. Generally, ItQP will be 1 in the later iterations, since theoretical analysis predicts that the correct active set will be identified near the solution (see Section 3).

               Note that ItQP may be greater than the Minor Iteration Limit if some iterations are required for the feasibility phase.

Step   is the step (alpha) taken along the computed search direction. On reasonably well-behaved problems, the unit step will be taken as the solution is approached.

Nfun   is the cumulative number of evaluations of the objective function needed for the linesearch. Evaluations needed for the estimation of the gradients by finite differences are not included. Nfun is printed as a guide to the amount of work required for the linesearch.

Merit   is the value of the augmented Lagrangian merit function (12) at the current iterate. This function will decrease at each iteration unless it was necessary to increase the penalty parameters (see Section 8.2). As the solution is approached, Merit will converge to the value of the objective function at the solution.

               If the QP subproblem does not have a feasible point (signified by I at the end of the current output line), the merit function is a large multiple of the constraint violations, weighted by the penalty parameters. During a sequence of major iterations with infeasible subproblems, the sequence of Merit values will decrease monotonically until either a feasible subproblem is obtained or E04UCF terminates with IFAIL = 3 (no feasible point could be found for the nonlinear constraints).

If no nonlinear constraints are present (i.e., NCNLN = 0), this entry contains Objective, the value of the objective function F(x). The objective function will decrease monotonically to its optimal value when there are no nonlinear constraints.

Bnd — is the number of simple bound constraints in the predicted active set.

Lin — is the number of general linear constraints in the predicted active set.

Nln — is the number of nonlinear constraints in the predicted active set (not printed if NCNLN is zero).

Nz — is the number of columns of Z (see Section 8.1). The value of Nz is the number of variables minus the number of constraints in the predicted active set; i.e., Nz = n-(Bnd + Lin + Nln).

Norm Gf — is the Euclidean norm of g_{FR}, the gradient of the objective function with respect to the free variables, i.e.,variables not currently held at a bound.

Norm Gz — is $||Z^T g_{FR}||$, the Euclidean norm of the projected gradient (see Section 8.1). Norm Gz will be approximately zero in the neighbourhood of a solution.

Cond H — is a lower bound on the condition number of the Hessian approximation H.

Cond Hz — is a lower bound on the condition number of the projected Hessian approximation H_z ($(H_z=Z^T H_{FR} Z=R_z^T R_z$; see (6) and (12) in Sections 3 and 8.1). The larger this number, the more difficult the problem.

Cond T — is a lower bound on the condition number of the matrix of predicted active constraints.

Norm C — is the Euclidean norm of the residuals of constraints that are violated or in the predicted active set (not printed if NCNLN is zero). Norm C will be approximately zero in the neighbourhood of a solution.

Penalty — is the Euclidean norm of the vector of penalty parameters used in the augumented Lagrangian merit function (not printed if NCNLN is zero).

Conv — is a three-letter indication of the status of the three convergence tests (8a)-(8c) defined in the description of the optional parameter Optimality Tolerance in Section 5.1 Each letter is T if the test is satisfied, and F otherwise. The three tests indicate whether:

(a) the sequence of iterates has converged;

(b) the projected gradient (Norm Gz) is sufficiently small; and

(c) the norm of the residuals of constraints in the predicted active set (Norm C) is small enough.

If any of these indicators is F when E04UCF terminates with IFAIL = 0, the user should check the solution carefully.

M — is printed if the Quasi-Newton update was modified to ensure that the Hessian approximation is positive-definite (see Section 8.3).

I — is printed if the QP subproblem has no feasible point.

C — is printed if central differences were used to compute the unspecified objective and constraint gradients. If the value of Step is zero, the switch to central differences was made because no lower point could be found in the linesearch. (In this case, the QP subproblem is resolved with the central-difference gradient and Jacobian.) If the value of Step is non-zero, central differences were computed because Norm Gz and Norm C imply

```
                that x is close to a Kuhn-Tucker point.

L               is printed if the linesearch has produced a
                relative change in x greater than the value
                defined by the optional parameter Step Limit. If
                this output occurs frequently during later
                iterations of the run, Step Limit should be set to
                a larger value.

R               is printed if the approximate Hessian has been
                refactorized. If the diagonal condition estimator
                of R indicates that the approximate Hessian is
                badly conditioned, the approximate Hessian is
                refactorized using column interchanges. If
                necessary, R is modified so that its diagonal
                condition estimator is bounded.

When Major Print Level = 1 or Major Print Level >= 10, the
summary printout at the end of execution of E04UCF includes a
listing of the status of every variable and constraint. Note that
default names are assigned to all variables and constraints.

The following describes the printout for each variable.

Varbl           gives the name (V) and index j=1,2,...,n of the
                variable.

State           gives the state of the variable in the predicted
                active set (FR if neither bound is in the active
                set, EQ if a fixed variable, LL if on its lower
                bound, UL if on its upper bound). If the variable
                is predicted to lie outside its upper or lower
                bound by more than the feasibility tolerance,
                State will be ++ or -- respectively. (The latter
                situation can occur only when there is no feasible
                point for the bounds and linear constraints.)

Value           is the value of the variable at the final
                iteration.

Lower bound     is the lower bound specified for the variable.
                (None indicates that BL(j)<=- BIGBND.)

Upper bound     is the upper bound specified for the variable.
                (None indicates that BL(j)>=BIGBND.)

Lagr Mult       is the value of the Lagrange-multiplier for the
                associated bound constraint. This will be zero if
                State is FR. If x is optimal, the multiplier
                should be non-negative if State is LL, and non-
```

```
              positive if State is UL.

Residual      is the difference between the variable Value and
              the nearer of its bounds BL(j) and BU(j).

The printout for general constraints is the same as for
variables, except for the following:

L Con      is the name (L) and index i, for i = 1,2,...,NCLIN of
           a linear constraint.

N Con      is the name (N) and index i, for i = 1,2,...,NCNLN of
           a nonlinear constraint.

6. Error Indicators and Warnings

Errors or warnings specified by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

The input data for E04UCF should always be checked (even if
E04UCF terminates with IFAIL=0).

Note that when Print Level>0, a short description of IFAIL is
printed.

Errors and diagnostics indicated by IFAIL, together with some
recommendations for recovery are indicated below.

IFAIL= 1
     The final iterate x satisfies the first-order Kuhn-Tucker
     conditions to the accuracy requested, but the sequence of
     iterates has not yet converged. E04UCF was terminated
     because no further improvement could be made in the merit
     function.

     This value of IFAIL may occur in several circumstances. The
     most common situation is that the user asks for a solution
     with accuracy that is not attainable with the given
     precision of the problem (as specified by Function Precision
     see Section 5). This condition will also occur if, by
     chance, an iterate is an 'exact' Kuhn-Tucker point, but the
     change in the variables was significant at the previous
     iteration. (This situation often happens when minimizing
     very simple functions, such as quadratics.)

     If the four conditions listed in Section 5 for IFAIL = 0 are
     satisfied, x is likely to be a solution of (1) even if IFAIL
     = 1.
```

IFAIL= 2

E04UCF has terminated without finding a feasible point for the linear constraints and bounds, which means that no feasible point exists for the given value of Linear Feasibility Tolerance (see Section 5.1). The user should check that there are no constraint redundancies. If the data for the constraints are accurate only to an absolute precision (sigma), the user should ensure that the value of the optional parameter Linear Feasibility Tolerance is greater than (sigma). For example, if all elements of A are of order unity and are accurate to only three decimal places, Linear Feasibility Tolerance should be at least 10^{-3}.

IFAIL= 3

No feasible point could be found for the nonlinear constraints. The problem may have no feasible solution. This means that there has been a sequence of QP subproblems for which no feasible point could be found (indicated by I at the end of each terse line of output). This behaviour will occur if there is no feasible point for the nonlinear constraints. (However, there is no general test that can determine whether a feasible point exists for a set of nonlinear constraints.) If the infeasible subproblems occur from the very first major iteration, it is highly likely that no feasible point exists. If infeasibilities occur when earlier subproblems have been feasible, small constraint inconsistencies may be present. The user should check the validity of constraints with negative values of ISTATE. If the user is convinced that a feasible point does exist, E04UCF should be restarted at a different starting point.

IFAIL= 4

The limiting number of iterations (determined by the optional parameter Major Iteration Limit see Section 5.1) has been reached.

If the algorithm appears to be making progress, Major Iteration Limit may be too small. If so, increase its value and rerun E04UCF (possibly using the Warm Start option). If the algorithm seems to be 'bogged down', the user should check for incorrect gradients or ill-conditioning as described below under IFAIL = 6.

Note that ill-conditioning in the working set is sometimes resolved automatically by the algorithm, in which case performing additional iterations may be helpful. However, ill-conditioning in the Hessian approximation tends to persist once it has begun, so that allowing additional iterations without altering R is usually inadvisable. If the quasi-Newton update of the Hessian approximation was modified during the latter iterations (i.e., an M occurs at the end of each terse line), it may be worthwhile to try a warm start at the final point as suggested above.

IFAIL= 6

x does not satisfy the first-order Kuhn-Tucker conditions, and no improved point for the merit function could be found during the final line search.

A sufficient decrease in the merit function could not be attained during the final line search. This sometimes occurs because an overly stringent accuracy has been requested, i.e., Optimality Tolerance is too small. In this case the user should apply the four tests described under IFAIL = 0 above to determine whether or not the final solution is acceptable (see Gill et al [10], for a discussion of the attainable accuracy).

If many iterations have occurred in which essentially no progress has been made and E04UCF has failed completely to move from the initial point then subroutines OBJFUN or CONFUN may be incorrect. The user should refer to comments below under IFAIL = 7 and check the gradients using the Verify parameter. Unfortunately, there may be small errors in the objective and constraint gradients that cannot be detected by the verification process. Finite-difference approximations to first derivatives are catastrophically affected by even small inaccuracies. An indication of this situation is a dramatic alteration in the iterates if the finite-difference interval is altered. One might also suspect this type of error if a switch is made to central differences even when Norm Gz and Norm C are large.

Another possibility is that the search direction has become inaccurate because of ill-conditioning in the Hessian approximation or the matrix of constraints in the working set; either form of ill-conditioning tends to be reflected in large values of ItQP (the number of iterations required to solve each QP subproblem).

If the condition estimate of the projected Hessian (Cond Hz) is extremely large, it may be worthwhile to rerun E04UCF from the final point with the Warm Start option. In this situation, ISTATE should be left unaltered and R should be reset to the identity matrix.

If the matrix of constraints in the working set is ill-conditioned (i.e., Cond T is extremely large), it may be

```
     helpful to run E04UCF with a relaxed value of the
     Feasibility Tolerance (Constraint dependencies are often
     indicated by wide variations in size in the diagonal
     elements of the matrix T, whose diagonals will be printed
     for Major Print Level >= 30).

IFAIL= 7
     The user-provided derivatives of the objective function
     and/or nonlinear constraints appear to be incorrect.

     Large errors were found in the derivatives of the objective
     function and/or nonlinear constraints. This value of IFAIL
     will occur if the verification process indicated that at
     least one gradient or Jacobian component had no correct
     figures. The user should refer to the printed output to
     determine which elements are suspected to be in error.

     As a first-step, the user should check that the code for the
     objective and constraint values is correct - for example, by
     computing the function at a point where the correct value is
     known. However, care should be taken that the chosen point
     fully tests the evaluation of the function. It is remarkable
     how often the values x=0 or x=1 are used to test function
     evaluation procedures, and how often the special properties
     of these numbers make the test meaningless.

     Special care should be used in this test if computation of
     the objective function involves subsidiary data communicated
     in COMMON storage. Although the first evaluation of the
     function may be correct, subsequent calculations may be in
     error because some of the subsidiary data has accidently
     been overwritten.

     Errors in programming the function may be quite subtle in
     that the function value is 'almost' correct. For example,
     the function may not be accurate to full precision because
     of the inaccurate calculation of a subsidiary quantity, or
     the limited accuracy of data upon which the function
     depends. A common error on machines where numerical
     calculations are usually performed in double precision is to
     include even one single-precision constant in the
     calculation of the function; since some compilers do not
     convert such constants to double precision, half the correct
     figures may be lost by such a seemingly trivial error.

IFAIL= 9
     An input parameter is invalid. The user should refer to the
     printed output to determine which parameter must be
     redefined.
```

IFAILOverflow

If the printed output before the overflow error contains a warning about serious ill-conditioning in the working set when adding the jth constraint, it may be possible to avoid the difficulty by increasing the magnitude of the optional parameter Linear Feasiblity Tolerance or Nonlinear Feasiblity Tolerance, and rerunning the program. If the message recurs even after this change, the offending linearly dependent constraint (with index 'j') must be removed from the problem. If overflow occurs in one of the user-supplied routines (e.g. if the nonlinear functions involve exponentials or singularities), it may help to specify tighter bounds for some of the variables (i.e., reduce the gap between appropriate l_j and u_j).

7. Accuracy

If IFAIL = 0 on exit then the vector returned in the array X is an estimate of the solution to an accuracy of approximately Feasiblity Tolerance (see Section 5.1), whose default value is $(epsilon)^{0.8}$, where (epsilon) is the machine precision (see X02AJF(*)).

8. Further Comments

In this section we give some further details of the method used by E04UCF.

8.1. Solution of the Quadratic Programming Subproblem

The search direction p is obtained by solving (4) using the method of E04NCF(*) (Gill et al [8]), which was specifically designed to be used within an SQP algorithm for nonlinear programming.

The method of E04UCF is a two-phase (primal) quadratic programming method. The two phases of the method are: finding an initial feasible point by minimizing the sum of infeasibilities (the feasibility phase), and minimizing the quadratic objective function within the feasible region (the optimality phase). The computations in both phases are perfomed by the same subroutines. The two-phase nature of the algorithm is reflected by changing the function being minimized from the sum of infeasibilities to the quadratic objective function.

In general, a quadratic program must be solved by iteration. Let $\bar{p}$ denote the current estimate of the solution of (4); the new iterate p is defined by

$$\bar{p}=p+\sigma d, \qquad (9)$$

where, as in (3), σ is a non-negative step length and d is a search direction.

At the beginning of each iteration of E04UCF, a working set is defined of constraints (general and bound) that are satisfied exactly. The vector d is then constructed so that the values of constraints in the working set remain unaltered for any move along d. For a bound constraint in the working set, this property is achieved by setting the corresponding component of d to zero, i.e., by fixing the variable at its bound. As before, the subscripts 'FX' and 'FR' denote selection of the components associated with the fixed and free variables.

Let C denote the sub-matrix of rows of

$$\begin{pmatrix} A_L \\ A_N \end{pmatrix}$$

corresponding to general constraints in the working set. The general constraints in the working set will remain unaltered if

$$C_{FR} d_{FR} = 0, \qquad (10)$$

which is equivalent to defining d_{FR} as

$$d_{FR} = Z d_z \qquad (11)$$

for some vector d_z, where Z is the matrix associated with the TQ factorization (5) of C_{FR}.

The definition of d_z in (11) depends on whether the current p is feasible. If not, d_z is zero except for a component γ in the jth position, where j and γ are chosen so that the sum of infeasibilities is decreasing along d. (For further details, see Gill et al [8].) In the feasible case, d_z satisfies the equations

$$R_z^T R_z d = -Z^T q_{FR}, \qquad (12)$$

where R_z is the Cholesky factor of $Z^T H_{FR} Z$ and q is the gradient of the quadratic objective function (q=g+Hp). (The vector $Z^T q_{FR}$ is the projected gradient of the QP.) With (12), P+d is the minimizer of the quadratic objective function subject to treating the constraints in the working set as equalities.

If the QP projected gradient is zero, the current point is a constrained stationary point in the subspace defined by the working set. During the feasiblity phase, the projected gradient will usually be zero only at a vertex (although it may vanish at non-vertices in the presence of constraint dependencies). During the optimality phase, a zero projected gradient implies that p minimizes the quadratic objective function when the constraints in the working set are treated as equalities. In either case, Lagrange multipliers are computed. Given a positive constant (delta) of the order of the machine precision, the Lagrange multiplier $(mu)_j$ corresponding to an inequality constraint in the working set at its upper bound is said to be optimal if $(mu)_j$ <=(delta) when the jth constraint is at its upper bound, or if $(mu)_j$ >=-(delta) when the associated constraint is at its lower bound. If any multiplier is non-optimal, the current objective function (either the true objective or the sum of infeasibilities) can be reduced by deleting the corresponding constraint from the working set.

If optimal multipliers occur during the feasibility phase and the sum of infeasibilities is non-zero, no feasible point exists. The QP algorithm will then continue iterating to determine the minimum sum of infeasibilities. At this point, the Lagrange multiplier $(mu)_j$ will satisfy -(1+(delta))<=$(mu)_j$ <=(delta) for an inequality constraint at its upper bound, and -(delta)<=$(mu)_j$ <=1+(delta) for an inequality at its lower bound. The Lagrange multiplier for an equality constraint will satisfy |$(mu)_j$ |<=1+(delta).

The choice of step length (σ) in the QP iteration (9) is based on remaining feasible with respect to the satisfied constraints. During the optimality phase, if p+d is feasible, (σ) will be taken as unity. (In this case, the projected gradient at $\bar{p}$ will be zero.) Otherwise, (σ) is set to $(\sigma)_M$, the step to the 'nearest'constraint, which is added to the working set at the next iteration.

Each change in the working set leads to a simple change to C_{FR}: if the status of a general constraint changes, a row of C_{FR} is altered; if a bound constraint enters or leaves the working set, a column of C_{FR} changes. Explicit representations are recurred of the matrices T, Q_{FR} and R, and of the vectors $Q^T q$ and $Q^T g$.

8.2. The Merit Function

After computing the search direction as described in Section 3, each major iteration proceeds by determining a step length (α) in (3) that produces a 'sufficient decrease' in the augmented Lagrangian merit function

$$L(x,(\lambda),s)=F(x)-\sum_i (\lambda)_i (c_i(x)-s_i) + \frac{1}{2}\sum_i (\rho)_i (c_i(x)-s_i)^2 , \qquad (13)$$

where x, (λ) and s vary during the linsearch. The summation terms in (13) involve only the nonlinear constraints. The vector (λ) is an estimate of the Lagrange multipliers for the nonlinear constraints of (1). The non-negative slack variables $\{s_i\}$ allow nonlinear inequality constraints to be treated without introducing discontinuities. The solution of the QP subproblem (4) provides a vector triple that serves as a direction of search for the three sets of variables. The non-negative vector (ρ) of penalty parameters is initialised to zero at the beginning of the first major iteration. Thereafter, selected components are increased whenever necessary to ensure descent for the merit function. Thus, the sequence of norms of (rho) (the printed quantity Penalty, see Section 5.2) is generally non-decreasing, although each $(rho)_i$ may be reduced a limited number of times.

The merit function (13) and its global convergence properties are described in Gill et al [9].

8.3. The Quasi-Newton Update

The matrix H in (4) is a positive-definite quasi-Newton approximation to the Hessian of the Lagrangian function. (For a review of quasi-Newton methods, see Dennis and Schnabel [3].) At the end of each major iteration, a new Hessian approximation $\bar{H}$ is defined as a rank-two modification of H. In E04UCF, the BFGS quasi-Newton update is used:

$$\bar{H}=H-\frac{1}{s^T Hs}Hss^T H+\frac{1}{y^T s}yy^T, \qquad (14)$$

where $s=\bar{x}-x$ (the change in x).

In E04UCF, H is required to be positive-definite. If H is positive-definite, $\bar{H}$ defined by (14) will be positive-definite if and only if $y^T s$ is positive (see, e.g. Dennis and More [1]). Ideally, y in (14) would be taken as y_L, the change in gradient of the Lagrangian function

$$y_L=\bar{g}-\bar{A}_N^T(mu)_N-g+A_N^T(mu)_N, \qquad (15)$$

where $(mu)_N$ denotes the QP multipliers associated with the nonlinear constraints of the original problem. If $y_L^T s$ is not sufficiently positive, an attempt is made to perform the update with a vector y of the form

m

$$y = y_L + \sum_{i=1}^{N} \omega_i \left(a_i(\bar{x}) c_i(\bar{x}) - a_i(x) c_i(x)\right),$$

where $\omega_i \geq 0$. If no such vector can be found, the update is perfomed with a scaled y_L; in this case, M is printed to indicate that the update is modified.

Rather than modifying H itself, the Cholesky factor of the transformed Hessian H_Q (6) is updated, where Q is the matrix from (5) associated with the active set of the QP subproblem. The update (13) is equivalent to the following update to H_Q:

$$\bar{H}_Q = H_Q - \frac{1}{s_Q^T H_Q s_Q} H_Q s_Q s_Q^T H_Q + \frac{1}{y_Q^T s_Q} y_Q y_Q^T, \qquad (16)$$

where $y_Q = Q^T y$, and $s_Q = Q^T s$. This update may be expressed as a rank-one update to R (see Dennis and Schnabel [2]).

9. Example

This section describes one version of the so-called 'hexagon' problem (a different formulation is given as Problem 108 in Hock and Schittkowski [11]). The problem is to determine the hexagon of maximum area such that no two of its vertices are more than one unit apart (the solution is not a regular hexagon).

All constraint types are included (bounds, linear, nonlinear), and the Hessian of the Lagrangian function is not positive-definite at the solution. The problem has nine variables, non-infinite bounds on seven of the variables, four general linear constraints, and fourteen nonlinear constraints.

The objective function is

$$F(x) = -x_2 x_6 + x_1 x_7 - x_3 x_7 - x_5 x_8 + x_4 x_9 + x_3 x_8.$$

The bounds on the variables are

$x_1 >= 0$, $-1 <= x_3 <= 1$, $x_5 >= 0$, $x_6 >= 0$, $x_7 >= 0$, $x_8 <= 0$, and $x_9 <= 0$.

Thus,

$$l_B = (0,-\infty,-1,-\infty,0,0,0,-\infty,-\infty)^T$$

$$u_B = (\infty,\infty,1,\infty,\infty,\infty,\infty,0,0)^T$$

The general linear constraints are

$x_2 - x_1 >= 0$, $x_3 - x_2 >= 0$, $x_3 - x_4 >= 0$, and $x_4 - x_5 >= 0$.

Hence,

$$l_L = \begin{pmatrix}0\\0\\0\\0\end{pmatrix},\quad A_L = \begin{pmatrix}-1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0\end{pmatrix} \text{ and } u_L = \begin{pmatrix}\infty\\ \infty\\ \infty\\ \infty\end{pmatrix}.$$

The nonlinear constraints are all of the form $c_i(x) <= 1$, for $i=1,2,...,14$; hence, all components of l_N are $-\infty$, and all components of u_N are 1. The fourteen functions $\{c_i(x)\}$ are

$$c_1(x) = x_1^2 + x_6^2,$$

$$c_2(x) = (x_2 - x_1)^2 + (x_7 - x_6)^2,$$

$$c_3(x) = (x_3 - x_1)^2 + x_6^2,$$

$$c_4(x) = (x_1 - x_4)^2 + (x_6 - x_8)^2,$$

2 2

$$c_5(x)=(x_1-x_5)+(x_6-x_9),$$

$$c_6(x)=x_2^2+x_7^2,$$

$$c_7(x)=(x_3-x_2)^2+x_7^2,$$

$$c_8(x)=(x_4-x_2)^2+(x_8-x_7)^2,$$

$$c_9(x)=(x_2-x_5)^2+(x_7-x_9)^2,$$

$$c_{10}(x)=(x_4-x_3)^2+x_8^2,$$

$$c_{11}(x)=(x_5-x_3)^2+x_9^2,$$

$$c_{12}(x)=x_4^2+x_8^2,$$

$$c_{13}(x)=(x_4-x_5)^2+(x_9-x_8)^2,$$

$$c_{14}(x)=x_5^2+x_9^2.$$

An optimal solution (to five figures) is

$$x^*=(0.060947,0.59765,1.0,0.59765,0.060947,0.34377,0.5,-0.5,0.34377)^T,$$

and $F(x^*)=-1.34996$. (The optimal objective function is unique, but is achieved for other values of x.) Five nonlinear constraints and one simple bound are active at x^*. The sample solution output is given later in this section, following the sample main program and problem definition.

Two calls are made to E04UCF in order to demonstrate some of its features. For the first call, the starting point is:

$$x_0 =(0.1,0.125,0.666666,0.142857,0.111111,0.2,0.25,-0.2,-0.25)^T .$$

All objective and constraint derivatives are specified in the user-provided subroutines OBJFN1 and CONFN1, i.e., the default option Derivative Level =3 is used.

On completion of the first call to E04UCF, the optimal variables are perturbed to produce the initial point for a second run in which the problem functions are defined by the subroutines OBJFN2 and CONFN2. To illustrate one of the finite-difference options in E04UCF, these routines are programmed so that the first six components of the objective gradient and the constant elements of the Jacobian matrix are not specified; hence, the option Derivative Level =0 is chosen. During computation of the finite-difference intervals, the constant Jacobian elements are identified and set, and E04UCF automatically increases the derivative level to 2.

The second call to E04UCF illustrates the use of the Warm Start Level option to utilize the final active set, nonlinear multipliers and approximate Hessian from the first run. Note that Hessian = Yes was specified for the first run so that the array R would contain the Cholesky factor of the approximate Hessian of the Lagrangian.

The two calls to E04UCF illustrate the alternative methods of assigning default parameters. (There is no special significance in the order of these assignments; an options file may just as easily be used to modify parameters set by E04UEF.)

The results are typical of those obtained from E04UCF when solving well behaved (non-trivial) nonlinear problems. The approximate Hessian and working set remain relatively well-conditioned. Similarly the penalty parameters remain small and approximately constant. The numerical results illustrate much of the theoretically predicted behaviour of a quasi-Newton SQP method. As x approaches the solution, only one minor iteration is perfomed per major iteration, and the Norm Gz and Norm C columns exhibit the fast linear convergence rate mentioned in Sections 5 and 6. Note that the constraint violations converge earlier than

```
        the projected gradient. The final values of the project gradient
        norm and constraint norm reflect the limiting accuracy of the two
        quantities. It is possible to achieve almost full precision in
        the constraint norm but only half precision in the projected
        gradient norm. Note that the final accuracy in the nonlinear
        constraints is considerably better than the feasibility
        tolerance, because the constraint violations are being refined
        during the last few iterations while the algorithm is working to
        reduce the projected gradient norm. In this problem, the
        constraint values and Lagrange multipliers at the solution are '
        well balanced', i.e., all the multipliers are approximately the
        same order of magnitude. The behaviour is typical of a well-
        scaled problem.

        The example program is not reproduced here. The source code for
        all example programs is distributed with the NAG Foundation
        Library software and should be available on-line.
\end{verbatim}
\endscroll
\end{page}
```

Supply optional parameters to E04UCF from file

— nage.ht —

```
\begin{page}{manpageXXe04udf}{NAG Documentation: e04udf}
\beginscroll
\begin{verbatim}



     E04UDF(3NAG)      Foundation Library (12/10/92)      E04UDF(3NAG)



          E04 -- Minimizing or Maximizing a Function                 E04UDF
                  E04UDF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose
```

To supply optional parameters to E04UCF from an external file.

2. Specification

```
       SUBROUTINE E04UDF (IOPTNS, INFORM)
       INTEGER          IOPTNS, INFORM
```

3. Description

E04UDF may be used to supply values for optional parameters to E04UCF. E04UDF reads an external file and each line of the file defines a single optional parameter. It is only necessary to supply values for those parameters whose values are to be different from their default values.

Each optional parameter is defined by a single character string of up to 72 characters, consisting of one or more items. The items associated with a given option must be separated by spaces, or equal signs (=). Alphabetic characters may be upper or lower case. The string

```
       Print level = 1
```

is an example of a string used to set an optional parameter. For each option the string contains one or more of the following items:

(a) A mandatory keyword.

(b) A phrase that qualifies the keyword.

(c) A number that specifies an INTEGER or real value. Such numbers may be up to 16 contiguous characters in Fortran 77's I, F, E or D formats, terminated by a space if this is not the last item on the line.

Blank strings and comments are ignored. A comment begins with an asterisk (*) and all subsequent characters in the string are regarded as part of the comment.

The file containing the options must start with begin and must finish with end An example of a valid options file is:

```
      Begin * Example options file
       Print level =10
      End
```

Normally each line of the file is printed as it is read, on the current advisory message unit (see X04ABF), but printing may be suppressed using the keyword nolist To suppress printing of begin, nolist must be the first option supplied as in the file:

```
Begin
  Nolist
  Print level = 10
End
```

Printing will automatically be turned on again after a call to E04UCF and may be turned on again at any time by the user by using the keyword list.

Optional parameter settings are preserved following a call to E04UCF, and so the keyword defaults is provided to allow the user to reset all the optional parameters to their default values prior to a subsequent call to E04UCF.

A complete list of optional parameters, their abbreviations, synonyms and default values is given in Section 5.1 of the document for E04UCF.

4. References

None.

5. Parameters

1: IOPTNS -- INTEGER Input
On entry: IOPTNS must be the unit number of the options file. Constraint: 0 <= IOPTNS <= 99.

2: INFORM -- INTEGER Output
On exit: INFORM will be zero, if an options file with the current structure has been read. Otherwise INFORM will be positive. Positive values of INFORM indicate that an options file may not have been successfully read as follows:

INFORM = 1
IOPTNS is not in the range [0,99].

INFORM = 2
begin was found, but end-of-file was found before end was found.

INFORM = 3
end-of-file was found before begin was found.

6. Error Indicators and Warnings

If a line is not recognised as a valid option, then a warning message is output on the current advisory message unit (X04ABF).

```
          7. Accuracy

          Not applicable.

          8. Further Comments

          E04UEF may also be used to supply optional parameters to E04UCF.

          9. Example

          See the example for E04UCF.

\end{verbatim}
\endscroll
\end{page}
```

Supply individual optional params to E04UCF

— nage.ht —

```
\begin{page}{manpageXXe04uef}{NAG Documentation: e04uef}
\beginscroll
\begin{verbatim}



     E04UEF(3NAG)      Foundation Library (12/10/92)      E04UEF(3NAG)



          E04 -- Minimizing or Maximizing a Function                E04UEF
                  E04UEF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          To supply individual optional parameters to E04UCF.

          2. Specification

                 SUBROUTINE E04UEF (STRING)
```

```
CHARACTER*(*)    STRING
```

3. Description

E04UEF may be used to supply values for optional parameters to E04UCF. It is only necessary to call E04UEF for those parameters whose values are to be different from their default values. One call to E04UEF sets one parameter value.

Each optional parameter is defined by a single character string of up to 72 characters, consisting of one or more items. The items associated with a given option must be separated by spaces, or equal signs (=). Alphabetic characters may be upper or lower case. The string

```
Print level = 1
```

is an example of a string used to set an optional parameter. For each option the string contains one or more of the following items:

(a) A mandatory keyword.

(b) A phrase that qualifies the keyword.

(c) A number that specifies an INTEGER or real value. Such numbers may be up to 16 contiguous characters in Fortran 77's I, F, E or D formats, terminated by a space if this is not the last item on the line.

Blank strings and comments are ignored. A comment begins with an asterisk (*) and all subsequent characters in the string are regarded as part of the comment.

Normally, each user-specified option is printed as it is defined, on the current advisory message unit (see X04ABF), but this printing may be suppressed using the keyword nolist Thus the statement

```
CALL E04UEF ('Nolist')
```

suppresses printing of this and subsequent options. Printing will automatically be turned on again after a call to E04UCF, and may be turned on again at any time by the user, by using the keyword list.

Optional parameter settings are preserved following a call to E04UCF, and so the keyword defaults is provided to allow the user to reset all the optional parameters to their default values by the statement,

```
      CALL E04UEF ('Defaults')

prior to a subsequent call to E04UCF.

A complete list of optional parameters, their abbreviations,
synonyms and default values is given in Section 5.1 of the
document for E04UCF.

4. References

None.

5. Parameters

 1:  STRING -- CHARACTER*(*)                                 Input
     On entry: STRING must be a single valid option string. See
     Section 3 above and Section 5.1 of the routine document for
     E04UCF. On entry: STRING must be a single valid option
     string. See Section 3 above and Section 5.1 of the routine
     document for E04UCF.

6. Error Indicators and Warnings

If the parameter STRING is not recognised as a valid option
string, then a warning message is output on the current advisory
message unit (X04ABF).

7. Accuracy

Not applicable.

8. Further Comments

E04UDF may also be used to supply optional parameters to E04UCF.

9. Example

See the example for E04UCF.
```

```
\end{verbatim}
\endscroll
\end{page}
```

Estimates of elements of the variance-covariance matrix

— nage.ht —

```
\begin{page}{manpageXXe04ycf}{NAG Documentation: e04ycf}
\beginscroll
\begin{verbatim}



     E04YCF(3NAG)      Foundation Library (12/10/92)      E04YCF(3NAG)



          E04 -- Minimizing or Maximizing a Function                 E04YCF
                  E04YCF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          E04YCF returns estimates of elements of the variance-covariance
          matrix of the estimated regression coefficients for a nonlinear
          least squares problem. The estimates are derived from the
          Jacobian of the function f(x) at the solution.

          This routine may be used following any one of the nonlinear
          least-squares routines E04FCF(*), E04FDF, E04GBF(*), E04GCF,
          E04GDF(*), E04GEF(*), E04HEF(*), E04HFF(*).

          2. Specification

                 SUBROUTINE E04YCF (JOB, M, N, FSUMSQ, S, V, LV, CJ, WORK,
                1                   IFAIL)
                 INTEGER          JOB, M, N, LV, IFAIL
                 DOUBLE PRECISION FSUMSQ, S(N), V(LV,N), CJ(N), WORK(N)

          3. Description

          E04YCF is intended for use when the nonlinear least-squares
                          T
          function, F(x)=f (x)f(x), represents the goodness of fit of a
          nonlinear model to observed data. The routine assumes that the
          Hessian of F(x), at the solution, can be adequately approximated
               T
          by 2J J, where J is the Jacobian of f(x) at the solution. The
```

estimated variance-covariance matrix C is then given by

$$C=\sigma^2 (J^T J)^{-1} \quad J^T J \text{ non-singular},$$

where σ^2 is the estimated variance of the residual at the

solution, x, given by

$$\sigma^2 = \frac{F(x)}{m-n},$$

m being the number of observations and n the number of variables.

The diagonal elements of C are estimates of the variances of the estimated regression coefficients. See the Chapter Introduction E04 and Bard [1] and Wolberg [2] for further information on the use of C.

When $J^T J$ is singular then C is taken to be

$$C=\sigma^2 (J^T J)^*,$$

where $(J^T J)^*$ is the pseudo-inverse of $J^T J$, but in this case the parameter IFAIL is returned as non-zero as a warning to the user that J has linear dependencies in its columns. The assumed rank of J can be obtained from IFAIL.

The routine can be used to find either the diagonal elements of C, or the elements of the jth column of C, or the whole of C.

E04YCF must be preceded by one of the nonlinear least-squares routines mentioned in Section 1, and requires the parameters FSUMSQ, S and V to be supplied by those routines. FSUMSQ is the

residual sum of squares F(x), and S and V contain the singular values and right singular vectors respectively in the singular value decomposition of J. S and V are returned directly by the comprehensive routines E04FCF(*), E04GBF(*), E04GDF(*) and E04HEF(*), but are returned as part of the workspace parameter W from the easy-to-use routines E04FDF, E04GCF, E04GEF(*) and

E04HFF(*). In the case of E04FDF, S starts at W(NS), where

NS=6*N+2*M+M*N+1+max(1,N*(N-1)/2)

and in the cases of the remaining easy-to-use routines, S starts at W(NS), where

NS=7*N+2*M+M*N+N*(N+1)/2+1+max(1,N*(N-1)/2)

The parameter V starts immediately following the elements of S, so that V starts at W(NV), where

NV=NS+N.

For all the easy-to-use routines the parameter LV must be supplied as N. Thus a call to E04YCF following E04FDF can be illustrated as

```
       CALL E04FDF (M, N, X, FSUMSQ, IW, LIW, W, LW, IFAIL)
       NS = 6*N + 2*M + M*N + 1 + MAX((1,(N*(N-1))/2)
       NV = NS + N
       CALL E04YCF (JOB, M, N, FSUMSQ, W(NS), W(NV),
      *             N, CJ, WORK, IFAIL)
```

where the parameters M, N, FSUMSQ and the $(n+n^2)$ elements W(NS), WS(NS+1),..., W(NV+N*N-1) must not be altered between the calls to E04FDF and E04YCF. The above illustration also holds for a call to E04YCF following a call to one of E04GCF, E04GEF(*), E04HFF(*) except that NS must be computed as

NS = 7*N + 2*M + M*N + (N*(N+1))/2 + 1 + MAX((1,N*(N-1))/2)

4. References

[1] Bard Y (1974) Nonlinear Parameter Estimation. Academic Press.

[2] Wolberg J R (1967) Prediction Analysis. Van Nostrand.

5. Parameters

1: JOB -- INTEGER Input
On entry: which elements of C are returned as follows:
JOB = -1
The n by n symmetric matrix C is returned.

JOB = 0
The diagonal elements of C are returned.

JOB > 0
The elements of column JOB of C are returned.
Constraint: -1 <= JOB <= N.

2: M -- INTEGER Input
On entry: the number m of observations (residuals $f_i(x)$).
Constraint: M >= N.

3: N -- INTEGER Input
On entry: the number n of variables (x_j). Constraint: 1 <= N <= M.

4: FSUMSQ -- DOUBLE PRECISION Input

On entry: the sum of squares of the residuals, F(x), at the

solution x, as returned by the nonlinear least-squares routine. Constraint: FSUMSQ >= 0.0.

5: S(N) -- DOUBLE PRECISION array Input
On entry: the n singular values of the Jacobian as returned by the nonlinear least-squares routine. See Section 3 for information on supplying S following one of the easy-to-use routines.

6: V(LV,N) -- DOUBLE PRECISION array Input/Output
On entry: the n by n right-hand orthogonal matrix (the right singular vectors) of J as returned by the nonlinear least-squares routine. See Section 3 for information on supplying V following one of the easy-to-use routines. On exit: when JOB >= 0 then V is unchanged.

When JOB = -1 then the leading n by n part of V is overwritten by the n by n matrix C. When E04YCF is called with JOB = -1 following an easy-to-use routine this means that C is returned, column by column, in the n^2 elements of W given by W(NV),W(NV+1),...,W(NV+N^2-1). (See Section 3 for the definition of NV).

7: LV -- INTEGER Input
On entry:
the first dimension of the array V as declared in the (sub)program from which E04YCF is called.

When V is passed in the workspace parameter W following one of the easy-to-use least-square routines, LV must be the value N.

8: CJ(N) -- DOUBLE PRECISION array Output
On exit: with JOB = 0, CJ returns the n diagonal elements of C.

With JOB = j>0, CJ returns the n elements of the jth column of C.

When JOB = -1, CJ is not referenced.

9: WORK(N) -- DOUBLE PRECISION array Workspace
When JOB = -1 or 0 then WORK is used as internal workspace.

When JOB > 0, WORK is not referenced.

10: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. Users who are unfamiliar with this parameter should refer to the Essential Introduction for details.

On exit: IFAIL = 0 unless the routine detects an error or gives a warning (see Section 6).

For this routine, because the values of output parameters may be useful even if IFAIL /=0 on exit, users are recommended to set IFAIL to -1 before entry. It is then essential to test the value of IFAIL on exit. To suppress the output of an error message when soft failure occurs, set IFAIL to 1.

6. Error Indicators and Warnings

Errors or warnings specified by the routine:

IFAIL= 1
On entry JOB < -1,

or JOB > N,

or N < 1,

or M < N,

or FSUMSQ < 0.0.

IFAIL= 2
The singular values are all zero, so that at the solution

the Jacobian matrix J has rank 0.

IFAIL> 2
At the solution the Jacobian matrix contains linear, or near linear, dependencies amongst its columns. In this case the required elements of C have still been computed based upon J having an assumed rank given by (IFAIL-2). The rank is computed by regarding singular values SV(j) that are not larger than 10*(epsilon)*SV(1) as zero, where (epsilon) is the machine precision (see X02AJF(*)). Users who expect near linear dependencies at the solution and are happy with this tolerance in determining rank should call E04YCF with IFAIL = 1 in order to prevent termination by P01ABF(*). It is then essential to test the value of IFAIL on exit from E04YCF.

IFAILOverflow
If overflow occurs then either an element of C is very large, or the singular values or singular vectors have been incorrectly supplied.

7. Accuracy

The computed elements of C will be the exact covariances corresponding to a closely neighbouring Jacobian matrix J.

8. Further Comments

When JOB = -1 the time taken by the routine is approximately proportional to n^3. When JOB >= 0 the time taken by the routine is approximately proportional to n^2.

9. Example

To estimate the variance-covariance matrix C for the least-squares estimates of x_1, x_2 and x_3 in the model

$$y=x_1+\frac{t_1}{x_2t_2+x_3t_3}$$

using the 15 sets of data given in the following table:

y	t_1	t_2	t_3
0.14	1.0	15.0	1.0

```
                         0.18  2.0 14.0 2.0
                         0.22  3.0 13.0 3.0
                         0.25  4.0 12.0 4.0
                         0.29  5.0 11.0 5.0
                         0.32  6.0 10.0 6.0
                         0.35  7.0  9.0 7.0
                         0.39  8.0  8.0 8.0
                         0.37  9.0  7.0 7.0
                         0.58 10.0  6.0 6.0
                         0.73 11.0  5.0 5.0
                         0.96 12.0  4.0 4.0
                         1.34 13.0  3.0 3.0
                         2.10 14.0  2.0 2.0
                         4.39 15.0  1.0 1.0

          The program uses (0.5,1.0,1.5) as the initial guess at the
          position of the minimum and computes the least-squares solution
          using E04FDF. See the routine document E04FDF for further
          information.

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

———-

22.5 nagf.ht

Linear Algebra

— nagf.ht —

```
\begin{page}{manpageXXf}{NAG Documentation: f}
\beginscroll
\begin{verbatim}


     F(3NAG)          Foundation Library (12/10/92)           F(3NAG)
```

Chapter F
Linear Algebra

1. Introduction

The F Chapters of the Library are concerned with linear algebra and cover a large area. This general introduction is intended to help users decide which particular F Chapter is relevant to their problem. There are F Chapters with the following titles:

F01 -- Matrix Factorizations

F02 -- Eigenvalues and Eigenvectors

F04 -- Simultaneous Linear Equations

F06 -- Linear Algebra Support Routines

F07 -- Linear Equations (LAPACK)

The principal problem areas addressed by the above Chapters are:

Systems of linear equations

Linear least-squares problems

Eigenvalue and singular value problems

The solution of these problems usually involves several matrix operations, such as a matrix factorization followed by the solution of the factorized form, and the routines for these operations themselves utilize lower level support routines; typically routines from Chapter F06. Most users will not normally need to be concerned with these support routines.

NAG has been involved in a project, called LAPACK [1], to develop a linear algebra package for modern high-performance computers and some of the routines developed within that project are incorporated into the Library as Chapter F07. It should be emphasised that, while the LAPACK project has been concerned with high-performance computers, the routines do not compromise efficiency on conventional machines.

For background information on numerical algorithms for the solution of linear algebra problems see Golub and Van Loan [4]. For some problem areas the user has the choice of selecting a single routine to solve the problem, a so-called Black Box routine, or selecting more than one routine to solve the problem, such as a factorization routine followed by a solve routine, so-called General Purpose routines. The following sections indicate which chapters are relevant to particular problem areas.

2. Linear Equations

The Black Box routines for solving linear equations of the form

$$Ax=b \quad \text{and} \quad AX=B,$$

where A is an n by n real or complex, non-singular matrix, are to be found in Chapter F04. Such equations can also be solved by selecting a General Purpose factorization routine from Chapter F01 and combining it with a solve routine in Chapter F04, or by selecting a factorization and a solve routine from Chapter F07.

There are routines to cater for a variety of types of matrix, including general, symmetric or Hermitian, symmetric or Hermitian positive definite, tridiagonal, skyline and sparse matrices.

In order to select the appropriate routine, users are recommended to consult the F04 Chapter Introduction in the first instance.

3. Linear Least-squares

Routines for solving linear least-squares problems of the form

$$\underset{x}{\text{minimize}}\ r^T r, \quad \text{where} \quad r=b-Ax,$$

where A is an m by n, possibly rank deficient, matrix are to be found in Chapter F04. Linear least-squares problems can also be solved by routines in the statistical Chapter G02.

In order to select the appropriate routine, users are recommended to consult the F04 Chapter Introduction in the first instance, but users with additional statistical requirements may prefer to consult the G02 Chapter Introduction.

4. Eigenvalue Problems and Singular Value Problems

Routines for solving standard matrix eigenvalue problems of the form

$$Ax=(\lambda)x,$$

where A is an n by n real or complex matrix, and generalized matrix eigenvalue problems of the form

Ax=(lambda)Bx

where B is also an n by n matrix are to be found in Chapter F02.

There are routines to cater for various types of matrices, including general, symmetric or Hermitian and sparse matrices.

Similarly, the routines for finding singular values and/or singular vectors of an m by n real or complex matrix A are to be found in Chapter F02.

In order to select the appropriate routine, users are recommended to consult the F02 Chapter Introduction in the first instance.

5. Matrix Factorizations

Routines for various sorts of matrix factorization are to be found in Chapters F01 and F07 together with associated transformation routines. In order to select the appropriate routine users are recommended to consult the F01 Chapter Introduction in the first instance.

6. Support Routines

Chapter F06 contains a variety of routines to perform elementary algebraic operations involving scalars, vectors and matrices, such as setting up a plane rotation, performing a dot product and computing a matrix-vector product. Chapter F06 contains routines that meet the specification of the BLAS (Basic Linear Algebra Subprograms) [5, 3, 2]. The routines in this chapter will not normally be required by the general user, but are intended for use by those who require to build specialist linear algebra modules. The BLAS are extensively used by other NAG Foundation Library routines.

References

[1] Anderson E, Bai Z, Bischof C, Demmel J, Dongarra J J, Du Croz J, Greenbaum A, Hammarling S, McKenney A, Ostrouchov S and Sorensen D (1992) LAPACK Users' Guide. SIAM Philadelphia.

[2] Dongarra J J, Du Croz J J, Duff I S and Hammarling S (1990) A Set of Level 3 Basic Linear Algebra Subprograms. ACM Trans. Math. Softw. 16 1--28.

[3] Dongarra J J, Du Croz J J, Hammarling S and Hanson R J (1988) An Extended Set of FORTRAN Basic Linear Algebra

```
          Subprograms. ACM Trans. Math. Softw. 14 1--32.

     [4]  Golub G H and Van Loan C F (1989) Matrix Computations (2nd
          Edition). Johns Hopkins University Press, Baltimore,
          Maryland.

     [5]  Lawson C L, Hanson R J, Kincaid D R and Krogh F T (1979)
          Basic Linear Algebra Subprograms for Fortran Usage. ACM
          Trans. Math. Softw. 5 308--325.

     [6]  Parlett B N (1980) The Symmetric Eigenvalue Problem.
          Prentice-Hall.

     [7]  Wilkinson J H (1965) The Algebraic Eigenvalue Problem.
          Clarendon Press.

     [8]  Wilkinson J H (1977) Some Recent Advances in Numerical
          Linear Algebra. The State of the Art in Numerical Analysis.
          (ed D A H Jacobs) Academic Press.

     [9]  Wilkinson J H (1978) Singular Value Decomposition -- Basic
          Aspects. Numerical Software -- Needs and Availability. (ed D
          A H Jacobs) Academic Press.

     [10] Wilkinson J H and Reinsch C (1971) Handbook for Automatic
          Computation II, Linear Algebra. Springer-Verlag.
```


```
\end{verbatim}
\endscroll
\end{page}
```

———-

Matrix Factorization

— **nagf.ht** —

```
\begin{page}{manpageXXf01}{NAG Documentation: f01}
\beginscroll
\begin{verbatim}


     F01(3NAG)         Foundation Library (12/10/92)         F01(3NAG)
```

Chapter F01
Matrix Factorization

1. Scope of the Chapter

This chapter provides facilities for matrix factorizations and associated transformations.

2. Background to the Problems

An n by n matrix may be factorized as

$$A=PLUQ^T,$$

where L and U are respectively lower and uper triangular matrices, and P and Q are permutation matrices. This is called an LU factorization. For general dense matrices it is usual to choose Q=I and to then choose P to ensure that the factorization is numerically stable. For sparse matrices, judicious choice of P and Q ensures numerical stability as well as maintaining as much sparsity as possible in the factors L and U. The LU factorization is normally used in connection with the solution of the linear equations

$$Ax=b,$$

whose solution, x, may then be obtained by solving in succession the simpler equations

$$Ly=P^T b, \quad Uz=y, \quad x=Qz$$

the first by forward substitution and the second by backward substitution. Routines to perform this solution are to be found in Chapter F04.

When A is symmetric positive-definite then we can choose $U=L^T$ and Q=P, to give the Cholesky factorization. This factorization is numerically stable without permutations, but in the sparse case the permutations can again be used to try to maintain sparsity. The Cholesky factorization is sometimes expressed as

$$A=PLDL^T P^T,$$

where D is a diagonal matrix with positive diagonal elements and

L is unit lower triangular.

The LU factorization can also be performed on rectangular matrices, but in this case it is more usual to perform a QR factorization. When A is an m by n (m>=n) matrix this is given by

$$A=Q\begin{pmatrix}R\\0\end{pmatrix},$$

where R is an n by n upper triangular matrix and Q is an orthogonal (unitary in the complex case) matrix.

3. Recommendations on Choice and Use of Routines

Routine F07ADF performs the LU factorization of a real m by n dense matrix.

The LU factorization of a sparse matrix is performed by routine F01BRF. Following the use of F01BRF, matrices with the same sparsity pattern may be factorized by routine F01BSF.

The Cholesky factorization of a real symmetric positive-definite dense matrix is performed by routine F07FDF.

Routine F01MCF performs the Cholesky factorization of a real symmetric positive-definite variable band (skyline) matrix, and a general sparse symmetric positive-definite matrix may be factorized using routine F01MAF.

The QR factorization of an m by n (m>=n) matrix is performed by routine F01QCF in the real case, and F01RCF in the complex case. Following the use of F01QCF, operations with Q may be performed using routine F01QDF and some, or all, of the columns of Q may be formed using routine F01QEF. Routines F01RDF and F01REF perform the same tasks following the use of F01RCF.

F01 -- Matrix Factorizations Contents -- F01

Chapter F01

Matrix Factorizations

```
                definite matrix

                   T
        F01MCF  LDL  factorization of real symmetric positive-definite
                variable-bandwidth matrix

        F01QCF  QR factorization of real m by n matrix (m>=n)

                                                            T
        F01QDF  Operations with orthogonal matrices, compute QB or Q B
                after factorization by F01QCF

        F01QEF  Operations with orthogonal matrices, form columns of Q
                after factorization by F01QCF

        F01RCF  QR factorization of complex m by n matrix (m>=n)

                                                         H
        F01RDF  Operations with unitary matrices, compute QB or Q B after
                factorization by F01RCF

        F01REF  Operations with unitary matrices, form columns of Q after
                factorization by F01RCF

\end{verbatim}
\endscroll
\end{page}
```

Factorizes a real sparse matrix

— nagf.ht —

```
\begin{page}{manpageXXf01brf}{NAG Documentation: f01brf}
\beginscroll
\begin{verbatim}



     F01BRF(3NAG)      Foundation Library (12/10/92)      F01BRF(3NAG)



        F01 -- Matrix Factorizations                               F01BRF
                F01BRF -- NAG Foundation Library Routine Document
```

Note: Before using this routine, please read the Users' Note for your implementation to check implementation-dependent details. The symbol (*) after a NAG routine name denotes a routine that is not included in the Foundation Library.

1. Purpose

F01BRF factorizes a real sparse matrix. The routine either forms the LU factorization of a permutation of the entire matrix, or, optionally, first permutes the matrix to block lower triangular form and then only factorizes the diagonal blocks.

2. Specification

```
       SUBROUTINE F01BRF (N, NZ, A, LICN, IRN, LIRN, ICN, PIVOT,
      1                   IKEEP, IW, W, LBLOCK, GROW, ABORT,
      2                   IDISP, IFAIL)
       INTEGER          N, NZ, LICN, IRN(LIRN), LIRN, ICN(LICN),
      1                 IKEEP(5*N), IW(8*N), IDISP(10), IFAIL
       DOUBLE PRECISION A(LICN), PIVOT, W(N)
       LOGICAL          LBLOCK, GROW, ABORT(4)
```

3. Description

Given a real sparse matrix A, this routine may be used to obtain the LU factorization of a permutation of A,

$$PAQ=LU$$

where P and Q are permutation matrices, L is unit lower triangular and U is upper triangular. The routine uses a sparse variant of Gaussian elimination, and the pivotal strategy is designed to compromise between maintaining sparsity and controlling loss of accuracy through round-off.

Optionally the routine first permutes the matrix into block lower triangular form and then only factorizes the diagonal blocks. For some matrices this gives a considerable saving in storage and execution time.

Extensive data checks are made; duplicated non-zeros can be accumulated.

The factorization is intended to be used by F04AXF to solve sparse systems of linear equations $Ax=b$ or $A^T x=b$. If several matrices of the same sparsity pattern are to be factorized, F01BSF should be used for the second and subsequent matrices.

The method is fully described by Duff [1].

4. References

[1] Duff I S (1977) MA28 -- a set of Fortran subroutines for sparse unsymmetric linear equations. A.E.R.E. Report R.8730. HMSO.

5. Parameters

1: N -- INTEGER Input
On entry: n, the order of the matrix A. Constraint: N > 0.

2: NZ -- INTEGER Input
On entry: the number of non-zero elements in the matrix A. Constraint: NZ > 0.

3: A(LICN) -- DOUBLE PRECISION array Input/Output
On entry: A(i), for i = 1,2,...,NZ must contain the non-zero elements of the sparse matrix A. They can be in any order since the routine will reorder them. On exit: the non-zero elements in the LU factorization. The array must not be changed by the user between a call of this routine and a call of F04AXF.

4: LICN -- INTEGER Input
On entry:
the dimension of the arrays A and ICN as declared in the (sub)program from which F01BRF is called.
Since the factorization is returned in A and ICN, LICN should be large enough to accommodate this and should ordinarily be 2 to 4 times as large as NZ. Constraint: LICN >= NZ.

5: IRN(LIRN) -- INTEGER array Input/Output
On entry: IRN(i), for i = 1,2,...,NZ must contain the row index of the non-zero element stored in A(i). On exit: the array is overwritten and is not needed for subsequent calls of F01BSF or F04AXF.

6: LIRN -- INTEGER Input
On entry:
the dimension of the array IRN as declared in the (sub)program from which F01BRF is called.
It need not be as large as LICN; normally it will not need to be very much greater than NZ. Constraint: LIRN >= NZ.

7: ICN(LICN) -- INTEGER array Input/Output
On entry: ICN(i), for i = 1,2,...,NZ must contain the column index of the non-zero element stored in A(i). On exit: the column indices of the non-zero elements in the

```
     factorization. The array must not be changed by the user
     between a call of this routine and subsequent calls of
     F01BSF or F04AXF.

  8: PIVOT -- DOUBLE PRECISION                          Input
     On entry: PIVOT should have a value in the range 0.0 <=
     PIVOT <= 0.9999 and is used to control the choice of pivots.
     If PIVOT < 0.0, the value 0.0 is assumed, and if PIVOT > 0.
     9999, the value 0.9999 is assumed. When searching a row for
     a pivot, any element is excluded which is less than PIVOT
     times the largest of those elements in the row available as
     pivots. Thus decreasing PIVOT biases the algorithm to
     maintaining sparsity at the expense of stability. Suggested
     value: PIVOT = 0.1 has been found to work well on test
     examples.

  9: IKEEP(5*N) -- INTEGER array                       Output
     On exit: indexing information about the factorization. The
     array must not be changed by the user between a call of this
     routine and calls of F01BSF or F04AXF.

 10: IW(8*N) -- INTEGER array                       Workspace

 11: W(N) -- DOUBLE PRECISION array                    Output
     On exit: if GROW = .TRUE., W(1) contains an estimate (an
     upper bound) of the increase in size of elements encountered
     during the factorization (see GROW); the rest of the array
     is used as workspace.

     If GROW = .FALSE., the array is not used.

 12: LBLOCK -- LOGICAL                                  Input
     On entry: if LBLOCK = .TRUE., the matrix is pre-ordered
     into block lower triangular form before the LU factorization
     is performed; otherwise the entire matrix is factorized.
     Suggested value: LBLOCK = .TRUE. unless the matrix is known
     to be irreducible.

 13: GROW -- LOGICAL                                    Input
     On entry: if GROW = .TRUE., then on exit W(1) contains an
     estimate (an upper bound) of the increase in size of
     elements encountered during the factorization. If the matrix
     is well-scaled (see Section 8.2), then a high value for W(1)
     indicates that the LU factorization may be inaccurate and
     the user should be wary of the results and perhaps increase
     the parameter PIVOT for subsequent runs (see Section 7).
     Suggested value: GROW = .TRUE..

 14: ABORT(4) -- LOGICAL array                          Input
     On entry:
```

if ABORT(1) = .TRUE., the routine will exit immediately on detecting a structural singularity (one that depends on the pattern of non-zeros) and return

IFAIL = 1; otherwise it will complete the factorization (see Section 8.3).

If ABORT(2) = .TRUE., the routine will exit immediately on detecting a numerical singularity (one that depends on the numerical values) and return IFAIL = 2; otherwise it will complete the factorization (see Section 8.3).

If ABORT(3) = .TRUE., the routine will exit immediately (with IFAIL = 5) when the arrays A and ICN are filled up by the previously factorized, active and unfactorized parts of the matrix; otherwise it continues so that better guidance on necessary array sizes can be given in IDISP(6) and IDISP(7), and will exit with IFAIL in the range 4 to 6. Note that there is always an immediate error exit if the array IRN is too small.

If ABORT(4) = .TRUE., the routine exits immediately (with IFAIL = 13) if it finds duplicate elements in the input matrix. If ABORT(4) = .FALSE., the routine proceeds using a value equal to the sum of the duplicate elements. In either case details of each duplicate element are output on the current advisory message unit (see X04ABF), unless suppressed by the value of IFAIL on entry.

Suggested values:

ABORT(1) = .TRUE.

ABORT(2) = .TRUE.

ABORT(3) = .FALSE.

ABORT(4) = .TRUE..

15: IDISP(10) -- INTEGER array Output

On exit: IDISP is used to communicate information about the factorization to the user and also between a call of F01BRF and subsequent calls to F01BSF or F04AXF.

IDISP(1) and IDISP(2), indicate the position in arrays A and ICN of the first and last elements in the LU factorization of the diagonal blocks. (IDISP(2) gives the number of non-zeros in the factorization.)

IDISP(3) and IDISP(4), monitor the adequacy of 'elbow

```
        room' in the arrays IRN and A/ICN respectively, by
        giving the number of times that the data in these
        arrays has been compressed during the factorization to
        release more storage. If either IDISP(3) or IDISP(4)
        is quite large (say greater than 10), it will probably
        pay the user to increase the size of the corresponding
        array(s) for subsequent runs. If either is very low or
        zero, then the user can perhaps save storage by
        reducing the size of the corresponding array(s).

        IDISP(5), gives an upper bound on the rank of the
        matrix.

        IDISP(6) and IDISP(7), give the minimum size of arrays
        IRN and A/ICN respectively which would enable a
        successful run on an identical matrix (but some '
        elbow-room' should be allowed - see Section 8).

        IDISP(8) to (10), are only used if LBLOCK = .TRUE..

        IDISP(8), gives the structural rank of the matrix.

        IDISP(9), gives the number of diagonal blocks.

        IDISP(10), gives the size of the largest diagonal
        block.

        IDISP(1) and IDISP(2), must not be changed by the user
        between a call of F01BRF and subsequent calls to
        F01BSF or F04AXF.

16:  IFAIL -- INTEGER                                 Input/Output
     For this routine, the normal use of IFAIL is extended to
     control the printing of error and warning messages as well
     as specifying hard or soft failure (see the Essential
     Introduction).

     Before entry, IFAIL must be set to a value with the decimal
     expansion cba, where each of the decimal digits c, b and a
     must have a value of 0 or 1.
     a=0   specifies hard failure, otherwise soft failure;

     b=0   suppresses error messages, otherwise error messages
           will be printed (see Section 6);

     c=0   suppresses warning messages, otherwise warning
           messages will be printed (see Section 6).
     The recommended value for inexperienced users is 110 (i.e.,
     hard failure with all messages printed).
```

Unless the routine detects an error (see Section 6), IFAIL contains 0 on exit.

6. Error Indicators and Warnings

Errors detected by the routine:

For each error, an explanatory error message is output on the current error message unit (as defined by X04AAF), unless suppressed by the value of IFAIL on entry.

IFAIL=-2
: Successful factorization of a numerically singular matrix (which may also be structurally singular) (see Section 8.3).

IFAIL=-1
: Successful factorization of a structurally singular matrix (see Section 8.3).

IFAIL= 1
: The matrix is structurally singular and the factorization has been abandoned (ABORT(1) was .TRUE. on entry).

IFAIL= 2
: The matrix is numerically singular and the factorization has been abandoned (ABORT(2) was .TRUE. on entry).

IFAIL= 3
: LIRN is too small: there is not enough space in the array IRN to continue the factorization. The user is recommended to try again with LIRN (and the length of IRN) equal to at least IDISP(6) + N/2.

IFAIL= 4
: LICN is much too small: there is much too little space in the arrays A and ICN to continue the factorization.

IFAIL= 5
: LICN is too small: there is not enough space in the arrays A and ICN to store the factorization. If ABORT(3) was .FALSE. on entry, the factorization has been completed but some of the LU factors have been discarded to create space, IDISP(7) then gives the minimum value of LICN (i.e., the minimum length of A and ICN) required for a successful factorization of the same matrix.

IFAIL= 6
: LICN and LIRN are both too small: effectively this is a combination of IFAIL = 3 and IFAIL = 5 (with ABORT(3) = . FALSE.).

IFAIL= 7
LICN is too small: there is not enough space in the arrays A and ICN for the permutation to block triangular form.

IFAIL= 8
On entry N <= 0.

IFAIL= 9
On entry NZ <= 0.

IFAIL= 10
On entry LICN < NZ.

IFAIL= 11
On entry LIRN < NZ.

IFAIL= 12
On entry an element of the input matrix has a row or column index (i.e., an element of IRN or ICN) outside the range 1 to N.

IFAIL= 13
Duplicate elements have been found in the input matrix and the factorization has been abandoned (ABORT(4) = .TRUE. on entry).

7. Accuracy

The factorization obtained is exact for a perturbed matrix whose (i,j)th element differs from a_{ij} by less than 3(epsilon)(rho)m_{ij} where (epsilon) is the machine precision, (rho) is the growth value returned in W(1) if GROW = .TRUE., and m_{ij} the number of Gaussian elimination operations applied to element (i,j). The value of m_{ij} is not greater than n and is usually much less. Small (rho) values therefore guarantee accurate results, but unfortunately large (rho) values may give a very pessimistic indication of accuracy.

8. Further Comments

8.1. Timing

The time required may be estimated very roughly from the number (tau) of non-zeros in the factorized form (output as IDISP(2)) and for this routine and its associates is

F01BRF: $5(\tau)^2/n$ units

F01BSF: $(\tau)^2/n$ units

F04AXF: $2(\tau)$ units

where our unit is the time for the inner loop of a full matrix code (e.g. solving a full set of equations takes about $\frac{1}{3}n^3$ units). Note that the faster F01BSF time makes it well worthwhile to use this for a sequence of problems with the same pattern.

It should be appreciated that (tau) varies widely from problem to problem. For network problems it may be little greater than NZ, the number of non-zeros in A; for discretisation of 2-dimensional and 3-dimensional partial differential equations it may be about $3n\log_2 n$ and $\frac{1}{2}n^{5/3}$, respectively.

The time taken to find the block lower triangular form (LBLOCK = it is not found (LBLOCK = .FALSE.). If the matrix is irreducible (IDISP(9) = 1 after a call with LBLOCK = .TRUE.) then this time is wasted. Otherwise, particularly if the largest block is small (IDISP(10)<<n), the consequent savings are likely to be greater.

The time taken to estimate growth (GROW = .TRUE.) is typically under 2% of the overall time.

The overall time may be substantially increased if there is inadequate 'elbow-room' in the arrays A, IRN and ICN. When the sizes of the arrays are minimal (IDISP(6) and IDISP(7)) it can execute as much as three times slower. Values of IDISP(3) and IDISP(4) greater than about 10 indicate that it may be worthwhile to increase array sizes.

8.2. Scaling

The use of a relative pivot tolerance PIVOT essentially presupposes that the matrix is well-scaled, i.e., that the matrix elements are broadly comparable in size. Practical problems are often naturally well-scaled but particular care is needed for problems containing mixed types of variables (for example millimetres and neutron fluxes).

8.3. Singular and Rectangular Systems

```
It is envisaged that this routine will almost always be called
for square non-singular matrices and that singularity indicates
an error condition. However, even if the matrix is singular it is
possible to complete the factorization. It is even possible for
F04AXF to solve a set of equations whose matrix is singular
provided the set is consistent.

Two forms of singularity are possible. If the matrix would be
singular for any values of the non-zeros (e.g. if it has a whole
row of zeros), then we say it is structurally singular, and
continue only if ABORT(1) = .FALSE.. If the matrix is non-
singular by virtue of the particular values of the non-zeros,
then we say that it is numerically singular and continue only if
ABORT(2) = .FALSE..

Rectangular matrices may be treated by setting N to the larger of
the number of rows and numbers of columns and setting ABORT(1) =

Note: the soft failure option should be used (last digit of IFAIL
= 1) if the user wishes to factorize singular matrices with ABORT
(1) or ABORT(2) set to .FALSE..

8.4. Duplicated Non-zeros

The matrix A may consist of a sum of contributions from different
sub-systems (for example finite elements). In such cases the user
may rely on this routine to perform assembly, since duplicated
elements are summed.

9. Example

To factorize the real sparse matrix:

                       ( 5  0  0  0 0  0)
                       ( 0  2 -1  2 0  0)
                       ( 0  0  3  0 0  0)
                       (-2  0  0  1 1  0).
                       (-1  0  0 -1 2 -3)
                       (-1 -1  0  0 0  6)

This example program simply prints out some information about the
factorization as returned by F01BRF in W(1) and IDISP. Normally
the call of F01BRF would be followed by a call of F04AXF (see
Example for F04AXF).

The example program is not reproduced here. The source code for
all example programs is distributed with the NAG Foundation
Library software and should be available on-line.
```

\end{verbatim}

```
\endscroll
\end{page}
```

———-

Factorizes a real sparse matrix

— nagf.ht —

```
\begin{page}{manpageXXf01bsf}{NAG Documentation: f01bsf}
\beginscroll
\begin{verbatim}



     F01BSF(3NAG)      Foundation Library (12/10/92)      F01BSF(3NAG)



          F01 -- Matrix Factorizations                               F01BSF
                  F01BSF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          F01BSF factorizes a real sparse matrix using the pivotal sequence
          previously obtained by F01BRF when a matrix of the same sparsity
          pattern was factorized.

          2. Specification

                 SUBROUTINE F01BSF (N, NZ, A, LICN, IVECT, JVECT, ICN,
                1                   IKEEP, IW, W, GROW, ETA, RPMIN, ABORT,
                2                   IDISP, IFAIL)
                 INTEGER          N, NZ, LICN, IVECT(NZ), JVECT(NZ), ICN
                1                 (LICN), IKEEP(5*N), IW(8*N), IDISP(2),
                2                 IFAIL
                 DOUBLE PRECISION A(LICN), W(N), ETA, RPMIN
                 LOGICAL          GROW, ABORT

          3. Description

          This routine accepts as input a real sparse matrix of the same
```

sparsity pattern as a matrix previously factorized by a call of F01BRF. It first applies to the matrix the same permutations as were used by F01BRF, both for permutation to block triangular form and for pivoting, and then performs Gaussian elimination to obtain the LU factorization of the diagonal blocks.

Extensive data checks are made; duplicated non-zeros can be accumulated.

The factorization is intended to be used by F04AXF to solve sparse systems of linear equations $Ax=b$ or $A^T x=b$.

F01BSF is much faster than F01BRF and in some applications it is expected that there will be many calls of F01BSF for each call of F01BRF.

The method is fully described in Duff [1].

4. References

[1] Duff I S (1977) MA28 -- a set of Fortran subroutines for sparse unsymmetric linear equations. A.E.R.E. Report R.8730. HMSO.

5. Parameters

1: N -- INTEGER Input
On entry: n, the order of the matrix A. Constraint: N > 0.

2: NZ -- INTEGER Input
On entry: the number of non-zeros in the matrix A.
Constraint: NZ > 0.

3: A(LICN) -- DOUBLE PRECISION array Input/Output
On entry: A(i), for i = 1,2,...,NZ must contain the non-zero elements of the sparse matrix A. They can be in any order since the routine will reorder them. On exit: the non-zero elements in the factorization. The array must not be changed by the user between a call of this routine and a call of F04AXF.

4: LICN -- INTEGER Input
On entry:
the dimension of the arrays A and ICN as declared in the (sub)program from which F01BSF is called.
It should have the same value as it had for F01BRF.
Constraint: LICN >= NZ.

5: IVECT(NZ) -- INTEGER array Input

```
 6: JVECT(NZ) -- INTEGER array                              Input
    On entry: IVECT(i) and JVECT(i), for i = 1,2,...,NZ must
    contain the row index and the column index respectively of
    the non-zero element stored in A(i).

 7: ICN(LICN) -- INTEGER array                              Input
    On entry: the same information as output by F01BRF. It must
    not be changed by the user between a call of this routine
    and a call of F04AXF.

 8: IKEEP(5*N) -- INTEGER array                             Input
    On entry: the same indexing information about the
    factorization as output from F01BRF. It must not be changed
    between a call of this routine and a call of F04AXF.

 9: IW(8*N) -- INTEGER array                            Workspace

10: W(N) -- DOUBLE PRECISION array                         Output
    On exit: if GROW = .TRUE., W(1) contains an estimate (an
    upper bound) of the increase in size of elements encountered
    during the factorization (see GROW); the rest of the array
    is used as workspace.

    If GROW = .FALSE., the array is not used.

11: GROW -- LOGICAL                                         Input
    On entry: if GROW = .TRUE., then on exit W(1) contains an
    estimate (an upper bound) of the increase in size of
    elements encountered during the factorization. If the matrix
    is well-scaled (see Section 8.2), then a high value for W(1)
    indicates that the LU factorization may be inaccurate and
    the user should be wary of the results and perhaps increase
    the parameter PIVOT for subsequent runs (see Section 7).

12: ETA -- DOUBLE PRECISION                                 Input
    On entry: the relative pivot threshold below which an error
    diagnostic is provoked and IFAIL is set to 7. If ETA is
    greater than 1.0, then no check on pivot size is made.
                                -4
    Suggested value: ETA = 10  .

13: RPMIN -- DOUBLE PRECISION                              Output
    On exit: if ETA is less than 1.0, then RPMIN gives the
    smallest ratio of the pivot to the largest element in the
    row of the corresponding upper triangular factor thus
    monitoring the stability of the factorization. If RPMIN is
    very small it may be advisable to perform a new
    factorization using F01BRF.
```

```
14:  ABORT -- LOGICAL                                        Input
     On entry: if ABORT = .TRUE., the routine exits immediately
     (with IFAIL = 8) if it finds duplicate elements in the input
     matrix. If ABORT = .FALSE., the routine proceeds using a
     value equal to the sum of the duplicate elements. In either
     case details of each duplicate element are output on the
     current advisory message unit (see X04ABF), unless
     suppressed by the value of IFAIL on entry. Suggested value:
     ABORT = .TRUE..

15:  IDISP(2) -- INTEGER array                               Input
     On entry: IDISP(1) and IDISP(2) must be unchanged since the
     previous call of F01BRF.

16:  IFAIL -- INTEGER                                 Input/Output
     For this routine, the normal use of IFAIL is extended to
     control the printing of error and warning messages as well
     as specifying hard or soft failure (see the Essential
     Introduction).

     Before entry, IFAIL must be set to a value with the decimal
     expansion cba, where each of the decimal digits c, b and a
     must have a value of 0 or 1.
     a=0   specifies hard failure, otherwise soft failure;

     b=0   suppresses error messages, otherwise error messages
           will be printed (see Section 6);

     c=0   suppresses warning messages, otherwise warning
           messages will be printed (see Section 6).
     The recommended value for inexperienced users is 110 (i.e.,
     hard failure with all messages printed).

     Unless the routine detects an error (see Section 6), IFAIL
     contains 0 on exit.

6. Error Indicators and Warnings

Errors detected by the routine:

For each error, an explanatory error message is output on the
current error message unit (as defined by X04AAF), unless
suppressed by the value of IFAIL on entry.

IFAIL= 1
     On entry N <= 0.

IFAIL= 2
     On entry NZ <= 0.
```

IFAIL= 3
On entry LICN < NZ.

IFAIL= 4
On entry an element of the input matrix has a row or column index (i.e., an element of IVECT or JVECT) outside the range 1 to N.

IFAIL= 5
The input matrix is incompatible with the matrix factorized by the previous call of F01BRF (see Section 8).

IFAIL= 6
The input matrix is numerically singular.

IFAIL= 7
A very small pivot has been detected (see Section 5, ETA). The factorization has been completed but is potentially unstable.

IFAIL= 8
Duplicate elements have been found in the input matrix and the factorization has been abandoned (ABORT = .TRUE. on entry).

7. Accuracy

The factorization obtained is exact for a perturbed matrix whose (i,j)th element differs from a_{ij} by less than 3(epsilon)(rho)m_{ij} where (epsilon) is the machine precision, (rho) is the growth value returned in W(1) if GROW = .TRUE., and m_{ij} the number of Gaussian elimination operations applied to element (i,j).

If (rho) = W(1) is very large or RPMIN is very small, then a fresh call of F01BRF is recommended.

8. Further Comments

If the user has a sequence of problems with the same sparsity pattern then this routine is recommended after F01BRF has been called for one such problem. It is typically 4 to 7 times faster but is potentially unstable since the previous pivotal sequence is used. Further details on timing are given in document F01BRF.

If growth estimation is performed (GROW = .TRUE.), then the time increases by between 5% and 10%. Pivot size monitoring (ETA <= 1.0) involves a similar overhead.

```
          We normally expect this routine to be entered with a matrix
          having the same pattern of non-zeros as was earlier presented to
          F01BRF. However there is no record of this pattern, but rather a
          record of the pattern including all fill-ins. Therefore we permit
          additional non-zeros in positions corresponding to fill-ins.

          If singular matrices are being treated then it is also required
          that the present matrix be sufficiently like the previous one for
          the same permutations to be suitable for factorization with the
          same set of zero pivots.

          9. Example

          To factorize the real sparse matrices

                                 ( 5  0  0  0 0  0)
                                 ( 0  2 -1  2 0  0)
                                 ( 0  0  3  0 0  0)
                                 (-2  0  0  1 1  0)
                                 (-1  0  0 -1 2 -3)
                                 (-1 -1  0  0 0  6)

          and

                                (10  0  0  0  0  0)
                                ( 0 12 -3 -1  0  0)
                                ( 0  0 15  0  0  0)
                                (-2  0  0 10 -1  0).
                                (-1  0  0 -5  1 -1)
                                (-1 -2  0  0  0  6)

          This example program simply prints the values of W(1) and RPMIN
          returned by F01BSF. Normally the calls of F01BRF and F01BSF would
          be followed by calls of F04AXF.

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Incomplete Cholesky factorization

— nagf.ht —

```
\begin{page}{manpageXXf01maf}{NAG Documentation: f01maf}
\beginscroll
\begin{verbatim}



     F01MAF(3NAG)      Foundation Library (12/10/92)      F01MAF(3NAG)



          F01 -- Matrix Factorizations                               F01MAF
                  F01MAF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          F01MAF computes an incomplete Cholesky factorization of a real
          sparse symmetric positive-definite matrix A.

          2. Specification

                 SUBROUTINE F01MAF (N, NZ, A, LICN, IRN, LIRN, ICN, DROPTL,
                1                   DENSW, WKEEP, IKEEP, IWORK, ABORT,
                2                   INFORM, IFAIL)
                 INTEGER          N, NZ, LICN, IRN(LIRN), LIRN, ICN(LICN),
                1                 IKEEP(2*N), IWORK(6*N), INFORM(4), IFAIL
                 DOUBLE PRECISION A(LICN), DROPTL, DENSW, WKEEP(3*N)
                 LOGICAL          ABORT(3)

          3. Description

          F01MAF computes an incomplete Cholesky factorization

                                        T T
                                  C=PLDL P ,   WAW=C+E

          for the sparse symmetric positive-definite matrix A, where P is a
          permutation matrix, L is a unit lower triangular matrix, D is a
          diagonal matrix with positive diagonal elements, E is an error
          matrix representing elements dropped during the factorization and
          diagonal elements that have been modified to ensure that C is
```

positive-definite, and W is a diagonal matrix, chosen to make the diagonal elements of WAW unity.

$W^{-1}CW^{-1}$ is a pre-conditioning matrix for A, and the factorization of C is intended to be used by F04MAF to solve systems of linear equations Ax=b.

The permutation matrix P is chosen to reduce the amount of fill-in that occurs in L and the user-supplied parameter DROPTL can also be used to control the amount of fill-in that occurs.

Full details on the factorization can be found in Munksgaard [1].

F01MAF is based on the Harwell Library routine MA31A.

4. References

[1] Munksgaard N (1980) Solving Sparse Symmetric Sets of Linear Equations by Pre-conditioned Conjugate Gradients. ACM Trans. Math. Softw. 6 206--219.

5. Parameters

1: N -- INTEGER Input

On entry: n, the order of the matrix A. Constraint: N >= 1.

2: NZ -- INTEGER Input

On entry: the number of non-zero elements in the upper triangular part of the matrix A, including the number of elements on the leading diagonal. Constraint: NZ >= N.

3: A(LICN) -- DOUBLE PRECISION array Input/Output

On entry: the first NZ elements of the array A must contain the non-zero elements of the upper triangular part of the sparse positive-definite symmetric matrix A, including the elements on the leading diagonal. On exit: the first (NZ-N) elements of A contain the elements above the diagonal of the matrix WAW, where W is a diagonal matrix whose ith diagonal element is $w_i=a_{ii}^{-1/2}$. These elements are returned in order by rows and the value returned in ICN(k) gives the column index of the element returned in A(k). The value w_i is returned in the ith element of the array WKEEP. The remaining LROW-NZ+N elements of A, where LROW is the value returned in INFORM(1), return details of the factorization for use by F04MAF.

4: LICN -- INTEGER Input

On entry:
the dimension of the array A as declared in the (sub)program from which F01MAF is called.
If fill-in is expected during the factorization, then a larger value of LICN will allow fewer elements to be dropped during the factorization, thus giving a more accurate factorization, which in turn will almost certainly mean that fewer iterations will be required by F04MAF. Constraint: LICN>=2*NZ.

5: IRN(LIRN) -- INTEGER array Input/Output
On entry: IRN(k), for k = 1,2,...,NZ must contain the row index of the non-zero element of the matrix A supplied in A (k). On exit: the first LCOL elements of IRN, where LCOL is the value returned in INFORM(2), return details of the factorization for use by F04MAF.

6: LIRN -- INTEGER Input
On entry:
the dimension of the array IRN as declared in the (sub)program from which F01MAF is called.
LIRN must be at least NZ, but, as with LICN, if fill-in is expected then a larger value of LIRN will allow a more accurate factorization. For this purpose LIRN should exceed NZ by the same amount that LICN exceeds 2*NZ. Constraint: LIRN >= NZ.

7: ICN(LICN) -- INTEGER array Input/Output
On entry: ICN(k), for k = 1,2,...,NZ must contain the column index of the non-zero element of the matrix A supplied in A (k). Thus a_{ij}=A(k), where i = IRN(k) and j = ICN(k). On exit: the first (NZ-N) elements of ICN give the column indices of the first (NZ-N) elements returned in A. The remaining LROW - NZ + N elements of ICN return details of the factorization for use by F04MAF.

8: DROPTL -- DOUBLE PRECISION Input/Output
On entry: a value in the range [-1.0,1.0] to be used as a tolerance in deciding whether or not to drop elements during the factorization. At the kth pivot step the element $a_{ij}^{(k+1)}$ is dropped if it would cause fill-in and if

$$|a_{ij}^{(k+1)}|<|DROPTL|*\sqrt{a_{ii}^{(k)}a_{jj}^{(k)}}.$$

If DROPTL is supplied as negative, then it is not altered during the factorization and so is unchanged on exit, but if DROPTL is supplied as positive then it may be altered by the routine with the aim of obtaining an accurate factorization in the space available. If DROPTL is supplied as -1.0, then no fill-in will occur during the factorization; and if DROPTL is supplied as 0.0 then a complete factorization is performed. On exit: may be overwritten with the value used by the routine in order to obtain an accurate factorization in the space available, if DROPTL > 0.0 on entry.

9: DENSW -- DOUBLE PRECISION Input/Output
On entry: a value in the range [0.0,1.0] to be used in deciding whether or not to regard the active part of the matrix at the kth pivot step as being full. If the ratio of non-zero elements to the total number of elements is greater than or equal to DENSW, then the active part is regarded as full. If DENSW < 1.0, then the storage used is likely to increase compared to the case where DENSW = 0, but the execution time is likely to decrease. Suggested value: DENSW = 0.8. On exit: if on entry DENSW is not in the range [0.0,1.0], then it is set to 0.8. Otherwise it is unchanged.

10: WKEEP(3*N) -- DOUBLE PRECISION array Output
On exit: information which must be passed unchanged to F04MAF. The first N elements contain the values w_i, for $i=1,2,...,n$, and the next N elements contain the diagonal elements of D.

11: IKEEP(2*N) -- INTEGER array Output
On exit: information which must be passed unchanged to F04MAF.

12: IWORK(6*N) -- INTEGER array Workspace

13: ABORT(3) -- LOGICAL array Input
On entry:

if ABORT(1) = .TRUE., the routine will exit immediately on detecting duplicate elements and return IFAIL = 5. Otherwise when ABORT(1) = .FALSE., the calculations will continue using the sum of the duplicate entries. In either case details of the duplicate elements are output on the current advisory message unit (see X04ABF), unless suppressed by the value of IFAIL on entry.

If ABORT(2) = .TRUE., the routine will exit immediately on detecting a zero or negative pivot element and return IFAIL = 6. Otherwise when ABORT(2) = .FALSE., the zero or negative pivot element will be

```
          modified to ensure positive-definiteness and a message
          will be printed on the current advisory message unit,
          unless suppressed by the value of IFAIL on entry.

          If ABORT(3) = .TRUE., the routine will exit
          immediately if the arrays A and ICN have been filled
          up and return IFAIL = 7. Otherwise when ABORT(3) = .
          FALSE., the data in the arrays is compressed to
          release more storage and a message will be printed on
          the current advisory message unit, unless suppressed
          by the value of IFAIL on entry. If DROPTL is positive
          on entry, it may be modified in order to allow a
          factorization to be completed in the available space.
     Suggested values:
          ABORT(1) = .TRUE.,

          ABORT(2) = .TRUE.,

          ABORT(3) = .TRUE..

 14: INFORM(4) -- INTEGER array                           Output
     On exit:
          INFORM(1) returns the number of elements of A and ICN
          that have been used by the routine. Thus at least the
          first INFORM(1) elements of A and of ICN must be
          supplied to F04MAF.

          Similarly, INFORM(2) returns the number of elements of
          IRN that have been used by the routine and so at least
          the first INFORM(2) elements must be supplied to
          F04MAF.

          INFORM(3) returns the number of entries supplied in A
          that corresponded to diagonal and duplicate elements.
          If no duplicate entries were found, then INFORM(3)
          will return the value of N.

          INFORM(4) returns the value k of the pivot step from
          which the active matrix was regarded as full.
     INFORM must be passed unchanged to F04MAF.

 15: IFAIL -- INTEGER                               Input/Output
     For this routine, the normal use of IFAIL is extended to
     control the printing of error and warning messages as well
     as specifying hard or soft failure (see the Essential
     Introduction).

     Before entry, IFAIL must be set to a value with the decimal
     expansion cba, where each of the decimal digits c, b and a
     must have a value of 0 or 1.
```

```
     a=0   specifies hard failure, otherwise soft failure;

     b=0   suppresses error messages, otherwise error messages
           will be printed (see Section 6);

     c=0   suppresses warning messages, otherwise warning
           messages will be printed (see Section 6).
     The recommended value for inexperienced users is 110 (i.e.,
     hard failure with all messages printed).

     Unless the routine detects an error (see Section 6), IFAIL
     contains 0 on exit.

6. Error Indicators and Warnings

Errors detected by the routine:

For each error, an explanatory error message is output on the
current error message unit (as defined by X04AAF), unless
suppressed by the value of IFAIL on entry.

IFAIL= 1
     On entry N < 1,

     or       NZ < N,

     or       LIRN < NZ,

     or       LICN<2*NZ.

IFAIL= 2
     One of the conditions 0 < IRN(k) <= ICN(k) <= N is not
     satisfied so that A(k) is not in the upper triangle of the
     matrix. No further computation is attempted.

IFAIL= 3
     One of the diagonal elements of the matrix A is zero or
     negative so that A is not positive-definite. No further
     computation is attempted.

IFAIL= 4
     The available space has been used and no further
     compressions are possible. The user should either increase
     DROPTL, or allocate more space to A, IRN and ICN.

     For all the remaining values of IFAIL the computations will
     continue in the case of soft failure, so that more than one
     advisory message may be printed.

IFAIL= 5
```

Duplicate elements have been detected and ABORT(1) = .TRUE..

IFAIL= 6
A zero or negative pivot element has been detected during the factorization and ABORT(2) = .TRUE..

This should not happen if A is an M-matrix (see Munksgaard [1]), but may occur for other types of positive-definite matrix.

IFAIL= 7
The available space has been used and ABORT(3) = .TRUE..

7. Accuracy

The accuracy of the factorization will be determined by the size of the elements that are dropped and the size of the modifications made to the diagonal elements. If these sizes are small then the computed factors will correspond to a matrix close to A and the number of iterations required by F04MAF will be small. The more incomplete the factorization, the higher the number of iterations required by F04MAF.

8. Further Comments

The time taken by the routine will depend upon the sparsity pattern of the matrix and the number of fill-ins that occur during the factorization. At the very least the time taken can be expected to be roughly proportional to n(tau), where (tau) is the number of non-zeros.
The routine is intended for use with positive-definite matrices, but the user is warned that it will not necessarily detect non-positive-definiteness. Indeed the routine may return a factorization that can satisfactorily be used by F04MAF even when A is not positive-definite, but this should not be relied upon as F04MAF may not converge.

9. Example

The example program illustrates the use of F01MAF in conjunction with F04MAF to solve the 16 linear equations Ax=b, where

```
(1 z     z                      )
(z 1 z     z                    )
(  z 1 z     z                  )
(    z 1 0     z                )
(z     0 1 z     z              )
(  z     z 1 z     z            )
(    z     z 1 z     z          )
(      z     z 1 0     z        )
```

```
                A=(        z     0 1 z     z      ).
                  (          z     z 1 z     z    )
                  (            z     z 1 z     z  )
                  (              z     z 1 0     z)
                  (                z     0 1 z    )
                  (                  z     z 1 z  )
                  (                    z     z 1 z)
                  (                      z     z 1)

 T ( 1  1  1  1  1      1  1      1  1  1  1  1)
b =( -  -  -  -  - 0 0  -  - 0 0  -  -  -  -  -),
   ( 2  4  4  2  4      4  4      4  2  4  4  2)

          1
where z=- -.
          4

The n by n matrix A arises in the solution of Laplace's equation
in a unit square, using a 5-point formula with a 6 by 6
discretisation, with unity on the boundaries.

The drop tolerance, DROPTL, is taken as 0.1, and the density
factor, DENSW, is taken as 0.8. The value IFAIL = 111 is used so
that advisory and error messages will be printed, but soft
failure would occur if IFAIL were returned as non-zero.

A relative accuracy of about 0.0001 is requested in the solution
from F04MAF, with a maximum of 50 iterations.

The example program for F02FJF illustrates the use of F01MAF and
F04MAF in solving an eigenvalue problem.

The example program is not reproduced here. The source code for
all example programs is distributed with the NAG Foundation
Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Cholesky factor of a symmetric positive-definite matrix

— nagf.ht —

```
\begin{page}{manpageXXf01mcf}{NAG Documentation: f01mcf}
```

```
\beginscroll
\begin{verbatim}


     F01MCF(3NAG)      Foundation Library (12/10/92)      F01MCF(3NAG)



          F01 -- Matrix Factorizations                              F01MCF
                  F01MCF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          F01MCF computes the Cholesky factorization of a real symmetric
          positive-definite variable-bandwidth matrix.

          2. Specification

                 SUBROUTINE F01MCF (N, A, LAL, NROW, AL, D, IFAIL)
                 INTEGER          N, LAL, NROW(N), IFAIL
                 DOUBLE PRECISION A(LAL), AL(LAL), D(N)

          3. Description

          This routine determines the unit lower triangular matrix L and
                                                                     T
          the diagonal matrix D in the Cholesky factorization A=LDL  of a
          symmetric positive-definite variable-bandwidth matrix A of order
          n. (Such a matrix is sometimes called a 'sky-line' matrix.)

          The matrix A is represented by the elements lying within the
          envelope of its lower triangular part, that is, between the first
          non-zero of each row and the diagonal (see Section 9 for an
          example). The width NROW(i) of the ith row is the number of
          elements between the first non-zero element and the element on
          the diagonal, inclusive. Although, of course, any matrix
          possesses an envelope as defined, this routine is primarily
          intended for the factorization of symmetric positive-definite
          matrices with an average bandwidth which is small compared with n
          (also see Section 8).

          The method is based on the property that during Cholesky
          factorization there is no fill-in outside the envelope.
```

The determination of L and D is normally the first of two steps in the solution of the system of equations Ax=b. The remaining step, viz. the solution of $LDL^T x=b$ may be carried out using F04MCF.

4. References

[1] Jennings A (1966) A Compact Storage Scheme for the Solution of Symmetric Linear Simultaneous Equations. Comput. J. 9 281--285.

[2] Wilkinson J H and Reinsch C (1971) Handbook for Automatic Computation II, Linear Algebra. Springer-Verlag.

5. Parameters

1: N -- INTEGER Input
On entry: n, the order of the matrix A. Constraint: N >= 1.

2: A(LAL) -- DOUBLE PRECISION array Input
On entry: the elements within the envelope of the lower triangle of the positive-definite symmetric matrix A, taken in row by row order. The following code assigns the matrix elements within the envelope to the correct elements of the array:

```
K = 0
DO 20 I = 1, N
DO 10 J = I-NROW(I)+1, I
K = K + 1
A(K) = matrix (I,J)
10    CONTINUE
20 CONTINUE
```

See also Section 8.

3: LAL -- INTEGER Input
On entry: the smaller of the dimensions of the arrays A and AL as declared in the calling (sub)program from which F01MCF is called. Constraint: LAL >= NROW(1) + NROW(2) +... + NROW(n).

4: NROW(N) -- INTEGER array Input
On entry: NROW(i) must contain the width of row i of the matrix A, i.e., the number of elements between the first (leftmost) non-zero element and the element on the diagonal, inclusive. Constraint: 1 <= NROW(i) <= i, for i=1,2,...,n.

5: AL(LAL) -- DOUBLE PRECISION array Output

On exit: the elements within the envelope of the lower triangular matrix L, taken in row by row order. The envelope of L is identical to that of the lower triangle of A. The unit diagonal elements of L are stored explicitly. See also Section 8.

6: D(N) -- DOUBLE PRECISION array Output

On exit: the diagonal elements of the the diagonal matrix D . Note that the determinant of A is equal to the product of these diagonal elements. If the value of the determinant is required it should not be determined by forming the product explicitly, because of the possibility of overflow or underflow. The logarithm of the determinant may safely be formed from the sum of the logarithms of the diagonal elements.

7: IFAIL -- INTEGER Input/Output

On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1

On entry N < 1,

or for some i, NROW(i) < 1 or NROW(i) > i,

or LAL < NROW(1) + NROW(2) +... + NROW(N).

IFAIL= 2

A is not positive-definite, or this property has been destroyed by rounding errors. The factorization has not been completed.

IFAIL= 3

A is not positive-definite, or this property has been destroyed by rounding errors. The factorization has been completed but may be very inaccurate (see Section 7).

7. Accuracy

If IFAIL = 0 on exit, then the computed L and D satisfy the relation $LDL^T=A+F$, where

$$\|F\|_2 \le k m^2 (\epsilon) \max_i a_{ii}$$

and

$$\|F\|_2 \le k m^2 (\epsilon) \|A\|_2 ,$$

where k is a constant of order unity, m is the largest value of NROW(i), and (epsilon) is the machine precision. See Wilkinson and Reinsch [2], pp 25--27, 54--55. If IFAIL = 3 on exit, then the factorization has been completed although the matrix was not positive-definite. However the factorization may be very inaccurate and should be used only with great caution. For instance, if it is used to solve a set of equations Ax=b using F04MCF, the residual vector b-Ax should be checked.

8. Further Comments

The time taken by the routine is approximately proportional to the sum of squares of the values of NROW(i).

The distribution of row widths may be very non-uniform without undue loss of efficiency. Moreover, the routine has been designed to be as competitive as possible in speed with routines designed for full or uniformly banded matrices, when applied to such matrices.

Unless otherwise stated in the Users' Note for your implementation, the routine may be called with the same actual array supplied for parameters A and AL, in which case L overwrites the lower triangle of A. However this is not standard Fortran 77 and may not work in all implementations.

9. Example

To obtain the Cholesky factorization of the symmetric matrix, whose lower triangle is:

$$\begin{pmatrix} 1 & & & & & \\ 2 & 5 & & & & \\ 0 & 3 & 13 & & & \\ 0 & 0 & 0 & 16 & & \\ 5 & 14 & 18 & 8 & 55 & \\ 0 & 0 & 0 & 24 & 17 & 77 \end{pmatrix}.$$

For this matrix, the elements of NROW must be set to 1, 2, 2, 1,

```
         5, 3, and the elements within the envelope must be supplied in
         row order as:

         1, 2, 5, 3, 13, 16, 5, 14, 18, 8, 55, 24, 17, 77.


         The example program is not reproduced here. The source code for
         all example programs is distributed with the NAG Foundation
         Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

QR factorization of the real m by n matrix A

— nagf.ht —

```
\begin{page}{manpageXXf01qcf}{NAG Documentation: f01qcf}
\beginscroll
\begin{verbatim}



     F01QCF(3NAG)      Foundation Library (12/10/92)      F01QCF(3NAG)



         F01 -- Matrix Factorizations                               F01QCF
                 F01QCF -- NAG Foundation Library Routine Document

         Note: Before using this routine, please read the Users' Note for
         your implementation to check implementation-dependent details.
         The symbol (*) after a NAG routine name denotes a routine that is
         not included in the Foundation Library.

         1. Purpose

         F01QCF finds the QR factorization of the real m by n matrix A,
         where m>=n.

         2. Specification

                SUBROUTINE F01QCF (M, N, A, LDA, ZETA, IFAIL)
                INTEGER          M, N, LDA, IFAIL
```

```
      DOUBLE PRECISION A(LDA,*), ZETA(*)
```

3. Description

The m by n matrix A is factorized as

$$A=Q\begin{pmatrix}R\\0\end{pmatrix} \quad \text{when } m>n,$$

$$A=QR \quad \text{when } m=n,$$

where Q is an m by m orthogonal matrix and R is an n by n upper triangular matrix. The factorization is obtained by Householder's method. The kth transformation matrix, Q_k, which is used to introduce zeros into the kth column of A is given in the form

$$Q_k=\begin{pmatrix}I & 0\\0 & T_k\end{pmatrix}$$

where

$$T_k=I-u_k u_k^T,$$

$$u_k=\begin{pmatrix}(zeta)_k\\z_k\end{pmatrix},$$

$(zeta)_k$ is a scalar and z_k is an (m-k) element vector. $(zeta)_k$ and z_k are chosen to annihilate the elements below the triangular part of A.

The vector u_k is returned in the kth element of the array ZETA and in the kth column of A, such that $(zeta)_k$ is in ZETA(k) and the elements of z_k are in A(k+1,k),...,A(m,k). The elements of R are returned in the upper triangular part of A.

Q is given by

$$Q=(Q_n Q_{n-1}...Q_1)^T.$$

```
                     n n-1    1

Good background descriptions to the QR factorization are given in
Dongarra et al [1] and Golub and Van Loan [2], but note that this
routine is not based upon LINPACK routine DQRDC.

4. References
```


```
[1]   Dongarra J J, Moler C B, Bunch J R and Stewart G W (1979)
      LINPACK Users' Guide. SIAM, Philadelphia.

[2]   Golub G H and Van Loan C F (1989) Matrix Computations (2nd
      Edition). Johns Hopkins University Press, Baltimore,
      Maryland.

[3]   Wilkinson J H (1965) The Algebraic Eigenvalue Problem.
      Oxford University Press.
```


```
5. Parameters

 1:  M -- INTEGER                                          Input
     On entry: m, the number of rows of A. Constraint: M >= N.

 2:  N -- INTEGER                                          Input
     On entry: n, the number of columns of A.

     When N = 0 then an immediate return is effected.
     Constraint: N >= 0.

 3:  A(LDA,*) -- DOUBLE PRECISION array             Input/Output
     Note: the second dimension of the array A must be at least
     max(1,n).
     On entry: the leading m by n part of the array A must
     contain the matrix to be factorized. On exit: the n by n
     upper triangular part of A will contain the upper triangular
     matrix R and the m by n strictly lower triangular part of A
     will contain details of the factorization as described in
     Section 3.

 4:  LDA -- INTEGER                                        Input
     On entry:
     the first dimension of the array A as declared in the
     (sub)program from which F01QCF is called.
     Constraint: LDA >= max(1,M).

 5:  ZETA(*) -- DOUBLE PRECISION array                    Output
     Note: the dimension of the array ZETA must be at least max
     (1,n) On exit: ZETA(k) contains the scalar (zeta)  for the k
                                                     k
     th transformation. If T =I then ZETA(k)=0.0, otherwise ZETA(
```

k) contains $(\zeta)^k_k$ as described in Section 3 and $(\zeta)_k$ is always in the range $(1.0, \sqrt{2.0})$.

6: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL=-1
On entry M < N,

or N < 0,

or LDA < M.

7. Accuracy

The computed factors Q and R satisfy the relation

$$Q\begin{pmatrix}R\\0\end{pmatrix}=A+E,$$

where

$$\|E\|\leq c(\epsilon)\|A\|,$$

and (ϵ) is the machine precision (see X02AJF(*)), c is a modest function of m and n and $\|.\|$ denotes the spectral (two) norm.

8. Further Comments

The approximate number of floating-point operations is given by $2n^2(3m-n)/3$.

Following the use of this routine the operations

$$B:=QB \quad \text{and} \quad B:=Q^T B,$$

where B is an m by k matrix, can be performed by calls to F01QDF. The operation B:=QB can be obtained by the call:

```
      IFAIL = 0
      CALL F01QDF('No transpose', 'Separate', M, N, A, LDA, ZETA,
     *            K, B, LDB, WORK, IFAIL)
```

and $B:=Q^T B$ can be obtained by the call:

```
      IFAIL = 0
      CALL F01QDF('Transpose', 'Separate', M, N, A, LDA, ZETA,
     *            K, B, LDB, WORK, IFAIL)
```

In both cases WORK must be a k element array that is used as workspace. If B is a one-dimensional array (single column) then the parameter LDB can be replaced by M. See F01QDF for further details.

The first k columns of the orthogonal matrix Q can either be obtained by setting B to the first k columns of the unit matrix and using the first of the above two calls, or by calling F01QEF, which overwrites the k columns of Q on the first k columns of the array A. Q is obtained by the call:

```
     CALL F01QEF('Separate', M, N, K, A, LDA, ZETA, WORK, IFAIL)
```

As above WORK must be a k element array. If k is larger than N, then A must have been declared to have at least k columns.

Operations involving the matrix R can readily be performed by the Level 2 BLAS routines DTRSV and DTRMV (see Chapter F06), but note that no test for near singularity of R is incorporated in DTRSV. If R is singular, or nearly singular then F02WUF(*) can be used to determine the singular value decomposition of R.

9. Example

To obtain the QR factorization of the 5 by 3 matrix

```
                      (2.0  2.5  2.5)
                      (2.0  2.5  2.5)
                    A=(1.6 -0.4  2.8).
```

```
                          (2.0 -0.5  0.5)
                          (1.2 -0.3 -2.9)

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

——–

$B := QB$ or $B := Q^T B$

— nagf.ht —

```
\begin{page}{manpageXXf01qdf}{NAG Documentation: f01qdf}
\beginscroll
\begin{verbatim}



     F01QDF(3NAG)      Foundation Library (12/10/92)      F01QDF(3NAG)



          F01 -- Matrix Factorizations                              F01QDF
                  F01QDF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          F01QDF performs one of the transformations

                                                T
                                 B:=QB or B:=Q B,

          where B is an m by ncolb real matrix and Q is an m by m
          orthogonal matrix, given as the product of Householder
          transformation matrices.

          This routine is intended for use following F01QCF or F01QFF(*).
```

2. Specification

```
      SUBROUTINE F01QDF (TRANS, WHERET, M, N, A, LDA, ZETA,
     1                   NCOLB, B, LDB, WORK, IFAIL)
      INTEGER          M, N, LDA, NCOLB, LDB, IFAIL
      DOUBLE PRECISION A(LDA,*), ZETA(*), B(LDB,*), WORK(*)
      CHARACTER*1      TRANS, WHERET
```

3. Description

Q is assumed to be given by

$$Q=(Q_n Q_{n-1} ...Q_1)^T ,$$

Q_k being given in the form

$$Q_k = \begin{pmatrix} I & 0 \\ 0 & T_k \end{pmatrix}$$

where

$$T_k =I-u_k u_k^T ,$$

$$u_k = \begin{pmatrix} (zeta)_k \\ z_k \end{pmatrix},$$

$(zeta)_k$ is a scalar and z_k is an (m-k) element vector. z_k must be supplied in the kth column of A in elements A(k+1,k),...,A(m,k) and $(zeta)_k$ must be supplied either in A(k,k) or in ZETA(k), depending upon the parameter WHERET.

To obtain Q explicitly B may be set to I and pre-multiplied by Q. This is more efficient than obtaining Q^T.

4. References

[1] Golub G H and Van Loan C F (1989) Matrix Computations (2nd Edition). Johns Hopkins University Press, Baltimore,

Maryland.

[2] Wilkinson J H (1965) The Algebraic Eigenvalue Problem. Oxford University Press.

5. Parameters

1: TRANS -- CHARACTER*1 Input

On entry: the operation to be performed as follows:

TRANS = 'N' (No transpose)

Perform the operation B:=QB.

TRANS = 'T' or 'C' (Transpose)

Perform the operation $B:=Q^T B$.

Constraint: TRANS must be one of 'N', 'T' or 'C'.

2: WHERET -- CHARACTER*1 Input

On entry: indicates where the elements of (zeta) are to be found as follows:

WHERET = 'I' (In A)

The elements of (zeta) are in A.

WHERET = 'S' (Separate)

The elements of (zeta) are separate from A, in ZETA.

Constraint: WHERET must be one of 'I' or 'S'.

3: M -- INTEGER Input

On entry: m, the number of rows of A. Constraint: M >= N.

4: N -- INTEGER Input

On entry: n, the number of columns of A.

When N = 0 then an immediate return is effected.

Constraint: N >= 0.

5: A(LDA,*) -- DOUBLE PRECISION array Input

Note: the second dimension of the array A must be at least max(1,N).

On entry: the leading m by n strictly lower triangular part of the array A must contain details of the matrix Q. In addition, when WHERET = 'I', then the diagonal elements of A must contain the elements of (zeta) as described under the argument ZETA below.

When WHERET = 'S', the diagonal elements of the array A are referenced, since they are used temporarily to store the $(zeta)_k$, but they contain their original values on return.

```
 6: LDA -- INTEGER                                          Input
    On entry:
    the first dimension of the array A as declared in the
    (sub)program from which F01QDF is called.
    Constraint: LDA >= max(1,M).

 7: ZETA(*) -- DOUBLE PRECISION array                       Input
    Note: when WHERET = 'S', the dimension of the array ZETA
    must be greater than or equal to max(1,N). On entry: if
    WHERET = 'S', the array ZETA must contain the elements of
    (zeta). If ZETA(k) = 0.0 then T  is assumed to be I
                                   k
    otherwise ZETA(k) is assumed to contain (zeta) .
                                                  k

    When WHERET = 'I', ZETA is not referenced.

 8: NCOLB -- INTEGER                                        Input
    On entry: ncolb, number of columns of B.

    When NCOLB = 0 then an immediate return is effected.
    Constraint: NCOLB >= 0.

 9: B(LDB,*) -- DOUBLE PRECISION array               Input/Output
    Note: the second dimension of the array B must be at least
    max(1,NCOLB).
    On entry: the leading m by ncolb part of the array B must
    contain the matrix to be transformed. On exit: B is
    overwritten by the transformed matrix.

10: LDB -- INTEGER                                          Input
    On entry:
    the first dimension of the array B as declared in the
    (sub)program from which F01QDF is called.
    Constraint: LDB >= max(1,M).

11: WORK(*) -- DOUBLE PRECISION array                   Workspace
    Note: the dimension of the array WORK must be at least
    max(1,NCOLB).

12: IFAIL -- INTEGER                                 Input/Output
    On entry: IFAIL must be set to 0, -1 or 1. For users not
    familiar with this parameter (described in the Essential
    Introduction) the recommended value is 0.

    On exit: IFAIL = 0 unless the routine detects an error (see
    Section 6).

6. Error Indicators and Warnings
```

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL=-1
On entry TRANS /= 'N', 'T' or 'C',

or WHERET /= 'I' or 'S',

or M < N,

or N < 0,

or LDA < M,

or NCOLB < 0,

or LDB < M.

7. Accuracy

Letting C denote the computed matrix $Q^T B$, C satisfies the relation

$$QC=B+E,$$

where

$$||E||<=c(\epsilon)||B||,$$

and (epsilon) the machine precision (see X02AJF(*)), c is a modest function of m and |||.||| denotes the spectral (two) norm. An equivalent result holds for the computed matrix QB. See also Section 7 of F01QCF.

8. Further Comments

The approximate number of floating-point operations is given by 2n(2m-n)ncolb.

9. Example

To obtain the matrix $Q^T B$ for the matrix B given by

```
   ( 1.1  0.00)
   ( 0.9  0.00)
B=( 0.6  1.32)
```

```
                         ( 0.0  1.10)
                         (-0.8 -0.26)

        following the QR factorization of the 5 by 3 matrix A given by

                       (2.0  2.5  2.5)
                       (2.0  2.5  2.5)
                     A=(1.6 -0.4  2.8).
                       (2.0 -0.5  0.5)
                       (1.2 -0.3 -2.9)

        The example program is not reproduced here. The source code for
        all example programs is distributed with the NAG Foundation
        Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

First ncolq columns of the real m by m orthogonal matrix

— nagf.ht —

```
\begin{page}{manpageXXf01qef}{NAG Documentation: f01qef}
\beginscroll
\begin{verbatim}



     F01QEF(3NAG)      Foundation Library (12/10/92)      F01QEF(3NAG)



          F01 -- Matrix Factorizations                            F01QEF
                  F01QEF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          F01QEF returns the first ncolq columns of the real m by m
          orthogonal matrix Q, where Q is given as the product of
```

Householder transformation matrices.

This routine is intended for use following F01QCF or F01QFF(*).

2. Specification

```
       SUBROUTINE F01QEF (WHERET, M, N, NCOLQ, A, LDA, ZETA,
      1                   WORK, IFAIL)
       INTEGER          M, N, NCOLQ, LDA, IFAIL
       DOUBLE PRECISION A(LDA,*), ZETA(*), WORK(*)
       CHARACTER*1      WHERET
```

3. Description

Q is assumed to be given by

$$Q=(Q_n Q_{n-1} ... Q_1)^T,$$

Q_k being given in the form

$$Q_k=\begin{pmatrix} I & 0 \\ 0 & T_k \end{pmatrix}$$

where

$$T_k=I-u_k u_k^T,$$

$$u_k=\begin{pmatrix} (zeta)_k \\ z_k \end{pmatrix},$$

$(zeta)_k$ is a scalar and z_k is an (m-k) element vector. z_k must be supplied in the kth column of A in elements A(k+1,k),...,A(m,k) and $(zeta)_k$ must be supplied either in A(k,k) or in ZETA(k), depending upon the parameter WHERET.

4. References

[1] Golub G H and Van Loan C F (1989) Matrix Computations (2nd Edition). Johns Hopkins University Press, Baltimore, Maryland.

[2] Wilkinson J H (1965) The Algebraic Eigenvalue Problem.
Oxford University Press.

5. Parameters

1: WHERET -- CHARACTER*1 Input
On entry: indicates where the elements of (zeta) are to be found as follows:
WHERET = 'I' (In A)
The elements of (zeta) are in A.

WHERET = 'S' (Separate)
The elements of (zeta) are separate from A, in ZETA.
Constraint: WHERET must be one of 'I' or 'S'.

2: M -- INTEGER Input
On entry: m, the number of rows of A. Constraint: M >= N.

3: N -- INTEGER Input
On entry: n, the number of columns of A. Constraint: N >= 0.

4: NCOLQ -- INTEGER Input
On entry: ncolq, the required number of columns of Q.
Constraint: 0 <= NCOLQ <= M.

When NCOLQ = 0 then an immediate return is effected.

5: A(LDA,*) -- DOUBLE PRECISION array Input/Output
Note: the second dimension of the array A must be at least max(1,N,NCOLQ).
On entry: the leading m by n strictly lower triangular part of the array A must contain details of the matrix Q. In addition, when WHERET = 'I', then the diagonal elements of A must contain the elements of (zeta) as described under the argument ZETA below. On exit: the first NCOLQ columns of the array A are overwritten by the first NCOLQ columns of the m by m orthogonal matrix Q. When N = 0 then the first NCOLQ columns of A are overwritten by the first NCOLQ columns of the identity matrix.

6: LDA -- INTEGER Input
On entry:
the first dimension of the array A as declared in the (sub)program from which F01QEF is called.
Constraint: LDA >= max(1,M).

7: ZETA(*) -- DOUBLE PRECISION array Input
Note: the dimension of the array ZETA must be at least

max(1,N).
On entry: with WHERET = 'S', the array ZETA must contain the elements of (zeta). If ZETA(k) = 0.0 then T_k is assumed to be I, otherwise ZETA(k) is assumed to contain $(zeta)_k$.

When WHERET = 'I', the array ZETA is not referenced.

8: WORK(*) -- DOUBLE PRECISION array Workspace
Note: the dimension of the array WORK must be at least max(1,NCOLQ).

9: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL=-1
On entry WHERET /= 'I' or 'S',

or M < N,

or N < 0,

or NCOLQ < 0 or NCOLQ > M,

or LDA < M.

7. Accuracy

The computed matrix Q satisfies the relation

Q=P+E,

where P is an exactly orthogonal matrix and

||E||<=c(epsilon)

(epsilon) is the machine precision (see X02AJF(*)), c is a modest

```
         function of m and |||.||| denotes the spectral (two) norm. See
         also Section 7 of F01QCF.

         8. Further Comments

         The approximate number of floating-point operations required is
         given by

              2
              -n{(3m-n)(2ncolq-n)-n(ncolq-n)},    ncolq>n,
              3

              2     2
              -ncolq (3m-ncolq),                  ncolq<=n.
              3

         9. Example

         To obtain the 5 by 5 orthogonal matrix Q following the QR
         factorization of the 5 by 3 matrix A given by

                                   (2.0  2.5  2.5)
                                   (2.0  2.5  2.5)
                                 A=(1.6 -0.4  2.8).
                                   (2.0 -0.5  0.5)
                                   (1.2 -0.3 -2.9)

         The example program is not reproduced here. The source code for
         all example programs is distributed with the NAG Foundation
         Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

QR factorization of the complex m by n matrix A

— nagf.ht —

```
\begin{page}{manpageXXf01rcf}{NAG Documentation: f01rcf}
\beginscroll
\begin{verbatim}
```

F01 -- Matrix Factorizations F01RCF

F01RCF -- NAG Foundation Library Routine Document

Note: Before using this routine, please read the Users' Note for your implementation to check implementation-dependent details. The symbol (*) after a NAG routine name denotes a routine that is not included in the Foundation Library.

1. Purpose

F01RCF finds the QR factorization of the complex m by n matrix A, where m>=n.

2. Specification

```
SUBROUTINE   F01RCF (M, N, A, LDA, THETA, IFAIL)
INTEGER              M, N, LDA, IFAIL
COMPLEX(KIND(1.0D0)) A(LDA,*), THETA(*)
```

3. Description

The m by n matrix A is factorized as

$$A=Q\begin{pmatrix}R\\0\end{pmatrix} \quad \text{when } m>n,$$

$$A=QR \quad \text{when } m=n,$$

where Q is an m by m unitary matrix and R is an n by n upper triangular matrix with real diagonal elements.

The factorization is obtained by Householder's method. The kth transformation matrix, Q_k, which is used to introduce zeros into the kth column of A is given in the form

$$Q_k=\begin{pmatrix}I & 0\\0 & T_k\end{pmatrix},$$

where

$$T_k=I-\gamma_k u_k u_k^H,$$

$$u_k = \begin{pmatrix} \zeta_k \\ z_k \end{pmatrix},$$

γ_k is a scalar for which $Re\ \gamma_k = 1.0$, ζ_k is a real scalar and z_k is an $(m-k)$ element vector. γ_k, ζ_k and z_k are chosen to annihilate the elements below the triangular part of A and to make the diagonal elements real.

The scalar γ_k and the vector u_k are returned in the kth element of the array THETA and in the kth column of A, such that θ_k, given by

$$\theta_k = (\zeta_k, Im(\gamma_k)),$$

is in THETA(k) and the elements of z_k are in $a_{k+1,k},\ldots,a_{m,k}$. The elements of R are returned in the upper triangular part of A.

Q is given by

$$Q = (Q_n Q_{n-1} \ldots Q_1)^H.$$

A good background description to the QR factorization is given in Dongarra et al [1], but note that this routine is not based upon LINPACK routine ZQRDC.

4. References

[1] Dongarra J J, Moler C B, Bunch J R and Stewart G W (1979) LINPACK Users' Guide. SIAM, Philadelphia.

[2] Wilkinson J H (1965) The Algebraic Eigenvalue Problem. Oxford University Press.

5. Parameters

1: M -- INTEGER Input
On entry: m, the number of rows of A. Constraint: M >= N.

2: N -- INTEGER Input

```
     On entry: n, the number of columns of A. Constraint: N >=
     0.

     When N = 0 then an immediate return is effected.

 3:  A(LDA,*) -- COMPLEX(KIND(1.0D0)) array          Input/Output
     Note: the second dimension of the array A must be at least
     max(1,N).
     On entry: the leading m by n part of the array A must
     contain the matrix to be factorized. On exit: the n by n
     upper triangular part of A will contain the upper triangular
     matrix R, with the imaginary parts of the diagonal elements
     set to zero, and the m by n strictly lower triangular part
     of A will contain details of the factorization as described
     above.

 4:  LDA -- INTEGER                                          Input
     On entry:
     the first dimension of the array A as declared in the
     (sub)program from which F01RCF is called.
     Constraint: LDA >= max(1,M).

 5:  THETA(*) -- COMPLEX(KIND(1.0D)) array                  Output
     Note: the dimension of the array THETA must be at least
     max(1,N).
     On exit: the scalar (theta)  for the kth transformation. If
                                k
     T =I then THETA(k) = 0.0; if
      k
                    ((alpha) 0)
                 T =(      0 I)   Re(alpha)<0.0,
                  k
     then THETA(k)=(alpha); otherwise THETA(k) contains THETA(k)
     as described in Section 3 and Re(THETA(k)) is always in the


     range (1.0, \/2.0).

 6:  IFAIL -- INTEGER                                 Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:
```

```
If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

IFAIL=-1
     On entry M < N,

     or       N < 0,

     or       LDA < M.

7. Accuracy
The computed factors Q and R satisfy the relation

                             (R)
                            Q(0)=A+E,

where

                   ||E||<=c(epsilon)||A||,

(epsilon) being the machine precision, c is a modest function of
m and n and ||.|| denotes the spectral (two) norm.

8. Further Comments

The approximate number of real floating-point operations is given
     2
by 8n (3m-n)/3.

Following the use of this routine the operations

                                       H
                       B:=QB  and  B:=Q B,

where B is an m by k matrix, can be performed by calls to F01RDF.
The operation B:=QB can be obtained by the call:


      IFAIL = 0
      CALL F01RDF('No conjugate', 'Separate', M, N, A, LDA, THETA,
     *            K, B, LDB, WORK, IFAIL)

        H
and B:=Q B can be obtained by the call:


      IFAIL = 0
      CALL F01RDF('Conjugate', 'Separate', M, N, A, LDA, THETA,
     *            K, B, LDB, WORK, IFAIL)
```

```
          In both cases WORK must be a k element array that is used as
          workspace. If B is a one-dimensional array (single column) then
          the parameter LDB can be replaced by M. See F01RDF for further
          details.

          The first k columns of the unitary matrix Q can either be
          obtained by setting B to the first k columns of the unit matrix
          and using the first of the above two calls, or by calling F01REF,
          which overwrites the k columns of Q on the first k columns of the
          array A. Q is obtained by the call:


                CALL F01REF('Separate', M, N, K, A, LDA, THETA, WORK, IFAIL)

          As above, WORK must be a k element array. If k is larger than n,
          then A must have been declared to have at least k columns.

          Operations involving the matrix R can readily be performed by the
          Level 2 BLAS routines ZTRSV and ZTRMV (see Chapter F06), but note
          that no test for near singularity of R is incorporated in ZTRSV.
          If R is singular, or nearly singular, then F02XUF(*) can be used
          to determine the singular value decomposition of R.

          9. Example

          To obtain the QR factorization of the 5 by 3 matrix

                             (  0.5i -0.5+1.5i -1.0+1.0i)
                             (0.4+0.3i 0.9+1.3i 0.2+1.4i)
                           A=(  0.4 -0.4+0.4i    1.8    ).
                             (0.3-0.4i 0.1+0.7i    0.0  )
                             ( -0.3i 0.3+0.3i   2.4i    )

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

———-

$B := QB$ **or** $B := Q^H B$

— nagf.ht —

```
\begin{page}{manpageXXf01rdf}{NAG Documentation: f01rdf}
\beginscroll
\begin{verbatim}



     F01RDF(3NAG)      Foundation Library (12/10/92)      F01RDF(3NAG)



          F01 -- Matrix Factorizations                              F01RDF
                  F01RDF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          F01RDF performs one of the transformations

                                             H
                                B:=QB or B:=Q B,

          where B is an m by ncolb complex matrix and Q is an m by m
          unitary matrix, given as the product of Householder
          transformation matrices.

          This routine is intended for use following F01RCF or F01RFF(*).

          2. Specification

                 SUBROUTINE F01RDF (TRANS, WHERET, M, N, A, LDA, THETA,
                1                   NCOLB, B, LDB, WORK, IFAIL)
                 INTEGER              M, N, LDA, NCOLB, LDB, IFAIL
                 COMPLEX(KIND(1.0D0)) A(LDA,*), THETA(*), B(LDB,*), WORK(*)
                 CHARACTER*1          TRANS, WHERET

          3. Description

          The unitary matrix Q is assumed to be given by

                                               H
                                Q=(Q Q   ...Q ) ,
                                    n n-1    1

          Q  being given in the form
           k
```

$$Q_k = \begin{pmatrix} I & 0 \\ 0 & T_k \end{pmatrix},$$

where

$$T_k = I - \gamma_k u_k u_k^H,$$

$$u_k = \begin{pmatrix} \zeta_k \\ z_k \end{pmatrix},$$

γ_k is a scalar for which $\mathrm{Re}\,\gamma_k = 1.0$, ζ_k is a real scalar and z_k is an (m-k) element vector.

z_k must be supplied in the kth column of A in elements $a_{k+1,k},\ldots,a_{m,k}$ and θ_k, given by

$$\theta_k = (\zeta_k, \mathrm{Im}\,\gamma_k),$$

must be supplied either in $a_{k,k}$ or in THETA(k), depending upon the parameter WHERET.

To obtain Q explicitly B may be set to I and pre-multiplied by Q. This is more efficient than obtaining Q^H. Alternatively, F01REF may be used to obtain Q overwritten on A.

4. References

[1] Wilkinson J H (1965) The Algebraic Eigenvalue Problem. Oxford University Press.

5. Parameters

1: TRANS -- CHARACTER*1 Input

On entry: the operation to be performed as follows:

TRANS = 'N' (No transpose)

Perform the operation B:=QB.

TRANS = 'C' (Conjugate transpose)

```
                                     H
      Perform the operation B:=Q B.
Constraint: TRANS must be one of 'N' or 'C'.

2:  WHERET -- CHARACTER*1                                    Input
    On entry: the elements of (theta) are to be found as
    follows:
    WHERET = 'I' (In A)
          The elements of (theta) are in A.

    WHERET = 'S' (Separate)
          The elements of (theta) are separate from A, in THETA.
    Constraint: WHERET must be one of 'I' or 'S'.

3:  M -- INTEGER                                             Input
    On entry: m, the number of rows of A. Constraint: M >= N.

4:  N -- INTEGER                                             Input
    On entry: n, the number of columns of A. Constraint: N >=
    0.

    When N = 0 then an immediate return is effected.

5:  A(LDA,*) -- COMPLEX(KIND(1.0D)) array                    Input
    Note: the second dimension of the array A must be at least
    max(1,N).
    On entry: the leading m by n strictly lower triangular part
    of the array A must contain details of the matrix Q. In
    addition, when WHERET = 'I', then the diagonal elements of A
    must contain the elements of (theta) as described under the
    argument THETA below.

    When WHERET = 'S', then the diagonal elements of the array A
    are referenced, since they are used temporarily to store the
    (zeta) , but they contain their original values on return.
          k

6:  LDA -- INTEGER                                           Input
    On entry:
    the first dimension of the array A as declared in the
    (sub)program from which F01RDF is called.
    Constraint: LDA >= max(1,M).

7:  THETA(*) -- COMPLEX(KIND(1.0D)) array                    Input
    Note: the dimension of the array THETA must be at least
    max(1,N).
    On entry: with WHERET = 'S', the array THETA must contain
    the elements of (theta). If THETA(k)=0.0 then T  is assumed
                                                   k
    to be I; if THETA(k)=(alpha), with Re(alpha)<0.0, then T  is
```

```
                                                              k
assumed to be of the form
                         ((alpha) 0)
                      T =(0       I);
                       k
otherwise THETA(k) is assumed to contain (theta)  given by
                                                k
(theta) =((zeta) ,Im(gamma) ).
       k       k          k

When WHERET = 'I', the array THETA is not referenced, and
may be dimensioned of length 1.
```

```
 8:  NCOLB -- INTEGER                                       Input
     On entry: ncolb, the number of columns of B. Constraint:
     NCOLB >= 0.
     When NCOLB = 0 then an immediate return is effected.

 9:  B(LDB,*) -- COMPLEX(KIND(1.0D)) array           Input/Output
     Note: the second dimension of the array B must be at least
     max(1,NCOLB).
     On entry: the leading m by ncolb part of the array B must
     contain the matrix to be transformed. On exit: B is
     overwritten by the transformed matrix.

10:  LDB -- INTEGER                                         Input
     On entry:
     the first dimension of the array B as declared in the
     (sub)program from which F01RDF is called.
     Constraint: LDB >= max(1,M).

11:  WORK(*) -- COMPLEX(KIND(1.0D)) array               Workspace
     Note: the dimension of the array WORK must be at least
     max(1,NCOLB).

12:  IFAIL -- INTEGER                                Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).
```

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

```
IFAIL=-1
     On entry TRANS /= 'N' or 'C',

     or       WHERET /= 'I' or 'S',

     or       M < N,

     or       N < 0,

     or       LDA < M,

     or       NCOLB < 0,

     or       LDB < M.
```

7. Accuracy

Letting C denote the computed matrix Q B, C satisfies the relation

```
                             QC=B+E,
```

where

```
                   ||E||<=c(epsilon)||B||,
```

(epsilon) being the machine precision, c is a modest function of m and |||.||| denotes the spectral (two) norm. An equivalent result holds for the computed matrix QB. See also Section 7 of F01RCF.

8. Further Comments

The approximate number of real floating-point operations is given by 8n(2m-n)ncolb.

9. Example

```
                      H
To obtain the matrix Q B for the matrix B given by

                     (-0.55+1.05i  0.45+1.05i)
                     ( 0.49+0.93i  1.09+0.13i)
                   B=( 0.56-0.16i  0.64+0.16i)
                     ( 0.39+0.23i -0.39-0.23i)
                     ( 1.13+0.83i -1.13+0.77i)

following the QR factorization of the 5 by 3 matrix A given by

                   (  0.5i   -0.5+1.5i -1.0+1.0i)
```

```
                         (0.4+0.3i  0.9+1.3i  0.2+1.4i)
                       A=(  0.4    -0.4+0.4i  1.8     ).
                         (0.3-0.4i  0.1+0.7i  0.0     )
                         ( -0.3i    0.3+0.3i      2.4i)

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

First ncolq columns of the complex m by m unitary matrix

— nagf.ht —

```
\begin{page}{manpageXXf01ref}{NAG Documentation: f01ref}
\beginscroll
\begin{verbatim}



     F01REF(3NAG)      Foundation Library (12/10/92)      F01REF(3NAG)



          F01 -- Matrix Factorizations                             F01REF
                  F01REF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          F01REF returns the first ncolq columns of the complex m by m
          unitary matrix Q, where Q is given as the product of Householder
          transformation matrices.

          This routine is intended for use following F01RCF.

          2. Specification
```

```
      SUBROUTINE F01REF (WHERET, M, N, NCOLQ, A, LDA, THETA,
     1                   WORK, IFAIL)
      INTEGER              M, N, NCOLQ, LDA, IFAIL
      COMPLEX(KIND(1.0D0)) A(LDA,*), THETA(*), WORK(*)
      CHARACTER*1          WHERET
```

3. Description

The unitary matrix Q is assumed to be given by

$$Q=(Q_n Q_{n-1} \ldots Q_1)^H,$$

Q_k being given in the form

$$Q_k=\begin{pmatrix} I & 0 \\ 0 & T_k \end{pmatrix},$$

where

$$T_k=I-(\gamma)_k u_k u_k^H,$$

$$u_k=\begin{pmatrix} (\zeta)_k \\ z_k \end{pmatrix},$$

$(\gamma)_k$ is a scalar for which $\mathrm{Re}\,(\gamma)_k=1.0$, $(\zeta)_k$ is a real scalar and z_k is an (m-k) element vector.

z_k must be supplied in the kth column of A in elements $a_{k+1,k},\ldots,a_{m,k}$ and $(\theta)_k$, given by

$$(\theta)_k=((\zeta)_k,\mathrm{Im}\,(\gamma)_k),$$

must be supplied either in $a_{k,k}$ or in THETA(k) depending upon the parameter WHERET.

4. References

[1] Wilkinson J H (1965) The Algebraic Eigenvalue Problem. Oxford University Press.

5. Parameters

1: WHERET -- CHARACTER*1 Input
On entry: the elements of (theta) are to be found as follows:
WHERET = 'I' (In A)
The elements of (theta) are in A.

WHERET = 'S' (Separate)
The elements of (theta) are separate from A, in THETA.
Constraint: WHERET must be one of 'I' or 'S'.

2: M -- INTEGER Input
On entry: m, the number of rows of A. Constraint: M >= N.

3: N -- INTEGER Input
On entry: n, the number of columns of A. Constraint: N >= 0.

4: NCOLQ -- INTEGER Input
On entry: ncolq, the required number of columns of Q.
Constraint: 0 <= NCOLQ <= M.

When NCOLQ = 0 then an immediate return is effected.

5: A(LDA,*) -- COMPLEX(KIND(1.0D)) array Input/Output
Note: the second dimension of the array A must be at least max(1,N,NCOLQ).
On entry: the leading m by n strictly lower triangular part of the array A must contain details of the matrix Q. In addition, when WHERET = 'I', then the diagonal elements of A must contain the elements of (theta) as described under the argument THETA below. On exit: the first NCOLQ columns of the array A are overwritten by the first NCOLQ columns of the m by m unitary matrix Q. When N = 0 then the first NCOLQ columns of A are overwritten by the first NCOLQ columns of the unit matrix.

6: LDA -- INTEGER Input
On entry:
the first dimension of the array A as declared in the (sub)program from which F01REF is called.
Constraint: LDA >= max(1,M).

7: THETA(*) -- COMPLEX(KIND(1.0D)) array Input

```
    Note: the dimension of the array THETA must be at least
    max(1,N).
    On entry: if WHERET = 'S', the array THETA must contain the
    elements of (theta). If THETA(k)=0.0 then T  is assumed to
                                               k
    be I; if THETA(k)=(alpha), with Re(alpha)<0.0, then T  is
                                                         k
    assumed to be of the form
                         ((alpha) 0)
                      T =(   0    I);
                       k
    otherwise THETA(k) is assumed to contain (theta)  given by
                                                    k
    (theta) =((zeta) ,Im(gamma) ).
           k        k          k

    When WHERET = 'I', the array THETA is not referenced.

 8: WORK(*) -- COMPLEX(KIND(1.0D)) array               Workspace
    Note: the dimension of the array WORK must be at least
    max(1,NCOLQ).

 9: IFAIL -- INTEGER                                Input/Output
    On entry: IFAIL must be set to 0, -1 or 1. For users not
    familiar with this parameter (described in the Essential
    Introduction) the recommended value is 0.

    On exit: IFAIL = 0 unless the routine detects an error (see
    Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

IFAIL=-1
    On entry WHERET /= 'I' or 'S',

    or       M < N,

    or       N < 0,

    or       NCOLQ < 0 or NCOLQ > M,

    or       LDA < M.

7. Accuracy
```

```
The computed matrix Q satisfies the relation

                             Q=P+E,

where P is an exactly unitary matrix and

                      ||E||<=c(epsilon),

(epsilon) being the machine precision, c is a modest function of
m and |||.||| denotes the spectral (two) norm. See also Section 7
of F01RCF.

8. Further Comments

The approximate number of real floating-point operations required
is given by

     8
     -n{(3m-n)(2ncolq-n)-n(ncolq-n)},  ncolq>n
     3

     8     2
     -ncolq (3m-ncolq),                ncolq<=n
     3

9. Example

To obtain the 5 by 5 unitary matrix Q following the QR
factorization of the 5 by 3 matrix A given by

                   (    0.5i -0.5+1.5i -1.0+1.4i)
                   (0.4+0.3i  0.9+1.3i  0.2+1.4i)
                 A=(0.4      -0.4+0.4i  1.8     ).
                   (0.3-0.4i  0.1+0.7i  0.0     )
                   (   -0.3i  0.3+0.3i      2.4i)

The example program is not reproduced here. The source code for
all example programs is distributed with the NAG Foundation
Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Eigenvalues and Eigenvectors

— nagf.ht —

```
\begin{page}{manpageXXf02}{NAG Documentation: f02}
\beginscroll
\begin{verbatim}



     F02(3NAG)        Foundation Library (12/10/92)        F02(3NAG)



          F02 -- Eigenvalues and Eigenvectors           Introduction -- F02
                                    Chapter F02
                           Eigenvalues and Eigenvectors

          1. Scope of the Chapter

          This chapter is concerned with computing

          --    eigenvalues and eigenvectors of a matrix

          --    eigenvalues and eigenvectors of generalized matrix
                eigenvalue problems

          --    singular values and singular vectors of a matrix.

          2. Background to the Problems

          2.1. Eigenvalue Problems

          In the most usual form of eigenvalue problem we are given a
          square n by n matrix A and wish to compute (lambda) (an
          eigenvalue) and x/=0 (an eigenvector) which satisfy the equation

                                    Ax=(lambda)x

          Such problems are called 'standard' eigenvalue problems in
          contrast to 'generalized' eigenvalue problems where we wish to
          satisfy the equation

                                    Ax=(lambda)Bx

          B also being a square n by n matrix.

          Section 2.1.1 and Section 2.1.2 discuss, respectively, standard
          and generalized eigenvalue problems where the matrices involved
```

are dense; Section 2.1.3 discusses both types of problem in the case where A and B are sparse (and symmetric).

2.1.1. Standard eigenvalue problems

Some of the routines in this chapter find all the n eigenvalues, some find all the n eigensolutions (eigenvalues and corresponding eigenvectors), and some find a selected group of eigenvalues and/or eigenvectors. The matrix A may be:

(i) general (real or complex)

(ii) real symmetric, or

(iii) complex Hermitian (so that if a_{ij}=(alpha)+i(beta) then a_{ji}=(alpha)-i(beta)).

In all cases the computation starts with a similarity transformation $S^{-1}AS=T$, where S is non-singular and is the product of fairly simple matrices, and T has an 'easier form' than A so that its eigensolutions are easily determined. The matrices A and T, of course, have the same eigenvalues, and if y is an eigenvector of T then Sy is the corresponding eigenvector of A.

In case (i) (general real or complex A), the selected form of T is an upper Hessenberg matrix (t_{ij}=0 if i-j>1) and S is the product of n-2 stabilised elementary transformation matrices. There is no easy method of computing selected eigenvalues of a Hessenberg matrix, so that all eigenvalues are always calculated. In the real case this computation is performed via the Francis QR algorithm with double shifts, and in the complex case by means of the LR algorithm. If the eigenvectors are required they are computed by back-substitution following the QR and LR algorithm.

In case (ii) (real and symmetric A) the selected simple form of T is a tridiagonal matrix (t_{ij}=0 if |i-j|>1), and S is the product of n-2 orthogonal Householder transformation matrices. If only selected eigenvalues are required, they are obtained by the method of bisection using the Sturm sequence property, and the corresponding eigenvectors of T are computed by inverse iteration. If all eigenvalues are required, they are computed from T via the QL algorithm (an adaptation of the QR algorithm), and the corresponding eigenvectors of T are the product of the transformations for the QL reduction. In all cases the corresponding eigenvectors of A are recovered from the computation of x=Sy.

In case (iii) (complex Hermitian A) analogous transformations as in case (ii) are used. T has complex elements in off-diagonal positions, but a simple diagonal similarity transformation is then used to produce a real tridiagonal form, after which the QL algorithm and succeeding methods described in the previous paragraph are used to complete the solution.

2.1.2. Generalized eigenvalue problems

Here we distinguish as a special case those problems in which both A and B are symmetric and B is positive-definite and well-conditioned with respect to inversion (i.e., all the eigenvalues of B are significantly greater than zero). Such problems can be satisfactorily treated by first reducing them to case (ii) of Section 2.1.1 and then using the methods described there to compute the eigensolutions. If B is factorized as LL^T (L lower triangular), then Ax=(lambda)Bx is equivalent to the standard symmetric problem Ry=(lambda)y, where $R=L^{-1}A(L^T)^{-1}$ and $y=L^T x$. After finding an eigenvector y of R, the required x is computed by back-substitution in $y=L^T x$.

For generalized problems of the form Ax=(lambda)Bx which do not fall into the special case, the QZ algorithm is provided.

In order to appreciate the domain in which this algorithm is appropriate we remark first that when B is non-singular the problem Ax=(lambda)Bx is fully equivalent to the problem $(B^{-1}A)x$=(lambda)x; both the eigenvalues and eigenvectors being the same. When A is non-singular Ax=(lambda)Bx is equivalent to the problem $(A^{-1}B)x$=(mu)x; the eigenvalues (mu) being the reciprocals of the required eigenvalues and the eigenvectors remaining the same. In theory then, provided at least one of the matrices A and B is non-singular, the generalized problem Ax=(lambda)Bx could be solved via the standard problem Cx=(lambda)x with an appropriate matrix C, and as far as economy of effort is concerned this is quite satisfactory. However, in practice, for this reduction to be satisfactory from the standpoint of numerical stability, one requires more than the mere non-singularity of A or B. It is necessary that $B^{-1}A$ (or $A^{-1}B$) should not only exist but that B (or A) should be well-conditioned with respect to inversion. The nearer B (or A) is to singularity the more unsatisfactory $B^{-1}A$ (or $A^{-1}B$) will be as a vehicle for determining the required eigenvalues. Unfortunately one cannot counter ill-conditioning in B (or A) by computing $B^{-1}A$ (or $A^{-1}B$) accurately to single precision using iterative refinement. Well-determined eigenvalues of the original Ax=(lambda)Bx may be poorly determined even by the correctly rounded version of $B^{-1}A$ (or $A^{-1}B$). The situation may in some instances be saved by the observation that if Ax=(lambda)Bx then (A-kB)x=((lambda)-k)Bx. Hence if A-kB is non-singular we may solve the standard problem $[(A-kB)^{-1}B]x=(mu)x$ and for numerical stability we require only that (A-kB) be well-conditioned with respect to inversion.

In practice one may well be in a situation where no k is known for which (A-kB) is well-conditioned with respect to inversion and indeed (A-kB) may be singular for all k. The QZ algorithm is designed to deal directly with the problem Ax=(lambda)Bx itself and its performance is unaffected by singularity or near-singularity of A, B or A-kB.

2.1.3. Sparse symmetric problems

If the matrices A and B are large and sparse (i.e., only a small proportion of the elements are non-zero), then the methods described in the previous Section are unsuitable, because in reducing the problem to a simpler form, much of the sparsity of the problem would be lost; hence the computing time and the storage required would be very large. Instead, for symmetric problems, the method of simultaneous iteration may be used to determine selected eigenvalues and the corresponding eigenvectors. The routine provided has been designed to handle both symmetric and generalized symmetric problems.

2.2. Singular Value Problems

The singular value decomposition of an m by n real matrix A is given by

$$A=QDP^T,$$

where Q is an m by m orthogonal matrix, P is an n by n orthogonal matrix and D is an m by n diagonal matrix with non-negative diagonal elements. The first k==min(m,n) columns of Q and P are the left- and right-hand singular vectors of A and the k diagonal elements of D are the singular values.

When A is complex then the singular value decomposition is given by

$$A=QDP^H,$$

where Q and P are unitary, P^H denotes the complex conjugate of P^T and D is as above for the real case.

If the matrix A has column means of zero, then AP is the matrix of principal components of A and the singular values are the square roots of the sample variances of the observations with respect to the principal components. (See also Chapter G03.)

Routines are provided to return the singular values and vectors of a general real or complex matrix.

3. Recommendations on Choice and Use of Routines

3.1. General Discussion

There is one routine, F02FJF, which is designed for sparse symmetric eigenvalue problems, either standard or generalized. The remainder of the routines are designed for dense matrices.

3.2. Eigenvalue and Eigenvector Routines

These reduce the matrix A to a simpler form by a similarity transformation $S^{-1}AS=T$ where T is an upper Hessenberg or tridiagonal matrix, compute the eigensolutions of T, and then recover the eigenvectors of A via the matrix S. The eigenvectors are normalised so that

$$\sum_{r=1}^{n} |x_r|^2=1$$

x_r being the rth component of the eigenvector x, and so that the element of largest modulus is real if x is complex. For problems of the type Ax=(lambda)Bx with A and B symmetric and B positive-definite, the eigenvectors are normalised so that $x^TBx=1$, x always being real for such problems.

3.3. Singular Value and Singular Vector Routines

These reduce the matrix A to real bidiagonal form, B say, by orthogonal transformations $Q^T AP=B$ in the real case, and by unitary transformations $Q^H AP=B$ in the complex case, and the singular values and vectors are computed via this bidiagonal form. The singular values are returned in descending order.

3.4. Decision Trees

(i) Eigenvalues and Eigenvectors

Please see figure in printed Reference Manual

(ii) Singular Values and Singular Vectors

Please see figure in printed Reference Manual

F02 -- Eigenvalues and Eigenvectors Contents -- F02

Chapter F02

Eigenvalues and Eigenvectors

F02AAF All eigenvalues of real symmetric matrix

F02ABF All eigenvalues and eigenvectors of real symmetric matrix

F02ADF All eigenvalues of generalized real symmetric-definite eigenproblem

F02AEF All eigenvalues and eigenvectors of generalized real symmetric-definite eigenproblem

F02AFF All eigenvalues of real matrix

F02AGF All eigenvalues and eigenvectors of real matrix

F02AJF All eigenvalues of complex matrix

F02AKF All eigenvalues and eigenvectors of complex matrix

F02AWF All eigenvalues of complex Hermitian matrix

```
          F02AXF  All eigenvalues and eigenvectors of complex Hermitian
                  matrix

          F02BBF  Selected eigenvalues and eigenvectors of real symmetric
                  matrix

          F02BJF  All eigenvalues and optionally eigenvectors of
                  generalized eigenproblem by QZ algorithm, real matrices

          F02FJF  Selected eigenvalues and eigenvectors of sparse symmetric
                  eigenproblem

          F02WEF  SVD of real matrix

          F02XEF  SVD of complex matrix

\end{verbatim}
\endscroll
\end{page}
```

———-

Calculates all the eigenvalues of a real symmetric matrix

— nagf.ht —

```
\begin{page}{manpageXXf02aaf}{NAG Documentation: f02aaf}
\beginscroll
\begin{verbatim}



     F02AAF(3NAG)      Foundation Library (12/10/92)      F02AAF(3NAG)



          F02 -- Eigenvalue and Eigenvectors                         F02AAF
                  F02AAF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          F02AAF calculates all the eigenvalues of a real symmetric matrix.
```

2. Specification

```
SUBROUTINE F02AAF (A, IA, N, R, E, IFAIL)
INTEGER          IA, N, IFAIL
DOUBLE PRECISION A(IA,N), R(N), E(N)
```

3. Description

This routine reduces the real symmetric matrix A to a real symmetric tridiagonal matrix using Householder's method. The eigenvalues of the tridiagonal matrix are then determined using the QL algorithm.

4. References

[1] Wilkinson J H and Reinsch C (1971) Handbook for Automatic Computation II, Linear Algebra. Springer-Verlag.

5. Parameters

1: A(IA,N) -- DOUBLE PRECISION array Input/Output
On entry: the lower triangle of the n by n symmetric matrix A. The elements of the array above the diagonal need not be set. On exit: the elements of A below the diagonal are overwritten, and the rest of the array is unchanged.

2: IA -- INTEGER Input
On entry:
the first dimension of the array A as declared in the (sub)program from which F02AAF is called.
Constraint: IA >= N.

3: N -- INTEGER Input
On entry: n, the order of the matrix A.

4: R(N) -- DOUBLE PRECISION array Output
On exit: the eigenvalues in ascending order.

5: E(N) -- DOUBLE PRECISION array Workspace

6: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

```
          Errors detected by the routine:

          IFAIL= 1
               Failure in F02AVF(*) indicating that more than 30*N
               iterations are required to isolate all the eigenvalues.

          7. Accuracy

          The accuracy of the eigenvalues depends on the sensitivity of the
          matrix to rounding errors produced in tridiagonalisation. For a
          detailed error analysis see Wilkinson and Reinsch [1] pp 222 and
          235.

          8. Further Comments

                                                                           3
          The time taken by the routine is approximately proportional to n

          9. Example

          To calculate all the eigenvalues of the real symmetric matrix:

                               ( 0.5  0.0  2.3 -2.6)
                               ( 0.0  0.5 -1.4 -0.7)
                               ( 2.3 -1.4  0.5  0.0).
                               (-2.6 -0.7  0.0  0.5)

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Eigenvalues and eigenvectors of a real symmetric matrix

— nagf.ht —

```
\begin{page}{manpageXXf02abf}{NAG Documentation: f02abf}
\beginscroll
\begin{verbatim}
```

F02 -- Eigenvalue and Eigenvectors F02ABF
F02ABF -- NAG Foundation Library Routine Document

Note: Before using this routine, please read the Users' Note for your implementation to check implementation-dependent details. The symbol (*) after a NAG routine name denotes a routine that is not included in the Foundation Library.

1. Purpose

F02ABF calculates all the eigenvalues and eigenvectors of a real symmetric matrix.

2. Specification

```
SUBROUTINE F02ABF (A, IA, N, R, V, IV, E, IFAIL)
INTEGER          IA, N, IV, IFAIL
DOUBLE PRECISION A(IA,N), R(N), V(IV,N), E(N)
```

3. Description

This routine reduces the real symmetric matrix A to a real symmetric tridiagonal matrix by Householder's method. The eigenvalues and eigenvectors are calculated using the QL algorithm.

4. References

[1] Wilkinson J H and Reinsch C (1971) Handbook for Automatic Computation II, Linear Algebra. Springer-Verlag.

5. Parameters

1: A(IA,N) -- DOUBLE PRECISION array Input
On entry: the lower triangle of the n by n symmetric matrix A. The elements of the array above the diagonal need not be set. See also Section 8.

2: IA -- INTEGER Input
On entry:
the first dimension of the array A as declared in the (sub)program from which F02ABF is called.
Constraint: IA >= N.

3: N -- INTEGER Input

On entry: n, the order of the matrix A.

4: R(N) -- DOUBLE PRECISION array Output
On exit: the eigenvalues in ascending order.

5: V(IV,N) -- DOUBLE PRECISION array Output
On exit: the normalised eigenvectors, stored by columns; the ith column corresponds to the ith eigenvalue. The eigenvectors are normalised so that the sum of squares of the elements is equal to 1.

6: IV -- INTEGER Input
On entry:
the first dimension of the array V as declared in the (sub)program from which F02ABF is called.
Constraint: IV >= N.

7: E(N) -- DOUBLE PRECISION array Workspace

8: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1
Failure in F02AMF(*) indicating that more than 30*N iterations are required to isolate all the eigenvalues.

7. Accuracy

The eigenvectors are always accurately orthogonal but the accuracy of the individual eigenvectors is dependent on their inherent sensitivity to changes in the original matrix. For a detailed error analysis see Wilkinson and Reinsch [1] pp 222 and 235.

8. Further Comments

The time taken by the routine is approximately proportional to n^3

Unless otherwise stated in the Users' Note for your implementation, the routine may be called with the same actual

```
         array supplied for parameters A and V, in which case the
         eigenvectors will overwrite the original matrix. However this is
         not standard Fortran 77, and may not work on all systems.

         9. Example

         To calculate all the eigenvalues and eigenvectors of the real
         symmetric matrix:

                              ( 0.5  0.0  2.3 -2.6)
                              ( 0.0  0.5 -1.4 -0.7)
                              ( 2.3 -1.4  0.5  0.0).
                              (-2.6 -0.7  0.0  0.5)

         The example program is not reproduced here. The source code for
         all example programs is distributed with the NAG Foundation
         Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Calculates all the eigenvalues of $Ax = \lambda Bx$

— nagf.ht —

```
\begin{page}{manpageXXf02adf}{NAG Documentation: f02adf}
\beginscroll
\begin{verbatim}



     F02ADF(3NAG)      Foundation Library (12/10/92)      F02ADF(3NAG)



         F02 -- Eigenvalue and Eigenvectors                         F02ADF
                 F02ADF -- NAG Foundation Library Routine Document

         Note: Before using this routine, please read the Users' Note for
         your implementation to check implementation-dependent details.
         The symbol (*) after a NAG routine name denotes a routine that is
         not included in the Foundation Library.

         1. Purpose
```

F02ADF calculates all the eigenvalues of Ax=(lambda)Bx, where A is a real symmetric matrix and B is a real symmetric positive-definite matrix.

2. Specification

```
SUBROUTINE F02ADF (A, IA, B, IB, N, R, DE, IFAIL)
INTEGER          IA, IB, N, IFAIL
DOUBLE PRECISION A(IA,N), B(IB,N), R(N), DE(N)
```

3. Description

The problem is reduced to the standard symmetric eigenproblem using Cholesky's method to decompose B into triangular matrices, $B=LL^T$, where L is lower triangular. Then Ax=(lambda)Bx implies $(L^{-1}AL^{-T})(L^Tx)=(\lambda)(L^Tx)$; hence the eigenvalues of Ax=(lambda)Bx are those of Py=(lambda)y where P is the symmetric matrix $L^{-1}AL^{-T}$. Householder's method is used to tridiagonalise the matrix P and the eigenvalues are then found using the QL algorithm.

4. References

[1] Wilkinson J H and Reinsch C (1971) Handbook for Automatic Computation II, Linear Algebra. Springer-Verlag.

5. Parameters

1: A(IA,N) -- DOUBLE PRECISION array Input/Output
On entry: the upper triangle of the n by n symmetric matrix A. The elements of the array below the diagonal need not be set. On exit: the lower triangle of the array is overwritten. The rest of the array is unchanged.

2: IA -- INTEGER Input
On entry:
the first dimension of the array A as declared in the (sub)program from which F02ADF is called.
Constraint: IA >= N.

3: B(IB,N) -- DOUBLE PRECISION array Input/Output
On entry: the upper triangle of the n by n symmetric positive-definite matrix B. The elements of the array below the diagonal need not be set. On exit: the elements below the diagonal are overwritten. The rest of the array is unchanged.

4: IB -- INTEGER Input
On entry:
the first dimension of the array B as declared in the
(sub)program from which F02ADF is called.
Constraint: IB >= N.

5: N -- INTEGER Input
On entry: n, the order of the matrices A and B.

6: R(N) -- DOUBLE PRECISION array Output
On exit: the eigenvalues in ascending order.

7: DE(N) -- DOUBLE PRECISION array Workspace

8: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not
familiar with this parameter (described in the Essential
Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see
Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1
Failure in F01AEF(*); the matrix B is not positive-definite
possibly due to rounding errors.

IFAIL= 2
Failure in F02AVF(*), more than 30*N iterations are required
to isolate all the eigenvalues.

7. Accuracy

In general this routine is very accurate. However, if B is ill-conditioned with respect to inversion, the eigenvalues could be inaccurately determined. For a detailed error analysis see Wilkinson and Reinsch [1] pp 310, 222 and 235.

8. Further Comments

The time taken by the routine is approximately proportional to n^3

9. Example

To calculate all the eigenvalues of the general symmetric

```
          eigenproblem Ax=(lambda) Bx where A is the symmetric matrix:

                                 (0.5  1.5  6.6   4.8)
                                 (1.5  6.5 16.2   8.6)
                                 (6.6 16.2 37.6   9.8)
                                 (4.8  8.6  9.8 -17.1)

          and B is the symmetric positive-definite matrix:

                                     (1  3  4  1)
                                     (3 13 16 11)
                                     (4 16 24 18).
                                     (1 11 18 27)

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

————-

Eigenvalues and eigenvectors of $Ax = \lambda Bx$

— nagf.ht —

```
\begin{page}{manpageXXf02aef}{NAG Documentation: f02aef}
\beginscroll
\begin{verbatim}



     F02AEF(3NAG)      Foundation Library (12/10/92)      F02AEF(3NAG)



          F02 -- Eigenvalue and Eigenvectors                        F02AEF
                  F02AEF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose
```

F02AEF calculates all the eigenvalues and eigenvectors of $Ax=(\lambda)Bx$, where A is a real symmetric matrix and B is a real symmetric positive-definite matrix.

2. Specification

```
       SUBROUTINE F02AEF (A, IA, B, IB, N, R, V, IV, DL, E, IFAIL)
       INTEGER          IA, IB, N, IV, IFAIL
       DOUBLE PRECISION A(IA,N), B(IB,N), R(N), V(IV,N), DL(N), E
      1                 (N)
```

3. Description

The problem is reduced to the standard symmetric eigenproblem using Cholesky's method to decompose B into triangular matrices $B=LL^T$, where L is lower triangular. Then $Ax=(\lambda)Bx$ implies $(L^{-1}AL^{-T})(L^Tx)=(\lambda)(L^Tx)$; hence the eigenvalues of $Ax=(\lambda)Bx$ are those of $Py=(\lambda)y$, where P is the symmetric matrix $L^{-1}AL^{-T}$. Householder's method is used to tridiagonalise the matrix P and the eigenvalues are found using the QL algorithm. An eigenvector z of the derived problem is related to an eigenvector x of the original problem by $z=L^Tx$. The eigenvectors z are determined using the QL algorithm and are normalised so that $z^Tz=1$; the eigenvectors of the original problem are then determined by solving $L^Tx=z$, and are normalised so that $x^TBx=1$.

4. References

[1] Wilkinson J H and Reinsch C (1971) Handbook for Automatic Computation II, Linear Algebra. Springer-Verlag.

5. Parameters

1: A(IA,N) -- DOUBLE PRECISION array Input/Output

On entry: the upper triangle of the n by n symmetric matrix A. The elements of the array below the diagonal need not be set. On exit: the lower triangle of the array is overwritten. The rest of the array is unchanged. See also Section 8.

```
 2:  IA -- INTEGER                                          Input
     On entry:
     the first dimension of the array A as declared in the
     (sub)program from which F02AEF is called.
     Constraint: IA >= N.

 3:  B(IB,N) -- DOUBLE PRECISION array               Input/Output
     On entry: the upper triangle of the n by n symmetric
     positive-definite matrix B. The elements of the array below
     the diagonal need not be set. On exit: the elements below
     the diagonal are overwritten. The rest of the array is
     unchanged.

 4:  IB -- INTEGER                                          Input
     On entry:
     the first dimension of the array B as declared in the
     (sub)program from which F02AEF is called.
     Constraint: IB >= N.

 5:  N -- INTEGER                                           Input
     On entry: n, the order of the matrices A and B.

 6:  R(N) -- DOUBLE PRECISION array                        Output
     On exit: the eigenvalues in ascending order.

 7:  V(IV,N) -- DOUBLE PRECISION array                     Output
     On exit: the normalised eigenvectors, stored by columns;
     the ith column corresponds to the ith eigenvalue. The
                                       T
     eigenvectors x are normalised so that x Bx=1. See also
     Section 8.

 8:  IV -- INTEGER                                          Input
     On entry:
     the first dimension of the array V as declared in the
     (sub)program from which F02AEF is called.
     Constraint: IV >= N.

 9:  DL(N) -- DOUBLE PRECISION array                    Workspace

10:  E(N) -- DOUBLE PRECISION array                     Workspace

11:  IFAIL -- INTEGER                                Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.
     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).

6. Error Indicators and Warnings
```

Errors detected by the routine:

IFAIL= 1
Failure in F01AEF(*); the matrix B is not positive-definite, possibly due to rounding errors.

IFAIL= 2
Failure in F02AMF(*); more than 30*N iterations are required to isolate all the eigenvalues.

7. Accuracy

In general this routine is very accurate. However, if B is ill-conditioned with respect to inversion, the eigenvectors could be inaccurately determined. For a detailed error analysis see Wilkinson and Reinsch [1] pp 310, 222 and 235.

8. Further Comments

The time taken by the routine is approximately proportional to n^3

Unless otherwise stated in the Users' Note for your implementation, the routine may be called with the same actual array supplied for parameters A and V, in which case the eigenvectors will overwrite the original matrix A. However this is not standard Fortran 77, and may not work on all systems.

9. Example

To calculate all the eigenvalues and eigenvectors of the general symmetric eigenproblem Ax=(lambda) Bx where A is the symmetric matrix:

```
(0.5  1.5  6.6   4.8)
(1.5  6.5 16.2   8.6)
(6.6 16.2 37.6   9.8)
(4.8  8.6  9.8 -17.1)
```

and B is the symmetric positive-definite matrix:

```
(1  3  4  1)
(3 13 16 11)
(4 16 24 18).
(1 11 18 27)
```

The example program is not reproduced here. The source code for all example programs is distributed with the NAG Foundation Library software and should be available on-line.

```
\end{verbatim}
\endscroll
\end{page}
```

Calculates all the eigenvalues of a real unsymmetric matrix

— nagf.ht —

```
\begin{page}{manpageXXf02aff}{NAG Documentation: f02aff}
\beginscroll
\begin{verbatim}



     F02AFF(3NAG)      Foundation Library (12/10/92)      F02AFF(3NAG)



          F02 -- Eigenvalue and Eigenvectors                         F02AFF
                  F02AFF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          F02AFF calculates all the eigenvalues of a real unsymmetric
          matrix.

          2. Specification

                 SUBROUTINE F02AFF (A, IA, N, RR, RI, INTGER, IFAIL)
                 INTEGER          IA, N, INTGER(N), IFAIL
                 DOUBLE PRECISION A(IA,N), RR(N), RI(N)

          3. Description

          The matrix A is first balanced and then reduced to upper
          Hessenberg form using stabilised elementary similarity
          transformations. The eigenvalues are then found using the QR
          algorithm for real Hessenberg matrices.
```

4. References

[1] Wilkinson J H and Reinsch C (1971) Handbook for Automatic Computation II, Linear Algebra. Springer-Verlag.

5. Parameters

```
 1:  A(IA,N) -- DOUBLE PRECISION array               Input/Output
     On entry: the n by n matrix A. On exit: the array is
     overwritten.

 2:  IA -- INTEGER                                          Input
     On entry:
     the dimension of the array A as declared in the (sub)program
     from which F02AFF is called.
     Constraint: IA >= N.

 3:  N -- INTEGER                                           Input
     On entry: n, the order of the matrix A.

 4:  RR(N) -- DOUBLE PRECISION array                       Output
     On exit: the real parts of the eigenvalues.

 5:  RI(N) -- DOUBLE PRECISION array                       Output
     On exit: the imaginary parts of the eigenvalues.

 6:  INTGER(N) -- INTEGER array                            Output
     On exit: INTGER(i) contains the number of iterations used
     to find the ith eigenvalue. If INTGER(i) is negative, the i
     th eigenvalue is the second of a pair found simultaneously.

     Note that the eigenvalues are found in reverse order,
     starting with the nth.

 7:  IFAIL -- INTEGER                                Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).
```

6. Error Indicators and Warnings

Errors detected by the routine:

```
IFAIL= 1
     More than 30*N iterations are required to isolate all the
     eigenvalues.
```

```
         7. Accuracy

         The accuracy of the results depends on the original matrix and
         the multiplicity of the roots. For a detailed error analysis see
         Wilkinson and Reinsch [1] pp 352 and 367.

         8. Further Comments

                                                                        3
         The time taken by the routine is approximately proportional to n

         9. Example

         To calculate all the eigenvalues of the real matrix:

                              (  1.5 0.1  4.5 -1.5)
                              (-22.5 3.5 12.5 -2.5)
                              ( -2.5 0.3  4.5 -2.5).
                              ( -2.5 0.1  4.5  2.5)

         The example program is not reproduced here. The source code for
         all example programs is distributed with the NAG Foundation
         Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Eigenvalues and eigenvectors of a real unsymmetric matrix

— nagf.ht —

```
\begin{page}{manpageXXf02agf}{NAG Documentation: f02agf}
\beginscroll
\begin{verbatim}



     F02AGF(3NAG)      Foundation Library (12/10/92)      F02AGF(3NAG)



         F02 -- Eigenvalue and Eigenvectors                        F02AGF
                 F02AGF -- NAG Foundation Library Routine Document
```

Note: Before using this routine, please read the Users' Note for your implementation to check implementation-dependent details. The symbol (*) after a NAG routine name denotes a routine that is not included in the Foundation Library.

1. Purpose

F02AGF calculates all the eigenvalues and eigenvectors of a real unsymmetric matrix.

2. Specification

```
       SUBROUTINE F02AGF (A, IA, N, RR, RI, VR, IVR, VI, IVI,
      1                   INTGER, IFAIL)
       INTEGER          IA, N, IVR, IVI, INTGER(N), IFAIL
       DOUBLE PRECISION A(IA,N), RR(N), RI(N), VR(IVR,N), VI
      1                 (IVI,N)
```

3. Description

The matrix A is first balanced and then reduced to upper Hessenberg form using real stabilised elementary similarity transformations. The eigenvalues and eigenvectors of the Hessenberg matrix are calculated using the QR algorithm. The eigenvectors of the Hessenberg matrix are back-transformed to give the eigenvectors of the original matrix A.

4. References

[1] Wilkinson J H and Reinsch C (1971) Handbook for Automatic Computation II, Linear Algebra. Springer-Verlag.

5. Parameters

1: A(IA,N) -- DOUBLE PRECISION array Input/Output
On entry: the n by n matrix A. On exit: the array is overwritten.

2: IA -- INTEGER Input
On entry:
the first dimension of the array A as declared in the (sub)program from which F02AGF is called.
Constraint: IA >= N.

3: N -- INTEGER Input
On entry: n, the order of the matrix A.

4: RR(N) -- DOUBLE PRECISION array Output
On exit: the real parts of the eigenvalues.

```
 5:  RI(N) -- DOUBLE PRECISION array                       Output
     On exit: the imaginary parts of the eigenvalues.

 6:  VR(IVR,N) -- DOUBLE PRECISION array                   Output
     On exit: the real parts of the eigenvectors, stored by
     columns. The ith column corresponds to the ith eigenvalue.
     The eigenvectors are normalised so that the sum of the
     squares of the moduli of the elements is equal to 1 and the
     element of largest modulus is real. This ensures that real
     eigenvalues have real eigenvectors.

 7:  IVR -- INTEGER                                         Input
     On entry:
     the first dimension of the array VR as declared in the
     (sub)program from which F02AGF is called.
     Constraint: IVR >= N.

 8:  VI(IVI,N) -- DOUBLE PRECISION array                   Output
     On exit: the imaginary parts of the eigenvectors, stored by
     columns. The ith column corresponds to the ith eigenvalue.

 9:  IVI -- INTEGER                                         Input
     On entry:
     the first dimension of the array VI as declared in the
     (sub)program from which F02AGF is called.
     Constraint: IVI >= N.

10:  INTGER(N) -- INTEGER array                            Output
     On exit: INTGER(i) contains the number of iterations used
     to find the ith eigenvalue. If INTGER(i) is negative, the i
     th eigenvalue is the second of a pair found simultaneously.

     Note that the eigenvalues are found in reverse order,
     starting with the nth.

11:  IFAIL -- INTEGER                               Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1
     More than 30*N iterations are required to isolate all the
     eigenvalues.
```

```
          7. Accuracy

          The accuracy of the results depends on the original matrix and
          the multiplicity of the roots. For a detailed error analysis see
          Wilkinson and Reinsch [1] pp 352 and 390.

          8. Further Comments

                                                                     3
          The time taken by the routine is approximately proportional to n

          9. Example

          To calculate all the eigenvalues and eigenvectors of the real
          matrix:

                               (  1.5 0.1  4.5 -1.5)
                               (-22.5 3.5 12.5 -2.5)
                               ( -2.5 0.3  4.5 -2.5).
                               ( -2.5 0.1  4.5  2.5)

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Calculates all the eigenvalues of a complex matrix

— nagf.ht —

```
\begin{page}{manpageXXf02ajf}{NAG Documentation: f02ajf}
\beginscroll
\begin{verbatim}



     F02AJF(3NAG)      Foundation Library (12/10/92)      F02AJF(3NAG)



          F02 -- Eigenvalue and Eigenvectors                        F02AJF
```

F02AJF -- NAG Foundation Library Routine Document

Note: Before using this routine, please read the Users' Note for your implementation to check implementation-dependent details. The symbol (*) after a NAG routine name denotes a routine that is not included in the Foundation Library.

1. Purpose

F02AJF calculates all the eigenvalues of a complex matrix.

2. Specification

```
       SUBROUTINE F02AJF (AR, IAR, AI, IAI, N, RR, RI, INTGER,
      1                   IFAIL)
       INTEGER          IAR, IAI, N, INTGER(N), IFAIL
       DOUBLE PRECISION AR(IAR,N), AI(IAI,N), RR(N), RI(N)
```

3. Description

The complex matrix A is first balanced and then reduced to upper Hessenberg form using stabilised elementary similarity transformations. The eigenvalues are then found using the modified LR algorithm for complex Hessenberg matrices.

4. References

[1] Wilkinson J H and Reinsch C (1971) Handbook for Automatic Computation II, Linear Algebra. Springer-Verlag.

5. Parameters

1: AR(IAR,N) -- DOUBLE PRECISION array Input/Output
On entry: the real parts of the elements of the n by n complex matrix A. On exit: the array is overwritten.

2: IAR -- INTEGER Input
On entry:
the first dimension of the array AR as declared in the (sub)program from which F02AJF is called.
Constraint: IAR >= N.

3: AI(IAI,N) -- DOUBLE PRECISION array Input/Output
On entry: the imaginary parts of the elements of the n by n complex matrix A. On exit: the array is overwritten.

4: IAI -- INTEGER Input
On entry:
the first dimension of the array AI as declared in the (sub)program from which F02AJF is called.

```
     Constraint: IAI >= N.

 5:  N -- INTEGER                                          Input
     On entry: n, the order of the matrix A.

 6:  RR(N) -- DOUBLE PRECISION array                      Output
     On exit: the real parts of the eigenvalues.

 7:  RI(N) -- DOUBLE PRECISION array                      Output
     On exit: the imaginary parts of the eigenvalues.

 8:  INTGER(N) -- INTEGER array                        Workspace

 9:  IFAIL -- INTEGER                               Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1
     More than 30*N iterations are required to isolate all the
     eigenvalues.

7. Accuracy

The accuracy of the results depends on the original matrix and
the multiplicity of the roots. For a detailed error analysis see
Wilkinson and Reinsch [1] pp 352 and 401.

8. Further Comments

                                                                3
The time taken by the routine is approximately proportional to n

9. Example

To calculate all the eigenvalues of the complex matrix:

           (-21.0-5.0i      24.60i 13.6+10.2i     4.0i)
           (     22.5i 26.00-5.00i  7.5-10.0i 2.5     )
           ( -2.0+1.5i  1.68+2.24i  4.5-5.0i  1.5+2.0i).
           (     -2.5i -2.60       -2.7+3.6i  2.5-5.0i)

The example program is not reproduced here. The source code for
```

```
         all example programs is distributed with the NAG Foundation
         Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

————-

Eigenvalues and eigenvectors of a complex matrix

— nagf.ht —

```
\begin{page}{manpageXXf02akf}{NAG Documentation: f02akf}
\beginscroll
\begin{verbatim}



     F02AKF(3NAG)      Foundation Library (12/10/92)      F02AKF(3NAG)



          F02 -- Eigenvalue and Eigenvectors                         F02AKF
                  F02AKF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          F02AKF calculates all the eigenvalues and eigenvectors of a
          complex matrix.

          2. Specification

                 SUBROUTINE F02AKF (AR, IAR, AI, IAI, N, RR, RI, VR, IVR,
                1                   VI, IVI, INTGER, IFAIL)
                 INTEGER          IAR, IAI, N, IVR, IVI, INTGER(N), IFAIL
                 DOUBLE PRECISION AR(IAR,N), AI(IAI,N), RR(N), RI(N), VR
                1                 (IVR,N), VI(IVI,N)

          3. Description

          The complex matrix A is first balanced and then reduced to upper
```

Hessenberg form by stabilised elementary similarity transformations. The eigenvalues and eigenvectors of the Hessenberg matrix are calculated using the LR algorithm. The eigenvectors of the Hessenberg matrix are back-transformed to give the eigenvectors of the original matrix.

4. References

[1] Wilkinson J H and Reinsch C (1971) Handbook for Automatic Computation II, Linear Algebra. Springer-Verlag.

5. Parameters

1: AR(IAR,N) -- DOUBLE PRECISION array Input/Output
On entry: the real parts of the elements of the n by n complex matrix A. On exit: the array is overwritten.

2: IAR -- INTEGER Input
On entry:
the first dimension of the array AR as declared in the (sub)program from which F02AKF is called.
Constraint: IAR >= N.

3: AI(IAI,N) -- DOUBLE PRECISION array Input/Output
On entry: the imaginary parts of the elements of the n by n complex matrix A. On exit: the array is overwritten.

4: IAI -- INTEGER Input
On entry:
the first dimension of the array AI as declared in the (sub)program from which F02AKF is called.
Constraint: IAI >= N.

5: N -- INTEGER Input
On entry: n, the order of the matrix A.

6: RR(N) -- DOUBLE PRECISION array Output
On exit: the real parts of the eigenvalues.

7: RI(N) -- DOUBLE PRECISION array Output
On exit: the imaginary parts of the eigenvalues.

8: VR(IVR,N) -- DOUBLE PRECISION array Output
On exit: the real parts of the eigenvectors, stored by columns. The ith column corresponds to the ith eigenvalue. The eigenvectors are normalised so that the sum of squares of the moduli of the elements is equal to 1 and the element of largest modulus is real.

9: IVR -- INTEGER Input

```
     On entry:
     the first dimension of the array VR as declared in the
     (sub)program from which F02AKF is called.
     Constraint: IVR >= N.

10:  VI(IVI,N) -- DOUBLE PRECISION array                  Output
     On exit: the imaginary parts of the eigenvectors, stored by
     columns. The ith column corresponds to the ith eigenvalue.

11:  IVI -- INTEGER                                        Input
     On entry:
     the first dimension of the array VI as declared in the
     (sub)program from which F02AKF is called.
     Constraint: IVI >= N.

12:  INTGER(N) -- INTEGER array                        Workspace

13:  IFAIL -- INTEGER                               Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1
     More than 30*N iterations are required to isolate all the
     eigenvalues.

7. Accuracy

The accuracy of the results depends on the conditioning of the
original matrix and the multiplicity of the roots. For a detailed
error analysis see Wilkinson and Reinsch [1] pp 352 and 390.

8. Further Comments

                                                               3
The time taken by the routine is approximately proportional to n

9. Example

To calculate all the eigenvalues and eigenvectors of the complex
matrix:

           (-21.0-5.0i      24.60i 13.6+10.2i     4.0i)
```

```
              (      22.5i 26.00-5.00i  7.5-10.0i 2.5     )
              ( -2.0+1.5i  1.68+2.24i  4.5-5.0i  1.5+2.0i).
              (      -2.5i -2.60       -2.7+3.6i  2.5-5.0i)

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Eigenvalues of a complex Hermitian matrix

— nagf.ht —

```
\begin{page}{manpageXXf02awf}{NAG Documentation: f02awf}
\beginscroll
\begin{verbatim}



     F02AWF(3NAG)      Foundation Library (12/10/92)      F02AWF(3NAG)



          F02 -- Eigenvalue and Eigenvectors                         F02AWF
                  F02AWF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          F02AWF calculates all the eigenvalues of a complex Hermitian
          matrix.

          2. Specification

                 SUBROUTINE F02AWF (AR, IAR, AI, IAI, N, R, WK1, WK2, WK3,
                1                   IFAIL)
                 INTEGER          IAR, IAI, N, IFAIL
                 DOUBLE PRECISION AR(IAR,N), AI(IAI,N), R(N), WK1(N),
```

```
     1                 WK2(N), WK3(N)
```

3. Description

The complex Hermitian matrix A is first reduced to a real tridiagonal matrix by n-2 unitary transformations, and a subsequent diagonal transformation. The eigenvalues are then derived using the QL algorithm, an adaptation of the QR algorithm.

4. References

[1] Peters G (1967) NPL Algorithms Library. Document No. F1/04/A.

[2] Wilkinson J H and Reinsch C (1971) Handbook for Automatic Computation II, Linear Algebra. Springer-Verlag.

5. Parameters

1: AR(IAR,N) -- DOUBLE PRECISION array Input/Output
On entry: the real parts of the elements of the lower triangle of the n by n complex Hermitian matrix A. Elements of the array above the diagonal need not be set. On exit: the array is overwritten.

2: IAR -- INTEGER Input
On entry:
the first dimension of the array AR as declared in the (sub)program from which F02AWF is called.
Constraint: IAR >= N.

3: AI(IAI,N) -- DOUBLE PRECISION array Input/Output
On entry: the imaginary parts of the elements of the lower triangle of the n by n complex Hermitian matrix A. Elements of the array above the diagonal need not be set. On exit: the array is overwritten.

4: IAI -- INTEGER Input
On entry:
the first dimension of the array AI as declared in the (sub)program from which F02AWF is called.
Constraint: IAI >= N.

5: N -- INTEGER Input
On entry: n, the order of the complex Hermitian matrix, A.

6: R(N) -- DOUBLE PRECISION array Output
On exit: the eigenvalues in ascending order.

```
 7:  WK1(N) -- DOUBLE PRECISION array                  Workspace

 8:  WK2(N) -- DOUBLE PRECISION array                  Workspace

 9:  WK3(N) -- DOUBLE PRECISION array                  Workspace

10:  IFAIL -- INTEGER                               Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1
     More than 30*N iterations are required to isolate all the
     eigenvalues.

7. Accuracy

For a detailed error analysis see Peters [1] page 3 and Wilkinson
and Reinsch [2] page 235.

8. Further Comments

                                                                3
The time taken by the routine is approximately proportional to n

9. Example

To calculate all the eigenvalues of the complex Hermitian matrix:

            (0.50        0.00       1.84+1.38i  2.08-1.56i)
            (0.00        0.50       1.12+0.84i -0.56+0.42i)
            (1.84-1.38i  1.12-0.84i 0.50        0.00      ).
            (2.08+1.56i -0.56-0.42i 0.00        0.50      )

The example program is not reproduced here. The source code for
all example programs is distributed with the NAG Foundation
Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

———-

Eigenvalues/eigenvectors complex Hermitian matrix

— nagf.ht —

```
\begin{page}{manpageXXf02axf}{NAG Documentation: f02axf}
\beginscroll
\begin{verbatim}



     F02AXF(3NAG)      Foundation Library (12/10/92)      F02AXF(3NAG)



          F02 -- Eigenvalue and Eigenvectors                        F02AXF
                  F02AXF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          F02AXF calculates all the eigenvalues and eigenvectors of a
          complex Hermitian matrix.

          2. Specification

                 SUBROUTINE F02AXF (AR, IAR, AI, IAI, N, R, VR, IVR, VI,
                1                   IVI, WK1, WK2, WK3, IFAIL)
                 INTEGER          IAR, IAI, N, IVR, IVI, IFAIL
                 DOUBLE PRECISION AR(IAR,N), AI(IAI,N), R(N), VR(IVR,N), VI
                1                 (IVI,N), WK1(N), WK2(N), WK3(N)

          3. Description

          The complex Hermitian matrix is first reduced to a real
          tridiagonal matrix by n-2 unitary transformations and a
          subsequent diagonal transformation. The eigenvalues and
          eigenvectors are then derived using the QL algorithm, an
          adaptation of the QR algorithm.

          4. References

          [1]   Peters G (1967) NPL Algorithms Library. Document No.
```

F2/03/A.

[2] Peters G (1967) NPL Algorithms Library. Document No. F1/04/A.

5. Parameters

```
 1:  AR(IAR,N) -- DOUBLE PRECISION array                    Input
     On entry: the real parts of the elements of the lower
     triangle of the n by n complex Hermitian matrix A. Elements
     of the array above the diagonal need not be set. See also
     Section 8.

 2:  IAR -- INTEGER                                         Input
     On entry:
     the first dimension of the array AR as declared in the
     (sub)program from which F02AXF is called.
     Constraint: IAR >= N.

 3:  AI(IAI,N) -- DOUBLE PRECISION array                    Input
     On entry: the imaginary parts of the elements of the lower
     triangle of the n by n complex Hermitian matrix A. Elements
     of the array above the diagonal need not be set. See also
     Section 8.

 4:  IAI -- INTEGER                                         Input
     On entry:
     the first dimension of the array AI as declared in the
     (sub)program from which F02AXF is called.
     Constraint: IAI >= N.

 5:  N -- INTEGER                                           Input
     On entry: n, the order of the matrix, A.

 6:  R(N) -- DOUBLE PRECISION array                        Output
     On exit: the eigenvalues in ascending order.

 7:  VR(IVR,N) -- DOUBLE PRECISION array                   Output
     On exit: the real parts of the eigenvectors, stored by
     columns. The ith column corresponds to the ith eigenvector.
     The eigenvectors are normalised so that the sum of the
     squares of the moduli of the elements is equal to 1 and the
     element of largest modulus is real. See also Section 8.

 8:  IVR -- INTEGER                                         Input
     On entry:
     the first dimension of the array VR as declared in the
     (sub)program from which F02AXF is called.
     Constraint: IVR >= N.
```

```
 9:  VI(IVI,N) -- DOUBLE PRECISION array                    Output
     On exit: the imaginary parts of the eigenvectors, stored by
     columns. The ith column corresponds to the ith eigenvector.
     See also Section 8.

10:  IVI -- INTEGER                                          Input
     On entry:
     the first dimension of the array VI as declared in the
     (sub)program from which F02AXF is called.
     Constraint: IVI >= N.

11:  WK1(N) -- DOUBLE PRECISION array                    Workspace

12:  WK2(N) -- DOUBLE PRECISION array                    Workspace

13:  WK3(N) -- DOUBLE PRECISION array                    Workspace

14:  IFAIL -- INTEGER                                 Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1
     More than 30*N iterations are required to isolate all the
     eigenvalues.

IFAIL= 2
     The diagonal elements of AI are not all zero, i.e., the
     complex matrix is not Hermitian.

7. Accuracy

The eigenvectors are always accurately orthogonal but the
accuracy of the individual eigenvalues and eigenvectors is
dependent on their inherent sensitivity to small changes in the
original matrix. For a detailed error analysis see Peters [1]
page 3 and [2] page 3.

8. Further Comments

                                                                3
The time taken by the routine is approximately proportional to n
```

```
          Unless otherwise stated in the implementation document, the
          routine may be called with the same actual array supplied for
          parameters AR and VR, and for AI and VI, in which case the
          eigenvectors will overwrite the original matrix A. However this
          is not standard Fortran 77, and may not work on all systems.

          9. Example

          To calculate the eigenvalues and eigenvectors of the complex
          Hermitian matrix:

                      (0.50        0.00       1.84+1.38i  2.08-1.56i)
                      (0.00        0.50       1.12+0.84i -0.56+0.42i)
                      (1.84-1.38i  1.12-0.84i 0.50        0.00      ).
                      (2.08+1.56i -0.56-0.42i 0.00        0.50      )

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Eigenvalues and eigenvectors of a real symmetric matrix

— nagf.ht —

```
\begin{page}{manpageXXf02bbf}{NAG Documentation: f02bbf}
\beginscroll
\begin{verbatim}



     F02BBF(3NAG)      Foundation Library (12/10/92)      F02BBF(3NAG)



          F02 -- Eigenvalue and Eigenvectors                         F02BBF
                  F02BBF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.
```

1. Purpose

F02BBF calculates selected eigenvalues and eigenvectors of a real symmetric matrix by reduction to tridiagonal form, bisection and inverse iteration, where the selected eigenvalues lie within a given interval.

2. Specification

```
       SUBROUTINE F02BBF (A, IA, N, ALB, UB, M, MM, R, V, IV, D,
      1                   E, E2, X, G, C, ICOUNT, IFAIL)
       INTEGER          IA, N, M, MM, IV, ICOUNT(M), IFAIL
       DOUBLE PRECISION A(IA,N), ALB, UB, R(M), V(IV,M), D(N), E
      1                 (N), E2(N), X(N,7), G(N)
       LOGICAL          C(N)
```

3. Description

The real symmetric matrix A is reduced to a symmetric tridiagonal matrix T by Householder's method. The eigenvalues which lie within a given interval [l,u], are calculated by the method of bisection. The corresponding eigenvectors of T are calculated by inverse iteration. A back-transformation is then performed to obtain the eigenvectors of the original matrix A.

4. References

[1] Wilkinson J H and Reinsch C (1971) Handbook for Automatic Computation II, Linear Algebra. Springer-Verlag.

5. Parameters

1: A(IA,N) -- DOUBLE PRECISION array Input/Output
On entry: the lower triangle of the n by n symmetric matrix A. The elements of the array above the diagonal need not be set. On exit: the elements of A below the diagonal are overwritten, and the rest of the array is unchanged.

2: IA -- INTEGER Input
On entry:
the first dimension of the array A as declared in the (sub)program from which F02BBF is called.
Constraint: IA >= N.

3: N -- INTEGER Input
On entry: n, the order of the matrix A.

4: ALB -- DOUBLE PRECISION Input

```
 5:  UB -- DOUBLE PRECISION                                 Input
     On entry: l and u, the lower and upper end-points of the
     interval within which eigenvalues are to be calculated.

 6:  M -- INTEGER                                           Input
     On entry: an upper bound for the number of eigenvalues
     within the interval.

 7:  MM -- INTEGER                                         Output
     On exit: the actual number of eigenvalues within the
     interval.

 8:  R(M) -- DOUBLE PRECISION array                        Output
     On exit: the eigenvalues, not necessarily in ascending
     order.

 9:  V(IV,M) -- DOUBLE PRECISION array                     Output
     On exit: the eigenvectors, stored by columns. The ith
     column corresponds to the ith eigenvalue. The eigenvectors
     are normalised so that the sum of the squares of the
     elements are equal to 1.

10:  IV -- INTEGER                                          Input
     On entry:
     the first dimension of the array V as declared in the
     (sub)program from which F02BBF is called.
     Constraint: IV >= N.

11:  D(N) -- DOUBLE PRECISION array                     Workspace

12:  E(N) -- DOUBLE PRECISION array                     Workspace

13:  E2(N) -- DOUBLE PRECISION array                    Workspace

14:  X(N,7) -- DOUBLE PRECISION array                   Workspace

15:  G(N) -- DOUBLE PRECISION array                     Workspace

16:  C(N) -- LOGICAL array                              Workspace

17:  ICOUNT(M) -- INTEGER array                            Output
     On exit: ICOUNT(i) contains the number of iterations for
     the ith eigenvalue.

18:  IFAIL -- INTEGER                                Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
```

```
     Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1
     M is less than the number of eigenvalues in the given
     interval. On exit MM contains the number of eigenvalues in
     the interval. Rerun with this value for M.

IFAIL= 2
     More than 5 iterations are required to determine any one
     eigenvector.

7. Accuracy

There is no guarantee of the accuracy of the eigenvectors as the
results depend on the original matrix and the multiplicity of the
roots. For a detailed error analysis see Wilkinson and Reinsch
[1] pp 222 and 436.

8. Further Comments

                                                                3
The time taken by the routine is approximately proportional to n

This subroutine should only be used when less than 25% of the
eigenvalues and the corresponding eigenvectors are required. Also
this subroutine is less efficient with matrices which have
multiple eigenvalues.

9. Example

To calculate the eigenvalues lying between -2.0 and 3.0, and the
corresponding eigenvectors of the real symmetric matrix:

                     ( 0.5  0.0  2.3 -2.6)
                     ( 0.0  0.5 -1.4 -0.7)
                     ( 2.3 -1.4  0.5  0.0).
                     (-2.6 -0.7  0.0  0.5)

The example program is not reproduced here. The source code for
all example programs is distributed with the NAG Foundation
Library software and should be available on-line.
\end{verbatim}
\endscroll
\end{page}
```

———-

Eigenvalues of generalized eigenproblem $Ax = \lambda Bx$

— nagf.ht —

```
\begin{page}{manpageXXf02bjf}{NAG Documentation: f02bjf}
\beginscroll
\begin{verbatim}



     F02BJF(3NAG)      Foundation Library (12/10/92)      F02BJF(3NAG)



          F02 -- Eigenvalue and Eigenvectors                          F02BJF
                  F02BJF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          F02BJF calculates all the eigenvalues and, if required, all the
          eigenvectors of the generalized eigenproblem  Ax=(lambda)Bx
          where A and B are real, square matrices, using the QZ algorithm.

          2. Specification

                 SUBROUTINE F02BJF (N, A, IA, B, IB, EPS1, ALFR, ALFI,
                1                   BETA, MATV, V, IV, ITER, IFAIL)
                 INTEGER          N, IA, IB, IV, ITER(N), IFAIL
                 DOUBLE PRECISION A(IA,N), B(IB,N), EPS1, ALFR(N), ALFI(N),
                1                 BETA(N), V(IV,N)
                 LOGICAL          MATV

          3. Description

          All the eigenvalues and, if required, all the eigenvectors of the
          generalized eigenproblem Ax=(lambda)Bx where A and B are real,
          square matrices, are determined using the QZ algorithm. The QZ
          algorithm consists of 4 stages:

          (a)   A is reduced to upper Hessenberg form and at the same time
                B is reduced to upper triangular form.
```

(b) A is further reduced to quasi-triangular form while the triangular form of B is maintained.

(c) The quasi-triangular form of A is reduced to triangular form and the eigenvalues extracted. This routine does not actually produce the eigenvalues $(lambda)_j$, but instead returns $(alpha)_j$ and $(beta)_j$ such that

$$(lambda)_j=(alpha)_j/(beta)_j, \quad j=1,2,...,.n$$

The division by $(beta)_j$ becomes the responsibility of the user's program, since $(beta)_j$ may be zero indicating an infinite eigenvalue. Pairs of complex eigenvalues occur with $(alpha)_j/(beta)_j$ and $(alpha)_{j+1}/(beta)_{j+1}$ complex conjugates, even though $(alpha)_j$ and $(alpha)_{j+1}$ are not conjugate.

(d) If the eigenvectors are required (MATV = .TRUE.), they are obtained from the triangular matrices and then transformed back into the original co-ordinate system.

4. References

[1] Moler C B and Stewart G W (1973) An Algorithm for Generalized Matrix Eigenproblems. SIAM J. Numer. Anal. 10 241--256.

[2] Ward R C (1975) The Combination Shift QZ Algorithm. SIAM J. Numer. Anal. 12 835--853.

[3] Wilkinson J H (1979) Kronecker's Canonical Form and the QZ Algorithm. Linear Algebra and Appl. 28 285--303.

5. Parameters

1: N -- INTEGER Input
On entry: n, the order of the matrices A and B.

2: A(IA,N) -- DOUBLE PRECISION array Input/Output
On entry: the n by n matrix A. On exit: the array is overwritten.

```
 3:  IA -- INTEGER                                          Input
     On entry:
     the first dimension of the array A as declared in the
     (sub)program from which F02BJF is called.
     Constraint: IA >= N.

 4:  B(IB,N) -- DOUBLE PRECISION array               Input/Output
     On entry: the n by n matrix B. On exit: the array is
     overwritten.

 5:  IB -- INTEGER                                          Input
     On entry:
     the first dimension of the array B as declared in the
     (sub)program from which F02BJF is called.
     Constraint: IB >= N.

 6:  EPS1 -- DOUBLE PRECISION                               Input
     On entry: the tolerance used to determine negligible
     elements. If EPS1 > 0.0, an element will be considered
     negligible if it is less than EPS1 times the norm of its
     matrix. If EPS1 <= 0.0, machine precision is used in place
     of EPS1. A positive value of EPS1 may result in faster
     execution but less accurate results.

 7:  ALFR(N) -- DOUBLE PRECISION array                     Output

 8:  ALFI(N) -- DOUBLE PRECISION array                     Output
     On exit: the real and imaginary parts of (alpha) , for
                                                      j
     j=1,2,...,n.

 9:  BETA(N) -- DOUBLE PRECISION array                     Output
     On exit: (beta) , for j=1,2,...,n.
                    j

10:  MATV -- LOGICAL                                        Input
     On entry: MATV must be set .TRUE. if the eigenvectors are
     required, otherwise .FALSE..

11:  V(IV,N) -- DOUBLE PRECISION array                     Output
     On exit: if MATV = .TRUE., then
          (i)if the jth eigenvalue is real, the jth column of V
          contains its eigenvector;

          (ii) if the jth and (j+1)th eigenvalues form a complex
          pair, the jth and (j+1)th columns of V contain the
          real and imaginary parts of the eigenvector associated
          with the first eigenvalue of the pair. The conjugate
          of this vector is the eigenvector for the conjugate
          eigenvalue.
```

Each eigenvector is normalised so that the component of largest modulus is real and the sum of squares of the moduli equal one.

If MATV = .FALSE., V is not used.

12: IV -- INTEGER Input
On entry:
the first dimension of the array V as declared in the (sub)program from which F02BJF is called.
Constraint: IV >= N.

13: ITER(N) -- INTEGER array Output
On exit: ITER(j) contains the number of iterations needed to obtain the jth eigenvalue. Note that the eigenvalues are obtained in reverse order, starting with the nth.

14: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= i
More than 30*N iterations are required to determine all the diagonal 1 by 1 or 2 by 2 blocks of the quasi-triangular form in the second step of the QZ algorithm. IFAIL is set to the index i of the eigenvalue at which this failure occurs. If the soft failure option is used, $(alpha)_j$ and $(beta)_j$ are correct for j=i+1,i+2,...,n, but V does not contain any correct eigenvectors.

7. Accuracy

The computed eigenvalues are always exact for a problem (A+E)x=(lambda)(B+F)x where ||E||/||A|| and ||F||/||B|| are both of the order of max(EPS1,(epsilon)), EPS1 being defined as in Section 5 and (epsilon) being the machine precision.

Note: interpretation of results obtained with the QZ algorithm often requires a clear understanding of the effects of small changes in the original data. These effects are reviewed in Wilkinson [3], in relation to the significance of small values of

```
          (alpha)  and (beta) . It should be noted that if (alpha)  and
                 j          j                                  j
          (beta)  are both small for any j, it may be that no reliance can
                j
          be placed on any of the computed eigenvalues
          (lambda) =(alpha) /(beta) . The user is recommended to study [3]
                  i        i       i
          and, if in difficulty, to seek expert advice on determining the
          sensitivity of the eigenvalues to perturbations in the data.

          8. Further Comments

                                                                       3
          The time taken by the routine is approximately proportional to n
          and also depends on the value chosen for parameter EPS1.

          9. Example

          To find all the eigenvalues and eigenvectors of Ax=(lambda) Bx
          where

                                 (3.9 12.5 -34.5 -0.5)
                                 (4.3 21.5 -47.5  7.5)
                               A=(4.3 21.5 -43.5  3.5)
                                 (4.4 26.0 -46.0  6.0)

          and

                                      (1 2 -3 1)
                                      (1 3 -5 4)
                                    B=(1 3 -4 3).
                                      (1 3 -4 4)

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Eigenvalues and eigenvectors of real sparse symmetric problem

— nagf.ht —

```
\begin{page}{manpageXXf02fjf}{NAG Documentation: f02fjf}
\beginscroll
\begin{verbatim}



     F02FJF(3NAG)      Foundation Library (12/10/92)      F02FJF(3NAG)



          F02 -- Eigenvalue and Eigenvectors                         F02FJF
                  F02FJF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          To find eigenvalues and eigenvectors of a real sparse symmetric
          or generalized symmetric eigenvalue problem.

          2. Specification

                 SUBROUTINE F02FJF (N, M, K, NOITS, TOL, DOT, IMAGE, MONIT,
                1                   NOVECS, X, NRX, D, WORK, LWORK, RWORK,
                2                   LRWORK, IWORK, LIWORK, IFAIL)
                 INTEGER          N, M, K, NOITS, NOVECS, NRX, LWORK,
                1                 LRWORK, IWORK(LIWORK), LIWORK, IFAIL
                 DOUBLE PRECISION TOL, DOT, X(NRX,K), D(K), WORK(LWORK),
                1                 RWORK(LRWORK)
                 EXTERNAL         DOT, IMAGE, MONIT

          3. Description

          F02FJF finds the m eigenvalues of largest absolute value and the
          corresponding eigenvectors for the real eigenvalue problem

                                  Cx=(lambda)x                           (1)

          where C is an n by n matrix such that

                                         T
                                     BC=C B                              (2)

          for a given positive-definite matrix B. C is said to be B-
          symmetric. Different specifications of C allow for the solution
          of a variety of eigenvalue problems. For example, when
```

$$C=A \text{ and } B=I \text{ where } A=A^T$$

the routine finds the m eigenvalues of largest absolute magnitude for the standard symmetric eigenvalue problem

$$Ax=(\lambda)x. \qquad (3)$$

The routine is intended for the case where A is sparse.

As a second example, when

$$C=B^{-1}A$$

where

$$A=A^T$$

the routine finds the m eigenvalues of largest absolute magnitude for the generalized symmetric eigenvalue problem

$$Ax=(\lambda)Bx. \qquad (4)$$

The routine is intended for the case where A and B are sparse.

The routine does not require C explicitly, but C is specified via a user-supplied routine IMAGE which, given an n element vector z, computes the image w given by

$$w=Cz.$$

For instance, in the above example, where $C=B^{-1}A$, routine IMAGE will need to solve the positive-definite system of equations Bw=Az for w.

To find the m eigenvalues of smallest absolute magnitude of (3) we can choose $C=A^{-1}$ and hence find the reciprocals of the required eigenvalues, so that IMAGE will need to solve Aw=z for w, and correspondingly for (4) we can choose $C=A^{-1}B$ and solve Aw=Bz for w.

A table of examples of choice of IMAGE is given in Table 3.1. It should be remembered that the routine also returns the corresponding eigenvectors and that B is positive-definite. Throughout A is assumed to be symmetric and, where necessary, non-singularity is also assumed.

Eigenvalues Required	Problem		
	Ax=(lambda)x	(B=I)Ax=(lambda)Bx	ABx=(lambda)x
Largest	Compute w=Az	Solve Bw=Az	Compute w=ABz
Smallest (Find 1/(lambda))	Solve Aw=z	Solve Aw=Bz	Solve Av=z, Bw=(nu)
Furthest from (sigma) (Find (lambda)-(sigma))	Compute w=(A-(sigma)I)z	Solve Bw=(A-(sigma)B)z	Compute w=(AB-(sigma)I)z
Closest to (sigma) (Find 1/((lambda)-(sigma)))	Solve (A-(sigma)I)w=z	Solve (A-(sigma)B)w=Bz	Solve (AB-(sigma)I)w=z

Table 3.1
The Requirement of IMAGE for Various Problems

The matrix B also need not be supplied explicitly, but is specified via a user-supplied routine DOT which, given n element vectors z and w, computes the generalized dot product $w^T Bz$.

F02FJF is based upon routine SIMITZ (see Nikolai [1]), which is itself a derivative of the Algol procedure ritzit (see Rutishauser [4]), and uses the method of simultaneous (subspace) iteration. (See Parlett [2] for description, analysis and advice on the use of the method.)

The routine performs simultaneous iteration on k>m vectors. Initial estimates to p<=k eigenvectors, corresponding to the p eigenvalues of C of largest absolute value, may be supplied by the user to F02FJF. When possible k should be chosen so that the kth eigenvalue is not too close to the m required eigenvalues, but if k is initially chosen too small then F02FJF may be re-entered, supplying approximations to the k eigenvectors found so far and with k then increased.

At each major iteration F02FJF solves an r by r (r<=k) eigenvalue sub-problem in order to obtain an approximation to the eigenvalues for which convergence has not yet occurred. This approximation is refined by Chebyshev acceleration.

4. References

[1] Nikolai P J (1979) Algorithm 538: Eigenvectors and eigenvalues of real generalized symmetric matrices by simultaneous iteration. ACM Trans. Math. Softw. 5 118--125.

[2] Parlett B N (1980) The Symmetric Eigenvalue Problem. Prentice-Hall.

[3] Rutishauser H (1969) Computational aspects of F L Bauer's simultaneous iteration method. Num. Math. 13 4--13.

[4] Rutishauser H (1970) Simultaneous iteration method for symmetric matrices. Num. Math. 16 205--223.

5. Parameters

1: N -- INTEGER Input
On entry: n, the order of the matrix C. Constraint: N >= 1.

2: M -- INTEGER Input/Output
On entry: m, the number of eigenvalues required.
'

Constraint: M >= 1. On exit: m, the number of eigenvalues actually found. It is equal to m if IFAIL = 0 on exit, and is less than m if IFAIL = 2, 3 or 4. See Section 6 and Section 8 for further information.

3: K -- INTEGER Input
On entry: the number of simultaneous iteration vectors to be used. Too small a value of K may inhibit convergence, while a larger value of K incurs additional storage and additional work per iteration. Suggested value: K = M + 4 will often be a reasonable choice in the absence of better information. Constraint: M < K <= N.

4: NOITS -- INTEGER Input/Output
On entry: the maximum number of major iterations (eigenvalue sub-problems) to be performed. If NOITS <= 0, then the value 100 is used in place of NOITS. On exit: the number of iterations actually performed.

5: TOL -- DOUBLE PRECISION Input
On entry: a relative tolerance to be used in accepting

eigenvalues and eigenvectors. If the eigenvalues are required to about t significant figures, then TOL should be set to about 10^{-t}. d_i is accepted as an eigenvalue as soon as two successive approximations to d_i differ by less than $(|\tilde{d}_i|*TOL)/10$, where $\tilde{d}_i$ is the latest aproximation to d_i. Once an eigenvalue has been accepted, then an eigenvector is accepted as soon as $(d_i f_i)/(d_i - d_k) < TOL$, where f_i is the normalised residual of the current approximation to the eigenvector (see Section 8 for further information). The values of the f_i and d_i can be printed from routine MONIT. If TOL is supplied outside the range ((epsilon), 1.0), where (epsilon) is the machine precision, then the value (epsilon) is used in place of TOL.

6: DOT -- DOUBLE PRECISION FUNCTION, supplied by the user. External Procedure

DOT must return the value $w^T Bz$ for given vectors w and z. For the standard eigenvalue problem, where B=I, DOT must return the dot product $w^T z$.

Its specification is:

```
       DOUBLE PRECISION FUNCTION DOT (IFLAG, N, Z, W,
      1                              RWORK, LRWORK,
      2                              IWORK, LIWORK)
       INTEGER          IFLAG, N, LRWORK, IWORK(LIWORK),
      1                 LIWORK
       DOUBLE PRECISION Z(N), W(N), RWORK(LRWORK)
```

1: IFLAG -- INTEGER Input/Output
On entry: IFLAG is always non-negative. On exit: IFLAG may be used as a flag to indicate a failure in the computation of $w^T Bz$. If IFLAG is negative on exit from DOT, then F02FJF will exit immediately with IFAIL set to IFLAG. Note that in this case DOT must still be assigned a value.

2: N -- INTEGER Input
On entry: the number of elements in the vectors z and w and the order of the matrix B.

3: Z(N) -- DOUBLE PRECISION array Input

On entry: the vector z for which $w^T Bz$ is required.

4: W(N) -- DOUBLE PRECISION array Input

On entry: the vector w for which $w^T Bz$ is required.

5: RWORK(LRWORK) -- DOUBLE PRECISION array User Workspace

6: LRWORK -- INTEGER Input

7: IWORK(LIWORK) -- INTEGER array User Workspace

8: LIWORK -- INTEGER Input

DOT is called from F02FJF with the parameters RWORK, LRWORK, IWORK and LIWORK as supplied to F02FJF. The user is free to use the arrays RWORK and IWORK to supply information to DOT and to IMAGE as an alternative to using COMMON.

DOT must be declared as EXTERNAL in the (sub)program from which F02FJF is called. Parameters denoted as Input must not be changed by this procedure.

7: IMAGE -- SUBROUTINE, supplied by the user. External Procedure

IMAGE must return the vector w=Cz for a given vector z.

Its specification is:

```
      SUBROUTINE IMAGE (IFLAG, N, Z, W, RWORK, LRWORK,
     1                   IWORK, LIWORK)
      INTEGER          IFLAG, N, LRWORK, IWORK(LIWORK),
     1                 LIWORK
      DOUBLE PRECISION Z(N), W(N), RWORK(LRWORK)
```

1: IFLAG -- INTEGER Input/Output

On entry: IFLAG is always non-negative. On exit: IFLAG may be used as a flag to indicate a failure in the computation of w. If IFLAG is negative on exit from IMAGE, then F02FJF will exit immediately with IFAIL set to IFLAG.

2: N -- INTEGER Input

On entry: n, the number of elements in the vectors w and z, and the order of the matrix C.

```
     3:  Z(N) -- DOUBLE PRECISION array                    Input
         On entry: the vector z for which Cz is required.

     4:  W(N) -- DOUBLE PRECISION array                   Output
         On exit: the vector w=Cz.

     5:  RWORK(LRWORK) -- DOUBLE PRECISION array  User Workspace

     6:  LRWORK -- INTEGER                                 Input

     7:  IWORK(LIWORK) -- INTEGER array           User Workspace

     8:  LIWORK -- INTEGER                                 Input
         IMAGE is called from F02FJF with the parameters RWORK,
         LRWORK, IWORK and LIWORK as supplied to F02FJF. The
         user is free to use the arrays RWORK and IWORK to
         supply information to IMAGE and DOT as an alternative
         to using COMMON.
     IMAGE must be declared as EXTERNAL in the (sub)program
     from which F02FJF is called. Parameters denoted as
     Input must not be changed by this procedure.

 8:  MONIT -- SUBROUTINE, supplied by the user.
                                              External Procedure
     MONIT is used to monitor the progress of F02FJF. MONIT may
     be the dummy subroutine F02FJZ if no monitoring is actually
     required. (F02FJZ is included in the NAG Foundation Library
     and so need not be supplied by the user. The routine name
     F02FJZ may be implementation dependent: see the Users' Note
     for your implementation for details.) MONIT is called after
     the solution of each eigenvalue sub-problem and also just
     prior to return from F02FJF. The parameters ISTATE and
     NEXTIT allow selective printing by MONIT.

     Its specification is:

            SUBROUTINE MONIT (ISTATE, NEXTIT, NEVALS,
           1                  NEVECS, K, F, D)
            INTEGER          ISTATE, NEXTIT, NEVALS, NEVECS,
           1                 K
            DOUBLE PRECISION F(K), D(K)

     1:  ISTATE -- INTEGER                                 Input
         On entry: ISTATE specifies the state of F02FJF and will
         have values as follows:
         ISTATE = 0
               No eigenvalue or eigenvector has just been
               accepted.
```

ISTATE = 1
One or more eigenvalues have been accepted since the last call to MONIT.

ISTATE = 2
One or more eigenvectors have been accepted since the last call to MONIT.

ISTATE = 3
One or more eigenvalues and eigenvectors have been accepted since the last call to MONIT.

ISTATE = 4
Return from F02FJF is about to occur.

2: NEXTIT -- INTEGER Input
On entry: the number of the next iteration.

3: NEVALS -- INTEGER Input
On entry: the number of eigenvalues accepted so far.

4: NEVECS -- INTEGER Input
On entry: the number of eigenvectors accepted so far.

5: K -- INTEGER Input
On entry: k, the number of simultaneous iteration vectors.

6: F(K) -- DOUBLE PRECISION array Input
On entry: a vector of error quantities measuring the state of convergence of the simultaneous iteration vectors. See the parameter TOL of F02FJF above and Section 8 for further details. Each element of F is initially set to the value 4.0 and an element remains at 4.0 until the corresponding vector is tested.

7: D(K) -- DOUBLE PRECISION array Input
On entry: D(i) contains the latest approximation to the absolute value of the ith eigenvalue of C.

MONIT must be declared as EXTERNAL in the (sub)program from which F02FJF is called. Parameters denoted as Input must not be changed by this procedure.

9: NOVECS -- INTEGER Input
On entry: the number of approximate vectors that are being supplied in X. If NOVECS is outside the range (0,K), then the value 0 is used in place of NOVECS.

10: X(NRX,K) -- DOUBLE PRECISION array Input/Output
On entry: if 0 < NOVECS <= K, the first NOVECS columns of X

```
     must contain approximations to the eigenvectors
     corresponding to the NOVECS eigenvalues of largest absolute
     value of C. Supplying approximate eigenvectors can be useful
     when reasonable approximations are known, or when the
     routine is being restarted with a larger value of K.
     Otherwise it is not necessary to supply approximate vectors,
     as simultaneous iteration vectors will be generated randomly
     by the routine. On exit: if IFAIL = 0, 2, 3 or 4, the first
     m' columns contain the eigenvectors corresponding to the
     eigenvalues returned in the first m' elements of D (see
     below); and the next k-m'-1 columns contain approximations
     to the eigenvectors corresponding to the approximate
     eigenvalues returned in the next k-m'-1 elements of D. Here
     m' is the value returned in M (see above), the number of
     eigenvalues actually found. The kth column is used as
     workspace.

11:  NRX -- INTEGER                                          Input
     On entry:
     the first dimension of the array X as declared in the
     (sub)program from which F02FJF is called.
     Constraint: NRX >= N.

12:  D(K) -- DOUBLE PRECISION array                         Output
     On exit: if IFAIL = 0, 2, 3 or 4, the first m' elements
     contain the first m' eigenvalues in decreasing order of
     magnitude; and the next k-m'-1 elements contain
     approximations to the next k-m'-1 eigenvalues. Here m' is
     the value returned in M (see above), the number of
     eigenvalues actually found. D(k) contains the value e where
     (-e,e) is the latest interval over which Chebyshev
     acceleration is performed.

13:  WORK(LWORK) -- DOUBLE PRECISION array              Workspace

14:  LWORK -- INTEGER                                        Input
     On entry: the length of the array WORK, as declared in the
     (sub)program from which F02FJF is called. Constraint:
     LWORK>=3*K+max(K*K,2*N).

15:  RWORK(LRWORK) -- DOUBLE PRECISION array       User Workspace
     RWORK is not used by F02FJF, but is passed directly to
     routines DOT and IMAGE and may be used to supply information
     to these routines.

16:  LRWORK -- INTEGER                                       Input
     On entry: the length of the array RWORK, as declared in the
     (sub)program from which F02FJF is called. Constraint: LRWORK
     >= 1.
```

```
17:  IWORK(LIWORK) -- INTEGER array                User Workspace
     IWORK is not used by F02FJF, but is passed directly to
     routines DOT and IMAGE and may be used to supply information
     to these routines.

18:  LIWORK -- INTEGER                                      Input
     On entry: the length of the array IWORK, as declared in the
     (sub)program from which F02FJF is called. Constraint: LIWORK
     >= 1.

19:  IFAIL -- INTEGER                                Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. Users who are
     unfamiliar with this parameter should refer to the Essential
     Introduction for details.

     On exit: IFAIL = 0 unless the routine detects an error or
     gives a warning (see Section 6).

     For this routine, because the values of output parameters
     may be useful even if IFAIL /=0 on exit, users are
     recommended to set IFAIL to -1 before entry. It is then
     essential to test the value of IFAIL on exit. To suppress
     the output of an error message when soft failure occurs, set
     IFAIL to 1.

6. Error Indicators and Warnings

Errors or warnings specified by the routine:


IFAIL< 0
     A negative value of IFAIL indicates an exit from F02FJF
     because the user has set IFLAG negative in DOT or IMAGE. The
     value of IFAIL will be the same as the user's setting of
     IFLAG.

IFAIL= 1
     On entry N < 1,

     or       M < 1,

     or       M >= K,

     or       K > N,

     or       NRX < N,

     or       LWORK <3*K+max(K*K*N),

     or       LRWORK < 1,
```

or LIWORK < 1.

IFAIL= 2

Not all the requested eigenvalues and vectors have been obtained. Approximations to the rth eigenvalue are oscillating rapidly indicating that severe cancellation is occurring in the rth eigenvector and so M is returned as (r-1). A restart with a larger value of K may permit convergence.

IFAIL= 3

Not all the requested eigenvalues and vectors have been obtained. The rate of convergence of the remaining eigenvectors suggests that more than NOITS iterations would be required and so the input value of M has been reduced. A restart with a larger value of K may permit convergence.

IFAIL= 4

Not all the requested eigenvalues and vectors have been obtained. NOITS iterations have been performed. A restart, possibly with a larger value of K, may permit convergence.

IFAIL= 5

This error is very unlikely to occur, but indicates that convergence of the eigenvalue sub-problem has not taken place. Restarting with a different set of approximate vectors may allow convergence. If this error occurs the user should check carefully that F02FJF is being called correctly.

7. Accuracy

Eigenvalues and eigenvectors will normally be computed to the accuracy requested by the parameter TOL, but eigenvectors corresponding to small or to close eigenvalues may not always be computed to the accuracy requested by the parameter TOL. Use of the routine MONIT to monitor acceptance of eigenvalues and eigenvectors is recommended.

8. Further Comments

The time taken by the routine will be principally determined by the time taken to solve the eigenvalue sub-problem and the time taken by the routines DOT and IMAGE. The time taken to solve an eigenvalue sub-problem is approximately proportional to nk^2. It is important to be aware that several calls to DOT and IMAGE may occur on each major iteration.

As can be seen from Table 3.1, many applications of F02FJF will require routine IMAGE to solve a system of linear equations. For example, to find the smallest eigenvalues of Ax=(lambda)Bx, IMAGE needs to solve equations of the form Aw=Bz for w and routines from Chapters F01 and F04 of the NAG Foundation Library will frequently be useful in this context. In particular, if A is a positive-definite variable band matrix, F04MCF may be used after A has been factorized by F01MCF. Thus factorization need be performed only once prior to calling F02FJF. An illustration of this type of use is given in the example program in Section 9.

An approximation $\tilde{d}_h$, to the ith eigenvalue, is accepted as soon as $\tilde{d}_h$ and the previous approximation differ by less than $|\tilde{d}_h|$*TOL/10. Eigenvectors are accepted in groups corresponding to clusters of eigenvalues that are equal, or nearly equal, in absolute value and that have already been accepted. If d_r is the last eigenvalue in such a group and we define the residual r_j as

$$r_j=Cx_j-y_r$$

where y_r is the projection of Cx_j, with respect to B, onto the space spanned by $x_1,x_2,...,x_r$ and x_j is the current approximation to the jth eigenvector, then the value f_i returned in MONIT is given by

$$f_i=\max||r_j||_B/||Cx_j||_B \qquad ||x||_B^2=x^TBx$$

and each vector in the group is accepted as an eigenvector if

$$(|d_r|f_r)/(|d_r|-e)<TOL$$

where e is the current approximation to $|\tilde{d}_k|$. The values of the f_i are systematically increased if the convergence criteria i

appear to be too strict. See Rutishauser [4] for further details.

The algorithm implemented by F02FJF differs slightly from SIMITZ (Nikolai [1]) in that the eigenvalue sub-problem is solved using the singular value decomposition of the upper triangular matrix R of the Gram-Schmidt factorization of Cx_r , rather than forming $R^T R$

9. Example

To find the four eigenvalues of smallest absolute value and corresponding eigenvectors for the generalized symmetric eigenvalue problem Ax=(lambda)Bx, where A and B are the 16 by 16 matrices

```
  (1 a     a                     )
  (a 1 a     a                   )
  (  a 1 a     a                 )
  (    a 1 a     a               )
  (a     a 1 a     a             )
  (  a     a 1 a     a           )
  (    a     a 1 a     a         )
  (      a     a 1 a     a       )
A=(        a     a 1 a     a     )
  (          a     a 1 a     a   )
  (            a     a 1 a     a )
  (              a     a 1 a     a)
  (                a     a 1 a   )
  (                  a     a 1 a )
  (                    a     a 1 a)
  (                      a     a 1)
```

where $a = -\frac{1}{4}$

```
  (1 b                            )
  (b 1 b                          )
  (  b 1 b                        )
  (    b 1 b                      )
  (      b 1 b                    )
  (        b 1 b                  )
  (          b 1 b                )
B=(            b 1 b              )
  (              b 1 b            )
  (                b 1 b          )
  (                  b 1 b        )
  (                    b 1 b      )
```

```
                    (                       b 1 b    )
                    (                         b 1 b  )
                    (                           b 1 b)

          1
where b=- -
          2

TOL is taken as 0.0001 and 6 iteration vectors are used. F01MAF
is used to factorize the matrix A, prior to calling F02FJF, and
F04MAF is used within IMAGE to solve the equations Aw=Bz for w.
Details of the factorization of A are passed from F01MAF to
F04MAF by means of the COMMON block BLOCK1.

Output from MONIT occurs each time ISTATE is non-zero. Note that
the required eigenvalues are the reciprocals of the eigenvalues
returned by F02FJF.

The example program is not reproduced here. The source code for
all example programs is distributed with the NAG Foundation
Library software and should be available on-line.
```

```
\end{verbatim}
\endscroll
\end{page}
```

———-

Singular value decomposition of a general real matrix

— nagf.ht —

```
\begin{page}{manpageXXf02wef}{NAG Documentation: f02wef}
\beginscroll
\begin{verbatim}



     F02WEF(3NAG)      Foundation Library (12/10/92)      F02WEF(3NAG)



          F02 -- Eigenvalue and Eigenvectors                         F02WEF
                  F02WEF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
```

The symbol (*) after a NAG routine name denotes a routine that is not included in the Foundation Library.

1. Purpose

F02WEF returns all, or part, of the singular value decomposition of a general real matrix.

2. Specification

```
       SUBROUTINE F02WEF (M, N, A, LDA, NCOLB, B, LDB, WANTQ, Q,
      1                   LDQ, SV, WANTP, PT, LDPT, WORK, IFAIL)
       INTEGER          M, N, LDA, NCOLB, LDB, LDQ, LDPT, IFAIL
       DOUBLE PRECISION A(LDA,*), B(LDB,*), Q(LDQ,*), SV(*), PT
      1                 (LDPT,*), WORK(*)
       LOGICAL          WANTQ, WANTP
```

3. Description

The m by n matrix A is factorized as

$$A=QDP^T,$$

where

$$D=\begin{pmatrix} S \\ 0 \end{pmatrix}, \qquad m>n,$$

$$D=S, \qquad m=n,$$

$$D=(S\ 0), \qquad m<n,$$

Q is an m by m orthogonal matrix, P is an n by n orthogonal matrix and S is a min(m,n) by min(m,n) diagonal matrix with non-negative diagonal elements, $sv_1,sv_2,...,sv_{\min(m,n)}$, ordered such that

$$sv_1>=sv_2>=...>=sv_{\min(m,n)}>=0.$$

The first min(m,n) columns of Q are the left-hand singular vectors of A, the diagonal elements of S are the singular values of A and the first min(m,n) columns of P are the right-hand singular vectors of A.

Either or both of the left-hand and right-hand singular vectors of A may be requested and the matrix C given by

$$C=Q^T B,$$

where B is an m by ncolb given matrix, may also be requested.

The routine obtains the singular value decomposition by first reducing A to upper triangular form by means of Householder transformations, from the left when m>=n and from the right when m<n. The upper triangular form is then reduced to bidiagonal form by Givens plane rotations and finally the QR algorithm is used to obtain the singular value decomposition of the bidiagonal form.

Good background descriptions to the singular value decomposition are given in Dongarra et al [1], Hammarling [2] and Wilkinson [3] DSVDC.

Note that if K is any orthogonal diagonal matrix so that

$$KK^T=I,$$

(so that K has elements +1 or -1 on the diagonal)

then

$$A=(QK)D(PK)^T$$

is also a singular value decomposition of A.

4. References

[1] Dongarra J J, Moler C B, Bunch J R and Stewart G W (1979) LINPACK Users' Guide. SIAM, Philadelphia.

[2] Hammarling S (1985) The Singular Value Decomposition in Multivariate Statistics. ACM Signum Newsletter. 20, 3 2--25.

[3] Wilkinson J H (1978) Singular Value Decomposition -- Basic Aspects. Numerical Software -- Needs and Availability. (ed D A H Jacobs) Academic Press.

5. Parameters

1: M -- INTEGER Input
On entry: the number of rows, m, of the matrix A.
Constraint: M >= 0.

When M = 0 then an immediate return is effected.

```
2:  N -- INTEGER                                          Input
    On entry: the number of columns, n, of the matrix A.
    Constraint: N >= 0.

    When N = 0 then an immediate return is effected.

3:  A(LDA,*) -- DOUBLE PRECISION array             Input/Output
    Note: the second dimension of the array A must be at least
    max(1,N).
    On entry: the leading m by n part of the array A must
    contain the matrix A whose singular value decomposition is
    required. On exit: if M >= N and WANTQ = .TRUE., then the
    leading m by n part of A will contain the first n columns of
    the orthogonal matrix Q.

    If M < N and WANTP = .TRUE., then the leading m by n part of
                                                              T
    A will contain the first m rows of the orthogonal matrix P .

    If M >= N and WANTQ = .FALSE. and WANTP = .TRUE., then the
    leading n by n part of A will contain the first n rows of
                            T
    the orthogonal matrix P .

    Otherwise the leading m by n part of A is used as internal
    workspace.

4:  LDA -- INTEGER                                        Input
    On entry:
    the first dimension of the array A as declared in the
    (sub)program from which F02WEF is called.
    Constraint: LDA >= max(1,M).

5:  NCOLB -- INTEGER                                      Input
    On entry: ncolb, the number of columns of the matrix B.
    When NCOLB = 0 the array B is not referenced. Constraint:
    NCOLB >= 0.

6:  B(LDB,*) -- DOUBLE PRECISION array             Input/Output
    Note: the second dimension of the array B must be at least
    max(1,ncolb) On entry: if NCOLB > 0, the leading m by ncolb
    part of the array B must contain the matrix to be
    transformed. On exit: B is overwritten by the m by ncolb
             T
    matrix  Q B.

7:  LDB -- INTEGER                                        Input
    On entry:
    the first dimension of the array B as declared in the
    (sub)program from which F02WEF is called.
```

```
     Constraint: if NCOLB > 0 then LDB >= max(1,M).

 8:  WANTQ -- LOGICAL                                        Input
     On entry: WANTQ must be .TRUE., if the left-hand singular
     vectors are required. If WANTQ = .FALSE., then the array Q
     is not referenced.

 9:  Q(LDQ,*) -- DOUBLE PRECISION array                     Output
     Note: the second dimension of the array Q must be at least
     max(1,M).
     On exit: if M < N and WANTQ = .TRUE., the leading m by m
     part of the array Q will contain the orthogonal matrix Q.
     Otherwise the array Q is not referenced.

10:  LDQ -- INTEGER                                          Input
     On entry:
     the first dimension of the array Q as declared in the
     (sub)program from which F02WEF is called.
     Constraint: if M < N and WANTQ = .TRUE., LDQ >= max(1,M).

11:  SV(*) -- DOUBLE PRECISION array                        Output
     Note: the length of SV must be at least min(M,N). On exit:
     the min(M,N) diagonal elements of the matrix S.

12:  WANTP -- LOGICAL                                        Input
     On entry: WANTP must be .TRUE. if the right-hand singular
     vectors are required. If WANTP = .FALSE., then the array PT
     is not referenced.

13:  PT(LDPT,*) -- DOUBLE PRECISION array                   Output
     Note: the second dimension of the array PT must be at least
     max(1,N).
     On exit: if M >= N and WANTQ and WANTP are .TRUE., the
     leading n by n part of the array PT will contain the
                         T
     orthogonal matrix P . Otherwise the array PT is not
     referenced.

14:  LDPT -- INTEGER                                         Input
     On entry:
     the first dimension of the array PT as declared in the
     (sub)program from which F02WEF is called.
     Constraint: if M >= N and WANTQ and WANTP are .TRUE., LDPT
     >= max(1,N).

15:  WORK(*) -- DOUBLE PRECISION array                      Output
     Note: the length of WORK must be at least max(1,lwork),
     where lwork must be as given in the following table:

     M >= N
```

```
WANTQ is .TRUE. and WANTP = .TRUE.
                 2
      lwork=max(N +5*(N-1),N+NCOLB,4)

WANTQ = .TRUE. and WANTP = .FALSE.
                 2
      lwork=max(N +4*(N-1),N+NCOLB,4)

WANTQ = .FALSE. and WANTP = .TRUE.
      lwork=max(3*(N-1),2) when NCOLB = 0

      lwork=max(5*(N-1),2) when NCOLB > 0

WANTQ = .FALSE. and WANTP = .FALSE.
      lwork=max(2*(N-1),2) when NCOLB = 0

      lwork=max(3*(N-1),2) when NCOLB > 0

M < N
WANTQ = .TRUE. and WANTP = .TRUE.
                 2
      lwork=max(M +5*(M-1),2)

WANTQ = .TRUE. and WANTP = .FALSE.
      lwork=max(3*(M-1),1)

WANTQ = .FALSE. and WANTP = .TRUE.
                 2
      lwork=max(M +3*(M-1),2) when NCOLB = 0

                 2
      lwork=max(M +5*(M-1),2) when NCOLB > 0

WANTQ = .FALSE. and WANTP = .FALSE.
      lwork=max(2*(M-1),1) when NCOLB = 0

      lwork=max(3*(M-1),1) when NCOLB > 0
On exit: WORK(min(M,N)) contains the total number of
iterations taken by the R algorithm.

The rest of the array is used as workspace.

16: IFAIL -- INTEGER                                Input/Output
    On entry: IFAIL must be set to 0, -1 or 1. For users not
    familiar with this parameter (described in the Essential
    Introduction) the recommended value is 0.

    On exit: IFAIL = 0 unless the routine detects an error (see
    Section 6).
```

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL=-1
One or more of the following conditions holds:

M < 0,

N < 0,

LDA < M,

NCOLB < 0,

LDB < M and NCOLB > 0,

LDQ < M and M < N and WANTQ = .TRUE.,

LDPT < N and M >= N and WANTQ = .TRUE., and WANTP = .TRUE..

IFAIL> 0
The QR algorithm has failed to converge in 50*min(m,n) iterations. In this case SV(1), SV(2),..., SV(IFAIL) may not have been found correctly and the remaining singular values may not be the smallest. The matrix A will nevertheless have been factorized as $A=QEP^T$, where the leading min(m,n) by min(m,n) part of E is a bidiagonal matrix with SV(1), SV(2), ..., SV(min(m,n)) as the diagonal elements and WORK(1), WORK(2),..., WORK(min(m,n)-1) as the super-diagonal elements.

This failure is not likely to occur.

7. Accuracy

The computed factors Q, D and P satisfy the relation

$$QDP^T=A+E,$$

where

$$||E||<=c(epsilon)||A||,$$

(epsilon) being the machine precision, c is a modest function of m and n and ||.|| denotes the spectral (two) norm. Note that

```
||A||=sv .
        1
```

8. Further Comments

Following the use of this routine the rank of A may be estimated by a call to the INTEGER FUNCTION F06KLF(*). The statement:

```
       IRANK = F06KLF(MIN(M, N), SV, 1, TOL)
```

returns the value (k-1) in IRANK, where k is the smallest integer for which SV(k)<tol*SV(1), where tol is the tolerance supplied in TOL, so that IRANK is an estimate of the rank of S and thus also of A. If TOL is supplied as negative then the machine precision is used in place of TOL.

9. Example

9.1. Example 1

To find the singular value decomposition of the 5 by 3 matrix

```
                          (2.0  2.5  2.5)
                          (2.0  2.5  2.5)
                        A=(1.6 -0.4  2.8)
                          (2.0 -0.5  0.5)
                          (1.2 -0.3 -2.9)
```

```
                          T
together with the vector Q b for the vector
```

```
                              ( 1.1)
                              ( 0.9)
                            b=( 0.6)
                              ( 0.0)
                              (-0.8)
```

The example program is not reproduced here. The source code for all example programs is distributed with the NAG Foundation Library software and should be available on-line.

9.2. Example 2

To find the singular value decomposition of the 3 by 5 matrix

```
                     (2.0 2.0  1.6  2.0  1.2)
                   A=(2.5 2.5 -0.4 -0.5 -0.3)
                     (2.5 2.5  2.8  0.5 -2.9)
```

The example program is not reproduced here. The source code for

```
         all example programs is distributed with the NAG Foundation
         Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

———-

Singular value decomposition of a general complex matrix

— nagf.ht —

```
\begin{page}{manpageXXf02xef}{NAG Documentation: f02xef}
\beginscroll
\begin{verbatim}



     F02XEF(3NAG)      Foundation Library (12/10/92)      F02XEF(3NAG)



          F02 -- Eigenvalue and Eigenvectors                         F02XEF
                  F02XEF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          F02XEF returns all, or part, of the singular value decomposition
          of a general complex matrix.

          2. Specification

                 SUBROUTINE F02XEF (M, N, A, LDA, NCOLB, B, LDB, WANTQ, Q,
                1                   LDQ, SV, WANTP, PH, LDPH, RWORK, CWORK,
                2                   IFAIL)
                 INTEGER                   M, N, LDA, NCOLB, LDB, LDQ, LDPH,
                1                          IFAIL
                 DOUBLE PRECISION          SV(*), RWORK(*)
                 COMPLEX(KIND=KIND(1.0D0)) A(LDA,*), B(LDB,*), Q(LDQ,*),
                1                          PH(LDPH,*), CWORK(*)
                 LOGICAL                   WANTQ, WANTP
```

3. Description

The m by n matrix A is factorized as

$$A=QDP^H,$$

where

$$D=\begin{pmatrix}S\\0\end{pmatrix} \qquad m>n,$$

$$D=S, \qquad m=n,$$

$$D=(S\ 0), \qquad m<n,$$

Q is an m by m unitary matrix, P is an n by n unitary matrix and S is a min(m,n) by min(m,n) diagonal matrix with real non-negative diagonal elements, $sv_1,sv_2,...,sv_{min(m,n)}$, ordered such that

$$sv_1>=sv_2>=...>=sv_{min(m,n)}>=0.$$

The first min(m,n) columns of Q are the left-hand singular vectors of A, the diagonal elements of S are the singular values of A and the first min(m,n) columns of P are the right-hand singular vectors of A.

Either or both of the left-hand and right-hand singular vectors of A may be requested and the matrix C given by

$$C=Q^H B,$$

where B is an m by ncolb given matrix, may also be requested.

The routine obtains the singular value decomposition by first reducing A to upper triangular form by means of Householder transformations, from the left when m>=n and from the right when m<n. The upper triangular form is then reduced to bidiagonal form by Givens plane rotations and finally the QR algorithm is used to obtain the singular value decomposition of the bidiagonal form.

Good background descriptions to the singular value decomposition are given in Dongarra et al [1], Hammarling [2] and Wilkinson [3] ZSVDC.

Note that if K is any unitary diagonal matrix so that

$$KK^H=I,$$

then

$$A=(QK)D(PK)^H$$

is also a singular value decomposition of A.

4. References

[1] Dongarra J J, Moler C B, Bunch J R and Stewart G W (1979) LINPACK Users' Guide. SIAM, Philadelphia.

[2] Hammarling S (1985) The Singular Value Decomposition in Multivariate Statistics. ACM Signum Newsletter. 20, 3 2--25.

[3] Wilkinson J H (1978) Singular Value Decomposition -- Basic Aspects. Numerical Software -- Needs and Availability. (ed D A H Jacobs) Academic Press.

5. Parameters

1: M -- INTEGER Input
On entry: the number of rows, m, of the matrix A.
Constraint: M >= 0.

When M = 0 then an immediate return is effected.

2: N -- INTEGER Input
On entry: the number of columns, n, of the matrix A.
Constraint: N >= 0.

When N = 0 then an immediate return is effected.

3: A(LDA,*) -- COMPLEX(KIND(1.0D)) array Input/Output
Note: the second dimension of the array A must be at least max(1,N).
On entry: the leading m by n part of the array A must contain the matrix A whose singular value decomposition is required. On exit: if M >= N and WANTQ = .TRUE., then the leading m by n part of A will contain the first n columns of the unitary matrix Q.
If M < N and WANTP = .TRUE., then the leading m by n part of A will contain the first m rows of the unitary matrix P^H.
will contain the first m rows of the unitary matrix P If M

>= N and WANTQ = .FALSE. and WANTP = .TRUE., then the leading n by n part of A will contain the first n rows of the unitary matrix P^H. Otherwise the leading m by n part of A is used as internal workspace.

4: LDA -- INTEGER Input
On entry:
the first dimension of the array A as declared in the (sub)program from which F02XEF is called.
Constraint: LDA >= max(1,M).

5: NCOLB -- INTEGER Input
On entry: ncolb, the number of columns of the matrix B. When NCOLB = 0 the array B is not referenced. Constraint: NCOLB >= 0.

6: B(LDB,*) -- COMPLEX(KIND(1.0D)) array Input/Output
Note: the second dimension of the array B must be at least max(1,NCOLB).
On entry: if NCOLB > 0, the leading m by ncolb part of the array B must contain the matrix to be transformed. On exit: B is overwritten by the m by ncolb matrix Q^H B.

7: LDB -- INTEGER Input
On entry:
the first dimension of the array B as declared in the (sub)program from which F02XEF is called.
Constraint: if NCOLB > 0, then LDB >= max(1,M).

8: WANTQ -- LOGICAL Input
On entry: WANTQ must be .TRUE. if the left-hand singular vectors are required. If WANTQ = .FALSE. then the array Q is not referenced.

9: Q(LDQ,*) -- COMPLEX(KIND(1.0D)) array Output
Note: the second dimension of the array Q must be at least max(1,M).
On exit: if M < N and WANTQ = .TRUE., the leading m by m part of the array Q will contain the unitary matrix Q. Otherwise the array Q is not referenced.

10: LDQ -- INTEGER Input
On entry:
the first dimension of the array Q as declared in the (sub)program from which F02XEF is called.
Constraint: if M < N and WANTQ = .TRUE., LDQ >= max(1,M).

11: SV(*) -- DOUBLE PRECISION array Output

Note: the length of SV must be at least min(M,N). On exit: the min(m,n) diagonal elements of the matrix S.

12: WANTP -- LOGICAL Input
On entry: WANTP must be .TRUE. if the right-hand singular vectors are required. If WANTP = .FALSE. then the array PH is not referenced.

13: PH(LDPH,*) -- DOUBLE PRECISION array Output
Note: the second dimension of the array PH must be at least max(1,N).
On exit: if M >= N and WANTQ and WANTP are .TRUE., the leading n by n part of the array PH will contain the unitary matrix P^H. Otherwise the array PH is not referenced.

14: LDPH -- INTEGER Input
On entry:
the first dimension of the array PH as declared in the (sub)program from which F02XEF is called.
Constraint: if M >= N and WANTQ and WANTP are .TRUE., LDPH >= max(1,N).

15: RWORK(*) -- DOUBLE PRECISION array Output
Note: the length of RWORK must be at least max(1,lrwork), where lrwork must satisfy:
lrwork=2*(min(M,N)-1) when
NCOLB = 0 and WANTQ and WANTP are .FALSE.,

lrwork=3*(min(M,N)-1) when
either NCOLB = 0 and WANTQ = .FALSE. and WANTP = .TRUE., or WANTP = .FALSE. and one or both of NCOLB > 0 and WANTQ = .TRUE.

lrwork=5*(min(M,N)-1)
otherwise.
On exit: RWORK(min(M,N)) contains the total number of iterations taken by the QR algorithm.

The rest of the array is used as workspace.

16: CWORK(*) -- COMPLEX(KIND(1.0D)) array Workspace
Note: the length of CWORK must be at least max(1,lcwork), where lcwork must satisfy:
lcwork=N+max(N^2,NCOLB) when
M >= N and WANTQ and WANTP are both .TRUE.

lcwork=N+max(N^2+N,NCOLB) when

M >= N and WANTQ = .TRUE., but WANTP = .FALSE.

lcwork=N+max(N,NCOLB) when
M >= N and WANTQ = .FALSE.

lcwork=M^2+M when
M < N and WANTP = .TRUE.

lcwork = M when
M < N and WANTP = .FALSE.

17: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL=-1
One or more of the following conditions holds:

M < 0,

N < 0,

LDA < M,

NCOLB < 0,

LDB < M and NCOLB > 0,

LDQ < M and M < N and WANTQ = .TRUE.,

LDPH < N and M >= N and WANTQ = .TRUE. and WANTP = .TRUE..

IFAIL> 0
The QR algorithm has failed to converge in 50*min(m,n) iterations. In this case SV(1), SV(2),..., SV(IFAIL) may not have been found correctly and the remaining singular values may not be the smallest. The matrix A will nevertheless have
H

been factorized as A=QEP where the leading min(m,n) by min(m,n) part of E is a bidiagonal matrix with SV(1), SV(2), ..., SV(min(m,n)) as the diagonal elements and RWORK(1), RWORK(2),..., RWORK(min(m,n)-1) as the super-diagonal elements.

This failure is not likely to occur.

7. Accuracy

The computed factors Q, D and P satisfy the relation

$$QDP^H = A+E,$$

where

$$||E|| \le c(\epsilon)||A||,$$

(epsilon) being the machine precision, c is a modest function of m and n and ||.|| denotes the spectral (two) norm. Note that $||A|| = sv_1$.

8. Further Comments

Following the use of this routine the rank of A may be estimated by a call to the INTEGER FUNCTION F06KLF(*). The statement:

```
        IRANK = F06KLF(MIN(M, N), SV, 1, TOL)
```

returns the value (k-1) in IRANK, where k is the smallest integer for which SV(k)<tol*SV(1), where tol is the tolerance supplied in TOL, so that IRANK is an estimate of the rank of S and thus also of A. If TOL is supplied as negative then the machine precision is used in place of TOL.

9. Example

9.1. Example 1

To find the singular value decomposition of the 5 by 3 matrix

```
                   (    0.5i -0.5+1.5i -1.0+1.0i)
                   (0.4+0.3i  0.9+1.3i  0.2+1.4i)
                 A=(0.4      -0.4+0.4i  1.8     )
                   (0.3-0.4i  0.1+0.7i  0.0     )
                   (   -0.3i  0.3+0.3i      2.4i)
```

```
                                H
together with the vector Q b for the vector

                           (-0.55+1.05i)
                           ( 0.49+0.93i)
                         b=( 0.56-0.16i)
                           ( 0.39+0.23i)
                           ( 1.13+0.83i)

The example program is not reproduced here. The source code for
all example programs is distributed with the NAG Foundation
Library software and should be available on-line.

9.2. Example 2

To find the singular value decompostition of the 3 by 5 matrix

         (     0.5i 0.4-0.3i  0.4      0.3+0.4i     0.3i)
       A=(-0.5-1.5i 0.9-1.3i -0.4-0.4i 0.1-0.7i 0.3-0.3i)
         (-1.0-1.0i 0.2-1.4i  1.8      0.0         -2.4i)

The example program is not reproduced here. The source code for
all example programs is distributed with the NAG Foundation
Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

———-

Simultaneous Linear Equations

— nagf.ht —

```
\begin{page}{manpageXXf04}{NAG Documentation: f04}
\beginscroll
\begin{verbatim}



     F04(3NAG)         Foundation Library (12/10/92)         F04(3NAG)



          F04 -- Simultaneous Linear Equations          Introduction -- F04
                                     Chapter F04
```

Simultaneous Linear Equations

1. Scope of the Chapter

This chapter, together with two routines in Chapter F07, is concerned with the solution of the matrix equation AX=B, where B may be a single vector or a matrix of multiple right-hand sides. The matrix A may be real, complex, symmetric, Hermitian positive-definite, or sparse. It may also be rectangular, in which case a least-squares solution is obtained.

2. Background to the Problems

A set of linear equations may be written in the form

$$Ax=b$$

where the known matrix A, with real or complex coefficients, is of size m by n, (m rows and n columns), the known right-hand vector b has m components (m rows and one column), and the required solution vector x has n components (n rows and one column). There may sometimes be p vectors b_i, i=1,2,...,p on the right-hand side and the equations may then be written as

$$AX=B$$

the required matrix X having as its p columns the solutions of $Ax_i=b_i$, i=1,2,...,p. Some routines deal with the latter case, but for clarity only the case p=1 is discussed here.

The most common problem, the determination of the unique solution of Ax=b, occurs when m=n and A is non-singular, that is rank(A)=n problem, discussed in Section 2.2 below, is the determination of the least-squares solution of Ax~=b, i.e., the determination of a vector x which minimizes the Euclidean length (two norm) of the residual vector r=b-Ax. The usual case has m>n and rank(A)=n, in which case x is unique.

2.1. Unique Solution of Ax=b

Most of the routines in this chapter, as well as two routines in Chapter F07, solve this particular problem. The solution is obtained by performing either an LU factorization, or a Cholesky factorization, as discussed in Section 2 of the F01 Chapter Introduction.

Two of the routines in this chapter use a process called iterative refinement to improve the initial solution in order to obtain a solution that is correct to working accuracy. It should be emphasised that if A and b are not known exactly then not all the figures in this solution may be meaningful. To be more precise, if x is the exact solution of the equations

$$Ax=b$$

and x is the solution of the perturbed equations

$$(A+E)x=b+e,$$

then, provided that $\kappa(A) \frac{\|E\|}{\|A\|} \le 1$,

$$\frac{\|x-x\|}{\|x\|} \le \frac{\kappa(A)}{1-\kappa(A)\frac{\|E\|}{\|A\|}}\left(\frac{\|E\|}{\|A\|} + \frac{\|e\|}{\|b\|} \right),$$

where $\kappa(A)=\|A\|\|A^{-1}\|$ is the condition number of A with respect to inversion. Thus, if A is ill-conditioned ($\kappa(A)$ is large), x may differ significantly from x. Often $\kappa(A) \frac{\|E\|}{\|A\|} \ll 1$ in which case the above bound effectively reduces to

$$\frac{\|x-x\|}{\|x\|} \le \kappa(A)\left(\frac{\|E\|}{\|A\|} + \frac{\|e\|}{\|b\|} \right).$$

2.2. The Least-squares Solution of Ax~=b

The least-squares problem is to find a vector x to minimize

$$r^T r, \quad \text{where } r=b-Ax.$$

When m>=n and rank(A)=n then the solution vector x is unique. For the cases where x is not unique the routines in this chapter obtain the minimal length solution, that is the vector x for which $x^T x$ is a minimum.

2.3. Calculating the Inverse of a Matrix

The routines in this chapter can also be used to calculate the inverse of a square matrix A by solving the equation

$$AX=I,$$

where I is the identity matrix.

3. Recommendations on Choice and Use of Routines

3.1. General Purpose Routines

Many of the routines in this chapter perform the complete solution of the required equations, but some of the routines, as well as the routines in Chapter F07, assume that a prior factorization has been performed, using the appropriate factorization routine from Chapter F01 or Chapter F07. These, so-called, general purpose routines can be useful when explicit information on the factorization is required, as well as the solution of the equations, or when the solution is required for multiple right-hand sides, or for a sequence of right-hand sides.

Note that some of the routines that perform a complete solution also allow multiple right-hand sides.

3.2. Iterative Refinement

The routines that perform iterative refinement are more costly than those that do not perform iterative refinement, both in terms of time and storage, and should only be used if the problem really warrants the additional accuracy provided by these routines. The storage requirements are approximately doubled, while the additional time is not usually prohibitive since the initial factorization is used at each iteration.

3.3. Sparse Matrix Routines

The routines for sparse matrices should usually be used only when the number of non-zero elements is very small, less than 10% of the total number of elements of A. Additionally, when the matrix is symmetric positive-definite the sparse routines should generally be used only when A does not have a (variable) band structure.

There are four routines for solving sparse linear equations, two for solving general real systems (F04AXF and F04QAF), one for solving symmetric positive-definite systems (F04MAF) and one for solving symmetric systems that may, or may not, be positive-definite (F04MBF). F04AXF and F04MAF utilise factorizations of the matrix A obtained by routines in Chapter F01, while the other two routines use iterative techniques and require a user-supplied function to compute matrix-vector products Ac and $A^T c$ for any given vector c. The routines requiring factorizations will usually be faster and the factorization can be utilised to solve for several right-hand sides, but the original matrix has to be explicitly supplied and is overwritten by the factorization, and the storage requirements will usually be substantially more than those of the iterative routines.

Routines F04MBF and F04QAF both allow the user to supply a pre-conditioner.

F04MBF can be used to solve systems of the form (A-(lambda)I)x=b, which can be useful in applications such as Rayleigh quotient iteration.

F04QAF also solves sparse least-squares problems and allows the solution of damped (regularized) least-squares problems.

3.4. Decision Trees

If at any stage the answer to a question is 'Don't know' this should be read as 'No'.

For those routines that need to be preceded by a factorization routine, the appropriate routine name is given in brackets after the name of the routine for solving the equations. Note also that you may be directed to a routine in Chapter F07.

3.4.1. Routines for unique solution of Ax=b

Please see figure in printed Reference Manual

3.4.2. Routines for Least-squares problems

```
                  Please see figure in printed Reference Manual



          F04 -- Simultaneous Linear Equations            Contents -- F04
          Chapter F04

          Eigenvalues and Eigenvectors

          F04ADF  Approximate solution of complex simultaneous linear
                  equations with multiple right-hand sides

          F04ARF  Approximate solution of real simultaneous linear
                  equations, one right-hand side

          F04ASF  Accurate solution of real symmetric positive-definite
                  simultaneous linear equations, one right-hand side

          F04ATF  Accurate solution of real simultaneous linear equations,
                  one right-hand side

          F04AXF  Approximate solution of real sparse simultaneous linear
                  equations (coefficient matrix already factorized by
                  F01BRF or F01BSF)

          F04FAF  Approximate solution of real symmetric positive-definite
                  tridiagonal simultaneous linear equations, one right-hand
                  side

          F04JGF  Least-squares (if rank = n) or minimal least-squares (if
                  rank <n) solution of m real equations in n unknowns, rank
                  <=n, m>=n

          F04MAF  Real sparse symmetric positive-definite simultaneous
                  linear equations (coefficient matrix already factorized)

          F04MBF  Real sparse symmetric simultaneous linear equations

          F04MCF  Approximate solution of real symmetric positive-definite
                  variable-bandwidth simultaneous linear equations
                  (coefficient matrix already factorized)

          F04QAF  Sparse linear least-squares problem, m real equations in
                  n unknowns

\end{verbatim}
\endscroll
```

```
\end{page}
```

———

Approximate solution of a set of complex linear equations

— nagf.ht —

```
\begin{page}{manpageXXf04adf}{NAG Documentation: f04adf}
\beginscroll
\begin{verbatim}



     F04ADF(3NAG)      Foundation Library (12/10/92)      F04ADF(3NAG)



          F04 -- Simultaneous Linear Equations                      F04ADF
                  F04ADF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          F04ADF calculates the approximate solution of a set of complex
          linear equations with multiple right-hand sides, using an LU
          factorization with partial pivoting.

          2. Specification

                 SUBROUTINE F04ADF (A, IA, B, IB, N, M, C, IC, WKSPCE,
                1                   IFAIL)
                 INTEGER              IA, IB, N, M, IC, IFAIL
                 DOUBLE PRECISION     WKSPCE(*)
                 COMPLEX(KIND(1.0D0)) A(IA,*), B(IB,*), C(IC,*)

          3. Description

          Given a set of complex linear equations AX=B, the routine first
          computes an LU factorization of A with partial pivoting, PA=LU,
          where P is a permutation matrix, L is lower triangular and U is
          unit upper triangular. The columns x of the solution X are found
          by forward and backward substitution in Ly=Pb and Ux=y, where b
```

is a column of the right-hand side matrix B.

4. References

[1] Wilkinson J H and Reinsch C (1971) Handbook for Automatic Computation II, Linear Algebra. Springer-Verlag.

5. Parameters

```
 1:  A(IA,*) -- COMPLEX(KIND(1.0D)) array            Input/Output
     Note: the second dimension of the array A must be at least
     max(1,N).
     On entry: the n by n matrix A. On exit: A is overwritten by
     the lower triangular matrix L and the off-diagonal elements
     of the upper triangular matrix U. The unit diagonal elements
     of U are not stored.

 2:  IA -- INTEGER                                          Input
     On entry:
     the first dimension of the array A as declared in the
     (sub)program from which F04ADF is called.
     Constraint: IA >= max(1,N).

 3:  B(IB,*) -- COMPLEX(KIND(1.0D)) array                   Input
     Note: the second dimension of the array B must be at least
     max(1,M).
     On entry: the n by m right-hand side matrix B. See also
     Section 8.

 4:  IB -- INTEGER                                          Input
     On entry:
     the first dimension of the array B as declared in the
     (sub)program from which F04ADF is called.
     Constraint: IB >= max(1,N).

 5:  N -- INTEGER                                           Input
     On entry: n, the order of the matrix A. Constraint: N >= 0.

 6:  M -- INTEGER                                           Input
     On entry: m, the number of right-hand sides. Constraint: M
     >= 0.

 7:  C(IC,*) -- COMPLEX(KIND(1.0D)) array                  Output
     Note: the second dimension of the array C must be at least
     max(1,M).
     On exit: the n by m solution matrix X. See also Section 8.

 8:  IC -- INTEGER                                          Input
     On entry:
     the first dimension of the array C as declared in the
```

(sub)program from which F04ADF is called.
Constraint: IC >= max(1,N).

9: WKSPCE(*) -- DOUBLE PRECISION array Workspace
Note: the dimension of the array WKSPCE must be at least max(1,N).

10: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL= 1
The matrix A is singular, possibly due to rounding errors.

IFAIL= 2
On entry N < 0,

or M < 0,

or IA < max(1,N),

or IB < max(1,N),

or IC < max(1,N).

7. Accuracy

The accuracy of the computed solution depends on the conditioning of the original matrix. For a detailed error analysis see Wilkinson and Reinsch [1] page 106.

8. Further Comments

The time taken by the routine is approximately proportional to n^3

Unless otherwise stated in the Users' Note for your implementation, the routine may be called with the same actual array supplied for parameters B and C, in which case the solution

```
          vectors will overwrite the right-hand sides. However this is not
          standard Fortran 77, and may not work on all systems.

          9. Example

          To solve the set of linear equations AX=B where

                                   (1   1+2i  2+10i)
                                 A=(1+i  3i -5+14i)
                                   (1+i  5i -8+20i)

          and


                                         (1)
                                       B=(0).
                                         (0)

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Approximate solution of a set of real linear equations

— nagf.ht —

```
\begin{page}{manpageXXf04arf}{NAG Documentation: f04arf}
\beginscroll
\begin{verbatim}



     F04ARF(3NAG)      Foundation Library (12/10/92)      F04ARF(3NAG)



          F04 -- Simultaneous Linear Equations                      F04ARF
                  F04ARF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
```

The symbol (*) after a NAG routine name denotes a routine that is not included in the Foundation Library.

1. Purpose

F04ARF calculates the approximate solution of a set of real linear equations with a single right-hand side, using an LU factorization with partial pivoting.

2. Specification

```
SUBROUTINE F04ARF (A, IA, B, N, C, WKSPCE, IFAIL)
INTEGER          IA, N, IFAIL
DOUBLE PRECISION A(IA,*), B(*), C(*), WKSPCE(*)
```

3. Description

Given a set of linear equations, Ax=b, the routine first computes an LU factorization of A with partial pivoting, PA=LU, where P is a permutation matrix, L is lower triangular and U is unit upper triangular. The approximate solution x is found by forward and backward substitution in Ly=Pb and Ux=y, where b is the right-hand side.

4. References

[1] Wilkinson J H and Reinsch C (1971) Handbook for Automatic Computation II, Linear Algebra. Springer-Verlag.

5. Parameters

1: A(IA,*) -- DOUBLE PRECISION array Input/Output
Note: the second dimension of the array A must be at least max(1,N).
On entry: the n by n matrix A. On exit: A is overwritten by the lower triangular matrix L and the off-diagonal elements of the upper triangular matrix U. The unit diagonal elements of U are not stored.

2: IA -- INTEGER Input
On entry:
the first dimension of the array A as declared in the (sub)program from which F04ARF is called.
Constraint: IA >= max(1,N).

3: B(*) -- DOUBLE PRECISION array Input
Note: the dimension of the array B must be at least max(1,N).
On entry: the right-hand side vector b.

4: N -- INTEGER Input
On entry: n, the order of the matrix A. Constraint: N >= 0.

5: C(*) -- DOUBLE PRECISION array Output
Note: the dimension of the array C must be at least max(1,N).
On exit: the solution vector x.

6: WKSPCE(*) -- DOUBLE PRECISION array Workspace
Note: the dimension of the array WKSPCE must be at least max(1,N).

7: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL= 1
The matrix A is singular, possibly due to rounding errors.

IFAIL= 2
On entry N < 0,

or IA < max(1,N).

7. Accuracy

The accuracy of the computed solution depends on the conditioning of the original matrix. For a detailed error analysis see Wilkinson and Reinsch [1] page 107.

8. Further Comments

The time taken by the routine is approximately proportional to n^3

Unless otherwise stated in the Users' Note for your implementation, the routine may be called with the same actual array supplied for parameters B and C, in which case the solution vector will overwrite the right-hand side. However this is not

```
         standard Fortran 77, and may not work on all systems.

         9. Example

         To solve the set of linear equations Ax=b where

                                    ( 33  16  72)
                                  A=(-24 -10 -57)
                                    ( -8  -4 -17)

         and


                                       (-359)
                                     b=( 281).
                                       (  85)

         The example program is not reproduced here. The source code for
         all example programs is distributed with the NAG Foundation
         Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Real symmetric positive-definite linear equations

— nagf.ht —

```
\begin{page}{manpageXXf04asf}{NAG Documentation: f04asf}
\beginscroll
\begin{verbatim}



     F04ASF(3NAG)      Foundation Library (12/10/92)      F04ASF(3NAG)



         F04 -- Simultaneous Linear Equations                      F04ASF
                 F04ASF -- NAG Foundation Library Routine Document

         Note: Before using this routine, please read the Users' Note for
         your implementation to check implementation-dependent details.
         The symbol (*) after a NAG routine name denotes a routine that is
```

not included in the Foundation Library.

1. Purpose

F04ASF calculates the accurate solution of a set of real symmetric positive-definite linear equations with a single right-hand side, Ax=b, using a Cholesky factorization and iterative refinement.

2. Specification

```
      SUBROUTINE F04ASF (A, IA, B, N, C, WK1, WK2, IFAIL)
      INTEGER          IA, N, IFAIL
      DOUBLE PRECISION A(IA,*), B(*), C(*), WK1(*), WK2(*)
```

3. Description

Given a set of real linear equations Ax=b, where A is a symmetric positive-definite matrix, the routine first computes a Cholesky factorization of A as $A=LL^T$ where L is lower triangular. An approximation to x is found by forward and backward substitution. The residual vector r=b-Ax is then calculated using additional precision and a correction d to x is found by solving $LL^T d=r$. x is then replaced by x+d, and this iterative refinement of the solution is repeated until machine accuracy is obtained.

4. References

[1] Wilkinson J H and Reinsch C (1971) Handbook for Automatic Computation II, Linear Algebra. Springer-Verlag.

5. Parameters

1: A(IA,*) -- DOUBLE PRECISION array Input/Output
Note: the second dimension of the array A must be at least max(1,N).
On entry: the upper triangle of the n by n positive-definite symmetric matrix A. The elements of the array below the diagonal need not be set. On exit: the elements of the array below the diagonal are overwritten; the upper triangle of A is unchanged.

2: IA -- INTEGER Input
On entry:
the first dimension of the array A as declared in the (sub)program from which F04ASF is called.
Constraint: IA >= max(1,N).

```
 3:  B(*) -- DOUBLE PRECISION array                         Input
     Note: the dimension of the array B must be at least
     max(1,N).
     On entry: the right-hand side vector b.

 4:  N -- INTEGER                                           Input
     On entry: n, the order of the matrix A. Constraint: N >= 0.

 5:  C(*) -- DOUBLE PRECISION array                        Output
     Note: the dimension of the array C must be at least
     max(1,N).
     On exit: the solution vector x.

 6:  WK1(*) -- DOUBLE PRECISION array                   Workspace
     Note: the dimension of the array WK1 must be at least
     max(1,N).

 7:  WK2(*) -- DOUBLE PRECISION array                   Workspace
     Note: the dimension of the array WK2 must be at least
     max(1,N).

 8:  IFAIL -- INTEGER                                Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

IFAIL= 1
     The matrix A is not positive-definite, possibly due to
     rounding errors.

IFAIL= 2
     Iterative refinement fails to improve the solution, i.e.,
     the matrix A is too ill-conditioned.

IFAIL= 3
     On entry N < 0,

     or       IA < max(1,N).

7. Accuracy
```

```
          The computed solutions should be correct to full machine
          accuracy. For a detailed error analysis see Wilkinson and Reinsch
          [1] page 39.

          8. Further Comments

                                                                      3
          The time taken by the routine is approximately proportional to n

          The routine must not be called with the same name for parameters
          B and C.

          9. Example

          To solve the set of linear equations Ax=b where

                                     (5  7  6  5)
                                     (7 10  8  7)
                                   A=(6  8 10  9)
                                     (5  7  9 10)

          and


                                         (23)
                                         (32)
                                       b=(33).
                                         (31)

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Set of real linear equations with a single right-hand side

— nagf.ht —

```
\begin{page}{manpageXXf04atf}{NAG Documentation: f04atf}
\beginscroll
\begin{verbatim}
```

F04ATF(3NAG)   Foundation Library (12/10/92)   F04ATF(3NAG)

F04 -- Simultaneous Linear Equations F04ATF
F04ATF -- NAG Foundation Library Routine Document

Note: Before using this routine, please read the Users' Note for your implementation to check implementation-dependent details. The symbol (*) after a NAG routine name denotes a routine that is not included in the Foundation Library.

1. Purpose

F04ATF calculates the accurate solution of a set of real linear equations with a single right-hand side, using an LU factorization with partial pivoting, and iterative refinement.

2. Specification

```
       SUBROUTINE F04ATF (A, IA, B, N, C, AA, IAA, WKS1, WKS2,
      1                   IFAIL)
       INTEGER          IA, N, IAA, IFAIL
       DOUBLE PRECISION A(IA,*), B(*), C(*), AA(IAA,*), WKS1(*),
      1                 WKS2(*)
```

3. Description

Given a set of real linear equations, Ax=b, the routine first computes an LU factorization of A with partial pivoting, PA=LU, where P is a permutation matrix, L is lower triangular and U is unit upper triangular. An approximation to x is found by forward and backward substitution in Ly=Pb and Ux=y. The residual vector r=b-Ax is then calculated using additional precision, and a correction d to x is found by solving LUd=r. x is replaced by x+d , and this iterative refinement of the solution is repeated until full machine accuracy is obtained.

4. References

[1] Wilkinson J H and Reinsch C (1971) Handbook for Automatic Computation II, Linear Algebra. Springer-Verlag.

5. Parameters

1: A(IA,*) -- DOUBLE PRECISION array Input
Note: the second dimension of the array A must be at least

```
     max(1,N).
     On entry: the n by n matrix A.

 2:  IA -- INTEGER                                        Input
     On entry:
     the first dimension of the array A as declared in the
     (sub)program from which F04ATF is called.
     Constraint: IA >= max(1,N).

 3:  B(*) -- DOUBLE PRECISION array                       Input
     Note: the dimension of the array B must be at least
     max(1,N).
     On entry: the right-hand side vector b.

 4:  N -- INTEGER                                         Input
     On entry: n, the order of the matrix A. Constraint: N >= 0.

 5:  C(*) -- DOUBLE PRECISION array                      Output
     Note: the dimension of the array C must be at least
     max(1,N).
     On exit: the solution vector x.

 6:  AA(IAA,*) -- DOUBLE PRECISION array                 Output
     Note: the second dimension of the array AA must be at least
     max(1,N).
     On exit: the triangular factors L and U, except that the
     unit diagonal elements of U are not stored.

 7:  IAA -- INTEGER                                       Input
     On entry:
     the first dimension of the array AA as declared in the
     (sub)program from which F04ATF is called.
     Constraint: IAA >= max(1,N).

 8:  WKS1(*) -- DOUBLE PRECISION array                Workspace
     Note: the dimension of the array WKS1 must be at least
     max(1,N).

 9:  WKS2(*) -- DOUBLE PRECISION array                Workspace
     Note: the dimension of the array WKS2 must be at least
     max(1,N).

10:  IFAIL -- INTEGER                              Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).
```

```
6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

IFAIL= 1
     The matrix A is singular, possibly due to rounding errors.

IFAIL= 2

     Iterative refinement fails to improve the solution, i.e.,
     the matrix A is too ill-conditioned.

IFAIL= 3
     On entry N < 0,

     or       IA < max(1,N),

     or       IAA < max(1,N).

7. Accuracy

The computed solutions should be correct to full machine
accuracy. For a detailed error analysis see Wilkinson and Reinsch
[1] page 107.

8. Further Comments

                                                                3
The time taken by the routine is approximately proportional to n

The routine must not be called with the same name for parameters
B and C.

9. Example

To solve the set of linear equations Ax=b where

                           ( 33  16  72)
                         A=(-24 -10 -57)
                           ( -8  -4 -17)

and


                              (-359)
                            b=( 281).
                              (  85)
```

```
        The example program is not reproduced here. The source code for
        all example programs is distributed with the NAG Foundation
        Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

———-

Solution of a set of real sparse linear equations

— nagf.ht —

```
\begin{page}{manpageXXf04axf}{NAG Documentation: f04axf}
\beginscroll
\begin{verbatim}



     F04AXF(3NAG)      Foundation Library (12/10/92)      F04AXF(3NAG)



        F04 -- Simultaneous Linear Equations                      F04AXF
                F04AXF -- NAG Foundation Library Routine Document

        Note: Before using this routine, please read the Users' Note for
        your implementation to check implementation-dependent details.
        The symbol (*) after a NAG routine name denotes a routine that is
        not included in the Foundation Library.

        1. Purpose

        F04AXF calculates the approximate solution of a set of real
        sparse linear equations with a single right-hand side, Ax=b or
         T
        A x=b, where A has been factorized by F01BRF or F01BSF.

        2. Specification

               SUBROUTINE F04AXF (N, A, LICN, ICN, IKEEP, RHS, W, MTYPE,
              1                   IDISP, RESID)
               INTEGER          N, LICN, ICN(LICN), IKEEP(5*N), MTYPE,
              1                 IDISP(2)
               DOUBLE PRECISION A(LICN), RHS(N), W(N), RESID
```

3. Description

To solve a system of real linear equations $Ax=b$ or $A^T x=b$, where A is a general sparse matrix, A must first be factorized by F01BRF or F01BSF. F04AXF then computes x by block forward or backward substitution using simple forward and backward substitution within each diagonal block.

The method is fully described in Duff [1].

4. References

[1] Duff I S (1977) MA28 -- a set of Fortran subroutines for sparse unsymmetric linear equations. A.E.R.E. Report R.8730. HMSO.

5. Parameters

1: N -- INTEGER Input
On entry: n, the order of the matrix A.

2: A(LICN) -- DOUBLE PRECISION array Input
On entry: the non-zero elements in the factorization of the matrix A, as returned by F01BRF or F01BSF.

3: LICN -- INTEGER Input
On entry:
the dimension of the arrays A and ICN as declared in the (sub)program from which F04AXF is called.

4: ICN(LICN) -- INTEGER array Input
On entry: the column indices of the non-zero elements of the factorization, as returned by F01BRF or F01BSF.

5: IKEEP(5*N) -- INTEGER array Input
On entry: the indexing information about the factorization, as returned by F01BRF or F01BSF.

6: RHS(N) -- DOUBLE PRECISION array Input/Output
On entry: the right-hand side vector b. On exit: RHS is overwritten by the solution vector x.

7: W(N) -- DOUBLE PRECISION array Workspace

8: MTYPE -- INTEGER Input
On entry: MTYPE specifies the task to be performed:
if MTYPE = 1, solve Ax=b,

if MTYPE /= 1, solve $A^T x=b$.

9: IDISP(2) -- INTEGER array Input
On entry: the values returned in IDISP by F01BRF.

10: RESID -- DOUBLE PRECISION Output
On exit: the value of the maximum residual,

$\max(|b_i - \sum_j a_{ij} x_j|)$, over all the unsatisfied equations, in case F01BRF or F01BSF has been used to factorize a singular or rectangular matrix.

6. Error Indicators and Warnings

None.

7. Accuracy

The accuracy of the computed solution depends on the conditioning of the original matrix. Since F04AXF is always used with either F01BRF or F01BSF, the user is recommended to set GROW = .TRUE. on entry to these routines and to examine the value of W(1) on exit (see the routine documents for F01BRF and F01BSF). For a detailed error analysis see Duff [1] page 17.

If storage for the original matrix is available then the error can be estimated by calculating the residual

$$r=b-Ax \quad (\text{or } b-A^T x)$$

and calling F04AXF again to find a correction (delta) for x by solving

$$A(\delta)=r \quad (\text{or } A^T(\delta)=r).$$

8. Further Comments

If the factorized form contains (tau) non-zeros (IDISP(2) = (tau)) then the time taken is very approximately 2(tau) times longer than the inner loop of full matrix code. Some advantage is taken of zeros in the right-hand side when solving $A^T x=b$ (MTYPE /= 1).

9. Example

```
          To solve the set of linear equations Ax=b where

                                ( 5  0  0  0 0  0)
                                ( 0  2 -1  2 0  0)
                                ( 0  0  3  0 0  0)
                              A=(-2  0  0  1 1  0)
                                (-1  0  0 -1 2 -3)
                                (-1 -1  0  0 0  6)

          and


                                       (15)
                                       (12)
                                       (18)
                                     b=( 3).
                                       (-6)
                                       ( 0)

          The non-zero elements of A and indexing information are read in
          by the program, as described in the document for F01BRF.

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Real symmetric positive-definite tridiagonal linear equations

— nagf.ht —

```
\begin{page}{manpageXXf04faf}{NAG Documentation: f04faf}
\beginscroll
\begin{verbatim}



     F04FAF(3NAG)      Foundation Library (12/10/92)      F04FAF(3NAG)



          F04 -- Simultaneous Linear Equations                       F04FAF
```

F04FAF -- NAG Foundation Library Routine Document

Note: Before using this routine, please read the Users' Note for your implementation to check implementation-dependent details. The symbol (*) after a NAG routine name denotes a routine that is not included in the Foundation Library.

1. Purpose

F04FAF calculates the approximate solution of a set of real symmetric positive-definite tridiagonal linear equations.

2. Specification

```
      SUBROUTINE F04FAF (JOB, N, D, E, B, IFAIL)
      INTEGER          JOB, N, IFAIL
      DOUBLE PRECISION D(N), E(N), B(N)
```

3. Description

F04FAF is based upon the Linpack routine DPTSL (see Dongarra et al [1]) and solves the equations

$$Tx=b,$$

where T is a real n by n symmetric positive-definite tridiagonal matrix, using a modified symmetric Gaussian elimination algorithm to factorize T as $T=MKM^T$, where K is diagonal and M is a matrix of multipliers as described in Section 8.

When the input parameter JOB is supplied as 1, then the routine assumes that a previous call to F04FAF has already factorized T; otherwise JOB must be supplied as 0.

4. References

[1] Dongarra J J, Moler C B, Bunch J R and Stewart G W (1979) LINPACK Users' Guide. SIAM, Philadelphia.

5. Parameters

1: JOB -- INTEGER Input

On entry: specifies the job to be performed by F04FAF as follows:

JOB = 0

The matrix T is factorized and the equations Tx=b are solved for x.

JOB = 1

```
          The matrix T is assumed to have already been
          factorized by a previous call to F04FAF with JOB = 0;
          the equations Tx=b are solved for x.

 2:  N -- INTEGER                                      Input
     On entry: n, the order of the matrix T. Constraint: N >= 1.

 3:  D(N) -- DOUBLE PRECISION array             Input/Output
     On entry: if JOB = 0, D must contain the diagonal elements
     of T. If JOB = 1, D must contain the diagonal matrix K, as
     returned by a previous call of F04FAF with JOB = 0. On
     exit: if JOB = 0, D is overwritten by the diagonal matrix K
     of the factorization. If JOB = 1, D is unchanged.

 4:  E(N) -- DOUBLE PRECISION array             Input/Output
     On entry: if JOB = 0, E must contain the super-diagonal
     elements of T, stored in E(2) to E(n). If JOB = 1, E must
     contain the off-diagonal elements of the matrix M, as
     returned by a previous call of F04FAF with JOB = 0. E(1) is
     not used. On exit: if JOB = 0, E(2) to E(n) are overwritten
     by the off-diagonal elements of the matrix M of the
     factorization. If JOB = 1, E is unchanged.

 5:  B(N) -- DOUBLE PRECISION array             Input/Output
     On entry: the right-hand side vector b. On exit: B is
     overwritten by the solution vector x.

 6:  IFAIL -- INTEGER                           Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1
     On entry N < 1,

     or       JOB /= 0 or 1.

IFAIL= 2
     The matrix T is either not positive-definite or is nearly
     singular. This failure can only occur when JOB = 0 and
     inspection of the elements of D will give an indication of
     why failure has occurred. If an element of D is close to
     zero, then T is probably nearly singular; if an element of D
```

is negative but not close to zero, then T is not positive-definite.

IFAILOverflow

If overflow occurs during the execution of this routine, then either T is very nearly singular or an element of the right-hand side vector b is very large. In this latter case the equations should be scaled so that no element of b is very large. Note that to preserve symmetry it is necessary to scale by a transformation of the form $(PTP^T)b=Px$, where P is a diagonal matrix.

IFAILUnderflow

Any underflows that occur during the execution of this routine are harmless.

7. Accuracy

The computed factorization (see Section 8) will satisfy the equation

$$MKM^T=T+E$$

where $||E||_p \le 2(\epsilon)||T||_p$, $p=1,F,\infty$,

(ϵ) being the machine precision. The computed solution of the equations $Tx=b$, say x, will satisfy an equation of the form

$$(T+F)x=b,$$

where F can be expected to satisfy a bound of the form

$$||F||\le(\alpha)(\epsilon)||T||,$$

(α) being a modest constant. This implies that the relative error in x satisfies

$$\frac{||x-x||}{}\le c(T)(\alpha)(\epsilon),$$

$||x||$

where c(T) is the condition number of T with respect to

inversion. Thus if T is nearly singular, x can be expected to have a large relative error.

8. Further Comments

The time taken by the routine is approximately proportional to n.

The routine eliminates the off-diagonal elements of T by simultaneously performing symmetric Gaussian elimination from the top and the bottom of T. The result is that T is factorized as

$$T=MKM^T,$$

where K is a diagonal matrix and M is a matrix of the form

$$M=\begin{pmatrix}
1 & 0 & 0 & .. & 0 & 0 & 0 & .. & 0 & 0 & 0 \\
m_2 & 1 & 0 & .. & 0 & 0 & 0 & .. & 0 & 0 & 0 \\
0 & m_3 & 1 & .. & 0 & 0 & 0 & .. & 0 & 0 & 0 \\
. & . & . & .. & . & . & . & .. & . & . & . \\
. & . & . & .. & . & . & . & .. & . & . & . \\
0 & 0 & 0 & .. & 1 & 0 & 0 & .. & 0 & 0 & 0 \\
0 & 0 & 0 & .. & m_{j+1} & 1 & m_{j+2} & .. & 0 & 0 & 0 \\
0 & 0 & 0 & .. & 0 & 0 & .. & 1 & 0 & 0 & 0 \\
. & . & . & .. & . & . & . & .. & . & . & . \\
. & . & . & .. & . & . & . & .. & . & . & . \\
0 & 0 & 0 & .. & 0 & 0 & 0 & .. & 1 & m_{n-1} & 0 \\
0 & 0 & 0 & .. & 0 & 0 & 0 & .. & 0 & 1 & m_n \\
0 & 0 & 0 & .. & 0 & 0 & 0 & . \; . & 0 & 0 & 1
\end{pmatrix}$$

j being the integer part of n/2. (For example when n=5,j=2.) The diagonal elements of K are returned in D with k_i in the ith element of D and m_i is returned in the ith element of E.

The routine fails with IFAIL = 2 if any diagonal element of K is non-positive. It should be noted that T may be nearly singular even if all the diagonal elements of K are positive, but in this case at least one element of K is almost certain to be small

```
          relative to |||T|||. If there is any doubt as to whether or not T
          is nearly singular, then the user should consider examining the
          diagonal elements of K.

          9. Example

          To solve the symmetric positive-definite equations

                                       Tx =b
                                         1  1

          and

                                       Tx =b
                                         2  2

          where

                             ( 4 -2  0  0 0)        ( 6)       (10)
                             (-2 10 -6  0 0)        ( 9)       ( 4)
                           T=( 0 -6 29 15 0),   b =( 2),   b =( 9).
                             ( 0  0 15 25 8)     1 (14)     2 (65)
                             ( 0  0  0  8 5)        ( 7)       (23)

          The equations are solved by two calls to F04FAF, the first with
          JOB = 0 and the second, using the factorization from the first
          call, with JOB = 1.

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Solution of a linear least-squares problem, $Ax = b$

— nagf.ht —

```
\begin{page}{manpageXXf04jgf}{NAG Documentation: f04jgf}
\beginscroll
\begin{verbatim}
```

F04 -- Simultaneous Linear Equations F04JGF
F04JGF -- NAG Foundation Library Routine Document

Note: Before using this routine, please read the Users' Note for your implementation to check implementation-dependent details. The symbol (*) after a NAG routine name denotes a routine that is not included in the Foundation Library.

1. Purpose

F04JGF finds the solution of a linear least-squares problem, Ax=b, where A is a real m by n (m>=n) matrix and b is an m element vector. If the matrix of observations is not of full rank, then the minimal least-squares solution is returned.

2. Specification

```
       SUBROUTINE F04JGF (M, N, A, NRA, B, TOL, SVD, SIGMA,
      1                   IRANK, WORK, LWORK, IFAIL)
       INTEGER          M, N, NRA, IRANK, LWORK, IFAIL
       DOUBLE PRECISION A(NRA,N), B(M), TOL, SIGMA, WORK(LWORK)
       LOGICAL          SVD
```

3. Description

The minimal least-squares solution of the problem Ax=b is the vector x of minimum (Euclidean) length which minimizes the length of the residual vector r=b-Ax.

The real m by n (m>=n) matrix A is factorized as

$$A=Q\begin{pmatrix}U\\0\end{pmatrix}$$

where Q is an m by m orthogonal matrix and U is an n by n upper triangular matrix. If U is of full rank, then the least-squares solution is given by

$$x=(U^{-1}\quad 0)Q^Tb.$$

If U is not of full rank, then the singular value decomposition of U is obtained so that U is factorized as

T

$$U=RDP^T,$$

where R and P are n by n orthogonal matrices and D is the n by n diagonal matrix

$$D=\mathrm{diag}(\sigma_1,\sigma_2,\ldots,\sigma_n),$$

with $\sigma_1 \geq \sigma_2 \geq \ldots \sigma_n \geq 0$, these being the singular values of A. If the singular values $\sigma_{k+1},\ldots,\sigma_n$ are negligible, but σ_k is not negligible, relative to the data errors in A, then the rank of A is taken to be k and the minimal least-squares solution is given by

$$x=P\begin{pmatrix} S^{-1} & 0 \\ 0 & 0 \end{pmatrix}\begin{pmatrix} R^T & 0 \\ 0 & I \end{pmatrix}Q^T b,$$

where $S=\mathrm{diag}(\sigma_1,\sigma_2,\ldots,\sigma_k)$.

This routine obtains the factorizations by a call to F02WDF(*).

The routine also returns the value of the standard error

$$\sigma = \sqrt{\frac{r^T r}{m-k}}, \quad \text{if } m>k,$$

$$= 0, \quad \text{if } m=k,\ r^T r \text{ being the residual sum of squares and } k \text{ the rank of } A.$$

4. References

[1] Lawson C L and Hanson R J (1974) Solving Least-squares Problems. Prentice-Hall.

5. Parameters

1: M -- INTEGER Input
On entry: m, the number of rows of A. Constraint: M >= N.

2: N -- INTEGER Input
On entry: n, the number of columns of A. Constraint: 1 <= N

<= M.

3: A(NRA,N) -- DOUBLE PRECISION array Input/Output
On entry: the m by n matrix A. On exit: if SVD is returned as .FALSE., A} is overwritten by details of the QU factorization of A (see F02WDF(*) for further details). If SVD is returned as .TRUE., the first n rows of A are overwritten by the right-hand singular vectors, stored by rows; and the remaining rows of the array are used as workspace.

4: NRA -- INTEGER Input
On entry:
the first dimension of the array A as declared in the (sub)program from which F04JGF is called.
Constraint: NRA >= M.

5: B(M) -- DOUBLE PRECISION array Input/Output
On entry: the right-hand side vector b. On exit: the first n elements of B contain the minimal least-squares solution vector x. The remaining m-n elements are used for workspace.

6: TOL -- DOUBLE PRECISION Input
On entry: a relative tolerance to be used to determine the rank of A. TOL should be chosen as approximately the largest relative error in the elements of A. For example, if the elements of A are correct to about 4 significant figures then TOL should be set to about $5*10^{-4}$. See Section 8 for a description of how TOL is used to determine rank. If TOL is outside the range ((epsilon),1.0), where (epsilon) is the machine precision, then the value (epsilon) is used in place of TOL. For most problems this is unreasonably small.

7: SVD -- LOGICAL Output
On exit: SVD is returned as .FALSE. if the least-squares solution has been obtained from the QU factorization of A. In this case A is of full rank. SVD is returned as .TRUE. if the least-squares solution has been obtained from the singular value decomposition of A.

8: SIGMA -- DOUBLE PRECISION Output

On exit: the standard error, i.e., the value $\sqrt{r^T r/(m-k)}$ when m>k, and the value zero when m=k. Here r is the residual vector b-Ax and k is the rank of A.

9: IRANK -- INTEGER Output

```
     On exit: k, the rank of the matrix A. It should be noted
     that it is possible for IRANK to be returned as n and SVD to
     be returned as .TRUE.. This means that the matrix U only
     just failed the test for non-singularity.

10:  WORK(LWORK) -- DOUBLE PRECISION array                 Output
     On exit: if SVD is returned as .FALSE., then the first n
     elements of WORK contain information on the QU factorization
     of A (see parameter A above and F02WDF(*)), and WORK(n+1)
                                              -1
     contains the condition number ||U|| ||U  ||  of the
                                        E       E
     upper triangular matrix U.

     If SVD is returned as .TRUE., then the first n elements of
     WORK contain the singular values of A arranged in descending
     order and WORK(n+1) contains the total number of iterations
     taken by the QR algorithm. The rest of WORK is used as
     workspace.

11:  LWORK -- INTEGER                                       Input
     On entry:
     the dimension of the array WORK as declared in the
     (sub)program from which F04JGF is called.
     Constraint: LWORK >= 4*N.

12:  IFAIL -- INTEGER                                Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1
     On entry N < 1,

     or       M < N,

     or       NRA < M,

     or       LWORK < 4*N.

IFAIL= 2
     The QR algorithm has failed to converge to the singular
     values in 50*N iterations. This failure can only happen when
```

the singular value decomposition is employed, but even then it is not likely to occur.

7. Accuracy

The computed factors Q, U, R, D and P^T satisfy the relations

$$Q\begin{pmatrix}U\\0\end{pmatrix}=A+E\ ,\quad Q\begin{pmatrix}R & 0\\0 & I\end{pmatrix}\begin{pmatrix}D\\0\end{pmatrix}P^T=A+F,$$

where

$$\|E\|_2 \le c_1 \epsilon \|A\|_2 ,$$

$$\|F\|_2 \le c_2 \epsilon \|A\|_2 ,$$

ϵ being the machine precision, and c_1 and c_2 being modest functions of m and n. Note that $\|A\|_2=\sigma_1$.

For a fuller discussion, covering the accuracy of the solution x see Lawson and Hanson [1], especially pp 50 and 95.

8. Further Comments

If the least-squares solution is obtained from the QU factorization then the time taken by the routine is approximately proportional to $n^2(3m-n)$. If the least-squares solution is obtained from the singular value decomposition then the time taken is approximately proportional to $n^2(3m+19n)$. The approximate proportionality factor is the same in each case.

This routine is column biased and so is suitable for use in paged environments.

Following the QU factorization of A the condition number

$$c(U)=\|U\|_E \|U^{-1}\|_E$$

is determined and if c(U) is such that

```
                         c(U)*TOL>1.0

          then U is regarded as singular and the singular values of A are
          computed. If this test is not satisfied, U is regarded as non-
          singular and the rank of A is set to n. When the singular values
          are computed the rank of A, say k, is returned as the largest
          integer such that

                           (sigma) >TOL*(sigma) ,
                                  k            1

          unless (sigma) =0 in which case k is returned as zero. That is,
                        1
          singular values which satisfy (sigma) <=TOL*(sigma)  are regarded
                                                 i            1
          as negligible because relative perturbations of order TOL can
          make such singular values zero.

          9. Example

          To obtain a least-squares solution for r=b-Ax, where

                           (0.05  0.05 0.25 -0.25)      (1)
                           (0.25  0.25 0.05 -0.05)      (2)
                           (0.35  0.35 1.75 -1.75)      (3)
                         A=(1.75  1.75 0.35 -0.35),   B=(4)
                           (0.30 -0.30 0.30  0.30)      (5)
                           (0.40 -0.40 0.40  0.40)      (6)

                                                  -4
          and the value TOL is to be taken as 5*10   .

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Sparse symmetric positive-definite system linear equations

— nagf.ht —

```
\begin{page}{manpageXXf04maf}{NAG Documentation: f04maf}
```

```
\beginscroll
\begin{verbatim}


     F04MAF(3NAG)      Foundation Library (12/10/92)      F04MAF(3NAG)



          F04 -- Simultaneous Linear Equations                       F04MAF
                  F04MAF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          To solve a sparse symmetric positive-definite system of linear
          equations, Ax=b, using a pre-conditioned conjugate gradient
          method, where A has been factorized by F01MAF.

          2. Specification

                 SUBROUTINE F04MAF (N, NZ, A, LICN, IRN, LIRN, ICN, B, ACC,
                1                   NOITS, WKEEP, WORK, IKEEP, INFORM,
                2                   IFAIL)
                 INTEGER           N, NZ, LICN, IRN(LIRN), LIRN, ICN(LICN),
                1                  NOITS(2), IKEEP(2*N), INFORM(4), IFAIL
                 DOUBLE PRECISION A(LICN), B(N), ACC(2), WKEEP(3*N), WORK
                1                 (3*N)

          3. Description

          F04MAF solves the n linear equations

                                     Ax=b,                             (1)

          where A is a sparse symmetric positive-definite matrix, following
          the incomplete Cholesky factorization by F01MAF, given by

                                      T T
                                C=PLDL P   , WAW=C+E,

          where P is a permutation matrix, L is a unit lower triangular
          matrix, D is a diagonal matrix with positive diagonal elements, E
          is an error matrix representing elements dropped during the
          factorization and diagonal elements that have been modified to
          ensure that C is positive-definite, and W is a diagonal matrix,
```

chosen to make the diagonal elements of WAW unity.

Equation (1) is solved by applying a pre-conditioned conjugate gradient method to the equations

$$(WAW)(W^{-1}x)=Wb, \qquad (2)$$

using C as the pre-conditioning matrix. Details of the conjugate gradient method are given in Munksgaard [1].

The iterative procedure is terminated if

$$||Wr||_2 <= (\eta), \qquad (3)$$

where r is the residual vector r=b-Ax, $||r||_2$ denotes the Euclidean length of r, (eta) is a user-supplied tolerance and x is the current approximation to the solution. Notice that

$$Wr=Wb-(WAW)(W^{-1}x)$$

so that Wr is the residual of the normalised equations (2).

F04MAF is based on the Harwell Library routine MA31B.

4. References

[1] Munksgaard N (1980) Solving Sparse Symmetric Sets of Linear Equations by Pre-conditioned Conjugate Gradients. ACM Trans. Math. Softw. 6 206--219.

5. Parameters

1: N -- INTEGER Input
On entry: n, the order of the matrix A. Constraint: N >= 1.

2: NZ -- INTEGER Input
On entry: the number of non-zero elements in the upper triangular part of the matrix A, including the number of elements on the leading diagonal. Constraint: NZ >= N.

3: A(LICN) -- DOUBLE PRECISION array Input
On entry: the first LROW elements, where LROW is the value supplied in INFORM(1), must contain details of the factorization, as returned by F01MAF.

4: LICN -- INTEGER Input

```
     On entry: the length of the array A, as declared in the
     (sub)program from which F04MAF is called. It need never be
     larger than the value of LICN supplied to F01MAF.
     Constraint: LICN >= INFORM(1).

 5:  IRN(LIRN) -- INTEGER array                             Input
     On entry: the first LCOL elements, where LCOL is the value
     supplied in INFORM(2), must contain details of the
     factorization, as returned by F01MAF.

 6:  LIRN -- INTEGER                                        Input
     On entry: the length of the array IRN, as declared in the
     (sub)program from which F04MAF is called. It need never be
     larger than the value of LIRN supplied to F01MAF.
     Constraint: LIRN >= INFORM(2).

 7:  ICN(LICN) -- INTEGER array                             Input
     On entry: the first LROW elements, where LROW is the value
     supplied in INFORM(1), must contain details of the
     factorization, as returned by F01MAF.

 8:  B(N) -- DOUBLE PRECISION array                  Input/Output
     On entry: the right-hand side vector b. On exit: B is
     overwritten by the solution vector x.

 9:  ACC(2) -- DOUBLE PRECISION array                Input/Output
     On entry: ACC(1) specifies the tolerance for convergence,
     (eta), in equation (3) of Section 3. If ACC(1) is outside
     the range [(epsilon),1], where (epsilon) is the machine
     precision, then the value (epsilon) is used in place of ACC
     (1). ACC(2) need not be set. On exit: ACC(2) contains the
     actual value of ||Wr||  at the final point. ACC(1) is
                           2
     unchanged.

10:  NOITS(2) -- INTEGER array                       Input/Output
     On entry: NOITS(1) specifies the maximum permitted number of
     iterations. If NOITS(1) < 1, then the value 100 is used in
     its place. NOITS(2) need not be set. On exit: NOITS(2)
     contains the number of iterations taken to converge. NOITS
     (1) is unchanged.

11:  WKEEP(3*N) -- DOUBLE PRECISION array                   Input
     On entry: WKEEP must be unchanged from the previous call of
     F01MAF.

12:  WORK(3*N) -- DOUBLE PRECISION array                   Output
     On exit: WORK(1) contains a lower bound for the condition
     number of A. The rest of the array is used for workspace.
```

```
13:  IKEEP(2*N) -- INTEGER array                           Input
     On entry: IKEEP must be unchanged from the previous call of
     F01MAF.

14:  INFORM(4) -- INTEGER array                            Input
     On entry: INFORM must be unchanged from the previous call of
     F01MAF.

15:  IFAIL -- INTEGER                               Input/Output
     For this routine, the normal use of IFAIL is extended to
     control the printing of error and warning messages as well
     as specifying hard or soft failure (see the Essential
     Introduction).

     Before entry, IFAIL must be set to a value with the decimal
     expansion cba, where each of the decimal digits c, b and a
     must have a value of 0 or 1.
     a=0   specifies hard failure, otherwise soft failure;

     b=0   suppresses error messages, otherwise error messages
           will be printed (see Section 6);

     c=0   suppresses warning messages, otherwise warning
           messages will be printed (see Section 6).
     The recommended value for inexperienced users is 110 (i.e.,
     hard failure with all messages printed).

     Unless the routine detects an error (see Section 6), IFAIL
     contains 0 on exit.

6. Error Indicators and Warnings

Errors detected by the routine:

For each error, an explanatory error message is output on the
current error message unit (as defined by X04AAF), unless
suppressed by the value of IFAIL on entry.

IFAIL= 1
     On entry N < 1,

     or       NZ < N,

     or       LICN < INFORM(1),

     or       LIRN < INFORM(2).

IFAIL= 2
     Convergence has not taken place within the requested NOITS
     (1) number of iterations. ACC(2) gives the value ||Wr|| ,
```

```
                                                          2
     for the final point. Either too few iterations have been
     allowed, or the requested convergence criterion cannot be
     met.

IFAIL= 3
     The matrix A is singular, or nearly singular. Singularity
     has been detected during the conjugate gradient iterations,
     so that the computations are not complete.

IFAIL= 4
     The matrix A is singular, or nearly singular. The message
     output on the current error message channel will include an
     estimate of the condition number of A. In the case of soft
     failure an approximate solution is returned such that the
     value ||Wr||  is given by ACC(2) and the estimate (a lower
                 2
     bound) of the condition number is returned in WORK(1).

7. Accuracy

On successful return, or on return with IFAIL = 2 or IFAIL = 4
the computed solution will satisfy equation (3) of Section 3,
with (eta) = ACC(2).

8. Further Comments

The time taken by the routine will depend upon the sparsity of
the factorization and the number of iterations required. The
number of iterations will be affected by the nature of the
factorization supplied by F01MAF. The more incomplete the
factorization, the higher the number of iterations required by
F04MAF.

When the solution of several systems of equations, all with the
same matrix of coefficients, A, is required, then F01MAF need be
called only once to factorize A. This is illustrated in the
context of an eigenvalue problem in the example program for
F02FJF.

9. Example

The example program illustrates the use of F01MAF in conjunction
with F04MAF to solve the 16 linear equations Ax=b, where

                 (1 a     a                       )
                 (a 1 a     a                     )
                 (  a 1 a     a                   )
                 (    a 1 0     a                 )
                 (a     0 1 a     a               )
```

```
                   ( a     a 1 a     a           )
                   (   a     a 1 a     a         )
                   (     a     a 1 0     a       )
                 A=(       a     0 1 a     a     )
                   (         a     a 1 a     a   )
                   (           a     a 1 a     a )
                   (             a     a 1 0     a)
                   (               a     0 1 a    )
                   (                 a     a 1 a  )
                   (                   a     a 1 a)
                   (                     a     a 1)

          1
where a=- -.
          4

         T (  1  1  1  1  1     1  1     1  1  1  1  1)
        b =(  -  -  -  -  - 0 0  -  - 0 0  -  -  -  -  -)
           (  2  4  4  2  4     4  4     4  2  4  4  2)

The n by n matrix A arises in the solution of Laplace's equation
in a unit-square, using a five-point formula with a 6 by 6
discretisation, with unity on the boundaries.

The drop tolerance, DROPTL, is taken as 0.1 and the density
factor, DENSW, is taken as 0.8. The value IFAIL = 111 is used so
that advisory and error messages will be printed, but soft
failure would occur if IFAIL were returned as non-zero.

A relative accuracy of about 0.0001 is requested in the solution
from F04MAF, with a maximum of 50 iterations.

The example program for F02FJF illustrates the use of routines
F01MAF and F04MAF in solving an eigenvalue problem.

The example program is not reproduced here. The source code for
all example programs is distributed with the NAG Foundation
Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Solves a system of real sparse symmetric linear equations

— **nagf.ht** —

```
\begin{page}{manpageXXf04mbf}{NAG Documentation: f04mbf}
\beginscroll
\begin{verbatim}



     F04MBF(3NAG)      Foundation Library (12/10/92)      F04MBF(3NAG)



          F04 -- Simultaneous Linear Equations                      F04MBF
                  F04MBF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          F04MBF solves a system of real sparse symmetric linear equations
          using a Lanczos algorithm.

          2. Specification

                 SUBROUTINE F04MBF (N, B, X, APROD, MSOLVE, PRECON, SHIFT,
                1                   RTOL, ITNLIM, MSGLVL, ITN, ANORM,
                2                   ACOND, RNORM, XNORM, WORK, RWORK,
                3                   LRWORK, IWORK, LIWORK, INFORM, IFAIL)
                 INTEGER          N, ITNLIM, MSGLVL, ITN, LRWORK, IWORK
                1                 (LIWORK), LIWORK, INFORM, IFAIL
                 DOUBLE PRECISION B(N), X(N), SHIFT, RTOL, ANORM, ACOND,
                1                 RNORM, XNORM, WORK(N,5), RWORK(LRWORK)
                 LOGICAL          PRECON
                 EXTERNAL         APROD, MSOLVE

          3. Description

          F04MBF solves the system of linear equations

                               (A-(lambda)I)x=b                        (3.1)

          where A is an n by n sparse symmetric matrix and (lambda) is a
          scalar, which is of course zero if the solution of the equations
```

$$Ax=b$$

is required. It should be noted that neither A nor $(A-\lambda I)$ need be positive-definite.

λ is supplied as the parameter SHIFT, and allows F04MBF to be used for finding eigenvectors of A in methods such as Rayleigh quotient iteration (see for example Lewis [1]), in which case λ will be an approximation to an eigenvalue of A and b an approximation to an eigenvector of A.

The routine also provides an option to allow pre-conditioning and this will often reduce the number of iterations required by F04MBF.

F04MBF is based upon algorithm SYMMLQ (see Paige and Saunders [2]) and solves the equations by an algorithm based upon the

Lanczos process. Details of the method are given in Paige and Saunders [2]. The routine does not require A explicitly, but A is specified via a user-supplied routine APROD which, given an n element vector c, must return the vector z given by

$$z=Ac.$$

The pre-conditioning option is based on the following reasoning. If A can be expressed in the form

$$A=I+B$$

where B is of rank ρ, then the Lanczos process converges (in exact arithmetic) in at most ρ iterations. If more generally A can be expressed in the form

$$A=M+C$$

where M is symmetric positive-definite and C has rank ρ, then

$$M^{-(1/2)}AM^{-(1/2)}=I+M^{-(1/2)}CM^{-(1/2)}$$

and $M^{-(1/2)}AM^{-(1/2)}$ also has rank ρ, and the Lanczos process applied to $M^{-(1/2)}AM^{-(1/2)}$ would again converge in at most ρ iterations. On a computer, the number of iterations may be greater than ρ, but the Lanczos process may still be expected to converge rapidly. F04MBF does not require $M^{-(1/2)}AM^{-(1/2)}$ to be formed explicitly, but implicitly solves the equations

$$M^{-(1/2)}(A-(\lambda)I)M^{-(1/2)}y=M^{-(1/2)}b \;,\; y=M^{1/2}x \qquad (3.2)$$

with the user being required to supply a routine MSOLVE to solve the equations

$$Mz=c. \qquad (3.3)$$

For the pre-conditioning option to be effective, it is desirable that equations (3.3) can be solved efficiently. The example program in Section 9 illustrates the use of this option.

If we let r denote the residual vector

$$r=b-(A-(\lambda)I)x$$

corresponding to an iterate x, then, when pre-conditioning has not been requested, the iterative procedure is terminated if it is estimated that

$$||r||<=tol.||A-(\lambda)I||.||x||, \qquad (3.4)$$

where tol is a user-supplied tolerance, ||r|| denotes the Euclidean length of the vector r and ||A|| denotes the Frobenius (Euclidean) norm of the matrix A. When pre-conditioning has been requested, the iterative procedure is terminated if it is estimated that

$$||M^{-(1/2)}r||<=tol.||M^{-(1/2)}(A-(\lambda)I)M^{-(1/2)}||.||M^{1/2}x||. \qquad (3.5)$$

Note that

$$M^{-(1/2)}r=(M^{-(1/2)}b)-M^{-(1/2)}(A-(\lambda)I)M^{-(1/2)}(M^{1/2}x)$$

so that $M^{-(1/2)}r$ is the residual vector corresponding to equation (3.2). The routine will also terminate if it is estimated that

$$||A-(\lambda)I||.||x||>=||b||/(\epsilon), \qquad (3.6)$$

where (ϵ) is the machine precision, when pre-conditioning has not been requested; or if it is estimated that

$$||M^{-(1/2)}(A-(\lambda)I)M^{-(1/2)}||.||M^{1/2}x||>=||M^{-(1/2)}b||/(\epsilon)$$

(3.7)

when pre-conditioning has been requested. If (3.6) is satisfied then x is almost certainly an eigenvector of A corresponding to the eigenvalue (lambda). If (lambda) was set to 0 (for the solution of Ax=b), then this condition simply means that A is effectively singular.

4. References

[1] Lewis J G (1977) Algorithms for sparse matrix eigenvalue problems. Technical Report STAN-CS-77-595. Computer Science Department, Stanford University.

[2] Paige C C and Saunders M A (1975) Solution of Sparse Indefinite Systems of Linear Equations. SIAM J. Numer. Anal. 12 617--629.

5. Parameters

1: N -- INTEGER Input
On entry: n, the order of the matrix A. Constraint: N >= 1.

2: B(N) -- DOUBLE PRECISION array Input
On entry: the right-hand side vector b.

3: X(N) -- DOUBLE PRECISION array Output
On exit: the solution vector x.

4: APROD -- SUBROUTINE, supplied by the user.
External Procedure
APROD must return the vector y=Ax for a given vector x.

Its specification is:

```
       SUBROUTINE APROD (IFLAG, N, X, Y, RWORK, LRWORK,
      1                  IWORK,LIWORK)
       INTEGER          IFLAG, N, LRWORK, LIWORK, IWORK
      1                 (LIWORK)
       DOUBLE PRECISION X(N), Y(N), RWORK(LRWORK)
```

1: IFLAG -- INTEGER Input/Output
On entry: IFLAG is always non-negative. On exit: IFLAG may be used as a flag to indicate a failure in the computation of Ax. If IFLAG is negative on exit from APROD, F04MBF will exit immediately with IFAIL set to IFLAG.

2: N -- INTEGER Input
On entry: n, the order of the matrix A.

```
 3:  X(N) -- DOUBLE PRECISION array                    Input
     On entry: the vector x for which Ax is required.

 4:  Y(N) -- DOUBLE PRECISION array                   Output
     On exit: the vector y=Ax.

 5:  RWORK(LRWORK) -- DOUBLE PRECISION array  User Workspace

 6:  LRWORK -- INTEGER                                 Input

 7:  IWORK(LIWORK) -- INTEGER array           User Workspace

 8:  LIWORK -- INTEGER                                 Input
     APROD is called from F04MBF with the parameters RWORK,
     LRWORK, IWORK and LIWORK as supplied to F04MBF. The
     user is free to use the arrays RWORK and IWORK to
     supply information to APROD and MSOLVE as an
     alternative to using COMMON.
APROD must be declared as EXTERNAL in the (sub)program
from which F04MBF is called. Parameters denoted as
Input must not be changed by this procedure.

5:  MSOLVE -- SUBROUTINE, supplied by the user.
                                          External Procedure
    MSOLVE is only referenced when PRECON is supplied as .TRUE..
    When PRECON is supplied as .FALSE., then F04MBF may be
    called with APROD as the actual argument for MSOLVE. When
    PRECON is supplied as .TRUE., then MSOLVE must return the
    solution y of the equations My=x for a given vector x, where
    M must be symmetric positive-definite.

    Its specification is:

           SUBROUTINE MSOLVE (IFLAG, N, X, Y, RWORK,
          1                   LRWORK, IWORK,LIWORK)
           INTEGER          IFLAG, N, LRWORK, LIWORK, IWORK
          1                 (LIWORK)
           DOUBLE PRECISION X(N), Y(N), RWORK(LRWORK)

 1:  IFLAG -- INTEGER                           Input/Output
     On entry: IFLAG is always non-negative. On exit: IFLAG
     may be used as a flag to indicate a failure in the
     solution of My=x.

     If IFLAG is negative on exit from MSOLVE, F04MBF will
     exit immediately with IFAIL set to IFLAG.

 2:  N -- INTEGER                                      Input
     On entry: n, the order of the matrix M.
```

```
         3:  X(N) -- DOUBLE PRECISION array                  Input
             On entry: the vector x for which the equations My=x are
             to be solved.

         4:  Y(N) -- DOUBLE PRECISION array                 Output
             On exit: the solution to the equations My=x.

         5:  RWORK(LRWORK) -- DOUBLE PRECISION array  User Workspace

         6:  LRWORK -- INTEGER                               Input

         7:  IWORK(LIWORK) -- INTEGER array           User Workspace

         8:  LIWORK -- INTEGER                               Input
             MSOLVE is called from F04MBF with the parameters RWORK,
             LRWORK, IWORK and LIWORK as supplied to F04MBF. The
             user is free to use the arrays RWORK and IWORK to
             supply information to APROD and MSOLVE as an
             alternative to using COMMON.
        MSOLVE must be declared as EXTERNAL in the (sub)program
        from which F04MBF is called. Parameters denoted as
        Input must not be changed by this procedure.

     6:  PRECON -- LOGICAL                                   Input
         On entry: PRECON specifies whether or not pre-conditioning
         is required. If PRECON = .TRUE., then pre-conditioning will
         be invoked and MSOLVE will be referenced by F04MBF; if
         PRECON = .FALSE., then MSOLVE is not referenced.

     7:  SHIFT -- DOUBLE PRECISION                           Input
         On entry: the value of (lambda). If the equations Ax=b are
         to be solved, then SHIFT must be supplied as zero.

     8:  RTOL -- DOUBLE PRECISION                            Input
         On entry: the tolerance for convergence, tol, of equation
         (3.4). RTOL should not normally be less than (epsilon),
         where (epsilon) is the machine precision.

     9:  ITNLIM -- INTEGER                                   Input
         On entry: an upper limit on the number of iterations. If
         ITNLIM <= 0, then the value N is used in place of ITNLIM.

    10:  MSGLVL -- INTEGER                                   Input
         On entry: the level of printing from F04MBF. If MSGLVL <= 0,
         then no printing occurs, but otherwise messages will be
         output on the advisory message channel (see X04ABF). A
         description of the printed output is given in Section 5.1
         below. The level of printing is determined as follows:
         MSGLVL <= 0
```

```
          No printing.

     MSGLVL = 1
          A brief summary is printed just prior to return from
          F04MBF.

     MSGLVL >= 2
          A summary line is printed periodically to monitor the
          progress of F04MBF, together with a brief summary just
          prior to return from F04MBF.

11:  ITN -- INTEGER                                      Output
     On exit: the number of iterations performed.

12:  ANORM -- DOUBLE PRECISION                           Output
     On exit: an estimate of ||A-(lambda)I|| when PRECON =
                    -(1/2)              -(1/2)
     .FALSE., and ||M      (A-(lambda)I)M      || when PRECON =
     .TRUE..

13:  ACOND -- DOUBLE PRECISION                           Output
     On exit: an estimate of the condition number of (A-
     (lambda)I) when PRECON = .FALSE., and of
      -(1/2)              -(1/2)
     M      (A-(lambda)I)M       when PRECON = .TRUE.. This will
     usually be a substantial under-estimate.

14:  RNORM -- DOUBLE PRECISION                           Output
     On exit: ||r||, where r=b-(A-(lambda)I)x and x is the
     solution returned in X.

15:  XNORM -- DOUBLE PRECISION                           Output
     On exit: ||x||, where x is the solution returned in X.

16:  WORK(5*N) -- DOUBLE PRECISION array              Workspace

17:  RWORK(LRWORK) -- DOUBLE PRECISION array     User Workspace
     RWORK is not used by F04MBF, but is passed directly to
     routines APROD and MSOLVE and may be used to pass
     information to these routines.

18:  LRWORK -- INTEGER                                    Input
     On entry: the length of the array RWORK as declared in the
     (sub)program from which F04MBF is called. Constraint: LRWORK
     >= 1.

19:  IWORK(LIWORK) -- INTEGER array              User Workspace
     IWORK is not used by F04MBF, but is passed directly to
     routines APROD and MSOLVE and may be used to pass
     information to these routines.
```

```
20:  LIWORK -- INTEGER                                       Input
     On entry: the length of the array IWORK as declared in the
     (sub)program from which F04MBF is called. Constraint: LIWORK
     >= 1.

21:  INFORM -- INTEGER                                      Output
     On exit: the reason for termination of F04MBF as follows:
     INFORM = 0
           The right-hand side vector b=0 so that the exact
           solution is x=0. No iterations are performed in this
           case.

     INFORM = 1
           The termination criterion of equation (3.4) has been
           satisfied with tol as the value supplied in RTOL.

     INFORM = 2
           The termination criterion of equation (3.4) has been
           satisfied with tol equal to (epsilon), where (epsilon)
           is the machine precision. The value supplied in RTOL
           must have been less than (epsilon) and was too small
           for the machine.

     INFORM = 3
           The termination criterion of equation (3.5) has been
           satisfied so that X is almost certainly an eigenvector
           of A corresponding to the eigenvalue SHIFT.
     The values INFORM = 4 and INFORM = 5 correspond to failure
     with IFAIL = 3 or IFAIL = 2 respectively (see Section 6) and
     when IFAIL is negative, INFORM will be set to the same
     negative value.

22:  IFAIL -- INTEGER                                 Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).

5.1. Description of the Printed Output

When MSGLVL > 0, then F04MBF will produce output (except in the
case where the routine fails with IFAIL = 1) on the advisory
message channel (see X04ABF ).

The following notation is used in the output.

   Output     Meaning
```

```
                -(1/2)
   RBAR        M      (b-(A-(lambda)I)x)=r

                -(1/2)                -(1/2)
   ABAR        M      (A-(lambda)I)M      =A

                1/2
   Y           M   x

   R           b-(A-(lambda)I)x

   NORM(A)     ||A||
```

Of course, when pre-conditioning has not been requested then the first three reduce to (b-(A-(lambda)I)x), (A-(lambda)I) and x respectively. When MSGLVL >= 2 then some initial information is printed and the following notation is used.

```
   Output      Meaning

                 T -1  1/2
   BETA1       (b M  b)   ==(beta)
                                  1

                          2  -(1/2)  T  -(1/2)  -(1/2)   -(1/2)
   ALFA1       (1/(beta) ) (M      b) (M      AM      )(M      b)
                        1
               ==(alpha)
                        1
```

and a summary line is printed periodically giving the following information:

```
   Output      Meaning

   ITN         Iteration number, k.

                                                 L         L
   X1(LQ)      The first element of the vector x , where x  is the
                                                 k         k
               current iterate. See Paige and Saunders [2] for
               details.

                                                 C         C
   X1(CG)      The first element of the vector x , where x  is the
                                                 k         k
               vector that would be obtained by conjugate
               gradients. See Paige and Saunders [2] for details.
```

NORM(RBAR) $||r||$, where r is as defined above and x is either x_k^L or x_k^C depending upon which is the best current approximation to the solution. (See LQ/CG below).

NORM(T) The value $||T_k||$, where T_k is the tridiagonal matrix of the Lanczos process. This increases

monotonically and is a lower bound on $||A||$.

COND(L) A monotonically increasing lower bound on the condition number of A, $||A||||(A)^{-1}||$.

LQ/CG L is printed if x_k^L is the best current approximation to the solution and C is printed otherwise.

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL< 0
A negative value of IFAIL indicates an exit from F04MBF because the user has set IFLAG negative in APROD or MSOLVE. The value of IFAIL will be the same as the user's setting of IFLAG.

IFAIL= 1
On entry N < 1,

or LRWORK < 1,

or LIWORK < 1.

IFAIL= 2
The pre-conditioning matrix M does not appear to be positive-definite. The user should check that MSOLVE is working correctly.

IFAIL= 3
The limit on the number of iterations has been reached. If IFAIL = 1 on entry then the latest approximation to the solution is returned in X and the values ANORM, ACOND, RNORM

and XNORM are also returned.

The value of INFORM contains additional information about the termination of the routine and users must examine INFORM to judge whether the routine has performed successfully for the problem in hand. In particular INFORM = 3 denotes that the matrix A-(lambda)I is effectively singular: if the purpose of calling F04MBF is to solve a system of equations Ax=b, then this condition must be regarded as a failure, but if the purpose is to compute an eigenvector, this result would be very satisfactory.

7. Accuracy

The computed solution x will satisfy the equation

```
                      r=b-(A-(lambda)I)x
```

where the value ||r|| is as returned in the parameter RNORM.

8. Further Comments

The time taken by the routine is likely to be principally determined by the time taken in APROD and, when pre-conditioning has been requested, in MSOLVE. Each of these routines is called once every iteration.

The time taken by the remaining operations in F04MBF is approximately proportional to n.

9. Example

To solve the 10 equations Ax=b given by

```
                  (2 1 0 0 0 0 0 0 0 3)      (6)
                  (1 2 1 0 0 0 0 0 0 0)      (4)
                  (0 1 2 1 0 0 0 0 0 0)      (4)
                  (0 0 1 2 1 0 0 0 0 0)      (4)
                  (0 0 0 1 2 1 0 0 0 0)      (4)
                A=(0 0 0 0 1 2 1 0 0 0),   b=(4).
                  (0 0 0 0 0 1 2 1 0 0)      (4)
                  (0 0 0 0 0 0 1 2 1 0)      (4)
                  (0 0 0 0 0 0 0 1 2 1)      (4)
                  (3 0 0 0 0 0 0 0 1 2)      (6)
```

The tridiagonal part of A is positive-definite and such tridiagonal equations can be solved efficiently by F04FAF. The form of A suggests that this tridiagonal part is a good candidate for the pre-conditioning matrix M and so we illustrate the use of F04MBF by pre-conditioning with the 10 by 10 matrix

```
                      (2 1 0 ... 0)
                      (1 2 1 ... 0)
                      (0 1 2 ... 0)
                    M=(. . .   .  ).
                      (. . .   .  )
                      (. . .   .  )
                      (0 0 0 ... 2)

          Since A-M has only 2 non-zero elements and is obviously of rank
          2, we can expect F04MBF to converge very quickly in this example.
          Of course, in practical problems we shall not usually be able to
          make such a good choice of M.

                                                        -5
          The example sets the tolerance RTOL = 10  .


          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Solution of a system of real linear equations

— nagf.ht —

```
\begin{page}{manpageXXf04mcf}{NAG Documentation: f04mcf}
\beginscroll
\begin{verbatim}



     F04MCF(3NAG)      Foundation Library (12/10/92)      F04MCF(3NAG)



          F04 -- Simultaneous Linear Equations                      F04MCF
                  F04MCF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
```

not included in the Foundation Library.

1. Purpose

F04MCF computes the approximate solution of a system of real linear equations with multiple right-hand sides, AX=B, where A is a symmetric positive-definite variable-bandwidth matrix, which has previously been factorized by F01MCF. Related systems may also be solved.

2. Specification

```
       SUBROUTINE F04MCF (N, AL, LAL, D, NROW, IR, B, NRB,
      1                   ISELCT, X, NRX, IFAIL)
       INTEGER          N, LAL, NROW(N), IR, NRB, ISELCT, NRX,
      1                 IFAIL
       DOUBLE PRECISION AL(LAL), D(N), B(NRB,IR), X(NRX,IR)
```

3. Description

The normal use of this routine is the solution of the systems AX=B, following a call of F01MCF to determine the Cholesky factorization $A=LDL^T$ of the symmetric positive-definite variable-bandwidth matrix A.

However, the routine may be used to solve any one of the following systems of linear algebraic equations:

(1) $LDL^TX = B$ (usual system),

(2) $LDX = B$ (lower triangular system),

(3) $DL^TX = B$ (upper triangular system),

(4) $LL^TX = B$

(5) $LX = B$ (unit lower triangular system),

(6) $L^TX = B$ (unit upper triangular system).

L denotes a unit lower triangular variable-bandwidth matrix of order n, D a diagonal matrix of order n, and B a set of right-hand sides.

The matrix L is represented by the elements lying within its envelope i.e., between the first non-zero of each row and the diagonal (see Section 9 for an example). The width NROW(i) of the ith row is the number of elements between the first non-zero element and the element on the diagonal inclusive.

4. References

[1] Wilkinson J H and Reinsch C (1971) Handbook for Automatic Computation II, Linear Algebra. Springer-Verlag.

5. Parameters

1: N -- INTEGER Input
On entry: n, the order of the matrix L. Constraint: N >= 1.

2: AL(LAL) -- DOUBLE PRECISION array Input
On entry: the elements within the envelope of the lower triangular matrix L, taken in row by row order, as returned by F01MCF. The unit diagonal elements of L must be stored explicitly.

3: LAL -- INTEGER Input
On entry:
the dimension of the array AL as declared in the (sub)program from which F04MCF is called.
Constraint: LAL >= NROW(1) + NROW(2) +... + NROW(n).

4: D(N) -- DOUBLE PRECISION array Input
On entry: the diagonal elements of the diagonal matrix D. D is not referenced if ISELCT >= 4.

5: NROW(N) -- INTEGER array Input
On entry: NROW(i) must contain the width of row i of L, i.e.,the number of elements between the first (leftmost) non-zero element and the element on the diagonal, inclusive.
Constraint: 1 <= NROW(i)<=i.

6: IR -- INTEGER Input
On entry: r, the number of right-hand sides. Constraint: IR >= 1.

7: B(NRB,IR) -- DOUBLE PRECISION array Input
On entry: the n by r right-hand side matrix B. See also Section 8.

8: NRB -- INTEGER Input
On entry:
the first dimension of the array B as declared in the (sub)program from which F04MCF is called.
Constraint: NRB >= N.

9: ISELCT -- INTEGER Input
On entry: ISELCT must specify the type of system to be solved, as follows:

ISELCT = 1: solve $LDL^T X = B$,

ISELCT = 2: solve $LDX = B$,

ISELCT = 3: solve $DL^T X = B$,

ISELCT = 4: solve $LL^T X = B$,

ISELCT = 5: solve $LX = B$,

ISELCT = 6: solve $L^T X = B$.

10: X(NRX,IR) -- DOUBLE PRECISION array Output
On exit: the n by r solution matrix X. See also Section 8.

11: NRX -- INTEGER Input
On entry:
the first dimension of the array X as declared in the (sub)program from which F04MCF is called.
Constraint: NRX >= N.

12: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1
On entry N < 1,

or for some i, NROW(i)<1 or NROW(i) > i,

or LAL < NROW(1) + NROW(2) +... + NROW(N).

IFAIL= 2

```
     On entry IR < 1,

     or       NRB < N,

     or       NRX < N.

IFAIL= 3
     On entry ISELCT < 1,

     or       ISELCT > 6.

IFAIL= 4
     The diagonal matrix D is singular, i.e., at least one of the
     elements of D is zero. This can only occur if ISELCT <= 3.

IFAIL= 5
     At least one of the diagonal elements of L is not equal to
     unity.
```

7. Accuracy

The usual backward error analysis of the solution of triangular system applies: each computed solution vector is exact for slightly perturbed matrices L and D, as appropriate (cf. Wilkinson and Reinsch [1] pp 25--27, 54--55).

8. Further Comments

The time taken by the routine is approximately proportional to pr, where

```
                 p=NROW(1)+NROW(2)+...+NROW(n).
```

Unless otherwise stated in the Users' Note for your implementation, the routine may be called with the same actual array supplied for the parameters B and X, in which case the solution matrix will overwrite the right-hand side matrix. However this is not standard Fortran 77 and may not work in all implementations.

9. Example

To solve the system of equations AX=B, where

```
                         (1  2  0  0  5  0)
                         (2  5  3  0 14  0)
                         (0  3 13  0 18  0)
                       A=(0  0  0 16  8 24)
                         (5 14 18  8 55 17)
```

```
                    (0  0  0 24 17 77)

   and

                         ( 6 -10)
                         (15 -21)
                         (11  -3)
                       B=( 0  24)
                         (51 -39)
                         (46  67)

   Here A is symmetric and positive-definite and must first be
   factorized by F01MCF.

   The example program is not reproduced here. The source code for
   all example programs is distributed with the NAG Foundation
   Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Solves sparse unsymmetric equations

— nagf.ht —

```
\begin{page}{manpageXXf04qaf}{NAG Documentation: f04qaf}
\beginscroll
\begin{verbatim}



     F04QAF(3NAG)      Foundation Library (12/10/92)      F04QAF(3NAG)



          F04 -- Simultaneous Linear Equations                      F04QAF
                  F04QAF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose
```

F04QAF solves sparse unsymmetric equations, sparse linear least-squares problems and sparse damped linear least-squares problems, using a Lanczos algorithm.

2. Specification

```
       SUBROUTINE F04QAF (M, N, B, X, SE, APROD, DAMP, ATOL,
      1                   BTOL, CONLIM, ITNLIM, MSGLVL, ITN,
      2                   ANORM, ACOND, RNORM, ARNORM, XNORM,
      3                   WORK, RWORK, LRWORK, IWORK, LIWORK,
      4                   INFORM, IFAIL)
       INTEGER          M, N, ITNLIM, MSGLVL, ITN, LRWORK, IWORK
      1                 (LIWORK), LIWORK, INFORM, IFAIL
       DOUBLE PRECISION B(M), X(N), SE(N), DAMP, ATOL, BTOL,
      1                 CONLIM, ANORM, ACOND, RNORM, ARNORM,
      2                 XNORM, WORK(N,2), RWORK(LRWORK)
       EXTERNAL         APROD
```

3. Description

F04QAF can be used to solve a system of linear equations

$$Ax=b \tag{3.1}$$

where A is an n by n sparse unsymmetric matrix, or can be used to solve linear least-squares problems, so that F04QAF minimizes the value (rho) given by

$$(\rho)=||r||, \quad r=b-Ax \tag{3.2}$$

where A is an m by n sparse matrix and ||r|| denotes the Euclidean length of r so that $||r||^2=r^Tr$. A damping parameter, (lambda), may be included in the least-squares problem in which case F04QAF minimizes the value (rho) given by

$$(\rho)^2=||r||^2+(\lambda)^2||x||^2 \tag{3.3}$$

(lambda) is supplied as the parameter DAMP and should of course be zero if the solution to problems (3.1) or (3.2) is required. Minimizing (rho) in (3.3) is often called ridge regression.

F04QAF is based upon algorithm LSQR (see Paige and Saunders [1] and [2]) and solves the problems by an algorithm based upon the Lanczos process. Details of the method are given in [1]. The routine does not require A explicitly, but A is specified via a user-supplied routine APROD which must perform the operations (^{T}

y+Ax) and (x+A y) for a given n element vector x and m element vector y. A parameter to APROD specifies which of the two operations is required on a given entry.

The routine also returns estimates of the standard errors of the sample regression coefficients (x_i, for $i=1,2,...,n$) given by the diagonal elements of the estimated variance-covariance matrix V. When problem (3.2) is being solved and A is of full rank, then V is given by

$$V=s^2 (A^T A)^{-1}, \quad s^2=(\rho)^2/(m-n), \quad m>n$$

and when problem (3.3) is being solved then V is given by

$$V=s^2 (A^T A+(\lambda)^2 I)^{-1}, \quad s^2=(\rho)^2/m, \quad (\lambda)\neq 0.$$

Let A denote the matrix

$$A=A, \quad (\lambda)=0 \; ; \; A=\begin{pmatrix} A \\ (\lambda) I \end{pmatrix}, \quad (\lambda)\neq 0, \qquad (3.4)$$

let r denote the residual vector

$$r=r, \quad (\lambda)=0 \; ; \; r=\begin{pmatrix} b \\ 0 \end{pmatrix}-Ax, \quad (\lambda)\neq 0 \qquad (3.5)$$

corresponding to an iterate x, so that $(\rho)=||r||$ is the function being minimized, and let $||A||$ denote the Frobenius (Euclidean) norm of A. Then the routine accepts x as a solution if it is estimated that one of the following two conditions is satisfied:

$$(\rho)\leq tol_1 ||A||.||x||+tol_2 ||b|| \qquad (3.6)$$

$$||A^T r||\leq tol_1 ||A||(\rho) \qquad (3.7)$$

where tol_1 and tol_2 are user-supplied tolerances which estimate the relative errors in A and b respectively. Condition (3.6) is appropriate for compatible problems where, in theory, we expect the residual to be zero and will be satisfied by an acceptable solution x to a compatible problem. Condition (3.7) is appropriate for incompatible systems where we do not expect the residual to be zero and is based upon the observation that, in theory,

$$A^T r=0$$

when x is a solution to the least-squares problem, and so (3.7) will be satisfied by an acceptable solution x to a linear least-squares problem.

The routine also includes a test to prevent convergence to solutions, x, with unacceptably large elements. This can happen if A is nearly singular or is nearly rank deficient. If we let

the singular values of A be

$$(\sigma)_1 >=(\sigma)_2 >=...>=(\sigma)_n >=0$$

then the condition number of A is defined as

$$cond(A)=(\sigma)_1 /(\sigma)_k$$

where $(\sigma)_k$ is the smallest non-zero singular value of A and

hence k is the rank of A. When k<n, then A is rank deficient, the least-squares solution is not unique and F04QAF will normally

converge to the minimal length solution. In practice A will not have exactly zero singular values, but may instead have small singular values that we wish to regard as zero.

The routine provides for this possibility by terminating if

$$\mathrm{cond}(A) \geq c_{lim} \qquad (3.8)$$

where c_{lim} is a user-supplied limit on the condition number of A. For problem (3.1) termination with this condition indicates that A is nearly singular and for problem (3.2) indicates that A is nearly rank deficient and so has near linear dependencies in its columns. In this case inspection of $||r||$, $||A^T r||$ and $||x||$, which are all returned by the routine, will indicate whether or not an acceptable solution has been found. Condition (3.8), perhaps in conjunction with (lambda)/=0, can be used to try and ' regularise' least-squares solutions. A full discussion of the stopping criteria is given in Section 6 of reference Paige and Saunders [1].

Introduction of a non-zero damping parameter (lambda) tends to reduce the size of the computed solution and to make its components less sensitive to changes in the data, and F04QAF is applicable when a value of (lambda) is known a priori. To have an effect, (lambda) should normally be at least \/(epsilon)||A|| where (epsilon) is the machine precision. For further discussion see Paige and Saunders [2] and the references given there.

Whenever possible the matrix A should be scaled so that the relative errors in the elements of A are all of comparable size. Such a scaling helps to prevent the least-squares problem from being unnecessarily sensitive to data errors and will normally reduce the number of iterations required. At the very least, in the absence of better information, the columns of A should be scaled to have roughly equal column length.

4. References

[1] Paige C C and Saunders M A (1982) LSQR: An Algorithm for Sparse Linear Equations and Sparse Least-squares. ACM Trans. Math. Softw. 8 43--71.

[2] Paige C C and Saunders M A (1982) ALGORITHM 583 LSQR: Sparse Linear Equations and Least-squares Problems. ACM Trans. Math. Softw. 8 195--209.

5. Parameters

1: M -- INTEGER Input
On entry: m, the number of rows of the matrix A.
Constraint: M >= 1.

2: N -- INTEGER Input
On entry: n, the number of columns of the matrix A.
Constraint: N >= 1.

3: B(M) -- DOUBLE PRECISION array Input/Output
On entry: the right-hand side vector b. On exit: the array is overwritten.

4: X(N) -- DOUBLE PRECISION array Output
On exit: the solution vector x.

5: SE(N) -- DOUBLE PRECISION array Output
On exit: the estimates of the standard errors of the components of x. Thus SE(i) contains an estimate of the element v_{ii} of the estimated variance-covariance matrix V.
The estimates returned in SE will be the lower bounds on the actual estimated standard errors, but will usually have at least one correct figure.

6: APROD -- SUBROUTINE, supplied by the user.
External Procedure

APROD must perform the operations $y:=y+Ax$ and $x:=x+A^T y$ for given vectors x and y.

Its specification is:

```
 SUBROUTINE APROD (MODE, M, N, X, Y, RWORK,
1                  LRWORK, IWORK, LIWORK)
 INTEGER          MODE, M, N, LRWORK, LIWORK,
1                 IWORK(LIWORK)
 DOUBLE PRECISION X(N), Y(M), RWORK(LRWORK)
```

1: MODE -- INTEGER Input/Output
On entry: MODE specifies which operation is to be performed:
If MODE = 1, then APROD must compute $y+Ax$.

If MODE = 2, then APROD must compute $x+A^T y$.
On exit: MODE may be used as a flag to indicate a

T

```
        failure in the computation of y+Ax or x+A y. If MODE is
        negative on exit from APROD, F04QAF will exit
        immediately with IFAIL set to MODE.

    2:  M -- INTEGER                                      Input
        On entry: m, the number of rows of A.

    3:  N -- INTEGER                                      Input
        On entry: n, the number of columns of A.

    4:  X(N) -- DOUBLE PRECISION array             Input/Output
        On entry: the vector x. On exit: if MODE = 1, X must be
        unchanged;

                                   T
        If MODE = 2, X must contain x+A y.

    5:  Y(M) -- DOUBLE PRECISION array             Input/Output
        On entry: the vector y. On exit: if MODE = 1, Y must
        contain y+Ax;

        If MODE = 2, Y must be unchanged.

    6:  RWORK(LRWORK) -- DOUBLE PRECISION array  User Workspace

    7:  LRWORK -- INTEGER                                 Input

    8:  IWORK(LIWORK) -- INTEGER array           User Workspace

    9:  LIWORK -- INTEGER                                 Input
        APROD is called from F04QAF with the parameters RWORK,
        LRWORK, IWORK and LIWORK as supplied to F04QAF. The
        user is free to use the arrays RWORK and IWORK to
        supply information to APROD as an alternative to using
        COMMON.
    APROD must be declared as EXTERNAL in the (sub)program
    from which F04QAF is called. Parameters denoted as
    Input must not be changed by this procedure.

 7: DAMP -- DOUBLE PRECISION                              Input
    On entry: the value (lambda). If either problem (3.1) or
    problem (3.2) is to be solved, then DAMP must be supplied as
    zero.

 8: ATOL -- DOUBLE PRECISION                              Input
    On entry: the tolerance, tol , of the convergence criteria
                                1
    (3.6) and (3.7); it should be an estimate of the largest
    relative error in the elements of A. For example, if the
    elements of A are correct to about 4 significant figures,
```

then ATOL should be set to about $5*10^{-4}$. If ATOL is supplied as less than (epsilon), where (epsilon) is the machine precision, then the value (epsilon) is used in place of ATOL.

9: BTOL -- DOUBLE PRECISION Input
On entry: the tolerance, tol_2, of the convergence criterion (3.6); it should be an estimate of the largest relative error in the elements of B. For example, if the elements of B are correct to about 4 significant figures, then BTOL should be set to about $5*10^{-4}$. If BTOL is supplied as less than (epsilon), where (epsilon) is the machine precision, then the value (epsilon) is used in place of BTOL.

10: CONLIM -- DOUBLE PRECISION Input
On entry: the value c_{lim} of equation (3.8); it should be an upper limit on the condition number of A. CONLIM should not normally be chosen much larger than 1.0/ATOL. If CONLIM is supplied as zero then the value 1.0/(epsilon), where (epsilon) is the machine precision, is used in place of CONLIM.

11: ITNLIM -- INTEGER Input
On entry: an upper limit on the number of iterations. If ITNLIM <= 0, then the value N is used in place of ITNLIM, but for ill-conditioned problems a higher value of ITNLIM is likely to be necessary.

12: MSGLVL -- INTEGER Input
On entry: the level of printing from F04QAF. If MSGLVL <= 0, then no printing occurs, but otherwise messages will be output on the advisory message channel (see X04ABF). A description of the printed output is given in Section 5.2 below. The level of printing is determined as follows:

MSGLVL <= 0
No printing.

MSGLVL = 1
A brief summary is printed just prior to return from F04QAF.

MSGLVL >= 2
A summary line is printed periodically to monitor the progress of F04QAF, together with a brief summary just

```
      prior to return from F04QAF.

 13:  ITN -- INTEGER                                        Output
      On exit: the number of iterations performed.

 14:  ANORM -- DOUBLE PRECISION                             Output


      On exit: an estimate of ||A|| for the matrix A of equation
      (3.4).

 15:  ACOND -- DOUBLE PRECISION                             Output


      On exit: an estimate of cond(A) which is a lower bound.

 16:  RNORM -- DOUBLE PRECISION                             Output


      On exit: an estimate of ||r|| for the residual,  r, of
      equation (3.5) corresponding to the solution x returned in


      X. Note that ||r|| is the function being minimized.

 17:  ARNORM -- DOUBLE PRECISION                            Output
                                     T
      On exit: an estimate of the ||A r|| corresponding to the
      solution x returned in X.

 18:  XNORM -- DOUBLE PRECISION                             Output
      On exit: an estimate of ||x|| for the solution x returned in
      X.

 19:  WORK(2*N) -- DOUBLE PRECISION array                Workspace

 20:  RWORK(LRWORK) -- DOUBLE PRECISION array       User Workspace
      RWORK is not used by F04QAF, but is passed directly to
      routine APROD and may be used to pass information to that
      routine.

 21:  LRWORK -- INTEGER                                      Input
      On entry: the length of the array RWORK as declared in the
      (sub)program from which F04QAF is called. Constraint: LRWORK
      >= 1.

 22:  IWORK(LIWORK) -- INTEGER array                User Workspace
      IWORK is not used by F04QAF, but is passed directly to
      routine APROD and may be used to pass information to that
      routine.
```

23: LIWORK -- INTEGER Input
On entry: the length of the array IWORK as declared in the (sub)program from which F04QAF is called. Constraint: LIWORK >= 1.

24: INFORM -- INTEGER Output
On exit: the reason for termination of F04QAF as follows:

INFORM = 0
The exact solution is x=0. No iterations are performed in this case.

INFORM = 1
The termination criterion of equation (3.6) has been satisfied with tol_1 and tol_2 as the values supplied in ATOL and BTOL respectively.

INFORM = 2
The termination criterion of equation (3.7) has been satisfied with tol_1 as the value supplied in ATOL.

INFORM = 3
The termination criterion of equation (3.6) has been satisfied with tol_1 and/or tol_2 as the value (epsilon), where (epsilon) is the machine precision. One or both of the values supplied in ATOL and BTOL must have been less than (epsilon) and was too small for this machine.

INFORM = 4
The termination criterion of equation (3.7) has been satisfied with tol_1 as the value (epsilon), where (epsilon) is the machine precision. The value supplied in ATOL must have been less than (epsilon) and was too small for this machine.

The values INFORM = 5, INFORM = 6 and INFORM = 7 correspond to failure with IFAIL = 2, IFAIL = 3 and IFAIL = 4 respectively (see Section 6) and when IFAIL is negative INFORM will be set to the same negative value.

25: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see

Section 6).

5.1. Description of the printed output

When MSGLVL > 0, then F04QAF will produce output (except in the case where the routine fails with IFAIL = 1) on the advisory message channel (see X04ABF).

When MSGLVL >= 2 then a summary line is printed periodically giving the following information:

Output	Meaning
ITN	Iteration number, k.
X(1)	The first element of the current iterate x_k.
FUNCTION	The current value of the function, (rho), being minimized.
COMPAT	An estimate of $\|\|r_k\|\|/\|\|b\|\|$, where r_k is the residual corresponding to x_k. This value should converge to zero (in theory) if and only if the problem is compatible. COMPAT decreases monotonically.
INCOMPAT	An estimate of $\|\|A^T r_k\|\|/(\|\|A\|\|\|\|r_k\|\|)$ which should converge to zero if and only if at the solution (rho) is non-zero. INCOMPAT is not usually monotonic.
NRM(ABAR)	A monotonically increasing estimate of $\|\|A\|\|$.
COND(ABAR)	A monotonically increasing estimate of the condition number cond(A).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL< 0

A negative value of IFAIL indicates an exit from F04QAF because the user has set MODE negative in APROD. The value of IFAIL will be the same as the user's setting of MODE.

IFAIL= 1

On entry M < 1,

or N < 1,

or LRWORK < 1,

or LIWORK < 1.

IFAIL= 2

The condition of equation (3.8) has been satisfied for the value of c_{lim} supplied in CONLIM. If this failure is unexpected the user should check that APROD is working correctly. Although conditions (3.6) or (3.7) have not been satisfied, the values returned in RNORM, ARNORM and XNORM may nevertheless indicate that an acceptable solution has been reached.

IFAIL= 3

The conditions of equation (3.8) has been satisified for the value c_{lim} =1.0/(epsilon), where (epsilon) is the machine precision. The matrix A is nearly singular or rank deficient and the problem is too ill-conditioned for this machine. If this failure is unexpected, the user should check that APROD is working correctly.

IFAIL= 4

The limit on the number of iterations has been reached. The number of iterations required by F04QAF and the condition of the matrix A can depend strongly on the scaling of the problem. Poor scaling of the rows and columns of A should be avoided whenever possible.

7. Accuracy

When the problem is compatible, the computed solution x will satisfy the equation

$$r=b-Ax,$$

where an estimate of $||r||$ is returned in the parameter RNORM. When the problem is incompatible, the computed solution x will satisfy the equation

$$A^T r=e,$$

where an estimate of $||e||$ is returned in the parameter ARNORM. See also Section 6.2 of Paige and Saunders [1].

8. Further Comments

The time taken by the routine is likely to be principally determined by the time taken in APROD, which is called twice on each iteration, once with MODE = 1 and once with MODE = 2. The time taken per iteration by the remaining operations in F04QAF is approximately proportional to max(m,n).

The Lanczos process will usually converge more quickly if A is pre-conditioned by a non-singular matrix M that approximates A in some sense and is also chosen so that equations of the form My=c can efficiently be solved for y. Some discussion of pre-conditioning in the context of symmetric matrices is given in Section 3 of the document for F04MBF. In the context of F04QAF, problem (3.1) is equivalent to

$$(AM^{-1})y=b, \quad Mx=y$$

and problem (3.2) is equivalent to minimizing

$$(\rho)=||r||, \quad r=b-(AM^{-1})y, \quad Mx=y.$$

Note that the normal matrix $(AM^{-1})^T(AM^{-1})=M^{-T}(A^TA)M^{-1}$ so that the pre-conditioning AM^{-1} is equivalent to the pre-conditioning $M^{-T}(A^TA)M^{-1}$ of the normal matrix A^TA.

Pre-conditioning can be incorporated into F04QAF simply by coding the routine APROD to compute $y+AM^{-1}x$ and $x+M^{-T}A^Ty$ in place of $y+Ax$ and $x+A^Ty$ respectively, and then solving the equations $Mx=y$ for x on return from F04QAF. $y+AM^{-1}x$ should be computed by solving $Mz=x$ for z and then computing $y+Az$, and $x+M^{-T}A^{T}y$ should be computed by solving $M^{T}z=A^{T}y$ for z and then forming $x+z$.

9. Example

To solve the linear least-squares problem

$$\text{minimize } \rho=\|r\|, \quad r=b-Ax$$

where A is the 13 by 12 matrix and b is the 13 element vector given by

$$A=\begin{pmatrix}
1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
-1 & 0 & -1 & 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0\\
0 & -1 & 0 & -1 & 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & -1 & 0 & 0 & -1 & 4 & -1 & 0 & -1 & 0\\
0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 4 & -1 & 0 & -1\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1\\
1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1
\end{pmatrix},$$

$$b=-h^{2}\begin{pmatrix}
0\\ 0\\ 0\\ 1\\ 1\\ 0\\ 0\\ 1\\ 1\\ 0\\ 0\\ 0\\ -h^{-3}
\end{pmatrix}$$

with $h=0.1$.

Such a problem can arise by considering the Neumann problem on a rectangle

```
(delta)u
--------=0
```

```
                   (delta)n

    (delta)u           2           (delta)u      /
    --------=0   (nabla) u=g(x,y)   --------=0    |u=1
    (delta)n                       (delta)n      /
                                                 c

                   (delta)u
                   --------=0
                   (delta)n

where C is the boundary of the rectangle, and discretising as
illustrated below with the square mesh



         Please see figure in printed Reference Manual

The 12 by 12 symmetric part of A represents the difference
equations and the final row comes from the normalising condition.
The example program has g(x,y)=1 at all the internal mesh points,
but apart from this is written in a general manner so that the
number of rows (NROWS) and columns (NCOLS) in the grid can
readily be altered.

The example program is not reproduced here. The source code for
all example programs is distributed with the NAG Foundation
Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Linear Algebra Support Routines

— **nagf.ht** —

```
\begin{page}{manpageXXf06}{NAG Documentation: f06}
\beginscroll
\begin{verbatim}


     F06(3NAG)         Foundation Library (12/10/92)         F06(3NAG)
```

Chapter F06

Linear Algebra Support Routines

Contents of this Introduction:

1. Scope of the Chapter

This Chapter is concerned with basic linear algebra routines which perform elementary algebraic operations involving scalars, vectors and matrices.

2. Background to the Problems

All the routines in this Chapter meet the specification of the Basic Linear Algebra Subprograms (BLAS) as described in Lawson et al [6], Dongarra et al [3] and [4]. The first reference describes a set of routines concerned with operations on scalars and vectors: these will be referred to here as the Level-0 and the Level-1 BLAS; the second reference describes a set of routines concerned with matrix-vector operations: these will be referred to here as the Level-2 BLAS; and the third reference describes a set of routines concerned with matrix-matrix operations: these will be referred to here as the Level-3 BLAS. The terminology reflects the number of operations involved. For example, a Level-2 routine involves $0(n^2)$ operations for an n by n matrix.

Table 1.1 indicates the naming scheme for the routines in this Chapter. The heading 'mixed type' is for routines where a mixture of data types is involved, such as a routine that returns the real Euclidean length of a complex vector.

		Level-0	Level-1	Level-2	Level-3
'real'	BLAS routine	F06A F	F06E F	F06P F	F06Y F
'complex'	BLAS routine	-	F06G F	F06S F	F06Z F

Table 1.1

The routines in this chapter do not have full routine documents, but instead are covered by some relevant background material, in Section 2.2, together with general descriptions, in Section 4, sufficient to enable their use. As this chapter is concerned only with basic linear algebra operations, the routines will not normally be required by the general user. The functionality of each routine is indicated in Section 3 so that those users requiring these routines to build specialist linear algebra modules can determine which routines are of interest.

2.1. The Use of BLAS Names

Many of the routines in other chapters of the Library call the BLAS in this chapter. These routines are usually called by the BLAS name and so, for correct operation of the Library, it is essential that users do not attempt to link their own versions of these routines. If users are in any doubt about how to avoid this, please consult your local support staff or the NAG Response Centre.

The BLAS names are used in order to make use of efficient implementations of the routines when these exist. Such implementations are stringently tested before being used, to ensure that they correctly meet the specification of the BLAS, and that they return the desired accuracy (see, for example, Dongarra et al. [3] and [4]).

2.2. Background Information

Most of the routines in this chapter implement straightforward scalar, vector and matrix operations that need no further explanation beyond a statement of the purpose of the routine. In this section we give some additional background information for those few cases where additional explanation may be necessary.

2.2.1. Real plane rotations

Two routines in the chapter are concerned with setting up and applying plane rotations. For further background information see Golub and Van Loan [5].

A plane rotation matrix for the (i,j) plane, R_{ij}, is an orthogonal matrix that is different from the unit matrix only in the elements r_{ii}, r_{jj}, r_{ij} and r_{ji}. If we put

$$R=\begin{pmatrix} r_{ii} & r_{ij} \\ r_{ji} & r_{jj} \end{pmatrix},$$

then, in the real case, it is usual to choose R_{ij} so that

$$R=\begin{pmatrix} c & s \\ -s & c \end{pmatrix}, \quad c=\cos(\theta), \quad s=\sin(\theta). \qquad (2.1)$$

The application of plane rotations is straightforward and needs no further elaboration, so further comment is made only on the construction of plane rotations.

The most common use of plane rotations is to choose c and s so that for given a and b,

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix}\begin{pmatrix} a \\ b \end{pmatrix}=\begin{pmatrix} d \\ 0 \end{pmatrix} \qquad (2.2)$$

In such an application the matrix R is often termed a Givens rotation matrix.

The BLAS routine F06AAF(*) (DROTG), see Lawson et al [6] and Dodson and Grimes [1, 2], computes c, s and d as

$$d=(\sigma)(a^2+b^2)^{1/2},$$
$$c=\begin{cases} a/d, & d\neq 0 \\ 1, & d=0 \end{cases}, \quad s=\begin{cases} b/d, & d\neq 0 \\ 0, & d=0 \end{cases} \qquad (2.3)$$

where $(\sigma)=\begin{cases} \mathrm{sign}\ a, & |a|>|b| \\ \mathrm{sign}\ b, & |a|<=|b| \end{cases}$.

The value z defined as

$$z=\begin{cases} s, & |s|<c \text{ or } c=0 \\ 1/c, & 0<|c|<=s \end{cases} \qquad (2.4)$$

is also computed and this enables c and s to be reconstructed from the single value z as

$$c=\begin{cases} 0, & z=1 \\ (1-z^2)^{1/2}, & |z|<1 \\ 1/z, & |z|>1, \end{cases} \quad s=\begin{cases} 1, & z=1 \\ z, & |z|<1 \\ (1-c^2), & |z|>1. \end{cases}$$

2.3. References

[1] Dodson D S and Grimes R G (1982) Remark on Algorithm 539. ACM Trans Math Softw. 8 403--404.

[2] Dodson D S and Grimes R G (1982) Remark on Algorithm 539. ACM Trans. Math. Softw. 9 140.

[3] Dongarra J J, Du Croz J J, Hammarling S and Hanson R J (1988) An Extended Set of FORTRAN Basic Linear Algebra Subprograms. ACM Trans. Math. Softw. 14 1--32.

[4] Dongarra J J, Du Croz J J, Duff I S and Hammarling S (1990) A Set of Level 3 Basic Linear Algebra Subprograms. ACM Trans. Math. Softw. 16 1--28.

[5] Golub G H and Van Loan C F (1989) Matrix Computations (2nd Edition). Johns Hopkins University Press, Baltimore, Maryland.

[6] Lawson C L, Hanson R J, Kincaid D R and Krogh F T (1979)

Basic Linear Algebra Subprograms for Fortran Usage. ACM
Trans. Math. Softw. 5 308--325.

3. Recommendations on Choice and Use of Routines

This section lists the routines in each of the categories Level-0 (scalar), Level-1 (vector), Level-2 (matrix-vector and matrix) and Level-3 (matrix-matrix). The corresponding double precision BLAS name is indicated in brackets.

Within each section routines are listed in alphabetic order of the fifth character in the routine name, so that corresponding real and complex routines may have adjacent entries.

3.1. The Level-0 Scalar Routine

The Level-0 routine performs the scalar operation of generating a plane rotation.

F06AAF (DROTG) generates a real plane rotation.

3.2. The Level-1 Vector Routines

The Level-1 routines perform operations on or between vectors, such as computing dot products and Euclidean lengths.

F06EAF (DDOT)   computes the dot product of two real vectors

F06GAF (ZDOTU) computes the dot product of two complex vectors (unconjugated)

F06GBF (ZDOTC) computes the dot product of two complex vectors (conjugated)

F06ECF (DAXPY) adds a scalar times a vector to another real vector

F06GCF (ZAXPY) adds a scalar times a vector to another complex vector

F06EDF (DSCAL) multiplies a real vector by a scalar

F06GDF (ZSCAL) multiplies a complex vector by a scalar

F06JDF (ZDSCAL) multiplies a complex vector by a real scalar

F06EFF (DCOPY) copies a real vector

F06GFF (ZCOPY) copies a complex vector

```
F06EGF (DSWAP)  swaps two real vectors

F06GGF (ZSWAP)  swaps two complex vectors

F06EJF (DNRM2)  computes the Euclidean length of a real vector

F06JJF (DZNRM2) computes the Euclidean length of a complex vector

F06EKF (DASUM)  sums the absolute values of the elements of a
                real vector

F06JKF (DZASUM) sums the absolute values of the elements of a
                complex vector

F06JLF (IDAMAX) finds the index of the element of largest
                absolute value of a real vector

F06JMF (IZAMAX) finds the index of the element of largest
                absolute value of a complex vector

F06EPF (DROT)   applies a real plane rotation

3.3. The Level-2 Matrix-vector Routines

The Level-2 routines perform matrix-vector operations, such as
forming the product between a matrix and a vector.

F06PAF (DGEMV)  computes a matrix-vector product;
                real general matrix

F06SAF (ZGEMV)  computes a matrix-vector product;
                complex general matrix

F06PBF (DGBMV)  computes a matrix-vector product;
                real general band matrix

F06SBF (ZGBMV)  computes a matrix-vector product;
                complex general band matrix

F06PCF (DSYMV)  computes a matrix-vector product;
                real symmetric matrix

F06SCF (ZHEMV)  computes a matrix-vector product;
                complex Hermitian matrix

F06PDF (DSBMV)  computes a matrix-vector product;
                real symmetric band matrix

F06SDF (ZHBMV)  computes a matrix-vector product;
                complex Hermitian band matrix
```

F06PEF	(DSPMV)	computes a matrix-vector product; real symmetric packed matrix
F06SEF	(ZHPMV)	computes a matrix-vector product; complex Hermitian packed matrix
F06PFF	(DTRMV)	computes a matrix-vector product; real triangular matrix
F06SFF	(ZTRMV)	computes a matrix-vector product; complex triangular matrix
F06PGF	(DTBMV)	computes a matrix-vector product; real triangular band matrix
F06SGF	(ZTBMV)	computes a matrix-vector product; complex triangular band matrix
F06PHF	(DTPMV)	computes a matrix-vector product; real triangular packed matrix
F06SHF	(ZTPMV)	computes a matrix-vector product; complex triangular packed matrix
F06PJF	(DTRSV)	solves a system of equations; real triangular coefficient matrix
F06SJF	(ZTRSV)	solves a system of equations; complex triangular coefficient matrix
F06PKF	(DTBSV)	solves a system of equations; real triangular band coefficient matrix
F06SKF	(ZTBSV)	solves a system of equations; complex triangular band coefficient matrix
F06PLF	(DTPSV)	solves a system of equations; real triangular packed coefficient matrix
F06SLF	(ZTPSV)	solves a system of equations; complex triangular packed coefficient matrix
F06PMF	(DGER)	performs a rank-one update; real general matrix
F06SMF	(ZGERU)	performs a rank-one update; complex general matrix (unconjugated vector)
F06SNF	(ZGERC)	performs a rank-one update;

```
                complex general matrix (conjugated vector)

F06PPF (DSYR)   performs a rank-one update;
                real symmetric matrix

F06SPF (ZHER)   performs a rank-one update;
                complex Hermitian matrix

F06PQF (DSPR)   performs a rank-one update;
                real symmetric packed matrix

F06SQF (ZHPR)   performs a rank-one update;
                complex Hermitian packed matrix

F06PRF (DSYR2)  performs a rank-two update;
                real symmetric matrix

F06SRF (ZHER2)  performs a rank-two update;
                complex Hermitian matrix

F06PSF (DSPR2)  performs a rank-two update;
                real symmetric packed matrix

F06SSF (ZHPR2)  performs a rank-two update;
                complex Hermitian packed matrix

3.4. The Level-3 Matrix-matrix Routines

The Level-3 routines perform matrix-matrix operations, such as
forming the product of two matrices.

F06YAF (DGEMM)  computes a matrix-matrix product; two real
                rectangular matrices

F06ZAF (ZGEMM)  computes a matrix-matrix product; two complex
                rectangular matrices

F06YCF (DSYMM)  computes a matrix-matrix product; one real
                symmetric matrix, one real rectangular matrix

F06ZCF (ZHEMM)  computes a matrix-matrix product; one complex
                Hermitian matrix, one complex rectangular matrix

F06YFF (DTRMM)  computes a matrix-matrix product; one real
                triangular matrix, one real rectangular matrix

F06ZFF (ZTRMM)  computes a matrix-matrix product; one complex
                triangular matrix, one complex rectangular matrix

F06YJF (DTRSM)  solves a system of equations with multiple right-
```

```
                 hand sides, real triangular coefficient matrix

F06ZJF (ZTRSM)   solves a system of equations with multiple right-
                 hand sides, complex triangular coefficient matrix

F06YPF (DSYRK)   performs a rank-k update of a real symmetric
                 matrix

F06ZPF (ZHERK)   performs a rank-k update of a complex hermitian
                 matrix

F06YRF (DSYR2K)  performs a rank-2k update of a real symmetric
                 matrix

F06ZRF (ZHER2K)  performs a rank-2k update of a complex Hermitian
                 matrix

F06ZTF (ZSYMM)   computes a matrix-matrix product: one complex
                 symmetric matrix, one complex rectangular matrix

F06ZUF (ZSYRK)   performs a rank-k update of a complex symmetric
                 matrix

F06ZWF (ZSYR2K)  performs a rank-2k update of a complex symmetric
                 matrix
```

4. Description of the F06 Routines

In this section we describe the purpose of each routine and give information on the parameter lists, where appropriate indicating their general nature. Usually the association between the routine arguments and the mathematical variables is obvious and in such cases a description of the argument is omitted.

Within each section, the parameter lists for all routines are presented, followed by the purpose of the routines and information on the parameter lists. The double precision BLAS names are given in ENTRY statements.

Within each section routines are listed in alphabetic order of the fifth character in the routine name, so that corresponding real and complex routines may have adjacent entries.

4.1. The Level-0 Scalar Routines

The scalar routines have no array arguments.

```
SUBROUTINE   F06AAF( A,B,C,S )
ENTRY        DROTG ( A,B,C,S )
```

```
DOUBLE PRECISION     A,B,C,S
```

F06AAF(*) generates the parameters c and s of a Givens rotation as defined by equations (2.3) and (2.4), from given a and b. On exit, A is overwritten by d and B is overwritten by z.

4.2. The Level-1 Vector Routines

The vector routines all have one or more one-dimensional arrays as arguments, each representing a vector.

The length of each vector, n, is represented by the argument N, and the routines may be called with non-positive values of N, in which case the routine returns immediately except for the functions, which set the function value to zero before returning.

In addition to the argument N, each array argument is also associated with an increment argument that immediately follows the array argument, and whose name consists of the three characters INC, followed by the name of the array. For example, a vector x will be represented by the two arguments X, INCX. The increment argument is the spacing (stride) in the array for which the elements of the vector occur. For instance, if INCX = 2, then the elements of x are in locations X(1),X(3),...,X(2*N-1) of the array X and the intermediate locations X(2),X(4),...,X(2*N-2) are not referenced.

Thus when INCX > 0, the vector element x_i is in the array element X(1+(i-1)*INCX). When INCX <= 0 the elements are stored in the reverse order so that the vector element x_i is in the array element X(1-(n-i)*INCX) and hence, in particular, the element x_n is in X(1). The declared length of the array X in the calling (sub)program must be at least (1+(N-1)*|INCX|).

Non-positive increments are permitted only for those routines that have more than one array argument. While zero increments are formally permitted for such routines, their use in Chapter F06 is strongly discouraged since the effect may be implementation dependent.

```
DOUBLE PRECISION FUNCTION F06EAF ( N,      X,INCX,Y,INCY )
DOUBLE PRECISION          DDOT
ENTRY                     DDOT   ( N,      X,INCX,Y,INCY )
INTEGER                            N,        INCX,  INCY
DOUBLE PRECISION                           X(*),  Y(*)
```

```
COMPLEX(KIND(1.0D0)) FUNCTION F06GAF ( N,      X,INCX,Y,INCY )
COMPLEX(KIND(1.0D0))          ZDOTU
ENTRY                         ZDOTU  ( N,      X,INCX,Y,INCY )
INTEGER                                N,        INCX,  INCY
COMPLEX(KIND(1.0D0))                           X(*),  Y(*)

COMPLEX(KIND(1.0D0)) FUNCTION F06GBF ( N,      X,INCX,Y,INCY )
COMPLEX(KIND(1.0D0))          ZDOTC
ENTRY                         ZDOTC  ( N,      X,INCX,Y,INCY )
INTEGER                                N,        INCX,  INCY
COMPLEX(KIND(1.0D0))                           X(*),  Y(*)

SUBROUTINE                 F06ECF ( N,ALPHA,X,INCX,Y,INCY )
ENTRY                      DAXPY  ( N,ALPHA,X,INCX,Y,INCY )
INTEGER                             N,        INCX,  INCY
DOUBLE PRECISION                      ALPHA,X(*),  Y(*)

SUBROUTINE                 F06GCF ( N,ALPHA,X,INCX,Y,INCY )
ENTRY                      ZAXPY  ( N,ALPHA,X,INCX,Y,INCY )
INTEGER                             N,        INCX,  INCY
COMPLEX(KIND(1.0D0))                  ALPHA,X(*),  Y(*)

SUBROUTINE                 F06EDF ( N,ALPHA,X,INCX )
ENTRY                      DSCAL  ( N,ALPHA,X,INCX )
INTEGER                             N,        INCX
DOUBLE PRECISION                      ALPHA,X(*)

SUBROUTINE                 F06GDF ( N,ALPHA,X,INCX )
ENTRY                      ZSCAL  ( N,ALPHA,X,INCX )
INTEGER                             N,        INCX
COMPLEX(KIND(1.0D0))                  ALPHA,X(*)

SUBROUTINE                 F06JDF ( N,ALPHA,X,INCX )
ENTRY                      ZDSCAL ( N,ALPHA,X,INCX )
INTEGER                             N,        INCX
DOUBLE PRECISION           ALPHA
COMPLEX(KIND(1.0D0))                        X(*)

SUBROUTINE                 F06EFF ( N,      X,INCX,Y,INCY )
ENTRY                      DCOPY  ( N,      X,INCX,Y,INCY )
INTEGER                             N,        INCX,  INCY
DOUBLE PRECISION                            X(*),  Y(*)

SUBROUTINE                 F06GFF ( N,      X,INCX,Y,INCY )
ENTRY                      ZCOPY  ( N,      X,INCX,Y,INCY )
INTEGER                             N,        INCX,  INCY
COMPLEX(KIND(1.0D0))                        X(*),  Y(*)

SUBROUTINE                 F06EGF ( N,      X,INCX,Y,INCY )
```

```
      ENTRY                    DSWAP  ( N,      X,INCX,Y,INCY )
      INTEGER                           N,        INCX,  INCY
      DOUBLE PRECISION                          X(*),  Y(*)

      SUBROUTINE               F06GGF ( N,      X,INCX,Y,INCY )
      ENTRY                    ZSWAP  ( N,      X,INCX,Y,INCY )
      INTEGER                           N,        INCX,  INCY
      COMPLEX(KIND(1.0D0))                      X(*),  Y(*)

      DOUBLE PRECISION FUNCTION F06EJF ( N,      X,INCX )
      DOUBLE PRECISION         DNRM2
      ENTRY                    DNRM2  ( N,      X,INCX )
      INTEGER                           N,        INCX
      DOUBLE PRECISION                          X(*)

      DOUBLE PRECISION FUNCTION F06JJF ( N,      X,INCX )
      DOUBLE PRECISION         DZNRM2
      ENTRY                    DZNRM2 ( N,      X,INCX )
      INTEGER                           N,        INCX
      COMPLEX(KIND(1.0D0))                      X(*)

      DOUBLE PRECISION FUNCTION F06EKF ( N,      X,INCX )
      DOUBLE PRECISION         DASUM
      ENTRY                    DASUM  ( N,      X,INCX )
      INTEGER                           N,        INCX
      DOUBLE PRECISION                          X(*)

      DOUBLE PRECISION FUNCTION F06JKF ( N,      X,INCX )
      DOUBLE PRECISION         DZASUM
      ENTRY                    DZASUM ( N,      X,INCX )
      INTEGER                           N,        INCX
      COMPLEX(KIND(1.0D0))                      X(*)

      INTEGER FUNCTION         F06JLF ( N,      X,INCX )
      INTEGER                  IDAMAX
      ENTRY                    IDAMAX ( N,      X,INCX )
      INTEGER                           N,        INCX
      DOUBLE PRECISION                           X(*)

      INTEGER FUNCTION         F06JMF ( N,      X,INCX )
      INTEGER                  IZAMAX
      ENTRY                    IZAMAX ( N,      X,INCX )
      INTEGER                           N,        INCX
      COMPLEX(KIND(1.0D0))                      X(*)

      SUBROUTINE               F06EPF ( N,      X,INCX,Y,INCY,C,S )
      ENTRY                    DROT   ( N,      X,INCX,Y,INCY,C,S )
      INTEGER                           N,        INCX,  INCY
      DOUBLE PRECISION                          X(*),  Y(*),  C,S
```

F06EAF(*) and F06GAF(*)

return the dot product $x^T y$.

F06GBF(*)

returns the dot product $x^H y$, where x^H denotes the complex conjugate of x^T.

F06ECF(*) and F06GCF(*)

perform the operation y<-(alpha)x+y, often called an axpy operation.

F06EDF(*), F06GDF(*) and F06JDF(*)

perform the operation x<-(alpha)x.

F06EFF(*) and F06GFF(*)

perform the operation y<-x.

F06EGF(*) and F06GGF(*)

perform the operation x<=>y, that is x and y are swapped.

F06EJF(*) and F06JJF(*)

return the value $||x||_2$ defined by $||x||_2 = \left(\sum_{i=1}^{n} |x_i|^2 \right)^{1/2}$.

F06EKF(*)

returns the value $||x||_1$ defined by $||x||_1 = \sum_{i=1}^{n} |x_i|$.

F06JKF(*)

returns the value asum defined by asum$= \sum^{n} (|\mathrm{Re}(x)|+|\mathrm{Im}(x)|)$.

```
                                        --        i        i
                                        i=1
```

F06JLF(*)

returns the first index j such that $|x_j|=\max_i |x_i|$.

F06JMF(*)

returns the first index j such that $|Re(x_j)|+|Im(x_j)|=\max_i (|(Re(x_i)|+|Im(x_i)|)$.

F06EPF(*)

performs the plane rotation $\begin{pmatrix} x^T \\ y^T \end{pmatrix} \leftarrow \begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} x^T \\ y^T \end{pmatrix}$.

4.3. The Level-2 Matrix-vector Routines

The matrix-vector routines all have one array argument representing a matrix; usually this is a two-dimensional array but in some cases the matrix is represented by a one-dimensional array.

The size of the matrix is determined by the arguments M and N for an m by n rectangular matrix; and by the argument N for an n by n symmetric, Hermitian, or triangular matrix. Note that it is permissible to call the routines with M or N = 0, in which case the routines exit immediately without referencing their array arguments. For band matrices, the bandwidth is determined by the arguments KL and KU for a rectangular matrix with kl sub-diagonals and ku super-diagonals; and by the argument K for a symmetric, Hermitian, or triangular matrix with k sub-diagonals and/or super-diagonals.

The description of the matrix consists either of the array name (A) followed by the first dimension of the array as declared in the calling (sub)program (LDA), when the matrix is being stored in a two-dimensional array; or the array name (AP) alone when the matrix is being stored as a (packed) vector. In the former case the actual array must contain at least ((n-1)d+l) elements, where d is the first dimension of the array, d>=l , and l=m for arrays representing general matrices, l=n for arrays representing symmetric, Hermitian and triangular matrices, l=kl+ku+1 for arrays representing general band matrices and l=k+1 for symmetric, Hermitian and triangular band matrices. For one-dimensional arrays representing matrices (packed storage) the actual array must contain at least $\frac{1}{2}n(n+1)$ elements.

As with the vector routines, vectors are represented by one-dimensional arrays together with a corresponding increment argument (see Section 4.2). The only difference is that for these routines a zero increment is not permitted.

When the vector x consists of k elements then the declared length of the array X in the calling (sub)program must be at least (1+(k-1)|INCX|).

The arguments that specify options are character arguments with the names TRANS, UPLO and DIAG. TRANS is used by the matrix-vector product routines as follows:

Value	Meaning
'N' or 'n'	Operate with the matrix
'T' or 't'	Operate with the transpose of the matrix
'C' or 'c'	Operate with the conjugate transpose of the matrix

In the real case the values 'T', 't', 'C' and 'c' have the same meaning.

UPLO is used by the Hermitian, symmetric, and triangular matrix routines to specify whether the upper or lower triangle is being referenced as follows:

Value	Meaning
'U' or 'u'	Upper triangle
'L' or 'l'	Lower triangle

DIAG is used by the triangular matrix routines to specify whether or not the matrix is unit triangular, as follows:

Value	Meaning
'U' or 'u'	Unit triangular
'N' or 'n'	Non-unit triangular

When DIAG is supplied as 'U' or 'u' the diagonal elements are not referenced.

It is worth noting that actual character arguments in Fortran may be longer than the corresponding dummy arguments. So that, for example, the value ’T’ for TRANS may be passed as ’TRANSPOSE’.

The routines for real symmetric and complex Hermitian matrices allow for the matrix to be stored in either the upper (UPLO = ’U to be packed in a one-dimensional array. In the latter case the upper triangle may be packed sequentially column by column (UPLO = ’U’), or the lower triangle may be packed sequentially column by column (UPLO = ’L’). Note that for real symmetric matrices packing the upper triangle by column is equivalent to packing the lower triangle by rows, and packing the lower triangle by columns is equivalent to packing the upper triangle by rows. (For complex Hermitian matrices the only difference is that the off-diagonal elements are conjugated.)

For triangular matrices the argument UPLO serves to define whether the matrix is upper (UPLO = ’U’) or lower (UPLO = ’L’) triangular. In packed storage the triangle has to be packed by column.

The band matrix routines allow storage so that the jth column of the matrix is stored in the jth column of the Fortran array. For a general band matrix the diagonal of the matrix is stored in the (ku+1)th row of the array. For a Hermitian or symmetric matrix either the upper triangle (UPLO = ’U’) may be stored in which case the leading diagonal is in the (k+1)th row of the array, or the lower triangle (UPLO = ’L’) may be stored in which case the leading diagonal is in the first row of the array. For an upper triangular band matrix (UPLO = ’U’) the leading diagonal is in the (k+1)th row of the array and for a lower triangular band matrix (UPLO = ’L’) the leading diagonal is in the first row.

For a Hermitian matrix the imaginary parts of the diagonal elements are of course zero and thus the imaginary parts of the corresponding Fortran array elements need not be set, but are assumed to be zero.

For packed triangular matrices the same storage layout is used whether or not DIAG = ’U’, i.e., space is left for the diagonal elements even if those array elements are not referenced.

Throughout the following sections A^H denotes the complex conjugate of A^T.

```
SUBROUTINE F06PAF( TRANS,M,N,ALPHA,A,LDA,X,INCX,BETA,Y,INCY )
ENTRY DGEMV      ( TRANS,M,N,ALPHA,A,LDA,X,INCX,BETA,Y,INCY )
CHARACTER*1        TRANS
INTEGER            M,N,LDA,INCX,INCY
DOUBLE PRECISION   ALPHA,A(LDA,*),X(*),BETA,Y(*)

SUBROUTINE F06SAF( TRANS,M,N,ALPHA,A,LDA,X,INCX,BETA,Y,INCY )
ENTRY ZGEMV      ( TRANS,M,N,ALPHA,A,LDA,X,INCX,BETA,Y,INCY )
CHARACTER*1          TRANS
INTEGER              M,N,LDA,INCX,INCY
COMPLEX(KIND(1.0D0)) ALPHA,A(LDA,*),X(*),BETA,Y(*)

SUBROUTINE F06PBF( TRANS,M,N,KL,KU,ALPHA,A,LDA,X,INCX,BETA,Y,INCY )
ENTRY DGBMV      ( TRANS,M,N,KL,KU,ALPHA,A,LDA,X,INCX,BETA,Y,INCY )
CHARACTER*1        TRANS
INTEGER            M,N,KL,KU,LDA,INCX,INCY
DOUBLE PRECISION   ALPHA,A(LDA,*),X(*),BETA,Y(*)

SUBROUTINE F06SBF( TRANS,M,N,KL,KU,ALPHA,A,LDA,X,INCX,BETA,Y,INCY )
ENTRY ZGBMV      ( TRANS,M,N,KL,KU,ALPHA,A,LDA,X,INCX,BETA,Y,INCY )
CHARACTER*1          TRANS
INTEGER              M,N,KL,KU,LDA,INCX,INCY
COMPLEX(KIND(1.0D0)) ALPHA,A(LDA,*),X(*),BETA,Y(*)

SUBROUTINE F06PCF( UPLO,N,ALPHA,A,LDA,X,INCX,BETA,Y,INCY )
ENTRY DSYMV      ( UPLO,N,ALPHA,A,LDA,X,INCX,BETA,Y,INCY )
CHARACTER*1        UPLO
INTEGER            N,LDA,INCX,INCY
DOUBLE PRECISION   ALPHA,A(LDA,*),X(*),BETA,Y(*)

SUBROUTINE F06SCF( UPLO,N,ALPHA,A,LDA,X,INCX,BETA,Y,INCY )
ENTRY ZHEMV      ( UPLO,N,ALPHA,A,LDA,X,INCX,BETA,Y,INCY )
CHARACTER*1          UPLO
INTEGER              N,LDA,INCX,INCY
COMPLEX(KIND(1.0D0)) ALPHA,A(LDA,*),X(*),BETA,Y(*)

SUBROUTINE F06PDF( UPLO,N,K,ALPHA,A,LDA,X,INCX,BETA,Y,INCY )
ENTRY DSBMV      ( UPLO,N,K,ALPHA,A,LDA,X,INCX,BETA,Y,INCY )
CHARACTER*1        UPLO
INTEGER            N,K,LDA,INCX,INCY
DOUBLE PRECISION   ALPHA,A(LDA,*),X(*),BETA,Y(*)

SUBROUTINE F06SDF( UPLO,N,K,ALPHA,A,LDA,X,INCX,BETA,Y,INCY )
ENTRY ZHBMV      ( UPLO,N,K,ALPHA,A,LDA,X,INCX,BETA,Y,INCY )
CHARACTER*1          UPLO
INTEGER              N,K,LDA,INCX,INCY
COMPLEX(KIND(1.0D0)) ALPHA,A(LDA,*),X(*),BETA,Y(*)

SUBROUTINE F06PEF( UPLO,N,ALPHA,AP,X,INCX,BETA,Y,INCY )
ENTRY DSPMV      ( UPLO,N,ALPHA,AP,X,INCX,BETA,Y,INCY )
```

```
CHARACTER*1        UPLO
INTEGER            N,INCX,INCY
DOUBLE PRECISION   ALPHA,AP(*),X(*),BETA,Y(*)

SUBROUTINE F06SEF( UPLO,N,ALPHA,AP,X,INCX,BETA,Y,INCY )
ENTRY ZHPMV      ( UPLO,N,ALPHA,AP,X,INCX,BETA,Y,INCY )
CHARACTER*1          UPLO
INTEGER              N,INCX,INCY
COMPLEX(KIND(1.0D0)) ALPHA,AP(*),X(*),BETA,Y(*)

SUBROUTINE F06PFF( UPLO,TRANS,DIAG,N,A,LDA,X,INCX )
ENTRY DTRMV      ( UPLO,TRANS,DIAG,N,A,LDA,X,INCX )
CHARACTER*1        UPLO,TRANS,DIAG
INTEGER            N,LDA,INCX
DOUBLE PRECISION   A(LDA,*),X(*)

SUBROUTINE F06SFF( UPLO,TRANS,DIAG,N,A,LDA,X,INCX )
ENTRY ZTRMV      ( UPLO,TRANS,DIAG,N,A,LDA,X,INCX )
CHARACTER*1          UPLO,TRANS,DIAG
INTEGER              N,LDA,INCX
COMPLEX(KIND(1.0D0)) A(LDA,*),X(*)

SUBROUTINE F06PGF( UPLO,TRANS,DIAG,N,K,A,LDA,X,INCX )
ENTRY DTBMV      ( UPLO,TRANS,DIAG,N,K,A,LDA,X,INCX )
CHARACTER*1        UPLO,TRANS,DIAG
INTEGER            N,K,LDA,INCX
DOUBLE PRECISION   A(LDA,*),X(*)

SUBROUTINE F06SGF( UPLO,TRANS,DIAG,N,K,A,LDA,X,INCX )
ENTRY ZTBMV      ( UPLO,TRANS,DIAG,N,K,A,LDA,X,INCX )
CHARACTER*1          UPLO,TRANS,DIAG
INTEGER              N,K,LDA,INCX
COMPLEX(KIND(1.0D0)) A(LDA,*),X(*)

SUBROUTINE F06PHF( UPLO,TRANS,DIAG,N,AP,X,INCX )
ENTRY DTPMV      ( UPLO,TRANS,DIAG,N,AP,X,INCX )
CHARACTER*1        UPLO,TRANS,DIAG
INTEGER            N,INCX
DOUBLE PRECISION   AP(*),X(*)

SUBROUTINE F06SHF( UPLO,TRANS,DIAG,N,AP,X,INCX )
ENTRY ZTPMV      ( UPLO,TRANS,DIAG,N,AP,X,INCX )
CHARACTER*1          UPLO,TRANS,DIAG
INTEGER              N,INCX
COMPLEX(KIND(1.0D0)) AP(*),X(*)

SUBROUTINE F06PJF( UPLO,TRANS,DIAG,N,A,LDA,X,INCX )
ENTRY DTRSV      ( UPLO,TRANS,DIAG,N,A,LDA,X,INCX )
CHARACTER*1        UPLO,TRANS,DIAG
INTEGER            N,LDA,INCX
```

```
DOUBLE PRECISION   A(LDA,*),X(*)

SUBROUTINE F06SJF( UPLO,TRANS,DIAG,N,A,LDA,X,INCX )
ENTRY ZTRSV      ( UPLO,TRANS,DIAG,N,A,LDA,X,INCX )
CHARACTER*1          UPLO,TRANS,DIAG
INTEGER              N,LDA,INCX
COMPLEX(KIND(1.0D0)) A(LDA,*),X(*)

SUBROUTINE F06PKF( UPLO,TRANS,DIAG,N,K,A,LDA,X,INCX )
ENTRY DTBSV      ( UPLO,TRANS,DIAG,N,K,A,LDA,X,INCX )
CHARACTER*1        UPLO,TRANS,DIAG
INTEGER            N,K,LDA,INCX
DOUBLE PRECISION   A(LDA,*),X(*)

SUBROUTINE F06SKF( UPLO,TRANS,DIAG,N,K,A,LDA,X,INCX )
ENTRY ZTBSV      ( UPLO,TRANS,DIAG,N,K,A,LDA,X,INCX )
CHARACTER*1          UPLO,TRANS,DIAG
INTEGER              N,K,LDA,INCX
COMPLEX(KIND(1.0D0)) A(LDA,*),X(*)

SUBROUTINE F06PLF( UPLO,TRANS,DIAG,N,AP,X,INCX )
ENTRY DTPSV      ( UPLO,TRANS,DIAG,N,AP,X,INCX )
CHARACTER*1        UPLO,TRANS,DIAG
INTEGER            N,INCX
DOUBLE PRECISION   AP(*),X(*)

SUBROUTINE F06SLF( UPLO,TRANS,DIAG,N,AP,X,INCX )
ENTRY ZTPSV      ( UPLO,TRANS,DIAG,N,AP,X,INCX )
CHARACTER*1          UPLO,TRANS,DIAG
INTEGER              N,INCX
COMPLEX(KIND(1.0D0)) AP(*),X(*)

SUBROUTINE F06PMF( M,N,ALPHA,X,INCX,Y,INCY,A,LDA )
ENTRY DGER       ( M,N,ALPHA,X,INCX,Y,INCY,A,LDA )
INTEGER            M,N,INCX,INCY,LDA
DOUBLE PRECISION   ALPHA,X(*),Y(*),A(LDA,*)

SUBROUTINE F06SMF( M,N,ALPHA,X,INCX,Y,INCY,A,LDA )
ENTRY ZGERU      ( M,N,ALPHA,X,INCX,Y,INCY,A,LDA )
INTEGER              M,N,INCX,INCY,LDA
COMPLEX(KIND(1.0D0)) ALPHA,X(*),Y(*),A(LDA,*)

SUBROUTINE F06SNF( M,N,ALPHA,X,INCX,Y,INCY,A,LDA )
ENTRY ZGERC      ( M,N,ALPHA,X,INCX,Y,INCY,A,LDA )
INTEGER              M,N,INCX,INCY,LDA
COMPLEX(KIND(1.0D0)) ALPHA,X(*),Y(*),A(LDA,*)

SUBROUTINE F06PPF( UPLO,N,ALPHA,X,INCX,A,LDA )
ENTRY DSYR       ( UPLO,N,ALPHA,X,INCX,A,LDA )
CHARACTER*1        UPLO
```

```
INTEGER            N,INCX,LDA
DOUBLE PRECISION   ALPHA,X(*),A(LDA,*)

SUBROUTINE F06SPF( UPLO,N,ALPHA,X,INCX,A,LDA )
ENTRY ZHER       ( UPLO,N,ALPHA,X,INCX,A,LDA )
CHARACTER*1          UPLO
INTEGER              N,INCX,LDA
DOUBLE PRECISION     ALPHA
COMPLEX(KIND(1.0D0)) X(*),A(LDA,*)

SUBROUTINE F06PQF( UPLO,N,ALPHA,X,INCX,AP )
ENTRY DSPR       ( UPLO,N,ALPHA,X,INCX,AP )
CHARACTER*1        UPLO
INTEGER                   N,INCX
DOUBLE PRECISION            ALPHA,X(*),AP(*)

SUBROUTINE F06SQF( UPLO,N,ALPHA,X,INCX,AP )
ENTRY ZHPR       ( UPLO,N,ALPHA,X,INCX,AP )
CHARACTER*1          UPLO
INTEGER              N,INCX
DOUBLE PRECISION     ALPHA
COMPLEX(KIND(1.0D0)) X(*),AP(*)

SUBROUTINE F06PRF( UPLO,N,ALPHA,X,INCX,Y,INCY,A,LDA )
ENTRY DSYR2      ( UPLO,N,ALPHA,X,INCX,Y,INCY,A,LDA )
CHARACTER*1        UPLO
INTEGER            N,INCX,INCY,LDA
DOUBLE PRECISION   ALPHA,X(*),Y(*),A(LDA,*)

SUBROUTINE F06SRF( UPLO,N,ALPHA,X,INCX,Y,INCY,A,LDA )
ENTRY ZHER2      ( UPLO,N,ALPHA,X,INCX,Y,INCY,A,LDA )
CHARACTER*1          UPLO
INTEGER              N,INCX,INCY,LDA
COMPLEX(KIND(1.0D0)) ALPHA,X(*),Y(*),A(LDA,*)

SUBROUTINE F06PSF( UPLO,N,ALPHA,X,INCX,Y,INCY,AP )
ENTRY DSPR2      ( UPLO,N,ALPHA,X,INCX,Y,INCY,AP )
CHARACTER*1        UPLO
INTEGER            N,INCX,INCY
DOUBLE PRECISION  ALPHA,X(*),Y(*),AP(*)

SUBROUTINE F06SSF( UPLO,N,ALPHA,X,INCX,Y,INCY,AP )
ENTRY ZHPR2      ( UPLO,N,ALPHA,X,INCX,Y,INCY,AP )
CHARACTER*1          UPLO
INTEGER              N,INCX,INCY
COMPLEX(KIND(1.0D0)) ALPHA,X(*),Y(*),AP(*)

F06PAF(*), F06SAF(*), F06PBF(*) and F06SBF(*)

perform the operation
```

```
    y<-(alpha)Ax+(beta)y, when TRANS = 'N',

               T
    y<-(alpha)A x+(beta)y, when TRANS = 'T',

               H
    y<-(alpha)A x+(beta)y, when TRANS = 'C',
```

where A is a general matrix for F06PAF(*) and F06SAF(*), and is a general band matrix for F06PBF(*) and F06SBF(*).

F06PCF(*), F06SCF(*), F06PEF(*), F06SEF(*), F06PDF(*) and F06SDF(*)

perform the operation

```
y<-(alpha)Ax+(beta)y
```

where A is symmetric and Hermitian for F06PCF(*) and F06SCF(*) respectively, is symmetric and Hermitian stored in packed form for F06PEF(*) and F06SEF(*) respectively, and is symmetric and Hermitian band for F06PDF(*) and F06SDF(*).

F06PFF(*), F06SFF(*), F06PHF(*), F06SHF(*), F06PGF(*) and F06SGF(*)

perform the operation

```
    x<-Ax,       when TRANS = 'N',

        T
    x<-A x,      when TRANS = 'T',

        H
    x<-A x,      when TRANS = 'C',
```

where A is a triangular matrix for F06PFF(*) and F06SFF(*), is a triangular matrix stored in packed form for F06PHF(*) and F06SHF(*), and is a triangular band matrix for F06PGF(*) and F06SGF(*).

F06PJF(*), F06SJF(*), F06PLF(*), F06SLF(*), F06PKF(*) and F06SKF(*)

solve the equations

```
    Ax=b,        when TRANS = 'N',

     T
```

```
    T
   A x=b,       when TRANS = 'T',

    H
   A x=b,       when TRANS = 'C',
```

where A is a triangular matrix for F06PJF(*) and F06SJF(*), is a triangular matrix stored in packed form for F06PLF(*) and F06SLF(*), and is a triangular band matrix for F06PKF(*) and F06SKF(*). The vector b must be supplied in the array X and is overwritten by the solution. It is important to note that no test for singularity is included in these routines.

F06PMF(*) and F06SMF(*)

```
                                  T
perform the operation A<-(alpha)xy +A, where A is a general
matrix.
```

F06SNF(*)

```
                                   H
performs the operation A<-(alpha)xy +A, where A is a general
complex matrix.
```

F06PPF(*) and F06PQF(*)

```
                                  T
perform the operation A<-(alpha)xx +A, where A is a symmetric
matrix for F06PPF(*) and is a symmetric matrix stored in packed
form for F06PQF(*).
```

F06SPF(*) and F06SQF(*)

```
                                  H
perform the operation A<-(alpha)xx +A, where A is an Hermitian
matrix for F06SPF(*) and is an Hermitian matrix stored in packed
form for F06SQF(*).
```

F06PRF(*) and F06PSF(*)

```
                                  T          T
perform the operation A<-(alpha)xy +(alpha)yx +A, where A is a
symmetric matrix for F06PRF(*) and is a symmetric matrix stored
in packed form for F06PSF(*).
```

F06SRF(*) and F06SSF(*)

```
                                  H          H
perform the operation A<-(alpha)xy +(alpha)yx +A, where A is an
Hermitian matrix for F06SRF(*) and is an Hermitian matrix stored
```

in packed form for F06SSF(*).

The following argument values are invalid:

Any value of the character arguments DIAG, TRANS, or UPLO whose meaning is not specified.

M < 0

N < 0

KL < 0

KU < 0

K < 0

LDA < M

LDA < KL + KU + 1

LDA < N for the routines involving full Hermitian, symmetric or triangular matrices

LDA < K + 1 for the routines involving band Hermitian, symmetric or triangular matrices

INCX = 0

INCY = 0

If a routine is called with an invalid value then an error message is output, on the error message unit (see X04AAF), giving the name of the routine and the number of the first invalid argument, and execution is terminated.

4.4. The Level-3 Matrix-matrix Routines

The matrix-matrix routines all have either two or three arguments representing a matrix, one of which is an input-output argument, and in each case the arguments are two-dimensional arrays.

The sizes of the matrices are determined by one or more of the arguments M, N and K. The size of the input-output array is always determined by the arguments M and N for a rectangular m by n matrix, and by the argument N for a square n by n matrix. It is permissible to call the routines with M or N = 0, in which case the routines exit immediately without referencing their array arguments.

Many of the routines perform an operation of the form

C<-P+(beta)C,

where P is the product of two matrices, or the sum of two such products. When the inner dimension of the matrix product is different from m or n it is denoted by K. Again it is permissible to call the routines with K = 0 and if M > 0, but K = 0, then the routines perform the operation

C<-(beta)C.

As with the Level-2 routines (see Section 4.3) the description of the matrix consists of the array name (A or B or C) followed by the first dimension (LDA or LDB or LDC).

The arguments that specify options are character arguments with the names SIDE, TRANSA, TRANSB, TRANS, UPLO and DIAG. UPLO and DIAG have the same values and meanings as for the Level-2 routines (see Section 4.3); TRANSA, TRANSB and TRANS have the same values and meanings as TRANS in the Level-2 routines, where TRANSA and TRANSB apply to the matrices A and B respectively. SIDE is used by the routines as follows:

Value	Meaning
'L'	Multiply general matrix by symmetric, Hermitian or triangular matrix on the left
'R'	Multiply general matrix by symmetric, Hermitian or triangular matrix on the right

The storage conventions for matrices are as for the Level-2 routines (see Section 4.3).

```
SUBROUTINE F06YAF( TRANSA,TRANSB,M,N,K,ALPHA,A,LDA,B,LDB,BETA,C,LDC )
ENTRY      DGEMM ( TRANSA,TRANSB,M,N,K,ALPHA,A,LDA,B,LDB,BETA,C,LDC )
CHARACTER          TRANSA,TRANSB
INTEGER            M,N,K,LDA,LDB,LDC
DOUBLE PRECISION   ALPHA,A(LDA,*),B(LDB,*),BETA,C(LDC,*)

SUBROUTINE F06ZAF( TRANSA,TRANSB,M,N,K,ALPHA,A,LDA,B,LDB,BETA,C,LDC )
ENTRY      ZGEMM ( TRANSA,TRANSB,M,N,K,ALPHA,A,LDA,B,LDB,BETA,C,LDC )
CHARACTER            TRANSA,TRANSB
INTEGER              M,N,K,LDA,LDB,LDC
COMPLEX(KIND(1.0D0)) ALPHA,A(LDA,*),B(LDB,*),BETA,C(LDC,*)

SUBROUTINE F06YCF( SIDE,UPLO,M,N,ALPHA,A,LDA,B,LDB,BETA,C,LDC )
ENTRY      DSYMM ( SIDE,UPLO,M,N,ALPHA,A,LDA,B,LDB,BETA,C,LDC )
```

```
CHARACTER          SIDE,UPLO
INTEGER            M,N,LDA,LDB,LDC
DOUBLE PRECISION   ALPHA,A(LDA,*),B(LDB,*),BETA,C(LDC,*)

SUBROUTINE F06ZCF( SIDE,UPLO,M,N,ALPHA,A,LDA,B,LDB,BETA,C,LDC )
ENTRY      ZHEMM ( SIDE,UPLO,M,N,ALPHA,A,LDA,B,LDB,BETA,C,LDC )
CHARACTER            SIDE,UPLO
INTEGER              M,N,LDA,LDB,LDC
COMPLEX(KIND(1.0D0)) ALPHA,A(LDA,*),B(LDB,*),BETA,C(LDC,*)

SUBROUTINE F06ZTF( SIDE,UPLO,M,N,ALPHA,A,LDA,B,LDB,BETA,C,LDC )
ENTRY      ZSYMM ( SIDE,UPLO,M,N,ALPHA,A,LDA,B,LDB,BETA,C,LDC )
CHARACTER            SIDE,UPLO
INTEGER              M,N,LDA,LDB,LDC
COMPLEX(KIND(1.0D0)) ALPHA,A(LDA,*),B(LDB,*),BETA,C(LDC,*)

SUBROUTINE F06YFF( SIDE,UPLO,TRANSA,DIAG,M,N,ALPHA,A,LDA,B,LDB )
ENTRY      DTRMM ( SIDE,UPLO,TRANSA,DIAG,M,N,ALPHA,A,LDA,B,LDB )
CHARACTER          SIDE,UPLO,TRANSA,DIAG
INTEGER            M,N,LDA,LDB
DOUBLE PRECISION   ALPHA,A(LDA,*),B(LDB,*)

SUBROUTINE F06ZFF( SIDE,UPLO,TRANSA,DIAG,M,N,ALPHA,A,LDA,B,LDB )
ENTRY      ZTRMM ( SIDE,UPLO,TRANSA,DIAG,M,N,ALPHA,A,LDA,B,LDB )
CHARACTER            SIDE,UPLO,TRANSA,DIAG
INTEGER              M,N,LDA,LDB
COMPLEX(KIND(1.0D0)) ALPHA,A(LDA,*),B(LDB,*)

SUBROUTINE F06YJF( SIDE,UPLO,TRANSA,DIAG,M,N,ALPHA,A,LDA,B,LDB )
ENTRY      DTRSM ( SIDE,UPLO,TRANSA,DIAG,M,N,ALPHA,A,LDA,B,LDB )
CHARACTER          SIDE,UPLO,TRANSA,DIAG
INTEGER            M,N,LDA,LDB
DOUBLE PRECISION   ALPHA,A(LDA,*),B(LDB,*)

SUBROUTINE F06ZJF( SIDE,UPLO,TRANSA,DIAG,M,N,ALPHA,A,LDA,B,LDB )
ENTRY      ZTRSM ( SIDE,UPLO,TRANSA,DIAG,M,N,ALPHA,A,LDA,B,LDB )
CHARACTER            SIDE,UPLO,TRANSA,DIAG
INTEGER              M,N,LDA,LDB
COMPLEX(KIND(1.0D0)) ALPHA,A(LDA,*),B(LDB,*)

SUBROUTINE F06YPF( UPLO,TRANS,N,K,ALPHA,A,LDA,BETA,C,LDC )
ENTRY      DSYRK ( UPLO,TRANS,N,K,ALPHA,A,LDA,BETA,C,LDC )
CHARACTER          UPLO,TRANS
INTEGER            N,K,LDA,LDC
DOUBLE PRECISION   ALPHA,A(LDA,*),BETA,C(LDC,*)

SUBROUTINE F06ZPF( UPLO,TRANS,N,K,ALPHA,A,LDA,BETA,C,LDC )
ENTRY      ZHERK ( UPLO,TRANS,N,K,ALPHA,A,LDA,BETA,C,LDC )
CHARACTER            UPLO,TRANS
INTEGER              N,K,LDA,LDC
```

```
DOUBLE PRECISION     ALPHA,BETA
COMPLEX(KIND(1.0D0)) A(LDA,*),C(LDC,*)

SUBROUTINE F06ZUF( UPLO,TRANS,N,K,ALPHA,A,LDA,BETA,C,LDC )
ENTRY      ZSYRK ( UPLO,TRANS,N,K,ALPHA,A,LDA,BETA,C,LDC )
CHARACTER            UPLO,TRANS
INTEGER              N,K,LDA,LDC
COMPLEX(KIND(1.0D0)) ALPHA,A(LDA,*),BETA,C(LDC,*)

SUBROUTINE F06YRF( UPLO,TRANS,N,K,ALPHA,A,LDA,B,LDB,BETA,C,LDC )
ENTRY      DSYR2K( UPLO,TRANS,N,K,ALPHA,A,LDA,B,LDB,BETA,C,LDC )
CHARACTER          UPLO,TRANS
INTEGER            N,K,LDA,LDB,LDC
DOUBLE PRECISION   ALPHA,A(LDA,*),B(LDB,*),BETA,C(LDC,*)

SUBROUTINE F06ZRF( UPLO,TRANS,N,K,ALPHA,A,LDA,B,LDB,BETA,C,LDC )
ENTRY      ZHER2K( UPLO,TRANS,N,K,ALPHA,A,LDA,B,LDB,BETA,C,LDC )
CHARACTER            UPLO,TRANS
INTEGER              N,K,LDA,LDB,LDC
DOUBLE PRECISION     BETA
COMPLEX(KIND(1.0D0)) ALPHA,A(LDA,*),B(LDB,*),C(LDC,*)

SUBROUTINE F06ZWF( UPLO,TRANS,N,K,ALPHA,A,LDA,B,LDB,BETA,C,LDC )
ENTRY      ZSYR2K( UPLO,TRANS,N,K,ALPHA,A,LDA,B,LDB,BETA,C,LDC )
CHARACTER            UPLO,TRANS
INTEGER              N,K,LDA,LDB,LDC
COMPLEX(KIND(1.0D0)) ALPHA,A(LDA,*),B(LDB,*),BETA,C(LDC,*)


F06YAF(*) and F06ZAF(*)

perform the operation indicated in the following table:

              TRANSA = 'N'    TRANSA = 'T'      TRANSA = 'C'

                                            T              H
TRANSB='N'   C <- (alpha)AB   C <- (alpha)A B   C <- (alpha)A B
                 +(beta)C        +(beta)C          +(beta)C
             A is m*k,        A is k*m,         A is k*m,
             B is k*n         B is k*n          B is k*n

                            T               T T              H T
TRANSB='T'   C <- (alpha)AB   C <- (alpha)A B   C <- (alpha)A B
                +(beta)C         +(beta)C          +(beta)C
             A is m*k,        A is k*m,         A is k*m,
             B is n*k         B is n*k          B is n*k

                            H               T H              H H
TRANSB='C'   C <- (alpha)AB   C <- (alpha)A B   C <- (alpha)A B
                +(beta)C         +(beta)C          +(beta)C
```

```
           A is m*k,        A is k*m,         A is k*m,
           B is n*k         B is n*k          B is n*k
```

where A and B are general matrices and C is a general m by n matrix.

F06YCF(*), F06ZCF(*) and F06ZTF(*) perform the operation

indicated in the following table:

```
    SIDE = 'L'               SIDE = 'R'

    C<-(alpha)AB+(beta)C     C<-(alpha)BA+(beta)C

    A is m*m                 B is m*n

    B is m*n                 A is n*n
```

where A is symmetric for F06YCF(*) and F06ZTF(*) and is Hermitian for F06ZCF(*), B is a general matrix and C is a general m by n matrix.

F06YFF(*) and F06ZFF(*) perform the operation indicated in the following table:

```
          TRANSA = 'N'       TRANSA = 'T'        TRANSA = 'C'

                                         T                   H
SIDE='L'  B <- (alpha)AB     B <- (alpha)A B     B <- (alpha)A B

          A is               A is                A is
          triangular         triangular          triangular
          m*m                m*m                 m*m

                                          T                   H
SIDE='R'  B <- (alpha)BA     B <- (alpha)BA      B <- (alpha)BA

          A is               A is                A is
          triangular         triangular          triangular
          n*n                n*n                 n*n
```

where B is a general m by n matrix.

F06YJF(*) and F06ZJF(*) solve the equations, indicated in the following table, for X:

```
          TRANSA = 'N'       TRANSA = 'T'       TRANSA = 'C'

                              T                  H
```

```
SIDE='L'  AX=(alpha)B        A X=(alpha)B       A X=(alpha)B

          A is               A is               A is
          triangular         triangular         triangular
          m*m                m*m                m*m

                               T                  H
SIDE='R'  XA=(alpha)B        XA =(alpha)B       XA =(alpha)B

          A is               A is               A is
          triangular         triangular         triangular
          n*n                n*n                n*n
```

```
where B is a general m by n matrix. The m by n solution matrix X
is overwritten on the array B. It is important to note that no
test for singularity is included in these routines.

F06YPF(*), F06ZPF(*) and F06ZUF(*) perform the operation
indicated in the following table:

          TRANS = 'N'        TRANS = 'T'        TRANS = 'C'

                        T                  T                  T
F06YPF    C <- (alpha)AA     C <- (alpha)A A    C <- (alpha)A A
              +(beta)C           +(beta)C           +(beta)C
                        T                  T
F06ZUF    C <- (alpha)AA     C <- (alpha)A A         --
              +(beta)C           +(beta)C
                        H                                     H
F06ZPF    C <- (alpha)AA          --            C <- (alpha)A A
              +(beta)C                              +(beta)C

          A is n*k           A is k*n           A is k*n

where A is a general matrix and C is an n by n symmetric matrix
for F06YPF(*) and F06ZUF(*), and is an n by n Hermitian matrix
for F06ZPF(*).

F06YRF(*), F06ZRF(*) and F06ZWF(*) perform the operation
indicated in the following table:


          TRANS = 'N'        TRANS = 'T'        TRANS = 'C'

                        T                  T                  T
F06YRF    C <- (alpha)AB     C <- (alpha)A B    C <- (alpha)A B
                        T                  T                  T
              +(alpha)BA         +(alpha)B A        +(alpha)B A
              +(beta)C           +(beta)C           +(beta)C
                        T                  T
```

```
F06ZWF    C <- (alpha)AB     C <- (alpha)A B           --
                        T                  T
              +(alpha)BA         +(alpha)B A
              +(beta)C           +(beta)C
                        H                                    H
F06ZRF    C <- (alpha)AB           --          C <- (alpha)A B
                        H                                    H
              +(alpha)BA                           +(alpha)B A
              +(beta)C                             +(beta)C

          A and B are n*k    A and B are k*n    A and B are k*n
```

where A and B are general matrices and C is an n by n symmetric matrix for F06YRF(*) and F06ZWF(*), and is an n by n Hermitian matrix for F06ZPF(*).

The following values of arguments are invalid:

Any value of the character arguments SIDE, TRANSA, TRANSB, TRANS, UPLO or DIAG, whose meaning is not specified.

M < 0

N < 0

K < 0

LDA < the number of rows in the matrix A.

LDB < the number of rows in the matrix B.

LDC < the number of rows in the matrix C.

If a routine is called with an invalid value then an error message is output, on the error message unit (see X04AAF), giving the name of the routine and the number of the first invalid argument, and execution is terminated.

F06 -- Linear Algebra Support Routines   Contents -- F06
Chapter F06

Linear Algebra Support Routines

F06EDF (DSCAL) Multiply real vector by scalar

F06EFF (DCOPY) Copy real vector

F06EGF (DSWAP) Swap two real vectors

F06EJF (DNRM2) Compute Euclidean norm of real vector

F06EKF (DASUM) Sum the absolute values of real vector elements

F06EPF (DROT) Apply real plane rotation

F06GAF (ZDOTU) Dot product of two complex vectors, unconjugated

F06GBF (ZDOTC) Dot product of two complex vectors, conjugated

F06GCF (ZAXPY) Add scalar times complex vector to complex vector

F06GDF (ZSCAL) Multiply complex vector by complex scalar

F06GFF (ZCOPY) Copy complex vector

F06GGF (ZSWAP) Swap two complex vectors

F06JDF (ZDSCAL) Multiply complex vector by real scalar

F06JJF (DZNRM2) Compute Euclidean norm of complex vector

F06JKF (DZASUM) Sum the absolute values of complex vector elements

F06JLF (IDAMAX) Index, real vector element with largest absolute value

F06JMF (IZAMAX) Index, complex vector element with largest absolute value

F06PAF (DGEMV) Matrix-vector product, real rectangular matrix

F06PBF (DGBMV) Matrix-vector product, real rectangular band matrix

F06PCF (DSYMV) Matrix-vector product, real symmetric matrix

F06PDF (DSBMV) Matrix-vector product, real symmetric band matrix

F06PEF (DSPMV) Matrix-vector product, real symmetric packed matrix

F06PFF (DTRMV) Matrix-vector product, real triangular matrix

F06PGF (DTBMV) Matrix-vector product, real triangular band matrix

F06PHF (DTPMV) Matrix-vector product, real triangular packed matrix

F06PJF (DTRSV) System of equations, real triangular matrix

F06PKF (DTBSV) System of equations, real triangular band matrix

F06PLF (DTPSV) System of equations, real triangular packed matrix

F06PMF (DGER) Rank-1 update, real rectangular matrix

F06PPF (DSYR) Rank-1 update, real symmetric matrix

F06PQF (DSPR) Rank-1 update, real symmetric packed matrix

F06PRF (DSYR2) Rank-2 update, real symmetric matrix

F06PSF (DSPR2) Rank-2 update, real symmetric packed matrix

F06SAF (ZGEMV) Matrix-vector product, complex rectangular matrix

F06SBF (ZGBMV) Matrix-vector product, complex rectangular band matrix

F06SCF (ZHEMV) Matrix-vector product, complex Hermitian matrix

F06SDF (ZHBMV) Matrix-vector product, complex Hermitian band matrix

F06SEF (ZHPMV) Matrix-vector product, complex Hermitian packed matrix

F06SFF (ZTRMV) Matrix-vector product, complex triangular matrix

F06SGF (ZTBMV) Matrix-vector product, complex triangular band matrix

F06SHF (ZTPMV) Matrix-vector product, complex triangular packed matrix

F06SJF (ZTRSV) System of equations, complex triangular matrix

F06SKF (ZTBSV) System of equations, complex triangular band matrix

```
F06SLF  (ZTPSV) System of equations, complex triangular packed
        matrix

F06SMF  (ZGERU) Rank-1 update, complex rectangular matrix,
        unconjugated vector

F06SNF  (ZGERC) Rank-1 update, complex rectangular matrix,
        conjugated vector

F06SPF  (ZHER) Rank-1 update, complex Hermitian matrix

F06SQF  (ZHPR) Rank-1 update, complex Hermitian packed matrix

F06SRF  (ZHER2) Rank-2 update, complex Hermitian matrix

F06SSF  (ZHPR2) Rank-2 update, complex Hermitian packed matrix

F06YAF  (DGEMM) Matrix-matrix product, two real rectangular
        matrices

F06YCF  (DSYMM) Matrix-matrix product, one real symmetric matrix,
        one real rectangular matrix

F06YFF  (DTRMM) Matrix-matrix product, one real triangular
        matrix, one real rectangular matrix

F06YJF  (DTRSM) Solves a system of equations with multiple right-
        hand sides, real triangular coefficient matrix

F06YPF  (DSYRK) Rank-k update of a real symmetric matrix

F06YRF  (DSYR2K) Rank-2k update of a real symmetric matrix

F06ZAF  (ZGEMM) Matrix-matrix product, two complex rectangular
        matrices

F06ZCF  (ZHEMM) Matrix-matrix product, one complex Hermitian
        matrix, one complex rectangular matrix

F06ZFF  (ZTRMM) Matrix-matrix product, one complex triangular
        matrix, one complex rectangular matrix

F06ZJF  (ZTRSM) Solves system of equations with multiple right-
        hand sides, complex triangular coefficient matrix

F06ZPF  (ZHERK) Rank-k update of a complex Hermitian matrix

F06ZRF  (ZHER2K) Rank-2k update of a complex Hermitian matrix

F06ZTF  (ZSYMM) Matrix-matrix product, one complex symmetric
```

```
                  matrix, one complex rectangular matrix

          F06ZUF  (ZSYRK) Rank-k update of a complex symmetric matrix

          F06ZWF  (ZSYR2K) Rank-2k update of a complex symmetric matrix

\end{verbatim}
\endscroll
\end{page}
```

Linear Equations (LAPACK)

— nagf.ht —

```
\begin{page}{manpageXXf07}{NAG Documentation: f07}
\beginscroll
\begin{verbatim}



     F07(3NAG)         Foundation Library (12/10/92)         F07(3NAG)



          F07 -- Linear Equations (LAPACK)              Introduction -- F07
                                     Chapter F07
                              Linear Equations (LAPACK)

          1. Scope of the Chapter

          This chapter provides four routines concerned with matrix
          factorization, and the solution of systems of linear equations
          following the matrix factorizations.

          2. Background to the Problems

          Background material, together with pointers to the routines in
          this chapter, are to be found in the F01 and F04 Chapter
          Introductions.

          3. Recommendations on Choice and Use of Routines

          The routines in this chapter are derived from the LAPACK project
          and may also be called using the LAPACK name, which is given in
          brackets following the F07 name in the following descriptions.
```

```
          Routine F07ADF (DGETRF) performs an LU factorization of a real m
          by n matrix A. Following the use of this routine, F07AEF (DGETRS)
          may be used to solve a system of n non-singular linear equations,
          with one or more right-hand sides.

          Routine F07FDF (DPOTRF) performs the Cholesky factorization of a
          real symmetric positive-definite matrix A. Following the use of
          this routine, F07FEF (DPOTRS) may be used to solve a system of
          symmetric positive-definite linear equations, with one or more
          right-hand sides.


          F07 -- Linear Equations (LAPACK)                  Contents -- F07
          Chapter F07

          Linear Equations (LAPACK)

          F07ADF  (DGETRF) LU factorization of real m by n matrix

          F07AEF  (DGETRS) Solution of real system of linear equations,
                  multiple right-hand sides, matrix already factorized by
                  F07ADF

          F07FDF  (DPOTRF) Cholesky factorization of real symmetric
                  positive-definite matrix

          F07FEF  (DPOTRS) Solution of real symmetric positive-definite
                  system of linear equations, multiple right-hand sides,
                  matrix already factorized by F07FDF

\end{verbatim}
\endscroll
\end{page}
```

Computes the LU factorization of a real m by n matrix

— nagf.ht —

```
\begin{page}{manpageXXf07adf}{NAG Documentation: f07adf}
\beginscroll
\begin{verbatim}
```

F07 -- Linear Equations (LAPACK) F07ADF

F07ADF -- NAG Foundation Library Routine Document

Note: Before using this routine, please read the Users' Note for your implementation to check implementation-dependent details. The symbol (*) after a NAG routine name denotes a routine that is not included in the Foundation Library.

1. Purpose

F07ADF (DGETRF) computes the LU factorization of a real m by n matrix.

2. Specification

```
SUBROUTINE F07ADF (M, N, A, LDA, IPIV, INFO)
ENTRY            M, N, A, LDA, IPIV, INFO
INTEGER          M, N, LDA, IPIV(*), INFO
DOUBLE PRECISION A(LDA,*)
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine forms the LU factorization of a real m by n matrix A as A=PLU, where P is a permutation matrix, L is lower triangular with unit diagonal elements (lower trapezoidal if m>n) and U is upper triangular (upper trapezoidal if m<n). Usually A is square (m=n), and both L and U are triangular. The routine uses partial pivoting, with row interchanges.

4. References

[1] Golub G H and Van Loan C F (1989) Matrix Computations (2nd Edition). Johns Hopkins University Press, Baltimore, Maryland.

5. Parameters

1: M -- INTEGER Input
On entry: m, the number of rows of the matrix A.
Constraint: M >= 0.

2: N -- INTEGER Input
On entry: n, the number of columns of the matrix A.

Constraint: N >= 0.

3: A(LDA,*) -- DOUBLE PRECISION array Input/Output
Note: the second dimension of the array A must be at least max(1,N).
On entry: the m by n matrix A. On exit: A is overwritten by the factors L and U; the unit diagonal elements of L are not stored.

4: LDA -- INTEGER Input
On entry:
the first dimension of the array A as declared in the (sub)program from which F07ADF is called.
Constraint: LDA >= max(1,M).

5: IPIV(*) -- INTEGER array Output
Note: the dimension of the array IPIV must be at least max(1,min(M,N)).
On exit: the pivot indices. Row i of the matrix A was interchanged with row IPIV(i) for i=1,2,...,min(m,n).

6: INFO -- INTEGER Output
On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0
If INFO = -i, the ith parameter has an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0
If INFO = i, u_{ii} is exactly zero. The factorization has been completed but the factor U is exactly singular, and division by zero will occur if it is subsequently used to solve a system of linear equations or to compute A^{-1}.

7. Accuracy

The computed factors L and U are the exact factors of a perturbed matrix A+E, where

|E|<=c(min(m,n))(epsilon)P|L||U|,

c(n) is a modest linear function of n, and (epsilon) is the machine precision.

```
          8. Further Comments

          The total number of floating-point operations is approximately
           2 3                            1 2                   1 2
           -n  if m=n (the usual case),  -n (3m-n) if m>n and  -m (3n-m) if
           3                              3                     3
          m<n.

          A call to this routine with m=n may be followed by calls to the
          routines:

                                                  T
               F07AEF (DGETRS) to solve AX=B or A X=B;

               F07AGF (DGECON)(*) to estimate the condition number of A;

               F07AJF (DGETRI)(*) to compute the inverse of A.

          The complex analogue of this routine is F07ARF (ZGETRF)(*).

          9. Example

          To compute the LU factorization of the matrix A, where

                              ( 1.80  2.88  2.05 -0.89)
                              ( 5.25 -2.95 -0.95 -3.80)
                            A=( 1.58 -2.69 -2.90 -1.04).
                              (-1.11 -0.66 -0.59  0.80)

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Solves a real system of linear equations

— nagf.ht —

```
\begin{page}{manpageXXf07aef}{NAG Documentation: f07aef}
\beginscroll
\begin{verbatim}
```

F07 -- Linear Equations (LAPACK) F07AEF

F07AEF -- NAG Foundation Library Routine Document

Note: Before using this routine, please read the Users' Note for your implementation to check implementation-dependent details. The symbol (*) after a NAG routine name denotes a routine that is not included in the Foundation Library.

1. Purpose

F07AEF (DGETRS) solves a real system of linear equations with multiple right-hand sides, $AX=B$ or $A^T X=B$, where A has been factorized by F07ADF (DGETRF).

2. Specification

```
       SUBROUTINE F07AEF (TRANS, N, NRHS, A, LDA, IPIV, B, LDB,
      1                   INFO)
       ENTRY            TRANS, N, NRHS, A, LDA, IPIV, B, LDB, INFO
       INTEGER          N, NRHS, LDA, IPIV(*), LDB, INFO
       DOUBLE PRECISION A(LDA,*), B(LDB,*)
       CHARACTER*1      TRANS
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

To solve a real system of linear equations $AX=B$ or $A^T X=B$, this routine must be preceded by a call to F07ADF (DGETRF)which computes the LU factorization of A as $A=PLU$. The solution is computed by forward and backward substitution.

If TRANS = 'N', the solution is computed by solving $PLY=B$ and then $UX=Y$.

If TRANS = 'T' or 'C', the solution is computed by solving $U^T Y=B$ and then $L^T P^T X=Y$.

4. References

[1] Golub G H and Van Loan C F (1989) Matrix Computations (2nd Edition). Johns Hopkins University Press, Baltimore, Maryland.

5. Parameters

1: TRANS -- CHARACTER*1 Input
On entry: indicates the form of the equations as follows:
if TRANS = 'N', then AX=B is solved for X;

if TRANS = 'T' or 'C', then $A^T X=B$ is solved for X.
Constraint: TRANS = 'N', 'T' or 'C'.

2: N -- INTEGER Input
On entry: n, the order of the matrix A. Constraint: N >= 0.

3: NRHS -- INTEGER Input
On entry: r, the number of right-hand sides. Constraint: NRHS >= 0.

4: A(LDA,*) -- DOUBLE PRECISION array Input
Note: the second dimension of the array A must be at least max(1,N).
On entry: the LU factorization of A, as returned by F07ADF (DGETRF).

5: LDA -- INTEGER Input
On entry:
the first dimension of the array A as declared in the (sub)program from which F07AEF is called.
Constraint: LDA >= max(1,N).

6: IPIV(*) -- INTEGER array Input
Note: the dimension of the array IPIV must be at least max(1,N).
On entry: the pivot indices, as returned by F07ADF (DGETRF).

7: B(LDB,*) -- DOUBLE PRECISION array Input/Output
Note: the second dimension of the array B must be at least max(1,NRHS).
On entry: the n by r right-hand side matrix B. On exit: the n by r solution matrix X.

8: LDB -- INTEGER Input
On entry:
the first dimension of the array B as declared in the (sub)program from which F07AEF is called.
Constraint: LDB >= max(1,N).

9: INFO -- INTEGER Output
On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0
If INFO = -i, the ith parameter has an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

For each right-hand side vector b, the computed solution x is the exact solution of a perturbed system of equations $(A+E)x=b$, where

$$|E| \le c(n)\epsilon P|L||U|,$$

$c(n)$ is a modest linear function of n, and ϵ is the machine precision.

If x is the true solution, then the computed solution $\hat{x}$ satisfies a forward error bound of the form

$$\frac{\|x-\hat{x}\|_{\infty}}{\|x\|_{\infty}} \le c(n)\,\mathrm{cond}(A,x)\,\epsilon$$

where $\mathrm{cond}(A,x)=\||A^{-1}||A||x|\|_{\infty}/\|x\|_{\infty} \le \mathrm{cond}(A)=\||A^{-1}||A|\|_{\infty} \le \kappa_{\infty}(A)$. Note that $\mathrm{cond}(A,x)$ can be much smaller than $\mathrm{cond}(A)$, and $\mathrm{cond}(A^T)$ can be much larger (or smaller) than $\mathrm{cond}(A)$.

Forward and backward error bounds can be computed by calling F07AHF (DGERFS)(*), and an estimate for $\kappa_{\infty}(A)$ can be obtained by calling F07AGF (DGECON)(*) with NORM ='I'.

8. Further Comments

```
          The total number of floating-point operations is approximately
            2
          2n r.

          This routine may be followed by a call to F07AHF (DGERFS)(*) to
          refine the solution and return an error estimate.

          The complex analogue of this routine is F07ASF (ZGETRS)(*).

          9. Example

          To solve the system of equations AX=B, where

                               ( 1.80  2.88  2.05 -0.89)
                               ( 5.25 -2.95 -0.95 -3.80)
                             A=( 1.58 -2.69 -2.90 -1.04)
                               (-1.11 -0.66 -0.59  0.80)

          and


                                    ( 9.52  18.47)
                                    (24.35   2.25)
                                  B=( 0.77 -13.28).
                                    (-6.22  -6.21)

          Here A is unsymmetric and must first be factorized by F07ADF
          (DGETRF)).

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Factorization of a real symmetric positive-definite matrix

— nagf.ht —

```
\begin{page}{manpageXXf07fdf}{NAG Documentation: f07fdf}
\beginscroll
\begin{verbatim}
```

F07 -- Linear Equations (LAPACK) F07FDF
F07FDF -- NAG Foundation Library Routine Document

Note: Before using this routine, please read the Users' Note for your implementation to check implementation-dependent details. The symbol (*) after a NAG routine name denotes a routine that is not included in the Foundation Library.

1. Purpose

F07FDF (DPOTRF) computes the Cholesky factorization of a real symmetric positive-definite matrix.

2. Specification

```
SUBROUTINE F07FDF (UPLO, N, A, LDA, INFO)
ENTRY            UPLO, N, A, LDA, INFO
INTEGER          N, LDA, INFO
DOUBLE PRECISION A(LDA,*)
CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine forms the Cholesky factorization of a real symmetric positive-definite matrix A either as $A=U^TU$ if UPLO = 'U' or $A=LL^T$ if UPLO = 'L', where U is an upper triangular matrix and L is lower triangular.

4. References

[1] Demmel J W (1989) On Floating-point Errors in Cholesky. LAPACK Working Note No. 14. University of Tennessee, Knoxville.

[2] Golub G H and Van Loan C F (1989) Matrix Computations (2nd Edition). Johns Hopkins University Press, Baltimore, Maryland.

5. Parameters

1: UPLO -- CHARACTER*1 Input

On entry: indicates whether the upper or lower triangular part of A is stored and how A is factorized, as follows:

if UPLO = 'U', then the upper triangular part of A is stored and A is factorized as $U^T U$, where U is upper triangular;

if UPLO = 'L', then the lower triangular part of A is stored and A is factorized as LL^T, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N -- INTEGER Input

On entry: n, the order of the matrix A. Constraint: N >= 0.

3: A(LDA,*) -- DOUBLE PRECISION array Input/Output

Note: the second dimension of the array A must be at least max(1,N).

On entry: the n by n symmetric positive-definite matrix A. If UPLO = 'U', the upper triangle of A must be stored and the elements of the array below the diagonal are not referenced; if UPLO = 'L', the lower triangle of A must be stored and the elements of the array above the diagonal are not referenced. On exit: the upper or lower triangle of A is overwritten by the Cholesky factor U or L as specified by UPLO.

4: LDA -- INTEGER Input

On entry:
the first dimension of the array A as declared in the (sub)program from which F07FDF is called.
Constraint: LDA >= max(1,N).

5: INFO -- INTEGER Output

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = -i, the ith parameter has an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

If INFO = i, the leading minor of order i is not positive-definite and the factorization could not be completed. Hence A itself is not positive-definite. This may indicate an error in forming the matrix A. To factorize a symmetric matrix which is not positive-definite, call F07MDF (DSYTRF)(*) instead.

7. Accuracy

If UPLO = 'U', the computed factor U is the exact factor of a perturbed matrix A+E, where

$$|E|\le c(n)(\epsilon)|U^T||U|,$$

c(n) is a modest linear function of n, and (epsilon) is the machine precision. If UPLO = 'L', a similar statement holds for the computed factor L. It follows that

$$|e_{ij}|\le c(n)(\epsilon)\sqrt{a_{ii}a_{jj}}.$$

8. Further Comments

The total number of floating-point operations is approximately $\frac{1}{3}n^3$.

A call to this routine may be followed by calls to the routines:

F07FEF (DPOTRS) to solve AX=B;

F07FGF (DPOCON)(*) to estimate the condition number of A;

F07FJF (DPOTRI)(*) to compute the inverse of A.

The complex analogue of this routine is F07FRF (ZPOTRF)(*).

9. Example

To compute the Cholesky factorization of the matrix A, where

$$A=\begin{pmatrix} 4.16 & -3.12 & 0.56 & -0.10\\ -3.12 & 5.03 & -0.83 & 1.18\\ 0.56 & -0.83 & 0.76 & 0.34\\ -0.10 & 1.18 & 0.34 & 1.18 \end{pmatrix}.$$

The example program is not reproduced here. The source code for all example programs is distributed with the NAG Foundation Library software and should be available on-line.

```
\end{verbatim}
\endscroll
\end{page}
```

———-

Real symmetric positive-definite system of linear equations

— nagf.ht —

```
\begin{page}{manpageXXf07fef}{NAG Documentation: f07fef}
\beginscroll
\begin{verbatim}



     F07FEF(3NAG)      Foundation Library (12/10/92)      F07FEF(3NAG)



          F07 -- Linear Equations (LAPACK)                          F07FEF
                  F07FEF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          F07FEF (DPOTRS) solves a real symmetric positive-definite system
          of linear equations with multiple right-hand sides, AX=B, where A
          has been factorized by F07FDF (DPOTRF).

          2. Specification

                 SUBROUTINE F07FEF (UPLO, N, NRHS, A, LDA, B, LDB, INFO)
                 ENTRY             UPLO, N, NRHS, A, LDA, B, LDB, INFO
                 INTEGER           N, NRHS, LDA, LDB, INFO
                 DOUBLE PRECISION A(LDA,*), B(LDB,*)
                 CHARACTER*1       UPLO

          The ENTRY statement enables the routine to be called by its
          LAPACK name.

          3. Description
```

To solve a real symmetric positive-definite system of linear equations AX=B, this routine must be preceded by a call to F07FDF (DPOTRF) which computes the Cholesky factorization of A. The solution X is computed by forward and backward substitution.

If UPLO = 'U', $A=U^TU$, where U is upper triangular; the solution X is computed by solving $U^TY=B$ and then UX=Y.

If UPLO = 'L', $A=LL^T$, where L is lower triangular; the solution X is computed by solving LY=B and then $L^TX=Y$.

4. References

[1] Golub G H and Van Loan C F (1989) Matrix Computations (2nd Edition). Johns Hopkins University Press, Baltimore, Maryland.

5. Parameters

1: UPLO -- CHARACTER*1 Input

On entry: indicates whether the upper or lower triangular part of A is stored and how A is factorized, as follows:

if UPLO = 'U', then $A=U^TU$ where U is upper triangular;

if UPLO = 'L', then $A=LL^T$ where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N -- INTEGER Input

On entry: n, the order of the matrix A. Constraint: N >= 0.

3: NRHS -- INTEGER Input

On entry: r, the number of right-hand sides. Constraint: NRHS >= 0.

4: A(LDA,*) -- DOUBLE PRECISION array Input

Note: the second dimension of the array A must be at least max(1,N).

On entry: the Cholesky factor of A, as returned by F07FDF (DPOTRF).

5: LDA -- INTEGER Input

On entry:
the first dimension of the array A as declared in the (sub)program from which F07FEF is called.

Constraint: LDA >=max(1,N).

6: B(LDB,*) -- DOUBLE PRECISION array Input/Output
Note: the second dimension of the array B must be at least max(1,NRHS).
On entry: the n by r right-hand side matrix B.

7: LDB -- INTEGER Input
On entry:
the first dimension of the array B as declared in the (sub)program from which F07FEF is called.
Constraint: LDB >=max(1,N).

8: INFO -- INTEGER Output
On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0
If INFO = -i, the ith parameter has an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

For each right-hand side vector b, the computed solution x is the exact solution of a perturbed system of equations (A+E)x=b, where

$$|E|\le c(n)(\epsilon)|U^T||U| \text{ if UPLO = 'U'},$$

$$|E|\le c(n)(\epsilon)|L||L^T| \text{ if UPLO = 'L'},$$

c(n) is a modest linear function of n, and (epsilon) is the machine precision.

If x is the true solution, then the computed solution $\hat{x}$ satisfies a forward bound of the form

$$\frac{\|x-\hat{x}\|_{\infty}}{\|x\|_{\infty}}\le c(n)\,cond(A,x)(\epsilon)$$

-1

```
where cond(A,x)=|||A  ||A||x|||     /||x||      <=
                              infty      infty
            -1
cond(A)=|||A  ||A|||     <=(kappa)     (A). Note that cond(A,x)
                    infty         infty
can be much smaller than cond(A).

Forward and backward error bounds can be computed by calling
F07FHF (DPORFS)(*), and an estimate for (kappa)     (A) (
                                               infty
=(kappa) (A)) can be obtained by calling F07FGF (DPOCON)(*).
        1

8. Further Comments

The total number of floating-point operations is approximately
  2
2n r.

This routine may be followed by a call to F07FHF (DPORFS)(*) to
refine the solution and return an error estimate.

The complex analogue of this routine is F07FSF (ZPOTRS)(*).

9. Example

To compute the Cholesky factorization of the matrix A, where

                    ( 4.16 -3.12  0.56 -0.10)
                    (-3.12  5.03 -0.83  1.18)
                  A=( 0.56 -0.83  0.76  0.34).
                    (-0.10  1.18  0.34  1.18)

and

                          (  8.70 8.30)
                          (-13.35 2.13)
                        B=(  1.89 1.61).
                          ( -4.14 5.00)

Here A is symmetric positive-definite and must first be
factorized by F07FDF (DPOTRF).

The example program is not reproduced here. The source code for
all example programs is distributed with the NAG Foundation
Library software and should be available on-line.
```

```
\end{verbatim}
\endscroll
\end{page}
```

```
\section{nagm.ht}
\pagetitle{manpageXXm01}{nagm.ht}{ Sorting}
\begin{chunk}{nagm.ht}
\begin{page}{manpageXXm01}{NAG Documentation: m01}
\beginscroll
\begin{verbatim}


     M01(3NAG)         Foundation Library (12/10/92)         M01(3NAG)



          M01 -- Sorting                                Introduction -- M01
                                    Chapter M01
                                      Sorting

          1. Scope of the Chapter

          This chapter is concerned with sorting numeric data. It handles
          only the simplest types of data structure and it is concerned
          only with internal sorting -- that is, sorting a set of data
          which can all be stored within the program.

          Users with large files of data or complex data structures to be
          sorted should use a comprehensive sorting program or package.

          2. Background to the Problems

          The usefulness of sorting is obvious (perhaps a little too
          obvious, since sorting can be expensive and is sometimes done
          when not strictly necessary). Sorting may traditionally be
          associated with data-processing and non-numerical programming,
          but it has many uses within the realm of numerical analysis, for
          example, to arrange eigenvalues in ascending order of absolute
          value and in the ranking of observations for nonparametric
          statistics.

          The general problem may be defined as follows. We are given N
          items of data


                                    R ,R ,...,R .
                                     1  2      N

          Each item R  contains a key K  which can be ordered relative to
                     i                 i
          any other key according to some specified criterion (for example,
          ascending numeric value). The problem is to determine a
```

permutation

```
                      p(1),p(2),...,p(N)
```

which puts the keys in order:

```
                    K     <=K     <=...<=K
                     p(1)    p(2)        p(N)
```

Sometimes we may wish actually to rearrange the items so that their keys are in order; for other purposes we may simply require a table of indices so that the items can be referred to in sorted order; or yet again we may require a table of ranks, that is, the positions of each item in the sorted order.

For example, given the single-character items, to be sorted into alphabetic order:

```
     E    B    A    D    C
```

the indices of the items in sorted order are

```
     3    2    5    4    1
```

and the ranks of the items are

```
     5    2    1    4    3
```

Indices may be converted to ranks, and vice versa, by simply computing the inverse permutation.

The items may consist solely of the key (each item may simply be a number). On the other hand, the items may contain additional information (for example, each item may be an eigenvalue of a matrix and its associated eigenvector, the eigenvalue being the key). In the latter case there may be many distinct items with equal keys, and it may be important to preserve the original order among them (if this is achieved, the sorting is called ' stable').

There are a number of ingenious algorithms for sorting. For a fascinating discussion of them, and of the whole subject, see Knuth [1].

2.1. References

[1] Knuth D E (1973) The Art of Computer Programming, Vol 3. Addison-Wesley.

3. Recommendations on Choice and Use of Routines

Four categories of routines are provided:

- routines which rearrange the data into sorted order (M01C-);

- routines which determine the ranks of the data, leaving the data unchanged (M01D-);

- routines which rearrange the data according to pre-determined ranks (M01E-);

- service routines (M01Z-).

Routines are provided for sorting double precision data only.

If the task is simply to rearrange a one-dimensional array of data into sorted order, then M01CAF should be used, since this requires no extra workspace and is faster than any other method.

For many applications it is in fact preferable to separate the task of determining the sorted order (ranking) from the task of rearranging data into a pre-determined order; the latter task may not need to be performed at all. Frequently it may be sufficient to refer to the data in sorted order via an index vector, without rearranging it. Frequently also one set of data (e.g. a column of a matrix) is used for determining a set of ranks, which are then applied to other data (e.g. the remaining columns of the matrix).

To determine the ranks of a set of data, use an M01D- routine. Routines are provided for ranking one-dimensional arrays, and for ranking rows or columns of two-dimensional arrays.

To create an index vector so that data can be referred to in sorted order, first call an M01D- routine to determine the ranks, and then call M01ZAF to convert the vector of ranks into an index vector.

To rearrange data according to pre-determined ranks: use M01EAF if the data is stored in a one-dimensional array; or if the data is stored in a more complicated structure, use an index vector to generate a new copy of the data in the desired order.

Examples of these operations can be found in the routine documents of the relevant routines.

4. Index

```
 Ranking:
     columns of a matrix, double precision numbers         M01DJF
```

```
          rows of a matrix, double precision numbers           M01DEF
          vector, double precision numbers                     M01DAF
     Rearranging (according to pre-determined ranks):
          vector, double precision numbers                     M01EAF
     Service routines:
          invert a permutation (ranks to indices or vice       M01ZAF
          versa)
     Sorting (i.e., rearranging into sorted order):
          vector, double precision numbers                     M01CAF


     M01 -- Sorting                                  Contents -- M01
     Chapter M01

     Sorting

     M01CAF  Sort a vector, double precision numbers

     M01DAF  Rank a vector, double precision numbers

     M01DEF  Rank rows of a matrix, double precision numbers

     M01DJF  Rank columns of a matrix, double precision numbers

     M01EAF  Rearrange a vector according to given ranks, double
             precision numbers

     M01ZAF  Invert a permutation

\end{verbatim}
\endscroll
\end{page}
```

Sort vector of double precision numbers

— nagm.ht —

```
\begin{page}{manpageXXm01caf}{NAG Documentation: m01caf}
\beginscroll
\begin{verbatim}


     M01CAF(3NAG)      Foundation Library (12/10/92)      M01CAF(3NAG)
```

M01 -- Sorting M01CAF

M01CAF -- NAG Foundation Library Routine Document

Note: Before using this routine, please read the Users' Note for your implementation to check implementation-dependent details. The symbol (*) after a NAG routine name denotes a routine that is not included in the Foundation Library.

1. Purpose

M01CAF rearranges a vector of double precision numbers into ascending or descending order.

2. Specification

```
SUBROUTINE M01CAF (RV, M1, M2, ORDER, IFAIL)
INTEGER          M1, M2, IFAIL
DOUBLE PRECISION RV(M2)
CHARACTER*1      ORDER
```

3. Description

M01CAF is based on Singleton's implementation of the 'median-of-three' Quicksort algorithm [2], but with two additional modifications. First, small subfiles are sorted by an insertion sort on a separate final pass (Sedgewick [1]). Second, if a subfile is partitioned into two very unbalanced subfiles, the larger of them is flagged for special treatment: before it is partitioned, its end-points are swapped with two random points within it; this makes the worst case behaviour extremely unlikely.

4. References

[1] Sedgewick R (1978) Implementing Quicksort Programs. Comm. ACM. 21 847--857.

[2] Singleton R C (1969) An Efficient Algorithm for Sorting with Minimal Storage: Algorithm 347. Comm. ACM. 12 185--187.

5. Parameters

1: RV(M2) -- DOUBLE PRECISION array Input/Output
On entry: elements M1 to M2 of RV must contain double precision values to be sorted. On exit: these values are rearranged into sorted order.

2: M1 -- INTEGER Input

```
    On entry: the index of the first element of RV to be sorted.
    Constraint: M1 > 0.

 3: M2 -- INTEGER                                          Input

    On entry: the index of the last element of RV to be sorted.
    Constraint: M2 >= M1.

 4: ORDER -- CHARACTER*1                                   Input
    On entry: if ORDER is 'A', the values will be sorted into
    ascending (i.e., non-decreasing) order; if ORDER is 'D',
    into descending order. Constraint: ORDER = 'A' or 'D'.

 5: IFAIL -- INTEGER                                Input/Output
    On entry: IFAIL must be set to 0, -1 or 1. For users not
    familiar with this parameter (described in the Essential
    Introduction) the recommended value is 0.

    On exit: IFAIL = 0 unless the routine detects an error (see
    Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

IFAIL= 1
    On entry M2 < 1,

    or       M1 < 1,

    or       M1 > M2.

IFAIL= 2
    On entry ORDER is not 'A' or 'D'.

7. Accuracy

Not applicable.

8. Further Comments

The average time taken by the routine is approximately
proportional to n*logn, where n = M2 - M1 + 1. The worst case
                         2
time is proportional to n  but this is extremely unlikely to
occur.
```

```
         9. Example

         The example program reads a list of double precision numbers and
         sorts them into ascending order.

         The example program is not reproduced here. The source code for
         all example programs is distributed with the NAG Foundation
         Library software and should be available on-line.
\end{verbatim}
\endscroll
\end{page}
```

Ranks a vector of double precision numbers

— nagm.ht —

```
\begin{page}{manpageXXm01daf}{NAG Documentation: m01daf}
\beginscroll
\begin{verbatim}



     M01DAF(3NAG)      Foundation Library (12/10/92)      M01DAF(3NAG)



         M01 -- Sorting                                            M01DAF
                 M01DAF -- NAG Foundation Library Routine Document

         Note: Before using this routine, please read the Users' Note for
         your implementation to check implementation-dependent details.
         The symbol (*) after a NAG routine name denotes a routine that is
         not included in the Foundation Library.

         1. Purpose

         M01DAF ranks a vector of double precision numbers in ascending or
         descending order.

         2. Specification

                SUBROUTINE M01DAF (RV, M1, M2, ORDER, IRANK, IFAIL)
                INTEGER          M1, M2, IRANK(M2), IFAIL
                DOUBLE PRECISION RV(M2)
                CHARACTER*1      ORDER
```

3. Description

M01DAF uses a variant of list-merging, as described by Knuth [1] pp 165-166. The routine takes advantage of natural ordering in the data, and uses a simple list insertion in a preparatory pass to generate ordered lists of length at least 10. The ranking is stable: equal elements preserve their ordering in the input data.

4. References

[1] Knuth D E (1973) The Art of Computer Programming, Vol 3. Addison-Wesley.

5. Parameters

1: RV(M2) -- DOUBLE PRECISION array Input
On entry: elements M1 to M2 of RV must contain double precision values to be ranked.

2: M1 -- INTEGER Input
On entry: the index of the first element of RV to be ranked. Constraint: M1 > 0.

3: M2 -- INTEGER Input
On entry: the index of the last element of RV to be ranked. Constraint: M2 >= M1.

4: ORDER -- CHARACTER*1 Input
On entry: if ORDER is 'A', the values will be ranked in ascending (i.e., non-decreasing) order; if ORDER is 'D', into descending order. Constraint: ORDER = 'A' or 'D'.

5: IRANK(M2) -- INTEGER array Output
On exit: elements M1 to M2 of IRANK contain the ranks of the corresponding elements of RV. Note that the ranks are in the range M1 to M2: thus, if RV(i) is the first element in the rank order, IRANK(i) is set to M1.

6: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

```
          If on entry IFAIL = 0 or -1, explanatory error messages are
          output on the current error message unit (as defined by X04AAF).

          IFAIL= 1
               On entry M2 < 1,

               or       M1 < 1,

               or       M1 > M2.

          IFAIL= 2
               On entry ORDER is not 'A' or 'D'.

          7. Accuracy

          Not applicable.

          8. Further Comments

          The average time taken by the routine is approximately
          proportional to n*logn, where n = M2 - M1 + 1.

          9. Example

          The example program reads a list of double precision numbers and
          ranks them in ascending order.

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.
\end{verbatim}
\endscroll
\end{page}
```

———-

Ranks the rows of a matrix of double precision numbers

— nagm.ht —

```
\begin{page}{manpageXXm01def}{NAG Documentation: m01def}
\beginscroll
\begin{verbatim}
```

M01 -- Sorting M01DEF

M01DEF -- NAG Foundation Library Routine Document

Note: Before using this routine, please read the Users' Note for your implementation to check implementation-dependent details. The symbol (*) after a NAG routine name denotes a routine that is not included in the Foundation Library.

1. Purpose

M01DEF ranks the rows of a matrix of double precision numbers in ascending or descending order.

2. Specification

```
       SUBROUTINE M01DEF (RM, LDM, M1, M2, N1, N2, ORDER, IRANK,
      1                   IFAIL)
       INTEGER          LDM, M1, M2, N1, N2, IRANK(M2), IFAIL
       DOUBLE PRECISION RM(LDM,N2)
       CHARACTER*1      ORDER
```

3. Description

M01DEF ranks rows M1 to M2 of a matrix, using the data in columns N1 to N2 of those rows. The ordering is determined by first ranking the data in column N1, then ranking any tied rows according to the data in column N1 + 1, and so on up to column N2.

M01DEF uses a variant of list-merging, as described by Knuth [1] pp 165-166. The routine takes advantage of natural ordering in the data, and uses a simple list insertion in a preparatory pass to generate ordered lists of length at least 10. The ranking is stable: equal rows preserve their ordering in the input data.

4. References

[1] Knuth D E (1973) The Art of Computer Programming, Vol 3. Addison-Wesley.

5. Parameters

1: RM(LDM,N2) -- DOUBLE PRECISION array Input

On entry: columns N1 to N2 of rows M1 to M2 of RM must contain double precision data to be ranked.

```
 2:  LDM -- INTEGER                                          Input
     On entry:
     the first dimension of the array RM as declared in the
     (sub)program from which M01DEF is called.
     Constraint: LDM >= M2.

 3:  M1 -- INTEGER                                           Input
     On entry: the index of the first row of RM to be ranked.
     Constraint: M1 > 0.

 4:  M2 -- INTEGER                                           Input
     On entry: the index of the last row of RM to be ranked.
     Constraint: M2 >= M1.

 5:  N1 -- INTEGER                                           Input
     On entry: the index of the first column of RM to be used.
     Constraint: N1 > 0.

 6:  N2 -- INTEGER                                           Input
     On entry: the index of the last column of RM to be used.
     Constraint: N2 >= N1.

 7:  ORDER -- CHARACTER*1                                    Input
     On entry: if ORDER is 'A', the rows will be ranked in
     ascending (i.e., non-decreasing) order; if ORDER is 'D',
     into descending order. Constraint: ORDER = 'A' or 'D'.

 8:  IRANK(M2) -- INTEGER array                             Output
     On exit: elements M1 to M2 of IRANK contain the ranks of the
     corresponding rows of RM. Note that the ranks are in the
     range M1 to M2: thus, if the ith row of RM is the first in
     the rank order, IRANK(i) is set to M1.

 9:  IFAIL -- INTEGER                                 Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

IFAIL= 1
     On entry M2 < 1,
```

```
     or      N2 < 1,

     or      M1 < 1,

     or      M1 > M2,

     or      N1 < 1,

     or      N1 > N2,

     or      LDM < M2.

IFAIL= 2
     On entry ORDER is not 'A' or 'D'.

7. Accuracy

Not applicable.

8. Further Comments

The average time taken by the routine is approximately
proportional to n*logn, where n = M2 - M1 + 1.

9. Example

The example program reads a matrix of double precision numbers
and ranks the rows in ascending order.

The example program is not reproduced here. The source code for
all example programs is distributed with the NAG Foundation
Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

———-

Ranks the columns of a matrix of double precision numbers

— nagm.ht —

```
\begin{page}{manpageXXm01djf}{NAG Documentation: m01djf}
\beginscroll
\begin{verbatim}
```

M01 -- Sorting M01DJF

M01DJF -- NAG Foundation Library Routine Document

Note: Before using this routine, please read the Users' Note for your implementation to check implementation-dependent details. The symbol (*) after a NAG routine name denotes a routine that is not included in the Foundation Library.

1. Purpose

M01DJF ranks the columns of a matrix of double precision numbers in ascending or descending order.

2. Specification

```
       SUBROUTINE M01DJF (RM, LDM, M1, M2, N1, N2, ORDER, IRANK,
      1                   IFAIL)
       INTEGER          LDM, M1, M2, N1, N2, IRANK(N2), IFAIL
       DOUBLE PRECISION RM(LDM,N2)
       CHARACTER*1      ORDER
```

3. Description

M01DJF ranks columns N1 to N2 of a matrix, using the data in rows M1 to M2 of those columns. The ordering is determined by first ranking the data in row M1, then ranking any tied columns according to the data in row M1 + 1, and so on up to row M2.

M01DJF uses a variant of list-merging, as described by Knuth [1] pp 165-166. The routine takes advantage of natural ordering in the data, and uses a simple list insertion in a preparatory pass to generate ordered lists of length at least 10. The ranking is stable: equal columns preserve their ordering in the input data.

4. References

[1] Knuth D E (1973) The Art of Computer Programming, Vol 3. Addison-Wesley.

5. Parameters

1: RM(LDM,N2) -- DOUBLE PRECISION array Input

On entry: rows M1 to M2 of columns N1 to N2 of RM must

```
    contain double precision data to be ranked.

 2: LDM -- INTEGER                                     Input
    On entry:
    the first dimension of the array RM as declared in the
    (sub)program from which M01DJF is called.
    Constraint: LDM >= M2.

 3: M1 -- INTEGER                                      Input
    On entry: the index of the first row of RM to be used.
    Constraint: M1 > 0.

 4: M2 -- INTEGER                                      Input
    On entry: the index of the last row of RM to be used.
    Constraint: M2 >= M1.

 5: N1 -- INTEGER                                      Input
    On entry: the index of the first column of RM to be ranked.
    Constraint: N1 > 0.

 6: N2 -- INTEGER                                      Input
    On entry: the index of the last column of RM to be ranked.
    Constraint: N2 >= N1.

 7: ORDER -- CHARACTER*1                               Input
    On entry: if ORDER is 'A', the columns will be ranked in
    ascending (i.e., non-decreasing) order; if ORDER is 'D',
    into descending order. Constraint: ORDER = 'A' or 'D'.

 8: IRANK(N2) -- INTEGER array                        Output
    On exit: elements N1 to N2 of IRANK contain the ranks of the
    corresponding columns of RM. Note that the ranks are in the
    range N1 to N2: thus, if the ith column of RM is the first
    in the rank order, IRANK(i) is set to N1.

 9: IFAIL -- INTEGER                            Input/Output
    On entry: IFAIL must be set to 0, -1 or 1. For users not
    familiar with this parameter (described in the Essential
    Introduction) the recommended value is 0.

    On exit: IFAIL = 0 unless the routine detects an error (see
    Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).
```

```
          IFAIL= 1
               On entry M2 < 1,

               or       N2 < 1,

               or       M1 < 1,

               or       M1 > M2,

               or       N1 < 1,

               or       N1 > N2,

               or       LDM < M2.

          IFAIL= 2
               On entry ORDER is not 'A' or 'D'.

          7. Accuracy

          Not applicable.

          8. Further Comments

          The average time taken by the routine is approximately
          proportional to n*logn, where n = N2 - N1 + 1.

          9. Example

          The example program reads a matrix of double precision numbers
          and ranks the columns in ascending order.

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

———-

Rearranges a vector of double precision numbers

— nagm.ht —

```
\begin{page}{manpageXXm01eaf}{NAG Documentation: m01eaf}
```

```
\beginscroll
\begin{verbatim}


     M01EAF(3NAG)      Foundation Library (12/10/92)      M01EAF(3NAG)



          M01 -- Sorting                                         M01EAF
                  M01EAF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          M01EAF rearranges a vector of double precision numbers into the
          order specified by a vector of ranks.

          2. Specification

                 SUBROUTINE M01EAF (RV, M1, M2, IRANK, IFAIL)
                 INTEGER          M1, M2, IRANK(M2), IFAIL
                 DOUBLE PRECISION RV(M2)

          3. Description

          M01EAF is designed to be used typically in conjunction with the
          M01D- ranking routines. After one of the M01D- routines has been
          called to determine a vector of ranks, M01EAF can be called to
          rearrange a vector of real numbers into the rank order. If the
          vector of ranks has been generated in some other way, then
          M01ZBF(*) should be called to check its validity before M01EAF is
          called.

          4. References

          None.

          5. Parameters

           1:  RV(M2) -- DOUBLE PRECISION array                Input/Output
               On entry: elements M1 to M2 of RV must contain double
               precision values to be rearranged. On exit: these values are
               rearranged into rank order. For example, if IRANK(i) = M1,
               then the initial value of RV(i) is moved to RV(M1).
```

```
 2:  M1 -- INTEGER                                          Input

 3:  M2 -- INTEGER                                          Input
     On entry: M1 and M2 must specify the range of the ranks
     supplied in IRANK and the elements of RV to be rearranged.
     Constraint: 0 < M1 <= M2.

 4:  IRANK(M2) -- INTEGER array                             Input
     On entry: elements M1 to M2 of IRANK must contain a
     permutation of the integers M1 to M2, which are interpreted
     as a vector of ranks.

 5:  IFAIL -- INTEGER                                Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

IFAIL= 1
     On entry M2 < 1,

     or       M1 < 1,

     or       M1 > M2.

IFAIL= 2
     Elements M1 to M2 of IRANK contain a value outside the range
     M1 to M2.

IFAIL= 3
     Elements M1 to M2 of IRANK contain a repeated value.

If IFAIL = 2 or 3, elements M1 to M2 of IRANK do not contain a
permutation of the integers M1 to M2. On exit, the contents of RV
may be corrupted. To check the validity of IRANK without the risk
of corrupting RV, use M01ZBF(*).

7. Accuracy

Not applicable.
```

```
        8. Further Comments

        The average time taken by the routine is approximately
        proportional to n, where n = M2 - M1 + 1.

        9. Example

        The example program reads a matrix of double precision numbers
        and rearranges its rows so that the elements of the kth column
        are in ascending order. To do this, the program first calls
        M01DAF to rank the elements of the kth column, and then calls
        M01EAF to rearrange each column into the order specified by the
        ranks. The value of k is read from the data-file.

        The example program is not reproduced here. The source code for
        all example programs is distributed with the NAG Foundation
        Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

———-

Inverts a permutation

— nagm.ht —

```
\begin{page}{manpageXXm01zaf}{NAG Documentation: m01zaf}
\beginscroll
\begin{verbatim}



     M01ZAF(3NAG)      Foundation Library (12/10/92)      M01ZAF(3NAG)



        M01 -- Sorting                                            M01ZAF
                M01ZAF -- NAG Foundation Library Routine Document

        Note: Before using this routine, please read the Users' Note for
        your implementation to check implementation-dependent details.
        The symbol (*) after a NAG routine name denotes a routine that is
        not included in the Foundation Library.

        1. Purpose
```

M01ZAF inverts a permutation, and hence converts a rank vector to an index vector, or vice versa.

2. Specification

```
       SUBROUTINE M01ZAF (IPERM, M1, M2, IFAIL)
       INTEGER          IPERM(M2), M1, M2, IFAIL
```

3. Description

There are two common ways of describing a permutation using an integer vector IPERM. The first uses ranks: IPERM(i) holds the position to which the ith data element should be moved in order to sort the data; in other words its rank in the sorted order. The second uses indices: IPERM(i) holds the current position of the data element which would occur in ith position in sorted order. For example, given the values

```
       3.5    5.9    2.9    0.5
```

to be sorted in ascending order, the ranks would be

```
       3      4      2      1
```

and the indices would be

```
       4      3      1      2
```

The M01D- routines generate ranks, and the M01E- routines require ranks to be supplied to specify the re-ordering. However if it is desired simply to refer to the data in sorted order without actually re-ordering them, indices are more convenient than ranks (see the example in Section 9).

M01ZAF can be used to convert ranks to indices, or indices to ranks, as the two permutations are inverses of one another.

4. References

None.

5. Parameters

1: IPERM(M2) -- INTEGER array Input/Output
On entry: elements M1 to M2 of IPERM must contain a permutation of the integers M1 to M2. On exit: these elements contain the inverse permutation of the integers M1 to M2.

```
 2:  M1 -- INTEGER                                       Input

 3:  M2 -- INTEGER                                       Input
     On entry: M1 and M2 must specify the range of elements used
     in the array IPERM and the range of values in the
     permutation, as specified under IPERM. Constraint: 0 < M1 <=
     M2.

 4:  IFAIL -- INTEGER                             Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

IFAIL= 1
     On entry M2 < 1,

     or       M1 < 1,
     or       M1 > M2.
IFAIL= 2
     Elements M1 to M2 of IPERM contain a value outside the range
     M1 to M2.

IFAIL= 3
     Elements M1 to M2 of IPERM contain a repeated value.

If IFAIL = 2 or 3, elements M1 to M2 of IPERM do not contain a
permutation of the integers M1 to M2; on exit these elements are
usually corrupted. To check the validity of a permutation without
the risk of corrupting it, use M01ZBF(*).

7. Accuracy

Not applicable.

8. Further Comments

None.

9. Example
```

```
          The example program reads a matrix of double precision numbers
          and prints its rows in ascending order as ranked by M01DEF. The
          program first calls M01DEF to rank the rows, and then calls
          M01ZAF to convert the rank vector to an index vector, which is
          used to refer to the rows in sorted order.

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.
\end{verbatim}
\endscroll
\end{page}
```

22.6 nags.ht

Approximations of Special Functions

— nags.ht —

```
\begin{page}{manpageXXs}{NAG Documentation: s}
\beginscroll
\begin{verbatim}



     S(3NAG)          Foundation Library (12/10/92)          S(3NAG)



          S -- Approximations of Special Functions        Introduction -- S
                                     Chapter S
                        Approximations of Special Functions

          1. Scope of the Chapter

          This chapter is concerned with the provision of some commonly
          occurring physical and mathematical functions.

          2. Background to the Problems

          The majority of the routines in this chapter approximate real-
          valued functions of a single real argument, and the techniques
          involved are described in Section 2.1. In addition the chapter
          contains routines for elliptic integrals (see Section 2.2),
```

Bessel and Airy functions of a complex argument (see Section 2.3), and exponential of a complex argument.

2.1. Functions of a Single Real Argument

Most of the routines for functions of a single real argument have been based on truncated Chebyshev expansions. This method of approximation was adopted as a compromise between the conflicting requirements of efficiency and ease of implementation on many different machine ranges. For details of the reasons behind this choice and the production and testing procedures followed in constructing this chapter see Schonfelder [7].

Basically if the function to be approximated is f(x), then for xis in [a,b] an approximation of the form

$$f(x)=g(x) {\sum_{r=0}}' C_r T_r(t)$$

is used, ($\sum'$ denotes, according to the usual convention, a summation in which the first term is halved), where g(x) is some suitable auxiliary function which extracts any singularities, asymptotes and, if possible, zeros of the function in the range in question and t=t(x) is a mapping of the general range [a,b] to the specific range [-1,+1] required by the Chebyshev polynomials, $T_r(t)$. For a detailed description of the properties of the Chebyshev polynomials see Clenshaw [5] and Fox and Parker [6].

The essential property of these polynomials for the purposes of function approximation is that $T_n(t)$ oscillates between +-1 and it takes its extreme values n+1 times in the interval [-1,+1]. Therefore, provided the coefficients C_r decrease in magnitude sufficiently rapidly the error made by truncating the Chebyshev expansion after n terms is approximately given by

$$E(t)\simeq C_n T_n(t)$$

That is the error oscillates between $\pm C_n$ and takes its extreme value n+1 times in the interval in question. Now this is just the condition that the approximation be a mini-max representation, one which minimizes the maximum error. By suitable choice of the interval, [a,b], the auxiliary function, g(x), and the mapping of the independent variable, t(x), it is almost always possible to obtain a Chebyshev expansion with rapid convergence and hence truncations that provide near mini-max polynomial approximations to the required function. The difference between the true mini-max polynomial and the truncated Chebyshev expansion is seldom sufficiently great to be of significance.

The evaluation of the Chebyshev expansions follows one of two methods. The first and most efficient, and hence most commonly used, works with the equivalent simple polynomial. The second method, which is used on the few occasions when the first method proves to be unstable, is based directly on the truncated Chebyshev series and uses backward recursion to evaluate the sum. For the first method, a suitably truncated Chebyshev expansion (truncation is chosen so that the error is less than the machine precision) is converted to the equivalent simple polynomial. That is we evaluate the set of coefficients b_r such that

$$y(t)=\sum_{r=0}^{n-1} b_r t^r = \sum_{r=0}^{n-1}{}' C_r T_r(t).$$

The polynomial can then be evaluated by the efficient Horner's method of nested multiplications,

$$y(t)=(b_0+t(b_1+t(b_2+...t(b_{n-2}+tb_{n-1})))...).$$

This method of evaluation results in efficient routines but for some expansions there is considerable loss of accuracy due to cancellation effects. In these cases the second method is used. It is well known that if

$$b_{n-1}=C_{n-1}$$

$$b_{n-2}=2tb_{n-1}+C_{n-2}$$

$$b_j=2tb_{j+1}-b_{j+2}+C_j, \quad j=n-3,n-4,...,0$$

then

```
--'         1
>  C T (t)= -(b -b )
--  r r     2  0  2
r=0
```

and this is always stable. This method is most efficiently
implemented by using three variables cyclically and explicitly
constructing the recursion.

That is,

```
(alpha) = C
           n-1
(beta) = 2t(alpha)+C
                    n-2
(gamma) = 2t(beta)-(alpha)+C
                            n-3
(alpha) = 2t(gamma)-(beta)+C
                            n-4
(beta) = 2t(alpha)-(gamma)+C
                            n-5
  ...
  ...
(alpha) = 2t(gamma)-(beta)+C   (say)
                            2
(beta) = 2t(alpha)-(gamma)+C
                            1
                        1
y(t) = t(beta)-(alpha)+ -C
                        2 0
```

The auxiliary functions used are normally functions compounded of
simple polynomial (usually linear) factors extracting zeros, and
the primary compiler-provided functions, sin, cos, ln, exp, sqrt,
which extract singularities and/or asymptotes or in some cases
basic oscillatory behaviour, leaving a smooth well-behaved
function to be approximated by the Chebyshev expansion which can
therefore be rapidly convergent.

The mappings of [a,b] to [-1,+1] used, range from simple linear
mappings to the case when b is infinite and considerable
improvement in convergence can be obtained by use of a bilinear
form of mapping. Another common form of mapping is used when the
function is even, that is it involves only even powers in its
expansion. In this case an approximation over the whole interval

```
                                    (x)2
[-a,a] can be provided using a mapping t=2(-) -1. This embodies
                                    (a)
```

the evenness property but the expansion in t involves all powers
and hence removes the necessity of working with an expansion with

half its coefficients zero.

For many of the routines an analysis of the error in principle is given, viz, if E and (nabla) are the absolute errors in function and argument and (epsilon) and (delta) are the corresponding relative errors, then

```
                        E~=|f'(x)|(nabla)

                        E~=|xf'(x)|(delta)

                                | xf'(x)|
                    (epsilon)~=| ------|(delta)
                                |  f(x) |
```

If we ignore errors that arise in the argument of the function by propagation of data errors etc and consider only those errors that result from the fact that a real number is being represented in the computer in floating-point form with finite precision, then (delta) is bounded and this bound is independent of the magnitude of x; e.g. on an 11-digit machine

```
                                     -11
                        |(delta)|<=10   .
```

(This of course implies that the absolute error (nabla)=x(delta) is also bounded but the bound is now dependent on x). However because of this the last two relations above are probably of more interest. If possible the relative error propagation is discussed; that is the behaviour of the error amplification factor |xf'(x)/f(x)| is described, but in some cases, such as near zeros of the function which cannot be extracted explicitly, absolute error in the result is the quantity of significance and here the factor |xf'(x)| is described. In general, testing of the functions has shown that their error behaviour follows fairly well these theoretical error behaviours. In regions, where the error amplification factors are less than or of the order of one, the errors are slightly larger than the above predictions. The errors are here limited largely by the finite precision of arithmetic in the machine but (epsilon) is normally no more than a few times greater than the bound on (delta). In regions where the amplification factors are large, order of ten or greater, the theoretical analysis gives a good measure of the accuracy obtainable.

It should be noted that the definitions and notations used for the functions in this chapter are all taken from Abramowitz and Stegun [1]. Users are strongly recommended to consult this book for details before using the routines in this chapter.

2.2. Approximations to Elliptic Integrals

The functions provided here are symmetrised variants of the classic elliptic integrals. These alternative definitions have been suggested by Carlson (see [2], [3] and [4]) and he also developed the basic algorithms used in this chapter.

The standard integral of the first kind is represented by

$$R_F(x,y,z)=\frac{1}{2}\int_0^{\infty}\frac{dt}{\sqrt{(t+x)(t+y)(t+z)}}$$

where x,y,z>=0 and at most one may be equal to zero.

The normalisation factor, $\frac{1}{2}$, is chosen so as to make

$$R_F(x,x,x)=1/\sqrt{x}.$$

If any two of the variables are equal, R_F degenerates into the second function

$$R_C(x,y)=R_F(x,y,y)=\frac{1}{2}\int_0^{\infty}\frac{dt}{\sqrt{t+x}(t+y)}$$

where the argument restrictions are now x>=0 and y/=0.

This function is related to the logarithm or inverse hyperbolic functions if 0<y<x, and to the inverse circular functions if 0<=x<=y.

The integrals of the second kind are defined by

$$R_D(x,y,z)=\frac{3}{2}\int^{\infty}\frac{dt}{\quad}$$

```
                           0      /                3
                                 \/ (t+x)(t+y)(t+z)
```

with z>0, x>=0 and y>=0 but only one of x or y may be zero.

The function is a degenerate special case of the integral of the third kind

```
                          infty
                        3 /                 dt
      R (x,y,z,(rho))= - |     ----------------------------
       J                2 /     (                  )
                          0     (\/(t+x)(t+y)(t+z))(t+(rho))
```

```
with (rho)/=0, x,y,z>=0 with at most one equality holding. Thus
R (x,y,z)=R (x,y,z,z). The normalisation of both these functions
 D         J
is chosen so that
```

```
                 R (x,x,x)=R (x,x,x,x)=1/(x\/x)
                  D         J
```

The algorithms used for all these functions are based on duplication theorems. These allow a recursion system to be established which constructs a new set of arguments from the old using a combination of arithmetic and geometric means. The value of the function at the original arguments can then be simply related to the value at the new arguments. These recursive reductions are used until the arguments differ from the mean by an amount small enough for a Taylor series about the mean to give sufficient accuracy when retaining terms of order less than six. Each step of the recurrences reduces the difference from the mean by a factor of four, and as the truncation error is of order six,

```
                                     -n
the truncation error goes like (4096)   , where n is the number of
iterations.
```

The above forms can be related to the more traditional canonical forms (see Abramowitz and Stegun [1], 17.2).

```
                2               2               2
If we write q=cos (phi),r=1-m.sin (phi),s=1+n.sin (phi), where
         1
0<(phi)<= -(pi), we have: the elliptic integral of the first
         2
kind:
```

```
              sin(phi)      -1/2      1-/2
```

$$F((\phi)|m)=\int_{0}(1-t^{2})\quad(1-mt^{2})\quad dt=\sin(\phi).R_{F}(q,r,1);$$

the elliptic integral of the second kind:

$$E((\phi)|m)=\int_{0}^{\sin(\phi)}(1-t^{2})^{-1/2}(1-mt^{2})^{-1/2}\,dt$$

$$=\sin(\phi).R_{F}(q,r,1)-\frac{1}{3}m.\sin^{3}(\phi).R_{D}(q,r,1)$$

the elliptic integral of the third kind:

$$(\Pi)(n;(\phi)|m)=\int_{0}^{\sin(\phi)}(1-t^{2})^{-1/2}(1-mt^{2})^{-1/2}(1+nt^{2})^{-1}\,dt$$

$$=\sin(\phi).R_{F}(q,r,1)-\frac{1}{3}n.\sin^{3}(\phi).R_{J}(q,r,1,s)$$

Also the complete elliptic integral of the first kind:

$$K(m)=\int_{0}^{(\pi)/2}(1-m.\sin^{2}(\theta))^{-1/2}\,d(\theta)=R_{F}(0,1-m,1);$$

the complete elliptic integral of the second kind:

$$E(m)=\int_{0}^{(\pi)/2}(1-m.\sin^{2}(\theta))^{1/2}\,d(\theta)=R_{F}(0,1-m,1)-\frac{1}{3}mR_{D}(0,1-m,1).$$

2.3. Bessel and Airy Functions of a Complex Argument

The routines for Bessel and Airy functions of a real argument are based on Chebyshev expansions, as described in Section 2.1. The routines for functions of a complex argument, however, use different methods. These routines relate all functions to the modified Bessel functions $I_{\nu}(z)$ and $K_{\nu}(z)$ computed in the right-half complex plane, including their analytic continuations. I_{ν} and K_{ν} are computed by different methods according to the values of zand ν. The methods include power series, asymptotic expansions and Wronskian evaluations. The relations between functions are based on well known formulae (see Abramowitz and Stegun [1]).

2.4. References

[1] Abramowitz M and Stegun I A (1968) Handbook of Mathematical Functions. Dover Publications.

[2] Carlson B C (1977) Special Functions of Applied Mathematics. Academic Press.

[3] Carlson B C (1965) On Computing Elliptic Integrals and Functions. J Math Phys. 44 36--51.

[4] Carlson B C (1977) Elliptic Integrals of the First Kind. SIAM J Math Anal. 8 231--242.

[5] Clenshaw C W (1962) Mathematical Tables. Chebyshev Series for Mathematical Functions. HMSO.

[6] Fox L and Parker I B (1968) Chebyshev Polynomials in Numerical Analysis. Oxford University Press.

[7] Schonfelder J L (1976) The Production of Special Function Routines for a Multi-Machine Library. Software Practice and Experience. 6(1)

3. Recommendations on Choice and Use of Routines

3.1. Elliptic Integrals

IMPORTANT ADVICE: users who encounter elliptic integrals in the course of their work are strongly recommended to look at transforming their analysis directly to one of the Carlson forms, rather than the traditional canonical Legendre forms. In general, the extra symmetry of the Carlson forms is likely to simplify the analysis, and these symmetric forms are much more stable to calculate.

The routine S21BAF for R_C is largely included as an auxiliary to the other routines for elliptic integrals. This integral essentially calculates elementary functions, e.g.

$$\ln x=(x-1).R_C\left(\left(\frac{1+x}{2}\right)^2,x\right),x>0;$$

$$\arcsin x=x.R_C(1-x^2,1),|x|<=1;$$

$$\mathrm{arcsinh}\,x=x.R_C(1+x^2,1),\quad \text{etc}$$

In general this method of calculating these elementary functions is not recommended as there are usually much more efficient specific routines available in the Library. However, S21BAF may be used, for example, to compute $\ln x/(x-1)$ when x is close to 1, without the loss of significant figures that occurs when $\ln x$ and $x-1$ are computed separately.

3.2. Bessel and Airy Functions

For computing the Bessel functions $J_{\nu}(x)$, $Y_{\nu}(x)$, $I_{\nu}(x)$ and $K_{\nu}(x)$ where x is real and $\nu=0$ or 1, special routines are provided, which are much faster than the more general routines that allow a complex argument and arbitrary real $\nu>=0$ functions and their derivatives Ai(x), Bi(x), Ai'(x), Bi'(x) for a real argument which are much faster than the routines for complex arguments.

3.3. Index

Airy function, Ai, real argument	S17AGF
Airy function, Ai', real argument	S17AJF
Airy function, Ai or Ai', complex argument, optionally scaled	S17DGF
Airy function, Bi, real argument	S17AHF
Airy function, Bi', real argument	S17AKF
Airy function, Bi or Bi', complex argument, optionally scaled	S17DHF
Bessel function, J_0, real argument	S17AEF
Bessel function, J_1, real argument	S17AFF
Bessel function, J_{ν}, complex argument, optionally	S17DEF

scaled	
Bessel function, Y_0, real argument	S17ACF
Bessel function, Y_1, real argument	S17ADF
Bessel function, $Y_{(nu)}$, complex argument, optionally scaled	S17DCF
Complement of the Error function	S15ADF
Cosine Integral	S13ACF
Elliptic integral, symmetrised, degenerate of 1st kind, R_C	S21BAF
Elliptic integral, symmetrised, of 1st kind, R_F	S21BBF
Elliptic integral, symmetrised, of 2nd kind, R_D	S21BCF
Elliptic integral, symmetrised, of 3rd kind, R_J	S21BDF
Erf, real argument	S15AEF
Erfc, real argument	S15ADF
Error function	S15AEF
Exponential, complex	S01EAF
Exponential Integral	S13AAF
Fresnel Integral, C	S20ADF
Fresnel Integral, S	S20ACF
Gamma function	S14AAF
Gamma function, incomplete	S14BAF
Generalized Factorial function	S14AAF
Hankel function $H^{(1)}_{(nu)}$ or $H^{(2)}_{(nu)}$, complex argument, optionally scaled	S17DLF
Incomplete Gamma function	S14BAF
Jacobian elliptic functions, sn, cn, dn	S21CAF
Kelvin function, bei x	S19ABF
Kelvin function, ber x	S19AAF
Kelvin function, kei x	S19ADF
Kelvin function, ker x	S19ACF
Logarithm of Gamma function	S14ABF
Modified Bessel function, I_0, real argument	S18AEF
Modified Bessel function, I_1, real argument	S18AFF
Modified Bessel function, $I_{(nu)}$, complex argument, optionally scaled	S18DEF
Modified Bessel function, K_0, real argument	S18ACF
Modified Bessel function, K , real argument	S18ADF

```
                              1
Modified Bessel function, K    , complex argument,         S18DCF
                             (nu)
optionally scaled
Sine integral                                              S13ADF


S -- Approximations of Special Functions            Contents -- S
Chapter S

Approximations of Special Functions


                                 z
S01EAF  Complex exponential, e

S13AAF  Exponential integral E (x)
                               1

S13ACF  Cosine integral Ci(x)

S13ADF  Sine integral Si(x)

S14AAF  Gamma function

S14ABF  Log Gamma function

S14BAF  Incomplete gamma functions P(a,x) and Q(a,x)

S15ADF  Complement of error function erfc x

S15AEF  Error function erf x

S17ACF  Bessel function Y (x)
                         0

S17ADF  Bessel function Y (x)
                         1

S17AEF  Bessel function J (x)
                         0

S17AFF  Bessel function J (x)
                         1

S17AGF  Airy function Ai(x)

S17AHF  Airy function Bi(x)

S17AJF  Airy function Ai'(x)
```

S17AKF Airy function Bi'(x)

S17DCF Bessel functions $Y_{(nu)+a}(z)$, real a>=0, complex z, (nu)=0,1,2,...

S17DEF Bessel functions $J_{(nu)+a}(z)$, real a>=0, complex z, (nu)=0,1,2,...

S17DGF Airy functions Ai(z) and Ai'(z), complex z

S17DHF Airy functions Bi(z) and Bi'(z), complex z

S17DLF Hankel functions $H^{(j)}_{(nu)+a}(z)$, j=1,2, real a>=0, complex z, (nu)=0,1,2,...

S18ACF Modified Bessel function $K_0(x)$

S18ADF Modified Bessel function $K_1(x)$

S18AEF Modified Bessel function $I_0(x)$

S18AFF Modified Bessel function $I_1(x)$

S18DCF Modified Bessel functions $K_{(nu)+a}(z)$, real a>=0, complex z, (nu)=0,1,2,...

S18DEF Modified Bessel functions $I_{(nu)+a}(z)$, real a>=0, complex z, (nu)=0,1,2,...

S19AAF Kelvin function ber x

S19ABF Kelvin function bei x

S19ACF Kelvin function ker x

S19ADF Kelvin function kei x

S20ACF Fresnel integral S(x)

S20ADF Fresnel integral C(x)

```
          S21BAF  Degenerate symmetrised elliptic integral of 1st kind
                  R (x,y)
                   C

          S21BBF  Symmetrised elliptic integral of 1st kind R (x,y,z)
                                                             F

          S21BCF  Symmetrised elliptic integral of 2nd kind R (x,y,z)
                                                             D

          S21BDF  Symmetrised elliptic integral of 3rd kind R (x,y,z,r)
                                                             J

\end{verbatim}
\endscroll
\end{page}
```

———-

Exponential function e^z , for complex z

— nags.ht —

```
\begin{page}{manpageXXs01eaf}{NAG Documentation: s01eaf}
\beginscroll
\begin{verbatim}



     S01EAF(3NAG)      Foundation Library (12/10/92)      S01EAF(3NAG)



          S01 -- Approximations of Special Functions                S01EAF
                  S01EAF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

                                                    z
          S01EAF evaluates the exponential function e , for complex z.
```

2. Specification

```
COMPLEX(KIND(1.0D0)) FUNCTION S01EAF (Z, IFAIL)
INTEGER                             IFAIL
COMPLEX(KIND(1.0D0))                Z
```

3. Description

This routine evaluates the exponential function e^z, taking care to avoid machine overflow, and giving a warning if the result cannot be computed to more than half precision. The function is evaluated as $e^z=e^x(\cos y+i\sin y)$, where x and y are the real and imaginary parts respectively of z.

Since cosy and siny are less than or equal to 1 in magnitude, it is possible that e^x may overflow although $e^x\cos y$ or $e^x\sin y$ does not. In this case the alternative formula $\mathrm{sign}(\cos y)e^{x+\ln|\cos y|}$ is used for the real part of the result, and $\mathrm{sign}(\sin y)e^{x+\ln|\sin y|}$ for the imaginary part. If either part of the result still overflows, a warning is returned through parameter IFAIL.

If Im z is too large, precision may be lost in the evaluation of siny and cosy. Again, a warning is returned through IFAIL.

4. References

None.

5. Parameters

1: Z -- COMPLEX(KIND(1.0D0)) Input
On entry: the argument z of the function.

2: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1

The real part of the result overflows, and is set to the largest safe number with the correct sign. The imaginary part of the result is meaningful.

IFAIL= 2

The imaginary part of the result overflows, and is set to the largest safe number with the correct sign. The real part of the result is meaningful.

IFAIL= 3

Both real and imaginary parts of the result overflow, and are set to the largest safe number with the correct signs.

IFAIL= 4

The computed result is accurate to less than half precision, due to the size of Im z.

IFAIL= 5

The computed result has no precision, due to the size of Im z, and is set to zero.

7. Accuracy

Accuracy is limited in general only by the accuracy of the Fortran intrinsic functions in the computation of siny, cosy and e^x, where x=Re z, y=Im z. As y gets larger, precision will probably be lost due to argument reduction in the evaluation of the sine and cosine functions, until the warning error IFAIL = 4 occurs when y gets larger than \/1/(epsilon), where (epsilon) is the machine precision. Note that on some machines, the intrinsic functions SIN and COS will not operate on arguments larger than about \/1/(epsilon), and so IFAIL can never return as 4.

In the comparatively rare event that the result is computed by the formulae $sign(cosy)e^{x+ln|cosy|}$ and $sign(siny)e^{x+ln|siny|}$, a further small loss of accuracy may be expected due to rounding errors in the logarithmic function.

8. Further Comments

None.

```
          9. Example

          The example program reads values of the argument z from a file,
          evaluates the function at each value of z and prints the results.

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

———-

Returns the value of the exponential integral $E(x)$

— nags.ht —

```
\begin{page}{manpageXXs13aaf}{NAG Documentation: s13aaf}
\beginscroll
\begin{verbatim}



     S13AAF(3NAG)      Foundation Library (12/10/92)      S13AAF(3NAG)



          S13 -- Approximations of Special Functions                S13AAF
                  S13AAF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          S13AAF returns the value of the exponential integral E (x), via
                                                                 1
          the routine name.

          2. Specification

                 DOUBLE PRECISION FUNCTION S13AAF (X, IFAIL)
                 INTEGER          IFAIL
```

DOUBLE PRECISION X

3. Description

The routine calculates an approximate value for

$$E_1(x)=\int_x^{\infty}\frac{e^{-u}}{u}\,du\ ,\ x>0.$$

For $0<x<=4$, the approximation is based on the Chebyshev expansion

$$E_1(x)=y(t)-\ln x=\sum_r{}' a_r T_r(t)-\ln x\ ,\ \text{where } t=\frac{1}{2}x-1.$$

For $x>4$,

$$E_1(x)=\frac{e^{-x}}{x}y(t)=\frac{e^{-x}}{x}\sum_r{}' a_r T_r(t),$$

$$\text{where}\quad t=-1.0+14.5/(x+3.25)=\frac{11.25-x}{3.25+x}.$$

In both cases, $-1<=t<=+1$.

To guard against producing underflows, if $x>x_{hi}$ the result is set directly to zero. For the value x_{hi} see the Users' Note for your implementation.

4. References

[1] Abramowitz M and Stegun I A (1965) Handbook of Mathematical Functions. Dover Publications. Ch. 26.

5. Parameters

1: X -- DOUBLE PRECISION Input
On entry: the argument x of the function. Constraint: X > 0.
0.

2: IFAIL -- INTEGER Input/Output
Before entry, IFAIL must be assigned a value. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

Unless the routine detects an error (see Section 6), IFAIL contains 0 on exit.

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1
The routine has been called with an argument less than or equal to zero for which the function is not defined. The result returned is zero.

7. Accuracy

If (delta) and (epsilon) are the relative errors in argument and result respectively, then in principle,

$$|\epsilon| \simeq \left| \frac{e^{-x}}{E_1(x)} \times \delta \right|$$

so the relative error in the argument is amplified in the result by at least a factor $e^{-x}/E_1(x)$. The equality should hold if (delta) is greater than the machine precision ((delta) due to data errors etc) but if (delta) is simply a result of round-off in the machine representation, it is possible that an extra figure may be lost in internal calculation and round-off.

The behaviour of this amplification factor is shown in Figure 1.

Figure 1
Please see figure in printed Reference Manual

It should be noted that, for small x, the amplification factor tends to zero and eventually the error in the result will be limited by machine precision.

For large x,

$$\epsilon \simeq x\delta = \Delta,$$

```
          the absolute error in the argument.

          8. Further Comments

          None.

          9. Example

          The example program reads values of the argument x from a file,
          evaluates the function at each value of x and prints the results.

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.
\end{verbatim}
\endscroll
\end{page}
```

Returns the value of the cosine integral

— nags.ht —

```
\begin{page}{manpageXXs13acf}{NAG Documentation: s13acf}
\beginscroll
\begin{verbatim}



     S13ACF(3NAG)      Foundation Library (12/10/92)      S13ACF(3NAG)



          S13 -- Approximations of Special Functions                S13ACF
                  S13ACF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          S13ACF returns the value of the cosine integral
```

$$Ci(x)=\gamma+\ln x+\int_0^x \frac{\cos u-1}{u}\,du, \quad x>0$$

via the routine name, where (gamma) denotes Euler's constant.

2. Specification

```
DOUBLE PRECISION FUNCTION S13ACF (X, IFAIL)
INTEGER          IFAIL
DOUBLE PRECISION X
```

3. Description

The routine calculates an approximate value for Ci(x).

For $0<x\le16$ it is based on the Chebyshev expansion

$$Ci(x)=\ln x+{\sum_{r=0}}' a_r T_r(t), \quad t=2\left(\frac{x}{16}\right)^2-1.$$

For $16<x<x_{hi}$ where the value of x_{hi} is given in the Users' Note for your implementation,

$$Ci(x)=\frac{f(x)\sin x}{x} \quad \frac{g(x)\cos x}{x^2}$$

where $f(x)={\sum_{r=0}}' f_r T_r(t)$ and $g(x)={\sum_{r=0}}' g_r T_r(t)$, $t=2\left(\frac{16}{x}\right)^2-1$.

For $x\ge x_{hi}$, $Ci(x)=0$ to within the accuracy possible (see Section 7).

4. References

[1] Abramowitz M and Stegun I A (1968) Handbook of Mathematical Functions. Dover Publications.

5. Parameters

1: X -- DOUBLE PRECISION Input
On entry: the argument x of the function. Constraint: X > 0.
0.

2: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1
The routine has been called with an argument less than or equal to zero for which the function is not defined. The result returned is zero.

7. Accuracy

If E and (epsilon) are the absolute and relative errors in the result and (delta) is the relative error in the argument then in principle these are related by

$$|E| \simeq |(\delta)\cos x| \quad \text{and} \quad |(\epsilon)| \simeq \left| \frac{(\delta)\cos x}{Ci(x)} \right|.$$

That is accuracy will be limited by machine precision near the origin and near the zeros of cosx, but near the zeros of Ci(x) only absolute accuracy can be maintained.

The behaviour of this amplification is shown in Figure 1.

Figure 1
Please see figure in printed Reference Manual

For large values of x, $Ci(x) \sim \frac{\sin x}{x}$ therefore
(epsilon)~(delta)xcotx and since (delta) is limited by the finite precision of the machine it becomes impossible to return results which have any relative accuracy. That is, when x>=1/(delta) we have that |Ci(x)|<=1/x~E and hence is not significantly different from zero.

```
        Hence x   is chosen such that for values of x>=x  , Ci(x) in
               hi                                      hi
        principle would have values less than the machine precision and
        so is essentially zero.

        8. Further Comments

        For details of the time taken by the routine see the Users' Note
        for your implementation.

        9. Example

        The example program reads values of the argument x from a file,
        evaluates the function at each value of x and prints the results.

        The example program is not reproduced here. The source code for
        all example programs is distributed with the NAG Foundation
        Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

———-

Returns the value of the sine integral

— nags.ht —

```
\begin{page}{manpageXXs13adf}{NAG Documentation: s13adf}
\beginscroll
\begin{verbatim}



     S13ADF(3NAG)      Foundation Library (12/10/92)      S13ADF(3NAG)



        S13 -- Approximations of Special Functions                 S13ADF
                S13ADF -- NAG Foundation Library Routine Document

        Note: Before using this routine, please read the Users' Note for
        your implementation to check implementation-dependent details.
        The symbol (*) after a NAG routine name denotes a routine that is
        not included in the Foundation Library.
```

1. Purpose

S13ADF returns the value of the sine integral

$$Si(x)=\int_0^x \frac{\sin u}{u}\,du,$$

via the routine name.

2. Specification

```
       DOUBLE PRECISION FUNCTION S13ADF (X, IFAIL)
       INTEGER          IFAIL
       DOUBLE PRECISION X
```

3. Description

The routine calculates an approximate value for Si(x).

For |x|<=16.0 it is based on the Chebyshev expansion

$$Si(x)=x\sum_{r=0}{}' a_r T_r(t)\;,\; t=2\left(\frac{x}{16}\right)^2-1.$$

For $16<|x|<x_{hi}$, where x_{hi} is an implementation dependent number,

$$Si(x)=\mathrm{sign}(x)\left\{\frac{\pi}{2}-\frac{f(x)\cos x}{x}-\frac{g(x)\sin x}{x^2}\right\}$$

where $f(x)=\sum_{r=0}{}' f_r T_r(t)$ and $g(x)=\sum_{r=0}{}' g_r T_r(t)$, $t=2\left(\frac{16}{x}\right)^2-1$.

For $|x|>=x_{hi}$, $Si(x)=\frac{1}{2}\pi\,\mathrm{sign}\,x$ to within machine precision.

4. References

[1] Abramowitz M and Stegun I A (1968) Handbook of Mathematical Functions. Dover Publications.

```
5. Parameters

 1:  X -- DOUBLE PRECISION                                  Input
     On entry: the argument x of the function.

 2:  IFAIL -- INTEGER                                Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).

6. Error Indicators and Warnings

There are no failure exits from this routine. The parameter IFAIL
has been included for consistency with other routines in this
chapter.

7. Accuracy

If (delta) and (epsilon) are the relative errors in the argument
and result, respectively, then in principle

                                | (delta)sinx|
                    |(epsilon)|~=| -----------|.
                                |    Si(x)   |

The equality may hold if (delta) is greater than the machine
precision ((delta) due to data errors etc) but if (delta) is
simply due to round-off in the machine representation, then since
the factor relating (delta) to (epsilon) is always less than one,
the accuracy will be limited by machine precision.

8. Further Comments

For details of the time taken by the routine see the Users' Note
for your implementation.

9. Example

The example program reads values of the argument x from a file,
evaluates the function at each value of x and prints the results.

The example program is not reproduced here. The source code for
all example programs is distributed with the NAG Foundation
Library software and should be available on-line.
```

\end{verbatim}

```
\endscroll
\end{page}
```

———-

Returns the value of the Gamma function

— nags.ht —

```
\begin{page}{manpageXXs14aaf}{NAG Documentation: s14aaf}
\beginscroll
\begin{verbatim}


     S14AAF(3NAG)      Foundation Library (12/10/92)      S14AAF(3NAG)



          S14 -- Approximations of Special Functions                S14AAF
                  S14AAF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          S14AAF returns the value of the Gamma function (Gamma)(x), via
          the routine name.

          2. Specification

                 DOUBLE PRECISION FUNCTION S14AAF (X, IFAIL)
                 INTEGER          IFAIL
                 DOUBLE PRECISION X

          3. Description

          This routine evaluates an approximation to the Gamma function
          (Gamma)(x). The routine is based on the Chebyshev expansion:

                              --'
                 (Gamma)(1+u)= >    a T (t) , where 0<=u<1 , t=2u-1,
                              --    r r
                             r=0
```

and uses the property (Gamma)(1+x)=x(Gamma)(x). If x=N+1+u where N is integral and 0<=u<1 then it follows that:

```
     for N>0  (Gamma)(x)=(x-1)(x-2)...(x-N)(Gamma)(1+u),

     for N=0  (Gamma)(x)=(Gamma)(1+u),

                                (Gamma)(1+u)
     for N<0  (Gamma)(x)= ---------------------.
                            x(x+1)(x+2)...(x-N-1)
```

There are four possible failures for this routine:

(i) if x is too large, there is a danger of overflow since (Gamma)(x) could become too large to be represented in the machine;

(ii) if x is too large and negative, there is a danger of underflow;

(iii) if x is equal to a negative integer, (Gamma)(x) would overflow since it has poles at such points;

(iv) if x is too near zero, there is again the danger of

```
                                                    1
     overflow on some machines. For small x, (Gamma)(x)~= -, and
                                                    x
```

on some machines there exists a range of non-zero but small values of x for which 1/x is larger than the greatest representable value.

4. References

[1] Abramowitz M and Stegun I A (1968) Handbook of Mathematical Functions. Dover Publications.

5. Parameters

1: X -- DOUBLE PRECISION Input
On entry: the argument x of the function. Constraint: X must not be a negative integer.

2: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

```
6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1
     The argument is too large. On soft failure the routine
     returns the approximate value of (Gamma)(x) at the nearest
     valid argument.

IFAIL= 2
     The argument is too large and negative. On soft failure the
     routine returns zero.

IFAIL= 3
     The argument is too close to zero. On soft failure the
     routine returns the approximate value of (Gamma)(x) at the
     nearest valid argument.

IFAIL= 4
     The argument is a negative integer, at which value
     (Gamma)(x) is infinite. On soft failure the routine returns
     a large positive value.

7. Accuracy

Let (delta) and (epsilon) be the relative errors in the argument
and the result respectively. If (delta) is somewhat larger than
the machine precision (i.e., is due to data errors etc), then
(epsilon) and (delta) are approximately related by:

                  (epsilon)~=|x(Psi)(x)|(delta)

(provided (epsilon) is also greater than the representation
                                               (Gamma)'(x)
error). Here (Psi)(x) is the digamma function  -----------.
                                               (Gamma)(x)
Figure 1 shows the behaviour of the error amplification factor
|x(Psi)(x)|.


                           Figure 1
         Please see figure in printed Reference Manual

If (delta) is of the same order as machine precision, then
rounding errors could make (epsilon) slightly larger than the
above relation predicts.

There is clearly a severe, but unavoidable, loss of accuracy for
arguments close to the poles of (Gamma)(x) at negative integers.
```

```
          However relative accuracy is preserved near the pole at x=0 right
          up to the point of failure arising from the danger of overflow.

          Also accuracy will necessarily be lost as x becomes large since
          in this region

                               (epsilon)~=(delta)xlnx.

          However since (Gamma)(x) increases rapidly with x, the routine
          must fail due to the danger of overflow before this loss of
          accuracy is too great. (e.g. for x=20, the amplification factor
          ~= 60.)

          8. Further Comments

          For details of the time taken by the routine see the Users' Note
          for your implementation.

          9. Example

          The example program reads values of the argument x from a file,
          evaluates the function at each value of x and prints the results.

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

———-

Returns a value for the logarithm of the Gamma function

— nags.ht —

```
\begin{page}{manpageXXs14abf}{NAG Documentation: s14abf}
\beginscroll
\begin{verbatim}



     S14ABF(3NAG)      Foundation Library (12/10/92)      S14ABF(3NAG)
```

S14 -- Approximations of Special Functions S14ABF

S14ABF -- NAG Foundation Library Routine Document

Note: Before using this routine, please read the Users' Note for your implementation to check implementation-dependent details. The symbol (*) after a NAG routine name denotes a routine that is not included in the Foundation Library.

1. Purpose

S14ABF returns a value for the logarithm of the Gamma function, $\ln \Gamma(x)$, via the routine name.

2. Specification

```
       DOUBLE PRECISION FUNCTION S14ABF (X, IFAIL)
       INTEGER          IFAIL
       DOUBLE PRECISION X
```

3. Description

This routine evaluates an approximation to $\ln \Gamma(x)$. It is based on two Chebyshev expansions.

For $0<x\le x_{small}$, $\ln \Gamma(x)=-\ln x$ to within machine accuracy.

For $x_{small}<x\le 15.0$, the recursive relation $\Gamma(1+x)=x\Gamma(x)$ is used to reduce the calculation to one involving $\Gamma(1+u)$, $0\le u<1$ which is evaluated as:

$$\Gamma(1+u)=\sum_{r=0}' a_r T_r(t), \quad t=2u-1.$$

Once $\Gamma(x)$ has been calculated, the required result is produced by taking the logarithm.

For $15.0<x\le x_{big}$,

$$\ln \Gamma(x)=\left(x-\frac{1}{2}\right)\ln x-x+\frac{1}{2}\ln 2\pi+y(x)/x$$

where $y(x)=\sum' b_r T_r(t)$, $t=2\left(\frac{15}{x}\right)^2-1$.

$r=0$

For $x_{big}<x<=x_{vbig}$ the term $y(x)/x$ is negligible and so its calculation is omitted.

For $x>x_{vbig}$ there is a danger of setting overflow so the routine must fail.

For x<=0.0 the function is not defined and the routine fails.

Note: x_{small} is calculated so that if $x<x_{small}$, (Gamma)(x)=1/x to within machine accuracy. x_{big} is calculated so that if $x>x_{big}$,

$$\ln(\Gamma)(x)=(x-\frac{1}{2})\ln x-x+\frac{1}{2}\ln 2(\pi)$$

to within machine accuracy. x_{vbig} is calculated so that $\ln(\Gamma)(x_{vbig})$ is close to the value returned by X02ALF(*).

4. References

[1] Abramowitz M and Stegun I A (1968) Handbook of Mathematical Functions. Dover Publications.

5. Parameters

1: X -- DOUBLE PRECISION Input
On entry: the argument x of the function. Constraint: X > 0. 0.

2: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1

X <= 0.0, the function is undefined. On soft failure, the routine returns zero.

IFAIL= 2
X is too large, the function would overflow. On soft failure, the routine returns the value of the function at the largest permissible argument.

7. Accuracy

Let (delta) and (epsilon) be the relative errors in the argument and result respectively, and E be the absolute error in the result.

If (delta) is somewhat larger than the relative machine precision, then

```
                                       |  x*(Psi)(x) |
  E~=|x*(Psi)(x)|(delta) and (epsilon)~=| ------------|(delta)
                                       | ln(Gamma)(x)|

                                       (Gamma)'(x)
where (Psi)(x) is the digamma function  -----------. Figure 1 and
                                       (Gamma)(x)
Figure 2 show the behaviour of these error amplification factors.
```

Figure 1
Please see figure in printed Reference Manual

Figure 2
Please see figure in printed Reference Manual

These show that relative error can be controlled, since except near x=1 or 2 relative error is attenuated by the function or at least is not greatly amplified.

```
                        (    1 )
For large x, (epsilon)~=(1+ ---)(delta) and for small x,
                        (   lnx)
             1
(epsilon)~= ---(delta).
            lnx
```

The function ln(Gamma)(x) has zeros at x=1 and 2 and hence relative accuracy is not maintainable near those points. However absolute accuracy can still be provided near those zeros as is shown above.

If however, (delta) is of the order of the machine precision,

```
        then rounding errors in the routine's internal arithmetic may
        result in errors which are slightly larger than those predicted
        by the equalities. It should be noted that even in areas where
        strong attenuation of errors is predicted the relative precision
        is bounded by the effective machine precision.

        8. Further Comments

        For details of the time taken by the routine see the Users' Note
        for your implementation.

        9. Example

        The example program reads values of the argument x from a file,
        evaluates the function at each value of x and prints the results.

        The example program is not reproduced here. The source code for
        all example programs is distributed with the NAG Foundation
        Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

———-

Incomplete gamma functions P(a,x) and Q(a,x)

— nags.ht —

```
\begin{page}{manpageXXs14baf}{NAG Documentation: s14baf}
\beginscroll
\begin{verbatim}



     S14BAF(3NAG)      Foundation Library (12/10/92)      S14BAF(3NAG)



        S14 -- Approximations of Special Functions                S14BAF
                S14BAF -- NAG Foundation Library Routine Document

        Note: Before using this routine, please read the Users' Note for
        your implementation to check implementation-dependent details.
        The symbol (*) after a NAG routine name denotes a routine that is
        not included in the Foundation Library.
```

1. Purpose

S14BAF computes values for the incomplete gamma functions P(a,x) and Q(a,x).

2. Specification

```
SUBROUTINE S14BAF (A, X, TOL, P, Q, IFAIL)
INTEGER          IFAIL
DOUBLE PRECISION A, X, TOL, P, Q
```

3. Description

This subroutine evaluates the incomplete gamma functions in the normalised form

```
                          x
                1         / a-1 -t
      P(a,x)= ---------- |t   e  dt,
              (Gamma)(a) /
                          0

                        infty
                1         /     a-1 -t
    Q(a,x)= ---------- |    t   e  dt,
            (Gamma)(a) /
                          x
```

with x>=0 and a>0, to a user-specified accuracy. With this normalisation, P(a,x)+Q(a,x)=1.

Several methods are used to evaluate the functions depending on the arguments a and x, the methods including Taylor expansion for P(a,x), Legendre's continued fraction for Q(a,x), and power series for Q(a,x). When both a and x are large, and a~=x, the uniform asymptotic expansion of Temme [3] is employed for greater efficiency - specifically, this expansion is used when a>=20 and 0.7a<=x<=1.4a.

Once either of P or Q is computed, the other is obtained by subtraction from 1. In order to avoid loss of relative precision in this subtraction, the smaller of P and Q is computed first.

This routine is derived from subroutine GAM in Gautschi [2].

4. References

[1]   Gautschi W (1979) A Computational Procedure for Incomplete
      Gamma Functions. ACM Trans. Math. Softw. 5 466--481.

[2] Gautschi W (1979) Algorithm 542: Incomplete Gamma Functions. ACM Trans. Math. Softw. 5 482--489.

[3] Temme N M (1987) On the Computation of the Incomplete Gamma Functions for Large Values of the Parameters. Algorithms for Approximation. (ed J C Mason and M G Cox) Oxford University Press.

5. Parameters

1: A -- DOUBLE PRECISION Input
On entry: the argument a of the functions. Constraint: A > 0.0.

2: X -- DOUBLE PRECISION Input
On entry: the argument x of the functions. Constraint: X >= 0.0.

3: TOL -- DOUBLE PRECISION Input
On entry: the relative accuracy required by the user in the results. If S14BAF is entered with TOL greater than 1.0 or less than machine precision, then the value of machine precision is used instead.

4: P -- DOUBLE PRECISION Output

5: Q -- DOUBLE PRECISION Output
On exit: the values of the functions P(a,x) and Q(a,x) respectively.

6: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1
On entry A <= 0.0.

IFAIL= 2
On entry X < 0.0.

IFAIL= 3

```
          Convergence of the Taylor series or Legendre continued
          fraction fails within 600 iterations. This error is
          extremely unlikely to occur; if it does, contact NAG.

     7. Accuracy

     There are rare occasions when the relative accuracy attained is
     somewhat less than that specified by parameter TOL. However, the
     error should never exceed more than one or two decimal places.
     Note also that there is a limit of 18 decimal places on the
     achievable accuracy, because constants in the routine are given
     to this precision.

     8. Further Comments

     The time taken for a call of S14BAF depends on the precision
     requested through TOL, and also varies slightly with the input
     arguments a and x.

     9. Example

     The example program reads values of the argument a and x from a
     file, evaluates the function and prints the results.

     The example program is not reproduced here. The source code for
     all example programs is distributed with the NAG Foundation
     Library software and should be available on-line.
\end{verbatim}
\endscroll
\end{page}
```

Returns the value of the complementary error function

— nags.ht —

```
\begin{page}{manpageXXs15adf}{NAG Documentation: s15adf}
\beginscroll
\begin{verbatim}


     S15ADF(3NAG)      Foundation Library (12/10/92)      S15ADF(3NAG)
```

S15 -- Approximations of Special Functions S15ADF
S15ADF -- NAG Foundation Library Routine Document

Note: Before using this routine, please read the Users' Note for your implementation to check implementation-dependent details. The symbol (*) after a NAG routine name denotes a routine that is not included in the Foundation Library.

1. Purpose

S15ADF returns the value of the complementary error function, erfcx, via the routine name.

2. Specification

```
DOUBLE PRECISION FUNCTION S15ADF (X, IFAIL)
INTEGER          IFAIL
DOUBLE PRECISION X
```

3. Description

The routine calculates an approximate value for the complement of the error function

$$\mathrm{erfc}\, x = \frac{2}{\sqrt{\pi}} \int_x^{\infty} e^{-u^2}\, du = 1 - \mathrm{erf}\, x.$$

For $x>=0$, it is based on the Chebyshev expansion

$$\mathrm{erfc}\, x = e^{-x^2} y(x),$$

where $y(x) = {\sum_{r=0}}' a_r T_r(t)$ and $t=(x-3.75)/(x+3.75)$, $-1<=t<=+1$.

For $x>=x_{hi}$, where there is a danger of setting underflow, the result is returned as zero.

For $x<0$, $\mathrm{erfc}\, x = 2 - e^{-x^2} y(|x|)$.

For $x<x_{low}<0$, the result is returned as 2.0 which is correct to within machine precision. The values of x_{hi} and x_{low} are given in the Users' Note for your implementation.

4. References

[1] Abramowitz M and Stegun I A (1968) Handbook of Mathematical Functions. Dover Publications.

5. Parameters

1: X -- DOUBLE PRECISION Input
On entry: the argument x of the function.

2: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

There are no failure exits from this routine. The parameter IFAIL has been included for consistency with other routines in this chapter.

7. Accuracy

If (delta) and (epsilon) are relative errors in the argument and result, respectively, then in principle

$$|\epsilon| \simeq \left| \frac{2xe^{-x^2}}{\sqrt{\pi}\,\mathrm{erfc}\,x}\delta \right|.$$

That is, the relative error in the argument, x, is amplified by a factor $\frac{2xe^{-x^2}}{\sqrt{\pi}\,\mathrm{erfc}\,x}$ in the result.

```
        \/(pi)erfc x
```

The behaviour of this factor is shown in Figure 1.

Figure 1

Please see figure in printed Reference Manual

```
                                                            2x
It should be noted that near x=0 this factor behaves as  ------



                                                          \/(pi)
and hence the accuracy is largely determined by the machine
precision. Also for large negative x, where the factor is
      2
    -x
  xe
~ ------, accuracy is mainly limited by machine precision.


  \/(pi)
                                                      2
However, for large positive x, the factor becomes ~2x  and to an
extent relative accuracy is necessarily lost. The absolute
accuracy E is given by

                            2
                          -x
                       2xe
                   E~= ------(delta)


                       \/(pi)
```

so absolute accuracy is guaranteed for all x.

8. Further Comments

For details of the time taken by the routine see the Users' Note for your implementation.

9. Example

The example program reads values of the argument x from a file, evaluates the function at each value of x and prints the results.

The example program is not reproduced here. The source code for all example programs is distributed with the NAG Foundation

```
         Library software and should be available on-line.
\end{verbatim}
\endscroll
\end{page}
```

———-

Returns the value of the error function erfx

— nags.ht —

```
\begin{page}{manpageXXs15aef}{NAG Documentation: s15aef}
\beginscroll
\begin{verbatim}



     S15AEF(3NAG)      Foundation Library (12/10/92)      S15AEF(3NAG)



          S15 -- Approximations of Special Functions                S15AEF
                  S15AEF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          S15AEF returns the value of the error function erfx, via the
          routine name.

          2. Specification

                 DOUBLE PRECISION FUNCTION S15AEF (X, IFAIL)
                 INTEGER          IFAIL
                 DOUBLE PRECISION X

          3. Description

          Evaluates the error function,

                                 2
                             x -t
                        2    /
```

```
     erf x =  ------ |e   dt.
                     /
              \/(pi) 0
```

For $|x|\le 2$,

$$\operatorname{erf} x = x \sum_{r=0}' a_r T_r(t), \text{ where } t = \frac{1}{2}x^2 - 1.$$

For $2 < |x| < x_{hi}$,

$$\operatorname{erf} x = \operatorname{sign} x \left\{ 1 - \frac{e^{-x^2}}{|x|\sqrt{\pi}} \sum_{r=0}' b_r T_r(t) \right\}, \text{ where } t = \frac{x-7}{x+3}.$$

For $|x| \ge x_{hi}$,

$$\operatorname{erf} x = \operatorname{sign} x.$$

x_{hi} is the value above which $\operatorname{erf} x = \pm 1$ within machine precision. Its value is given in the Users' Note for your implementation.

4. References

[1] Abramowitz M and Stegun I A (1968) Handbook of Mathematical Functions. Dover Publications.

5. Parameters

1: X -- DOUBLE PRECISION — Input
On entry: the argument x of the function.

2: IFAIL -- INTEGER — Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

There are no failure exits from this routine. The parameter IFAIL has been included for consistency with other routines in this chapter.

```
         7. Accuracy

         On a machine with approximately 11 significant figures the
         routine agrees with available tables to 10 figures and
         consistency checking with S15ADF of the relation

                                erf x+erfc x=1.0

         shows errors in at worst the 11th figure.

         8. Further Comments

         None.

         9. Example

         The example program reads values of the argument x from a file,
         evaluates the function at each value of x and prints the results.

         The example program is not reproduced here. The source code for
         all example programs is distributed with the NAG Foundation
         Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Returns the value of the Bessel Function $Y_0(x)$

— **nags.ht** —

```
\begin{page}{manpageXXs17acf}{NAG Documentation: s17acf}
\beginscroll
\begin{verbatim}



     S17ACF(3NAG)      Foundation Library (12/10/92)      S17ACF(3NAG)



         S17 -- Approximations of Special Functions                 S17ACF
                S17ACF -- NAG Foundation Library Routine Document
```

Note: Before using this routine, please read the Users' Note for your implementation to check implementation-dependent details. The symbol (*) after a NAG routine name denotes a routine that is not included in the Foundation Library.

1. Purpose

S17ACF returns the value of the Bessel Function $Y_0(x)$, via the routine name.

2. Specification

```
DOUBLE PRECISION FUNCTION S17ACF (X, IFAIL)
INTEGER          IFAIL
DOUBLE PRECISION X
```

3. Description

This routine evaluates an approximation to the Bessel Function of the second kind $Y_0(x)$.

Note: $Y_0(x)$ is undefined for $x<=0$ and the routine will fail for such arguments.

The routine is based on four Chebyshev expansions:

For $0<x<=8$,

$$Y_0(x)=\frac{2}{\pi}\ln x \sum_{r=0}{}' a_r T_r(t)+\sum_{r=0}{}' b_r T_r(t), \text{ with } t=2\left(\frac{x}{8}\right)^2-1,$$

For $x>8$,

$$Y_0(x)=\sqrt{\frac{2}{\pi x}}\left\{P_0(x)\sin\left(x-\frac{\pi}{4}\right)+Q_0(x)\cos\left(x-\frac{\pi}{4}\right)\right\}$$

where $P_0(x)=\sum_{r=0}{}' c_r T_r(t)$,

$$8\sum{}' \qquad \left(8\right)^2$$

```
and Q (x)= - >   d T (t), with t=2( -) -1.
     0     x --   r r            ( x)
             r=0
```

For x near zero, $Y_0(x) \simeq \frac{2}{\pi}\left(\ln\left(\frac{x}{2}\right)+\gamma\right)$, where γ denotes Euler's constant. This approximation is used when x is sufficiently small for the result to be correct to machine precision.

For very large x, it becomes impossible to provide results with any reasonable accuracy (see Section 7), hence the routine fails. Such arguments contain insufficient information to determine the

phase of oscillation of $Y_0(x)$; only the amplitude, $\sqrt{\frac{2}{x}}$, can be determined and this is returned on soft failure. The range for which this occurs is roughly related to the machine precision: the routine will fail if x>~1/machine precision (see the Users' Note for your implementation for details).

4. References

[1] Abramowitz M and Stegun I A (1968) Handbook of Mathematical Functions. Dover Publications.

[2] Clenshaw C W (1962) Mathematical Tables. Chebyshev Series for Mathematical Functions. HMSO.

5. Parameters

1: X -- DOUBLE PRECISION Input
On entry: the argument x of the function. Constraint: X > 0.
0.

2: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1

X is too large. On soft failure the routine returns the amplitude of the Y_0 oscillation, $\sqrt{2/(\pi)x}$.

IFAIL= 2

X <= 0.0, Y_0 is undefined. On soft failure the routine returns zero.

7. Accuracy

Let δ be the relative error in the argument and E be the absolute error in the result. (Since $Y_0(x)$ oscillates about zero, absolute error and not relative error is significant, except for very small x.)

If δ is somewhat larger than the machine representation error (e.g. if δ is due to data errors etc), then E and δ are approximately related by

$$E \simeq |xY_1(x)|\delta$$

(provided E is also within machine bounds). Figure 1 displays the behaviour of the amplification factor $|xY_1(x)|$.

Figure 1
Please see figure in printed Reference Manual

However, if δ is of the same order as the machine representation errors, then rounding errors could make E slightly larger than the above relation predicts.

For very small x, the errors are essentially independent of δ and the routine should provide relative accuracy bounded by the machine precision.

For very large x, the above relation ceases to apply. In this region, $Y_0(x) \simeq \sqrt{\frac{2}{\pi x}}\sin\left(x-\frac{\pi}{4}\right)$. The amplitude $\sqrt{\frac{2}{\pi x}}$ can be calculated with reasonable accuracy for all x, but $\sin\left(x-\frac{\pi}{4}\right)$ cannot. If $x-\frac{\pi}{4}$ is written as $2N\pi+\theta$

```
   (    4  )                 4
                                                    (   (pi))
where N is an integer and 0<=(theta)<2(pi), then sin(x- ----) is
                                                    (    4  )
                                        -1
determined by (theta) only. If x>~(delta)  , (theta) cannot be
determined with any accuracy at all. Thus if x is greater than,
or of the order of the inverse of machine precision, it is
impossible to calculate the phase of Y (x) and the routine must
                                      0
fail.

8. Further Comments

For details of the time taken by the routine see the Users' Note
for your implementation.

9. Example

The example program reads values of the argument x from a file,
evaluates the function at each value of x and prints the results.

The example program is not reproduced here. The source code for
all example programs is distributed with the NAG Foundation
Library software and should be available on-line.
```

```
\end{verbatim}
\endscroll
\end{page}
```

Returns the value of the Bessel Function $Y_1(x)$

— **nags.ht** —

```
\begin{page}{manpageXXs17adf}{NAG Documentation: s17adf}
\beginscroll
\begin{verbatim}



     S17ADF(3NAG)      Foundation Library (12/10/92)      S17ADF(3NAG)



          S17 -- Approximations of Special Functions                S17ADF
```

S17ADF -- NAG Foundation Library Routine Document

Note: Before using this routine, please read the Users' Note for your implementation to check implementation-dependent details. The symbol (*) after a NAG routine name denotes a routine that is not included in the Foundation Library.

1. Purpose

S17ADF returns the value of the Bessel Function $Y_1(x)$, via the routine name.

2. Specification

```
DOUBLE PRECISION FUNCTION S17ADF (X, IFAIL)
INTEGER          IFAIL
DOUBLE PRECISION X
```

3. Description

This routine evaluates an approximation to the Bessel Function of the second kind $Y_1(x)$.

Note: $Y_1(x)$ is undefined for $x<=0$ and the routine will fail for such arguments.

The routine is based on four Chebyshev expansions:

For $0<x<=8$,

$$Y_1(x)=\frac{2}{\pi}\ln x\,\frac{x}{8}\sum_{r=0}{}' a_r T_r(t)-\frac{2}{\pi x}+\frac{x}{8}\sum_{r=0}{}' b_r T_r(t)\text{ , with } t=2\left(\frac{x}{8}\right)^2-1;$$

For $x>8$,

$$Y_1(x)=\sqrt{\frac{2}{\pi x}}\left\{P_1(x)\sin\left(x-3\frac{\pi}{4}\right)+Q_1(x)\cos\left(x-3\frac{\pi}{4}\right)\right\}$$

$$\text{where } P_1(x)=\sum_{r=0}' c_r T_r(t),$$

$$\text{and } Q_1(x)=\frac{8}{x}\sum_{r=0}' d_r T_r(t), \text{ with } t=2\left(\frac{8}{x}\right)^2-1.$$

For x near zero, $Y_1(x) \simeq -\frac{2}{(pi)x}$. This approximation is used when x is sufficiently small for the result to be correct to machine precision. For extremely small x, there is a danger of overflow in calculating $-\frac{2}{(pi)x}$ and for such arguments the routine will fail.

For very large x, it becomes impossible to provide results with any reasonable accuracy (see Section 7), hence the routine fails. Such arguments contain insufficient information to determine the

phase of oscillation of $Y_1(x)$, only the amplitude, $\sqrt{\frac{2}{(pi)x}}$, can be determined and this is returned on soft failure. The range for which this occurs is roughly related to machine precision; the routine will fail if x>~1/machine precision (see the Users' Note for your implementation for details).

4. References

[1] Abramowitz M and Stegun I A (1968) Handbook of Mathematical Functions. Dover Publications.

[2] Clenshaw C W (1962) Mathematical Tables. Chebyshev Series for Mathematical Functions. HMSO.

5. Parameters

1: X -- DOUBLE PRECISION Input
On entry: the argument x of the function. Constraint: X > 0. 0.

2: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential

Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1

X is too large. On soft failure the routine returns the amplitude of the Y_1 oscillation, $\sqrt{\frac{2}{(pi)x}}$.

IFAIL= 2

X <= 0.0, Y_1 is undefined. On soft failure the routine returns zero.

IFAIL= 3

X is too close to zero, there is a danger of overflow. On soft failure, the routine returns the value of $Y_1(x)$ at the smallest valid argument.

7. Accuracy

Let (delta) be the relative error in the argument and E be the absolute error in the result. (Since $Y_1(x)$ oscillates about zero, absolute error and not relative error is significant, except for very small x.)

If (delta) is somewhat larger than the machine precision (e.g. if (delta) is due to data errors etc), then E and (delta) are approximately related by:

$$E \simeq |xY_0(x)-Y_1(x)|(delta)$$

(provided E is also within machine bounds). Figure 1 displays the behaviour of the amplification factor $|xY_0(x)-Y_1(x)|$.

Figure 1
Please see figure in printed Reference Manual

```
However, if (delta) is of the same order as machine precision,
then rounding errors could make E slightly larger than the above
relation predicts.

For very small x, absolute error becomes large, but the relative
error in the result is of the same order as (delta).

For very large x, the above relation ceases to apply. In this
                2(pi)   (   3(pi))                2(pi)
region, Y (x)~= -----sin(x- -----). The amplitude  ----- can be
         1        x     (     4  )                  x
                                                     (   3(pi))
calculated with reasonable accuracy for all x, but sin(x- -----)
                                                     (     4  )
              3(pi)
cannot. If x- ----- is written as 2N(pi)+(theta) where N is an
                4
                                      (   3(pi))
integer and 0<=(theta)<2(pi), then sin(x- -----) is determined by
                                      (     4  )
                          -1
(theta) only. If x>(delta)  , (theta) cannot be determined with
any accuracy at all. Thus if x is greater than, or of the order
of, the inverse of the machine precision, it is impossible to
calculate the phase of Y (x) and the routine must fail.
                        1

8. Further Comments

For details of the time taken by the routine see the Users' Note
for your implementation.

9. Example

The example program reads values of the argument x from a file,
evaluates the function at each value of x and prints the results.

The example program is not reproduced here. The source code for
all example programs is distributed with the NAG Foundation
Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Returns the value of the Bessel Function $J_0(x)$

— nags.ht —

```
\begin{page}{manpageXXs17aef}{NAG Documentation: s17aef}
\beginscroll
\begin{verbatim}



     S17AEF(3NAG)      Foundation Library (12/10/92)      S17AEF(3NAG)



          S17 -- Approximations of Special Functions                 S17AEF
                  S17AEF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          S17AEF returns the value of the Bessel Function J (x), via the
                                                           0
          routine name.

          2. Specification

                 DOUBLE PRECISION FUNCTION S17AEF (X, IFAIL)
                 INTEGER          IFAIL
                 DOUBLE PRECISION X

          3. Description

          This routine evaluates an approximation to the Bessel Function of
          the first kind J (x).
                          0

          Note: J (-x)=J (x), so the approximation need only consider x>=0.
                 0      0

          The routine is based on three Chebyshev expansions:

          For 0<x<=8,                           2
                     --'                    ( x)
                J (x)= >   a T (t), with t=2( -) -1.
                 0     --   r r             ( 8)
```

$r=0$

For $x>8$,

$$J_0(x)=\sqrt{\frac{2}{\pi x}}\left\{P_0(x)\cos\left(x-\frac{\pi}{4}\right)-Q_0(x)\sin\left(x-\frac{\pi}{4}\right)\right\}$$

where $P_0(x) = \sum_{r=0}' b_r T_r(t)$,

and $Q_0(x) = \frac{8}{x}\sum_{r=0}' c_r T_r(t)$, with $t=2\left(\frac{8}{x}\right)^2-1$.

For x near zero, $J_0(x)\simeq 1$. This approximation is used when x is sufficiently small for the result to be correct to machine precision.

For very large x, it becomes impossible to provide results with any reasonable accuracy (see Section 7), hence the routine fails. Such arguments contain insufficient information to determine the phase of oscillation of $J_0(x)$; only the amplitude, $\sqrt{\frac{2}{\pi|x|}}$, can be determined and this is returned on soft failure. The range for which this occurs is roughly related to the machine precision; the routine will fail if $|x|\gtrsim 1/\text{machine precision}$ (see the Users' Note for your implementation).

4. References

[1] Abramowitz M and Stegun I A (1968) Handbook of Mathematical Functions. Dover Publications.

[2] Clenshaw C W (1962) Mathematical Tables. Chebyshev Series for Mathematical Functions. HMSO.

5. Parameters

1: X -- DOUBLE PRECISION Input
On entry: the argument x of the function.

2: IFAIL -- INTEGER Input/Output

On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1

X is too large. On soft failure the routine returns the amplitude of the J_0 oscillation, $\sqrt{\frac{2}{\pi|x|}}$.

7. Accuracy

Let δ be the relative error in the argument and E be the absolute error in the result. (Since $J_0(x)$ oscillates about zero, absolute error and not relative error is significant.)

If δ is somewhat larger than the machine precision (e.g. if δ is due to data errors etc), then E and δ are approximately related by:

$$E \simeq |xJ_1(x)|\delta$$

(provided E is also within machine bounds). Figure 1 displays the behaviour of the amplification factor $|xJ_1(x)|$.

Figure 1
Please see figure in printed Reference Manual

However, if δ is of the same order as machine precision, then rounding errors could make E slightly larger than the above relation predicts.

For very large x, the above relation ceases to apply. In this

/   2   ((pi))

```
         region, J (x)~=  /  -------cos(x- ----). The amplitude
                  0     \/   (pi)|x|   (    4  )


            /    2
           /  ------- can be calculated with reasonable accuracy for all x
         \/   (pi)|x|
                (   (pi))              (pi)
         but cos(x- ----) cannot. If x- ---- is written as
                (    4  )               4
         2N(pi)+(theta) where N is an integer and 0 <= (theta) < 2(pi),
                 (   (pi))                                          -1
         then cos(x- ----) is determined by (theta) only. If x>~(delta)  ,
                 (    4  )
         (theta) cannot be determined with any accuracy at all. Thus if x
         is greater than, or of the order of, the inverse of the machine
         precision, it is impossible to calculate the phase of J (x) and
                                                                0
         the routine must fail.

         8. Further Comments

         For details of the time taken by the routine see the Users' Note
         for your implementation.

         9. Example

         The example program reads values of the argument x from a file,
         evaluates the function at each value of x and prints the results.

         The example program is not reproduced here. The source code for
         all example programs is distributed with the NAG Foundation
         Library software and should be available on-line.
\end{verbatim}
\endscroll
\end{page}
```

Returns the value of the Bessel Function $J_1(x)$

— nags.ht —

```
\begin{page}{manpageXXs17aff}{NAG Documentation: s17aff}
\beginscroll
\begin{verbatim}
```

S17 -- Approximations of Special Functions S17AFF

S17AFF -- NAG Foundation Library Routine Document

Note: Before using this routine, please read the Users' Note for your implementation to check implementation-dependent details. The symbol (*) after a NAG routine name denotes a routine that is not included in the Foundation Library.

1. Purpose

S17AFF returns the value of the Bessel Function $J_1(x)$, via the routine name.

2. Specification

```
DOUBLE PRECISION FUNCTION S17AFF (X, IFAIL)
INTEGER          IFAIL
DOUBLE PRECISION X
```

3. Description

This routine evaluates an approximation to the Bessel Function of the first kind $J_1(x)$.

Note: $J_1(-x)=-J_1(x)$, so the approximation need only consider $x>=0$

The routine is based on three Chebyshev expansions:

For $0<x<=8$,

$$J_1(x)=\frac{x}{8}{\sum_{r=0}}' a_r T_r(t), \text{ with } t=2\left(\frac{x}{8}\right)^2-1.$$

For $x>8$,

$$J_1(x)=\sqrt{\frac{2}{\pi x}}\left\{P_1(x)\cos\left(x-\frac{3\pi}{4}\right)-Q_1(x)\sin\left(x-\frac{3\pi}{4}\right)\right\}$$

$$\text{where} P_1(x)= \sideset{}{'}\sum_{r=0} b_r T_r(t),$$

$$\text{and } Q_1(x)= \frac{8}{x} \sideset{}{'}\sum_{r=0} c_r T_r(t), \text{ with } t=2\left(\frac{8}{x}\right)^2 -1.$$

For x near zero, $J_1(x)\simeq \frac{x}{2}$. This approximation is used when x is sufficiently small for the result to be correct to machine precision.

For very large x, it becomes impossible to provide results with any reasonable accuracy (see Section 7), hence the routine fails. Such arguments contain insufficient information to determine the

phase of oscillation of $J_1(x)$; only the amplitude, $\sqrt{\frac{2}{(pi)|x|}}$, can be determined and this is returned on soft failure. The range for which this occurs is roughly related to the machine precision; the routine will fail if $|x|>\sim 1/\text{machine precision}$ (see the Users' Note for your implementation for details).

4. References

[1] Abramowitz M and Stegun I A (1968) Handbook of Mathematical Functions. Dover Publications.

[2] Clenshaw C W (1962) Mathematical Tables. Chebyshev Series for Mathematical Functions. HMSO.

5. Parameters

1: X -- DOUBLE PRECISION Input
On entry: the argument x of the function.

2: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1

X is too large. On soft failure the routine returns the amplitude of the J_1 oscillation, $\sqrt{\frac{2}{\pi|x|}}$.

7. Accuracy

Let δ be the relative error in the argument and E be the absolute error in the result. (Since $J_1(x)$ oscillates about zero, absolute error and not relative error is significant.)

If δ is somewhat larger than machine precision (e.g. if δ is due to data errors etc), then E and δ are approximately related by:

$$E \simeq |xJ_0(x)-J_1(x)|\delta$$

(provided E is also within machine bounds). Figure 1 displays the behaviour of the amplification factor $|xJ_0(x)-J_1(x)|$.

Figure 1
Please see figure in printed Reference Manual

However, if δ is of the same order as machine precision, then rounding errors could make E slightly larger than the above relation predicts.

For very large x, the above relation ceases to apply. In this region, $J_1(x) \simeq \sqrt{\frac{2}{\pi|x|}}\cos\left(x-\frac{3\pi}{4}\right)$. The amplitude

$\sqrt{\frac{2}{\pi|x|}}$ can be calculated with reasonable accuracy for all x

(3(pi)) 3(pi)

```
          but cos(x- -----) cannot. If x- ----- is written as
                 (     4  )                4
          2N(pi)+(theta) where N is an integer and 0<=(theta)<2(pi), then
             (   3(pi))                                          -1
          cos(x- -----) is determined by (theta) only. If x>~(delta)  ,
             (     4  )
          (theta) cannot be determined with any accuracy at all. Thus if x
          is greater than, or of the order of, machine precision, it is
          impossible to calculate the phase of J (x) and the routine must
                                                 1
          fail.

          8. Further Comments

          For details of the time taken by the routine see the Users' Note
          for your implementation.

          9. Example

          The example program reads values of the argument x from a file,
          evaluates the function at each value of x and prints the results.

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Returns a value for the Airy function, $Ai(x)$

— nags.ht —

```
\begin{page}{manpageXXs17agf}{NAG Documentation: s17agf}
\beginscroll
\begin{verbatim}



     S17AGF(3NAG)      Foundation Library (12/10/92)      S17AGF(3NAG)



          S17 -- Approximations of Special Functions                S17AGF
```

S17AGF -- NAG Foundation Library Routine Document

Note: Before using this routine, please read the Users' Note for your implementation to check implementation-dependent details. The symbol (*) after a NAG routine name denotes a routine that is not included in the Foundation Library.

1. Purpose

S17AGF returns a value for the Airy function, Ai(x), via the routine name.

2. Specification

```
       DOUBLE PRECISION FUNCTION S17AGF (X, IFAIL)
       INTEGER          IFAIL
       DOUBLE PRECISION X
```

3. Description

This routine evaluates an approximation to the Airy function, Ai(x). It is based on a number of Chebyshev expansions:

For $x<-5$,

$$Ai(x)=\frac{a(t)\sin z-b(t)\cos z}{(-x)^{1/4}}$$

where $z=\frac{\pi}{4}+\frac{2}{3}\sqrt{-x^3}$, and a(t) and b(t) are expansions in the variable $t=-2\left(\frac{5}{x}\right)^3-1$.

For $-5<=x<=0$,

$$Ai(x)=f(t)-xg(t),$$

where f and g are expansions in $t=-2\left(\frac{x}{5}\right)^3-1$.

For $0<x<4.5$,

$$Ai(x)=e^{-3x/2}y(t),$$

where y is an expansion in t=4x/9-1.

For 4.5<=x<9,

$$Ai(x)=e^{-5x/2}u(t),$$

where u is an expansion in t=4x/9-3.

For x>=9,

$$Ai(x)=\frac{e^{-z}v(t)}{x^{1/4}},$$

where $z=\frac{2}{3}\sqrt{x^3}$ and v is an expansion in $t=2\left(\frac{18}{z}\right)-1$.

For |x|< the machine precision, the result is set directly to Ai(0). This both saves time and guards against underflow in intermediate calculations.

For large negative arguments, it becomes impossible to calculate the phase of the oscillatory function with any precision and so the routine must fail. This occurs if $x<-\left(\frac{3}{2(\epsilon)}\right)^{2/3}$, where (epsilon) is the machine precision.

For large positive arguments, where Ai decays in an essentially exponential manner, there is a danger of underflow so the routine must fail.

4. References

[1] Abramowitz M and Stegun I A (1968) Handbook of Mathematical Functions. Dover Publications.

5. Parameters

1: X -- DOUBLE PRECISION Input

```
     On entry: the argument x of the function.

 2:  IFAIL -- INTEGER                                Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.
     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1
     X is too large and positive. On soft failure, the routine
     returns zero.

IFAIL= 2
     X is too large and negative. On soft failure, the routine
     returns zero.

7. Accuracy

For negative arguments the function is oscillatory and hence
absolute error is the appropriate measure. In the positive region
the function is essentially exponential-like and here relative
error is appropriate. The absolute error, E, and the relative
error, (epsilon), are related in principle to the relative error
in the argument, (delta), by

                                     | xAi'(x)|
       E~=|xAi'(x)|(delta), (epsilon)~=| -------|(delta).
                                     |  Ai(x) |

In practice, approximate equality is the best that can be
expected. When (delta), (epsilon) or E is of the order of the
machine precision, the errors in the result will be somewhat
larger.

For small x, errors are strongly damped by the function and hence
will be bounded by the machine precision.

For moderate negative x, the error behaviour is oscillatory but
the amplitude of the error grows like

                                      5/4
                         (    E   )  |x|
               amplitude ( -------)~ ------.
                         ( (delta))
                                    \/(pi)
```

```
                                                             2  /   3
          However the phase error will be growing roughly like  -\/ |x|
                                                             3
          and hence all accuracy will be lost for large negative arguments
          due to the impossibility of calculating sin and cos to any


                       2  /   3     1
          accuracy if  -\/ |x| > -------.
                       3         (delta)

          For large positive arguments, the relative error amplification is
          considerable:



                                    (epsilon)   / 3
                                    ---------~\/ x .
                                     (delta)

          This means a loss of roughly two decimal places accuracy for
          arguments in the region of 20. However very large arguments are
          not possible due to the danger of setting underflow and so the
          errors are limited in practice.

          8. Further Comments

          None.

          9. Example

          The example program reads values of the argument x from a file,
          evaluates the function at each value of x and prints the results.

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.
\end{verbatim}
\endscroll
\end{page}
```

Returns a value of the Airy function, $Bi(x)$

— nags.ht —

```
\begin{page}{manpageXXs17ahf}{NAG Documentation: s17ahf}
\beginscroll
\begin{verbatim}



     S17AHF(3NAG)      Foundation Library (12/10/92)      S17AHF(3NAG)



          S17 -- Approximations of Special Functions                 S17AHF
                  S17AHF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          S17AHF returns a value of the Airy function, Bi(x), via the
          routine name.

          2. Specification

                 DOUBLE PRECISION FUNCTION S17AHF (X, IFAIL)
                 INTEGER          IFAIL
                 DOUBLE PRECISION X

          3. Description

          This routine evaluates an approximation to the Airy function
          Bi(x). It is based on a number of Chebyshev expansions.

          For x<-5,

                                     a(t)cosz+b(t)sinz
                              Bi(x)= -----------------,
                                             1/4
                                          (-x)



                   (pi)  2  /  3
          where z= ----+ -\/ -x  and a(t) and b(t) are expansions in the
```

4 3

variable $t=-2\left(\frac{5}{x}\right)^3-1$.

For $-5<=x<=0$,

$$Bi(x)=\sqrt{3}(f(t)+xg(t)),$$

where f and g are expansions in $t=-2\left(\frac{x}{5}\right)^3-1$.

For $0<x<4.5$,

$$Bi(x)=e^{11x/8}y(t),$$

where y is an expansion in $t=4x/9-1$.

For $4.5<=x<=9$,

$$Bi(x)=e^{5x/2}v(t),$$

where v is an expansion in $t=4x/9-3$.

For $x>=9$,

$$Bi(x)=\frac{e^z u(t)}{x^{1/4}},$$

where $z=\frac{2}{3}\sqrt{x^3}$ and u is an expansion in $t=2\left(\frac{18}{z}\right)-1$.

For |x|< the machine precision, the result is set directly to Bi(0). This both saves time and avoids possible intermediate underflows.

For large negative arguments, it becomes impossible to calculate the phase of the oscillating function with any accuracy so the routine must fail. This occurs if $x<-\left(\frac{3}{2(\epsilon)}\right)^{2/3}$, where (epsilon) is the machine precision.

For large positive arguments, there is a danger of causing overflow since Bi grows in an essentially exponential manner, so the routine must fail.

4. References

[1] Abramowitz M and Stegun I A (1968) Handbook of Mathematical Functions. Dover Publications.

5. Parameters

1: X -- DOUBLE PRECISION Input
On entry: the argument x of the function.

2: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1
X is too large and positive. On soft failure, the routine returns zero.

IFAIL= 2
X is too large and negative. On soft failure, the routine returns zero.

7. Accuracy

For negative arguments the function is oscillatory and hence absolute error is the appropriate measure. In the positive region the function is essentially exponential-like and here relative error is appropriate. The absolute error, E, and the relative error, (epsilon), are related in principle to the relative error in the argument, (delta), by

$$| xBi'(x)|$$

```
E~=|xBi'(x)|(delta), (epsilon)~=| -------|(delta).
                                |  Bi(x) |
```

In practice, approximate equality is the best that can be
expected. When (delta), (epsilon) or E is of the order of the
machine precision, the errors in the result will be somewhat
larger.

For small x, errors are strongly damped and hence will be bounded
essentially by the machine precision.

For moderate to large negative x, the error behaviour is clearly
oscillatory but the amplitude of the error grows like amplitude

```
              5/4
(    E   )  |x|
( -------)~ ------.
( (delta))
            \/(pi)
```

```
                                                    2  /   3
However the phase error will be growing roughly as  -\/ |x|  and
                                                    3
```

hence all accuracy will be lost for large negative arguments.

This is due to the impossibility of calculating sin and cos to

```
                 2  /   3     1
any accuracy if  -\/ |x| > -------.
                 3         (delta)
```

For large positive arguments, the relative error amplification is
considerable:

```
(epsilon)   / 3
---------~\/ x .
 (delta)
```

This means a loss of roughly two decimal places accuracy for
arguments in the region of 20. However very large arguments are
not possible due to the danger of causing overflow and errors are
therefore limited in practice.

8. Further Comments

For details of the time taken by the routine see the Users' Note

```
          for your implementation.

          9. Example

          The example program reads values of the argument x from a file,
          evaluates the function at each value of x and prints the results.

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.
\end{verbatim}
\endscroll
\end{page}
```

Value of the derivative of the Airy function $Ai(x)$

— nags.ht —

```
\begin{page}{manpageXXs17ajf}{NAG Documentation: s17ajf}
\beginscroll
\begin{verbatim}



     S17AJF(3NAG)      Foundation Library (12/10/92)      S17AJF(3NAG)



          S17 -- Approximations of Special Functions                 S17AJF
                  S17AJF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          S17AJF returns a value of the derivative of the Airy function
          Ai(x), via the routine name.

          2. Specification

                 DOUBLE PRECISION FUNCTION S17AJF (X, IFAIL)
                 INTEGER          IFAIL
```

```
      DOUBLE PRECISION X
```

3. Description

This routine evaluates an approximation to the derivative of the Airy function Ai(x). It is based on a number of Chebyshev expansions.

For $x<-5$,

$$Ai'(x)=\sqrt[4]{-x}\left[a(t)\cos z+\frac{b(t)}{(\zeta)}\sin z\right],$$

where $z=\frac{\pi}{4}+(\zeta)$, $(\zeta)=\frac{2}{3}\sqrt{-x^3}$ and $a(t)$ and $b(t)$ are expansions in variable $t=-2\left(\frac{5}{x}\right)^3-1$.

For $-5\le x\le 0$,

$$Ai'(x)=x^2f(t)-g(t),$$

where f and g are expansions in $t=-2\left(\frac{x}{5}\right)^3-1$.

For $0<x<4.5$,

$$Ai'(x)=e^{-11x/8}y(t),$$

where $y(t)$ is an expansion in $t=4\left(\frac{x}{9}\right)-1$.

For $4.5\le x<9$,

$$Ai'(x)=e^{-5x/2}v(t),$$

where $v(t)$ is an expansion in $t=4\left(\frac{x}{9}\right)-3$.

For $x>=9$,

$$Ai'(x)=\sqrt[4]{x}e^{-z}u(t),$$

where $z=\frac{2}{3}\sqrt{x^3}$ and $u(t)$ is an expansion in $t=2\left(\frac{18}{z}\right)-1$.

For $|x|<$ the square of the machine precision, the result is set directly to $Ai'(0)$. This both saves time and avoids possible intermediate underflows.

For large negative arguments, it becomes impossible to calculate a result for the oscillating function with any accuracy and so the routine must fail. This occurs for $x<-\left(\frac{\sqrt{\pi}}{\epsilon}\right)^{4/7}$, where ϵ is the machine precision.

For large positive arguments, where Ai' decays in an essentially exponential manner, there is a danger of underflow so the routine must fail.

4. References

[1] Abramowitz M and Stegun I A (1968) Handbook of Mathematical Functions. Dover Publications.

5. Parameters

1: X -- DOUBLE PRECISION Input
On entry: the argument x of the function.

2: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1
X is too large and positive. On soft failure, the routine returns zero.

IFAIL= 2
X is too large and negative. On soft failure, the routine returns zero.

7. Accuracy

For negative arguments the function is oscillatory and hence absolute error is the appropriate measure. In the positive region the function is essentially exponential in character and here relative error is needed. The absolute error, E, and the relative error, (epsilon), are related in principle to the relative error in the argument, (delta), by

$$E \simeq |x^2 Ai(x)|\delta \qquad \epsilon \simeq \left|\frac{x^2 Ai(x)}{Ai'(x)}\right|\delta.$$

In practice, approximate equality is the best that can be expected. When (delta), (epsilon) or E is of the order of the machine precision, the errors in the result will be somewhat larger.

For small x, positive or negative, errors are strongly attenuated by the function and hence will be roughly bounded by the machine precision.

For moderate to large negative x, the error, like the function, is oscillatory; however the amplitude of the error grows like

$$\frac{|x|^{7/4}}{\sqrt{\pi}}.$$

Therefore it becomes impossible to calculate the function with

any accuracy if $|x|^{7/4} > \frac{\sqrt{\pi}}{\delta}$.

For large positive x, the relative error amplification is considerable:

```
                                   (epsilon)    / 3
                                   ---------~=\/ x .
                                    (delta)

          However, very large arguments are not possible due to the danger
          of underflow. Thus in practice error amplification is limited.

          8. Further Comments

          None.

          9. Example

          The example program reads values of the argument x from a file,
          evaluates the function at each value of x and prints the results.

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.
\end{verbatim}
\endscroll
\end{page}
```

Value for the derivative of the Airy function $Bi(x)$

— nags.ht —

```
\begin{page}{manpageXXs17akf}{NAG Documentation: s17akf}
\beginscroll
\begin{verbatim}



     S17AKF(3NAG)      Foundation Library (12/10/92)      S17AKF(3NAG)



          S17 -- Approximations of Special Functions                 S17AKF
                  S17AKF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
```

The symbol (*) after a NAG routine name denotes a routine that is not included in the Foundation Library.

1. Purpose

S17AKF returns a value for the derivative of the Airy function Bi(x), via the routine name.

2. Specification

```
DOUBLE PRECISION FUNCTION S17AKF (X, IFAIL)
INTEGER          IFAIL
DOUBLE PRECISION X
```

3. Description

This routine calculates an approximate value for the derivative of the Airy function Bi(x). It is based on a number of Chebyshev expansions.

For $x<-5$,

$$Bi'(x)=\sqrt[4]{-x}\left[-a(t)\sin z+\frac{b(t)}{\zeta}\cos z\right],$$

where $z=\frac{\pi}{4}+\zeta$, $\zeta=\frac{2}{3}\sqrt{-x^3}$ and $a(t)$ and $b(t)$ are expansions in the variable $t=-2\left(\frac{5}{x}\right)^3-1$.

For $-5\le x\le 0$,

$$Bi'(x)=\sqrt{3}(x^2f(t)+g(t)),$$

where f and g are expansions in $t=-2\left(\frac{x}{5}\right)^3-1$.

For $0<x<4.5$,

$$Bi'(x)=e^{3x/2}y(t),$$

where $y(t)$ is an expansion in $t=4x/9-1$.

For $4.5<=x<9$,

$$Bi'(x)=e^{21x/8}u(t),$$

where $u(t)$ is an expansion in $t=4x/9-3$.

For $x>=9$,

$$Bi'(x)=\sqrt[4]{x}e^{z}v(t),$$

where $z=\frac{2}{3}\sqrt{x^3}$ and $v(t)$ is an expansion in $t=2\left(\frac{18}{z}\right)-1$.

For $|x|<$ the square of the machine precision, the result is set directly to $Bi'(0)$. This saves time and avoids possible underflows in calculation.

For large negative arguments, it becomes impossible to calculate a result for the oscillating function with any accuracy so the routine must fail. This occurs for $x<-\left(\frac{\sqrt{\pi}}{\epsilon}\right)^{4/7}$, where ϵ is the machine precision.

For large positive arguments, where Bi' grows in an essentially exponential manner, there is a danger of overflow so the routine must fail.

4. References

[1] Abramowitz M and Stegun I A (1968) Handbook of Mathematical Functions. Dover Publications.

5. Parameters

1: X -- DOUBLE PRECISION — Input
On entry: the argument x of the function.

2: IFAIL -- INTEGER — Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential

Introduction) the recommended value is 0.
On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1
X is too large and positive. On soft failure the routine returns zero.

IFAIL= 2
X is too large and negative. On soft failure the routine returns zero.

7. Accuracy

For negative arguments the function is oscillatory and hence absolute error is appropriate. In the positive region the function has essentially exponential behaviour and hence relative error is needed. The absolute error, E, and the relative error (epsilon), are related in principle to the relative error in the argument (delta), by

```
                                     |  2     |
           2                         | x Bi(x)|
      E~=|x Bi(x)|(delta)    (epsilon)~=| -------|(delta).
                                     | Bi'(x) |
```

In practice, approximate equality is the best that can be expected. When (delta), (epsilon) or E is of the order of the machine precision, the errors in the result will be somewhat larger.

For small x, positive or negative, errors are strongly attenuated by the function and hence will effectively be bounded by the machine precision.

For moderate to large negative x, the error is, like the function, oscillatory. However, the amplitude of the absolute

```
                     7/4
                  |x|
error grows like  ------. Therefore it becomes impossible to


                  \/(pi)


                                                 7/4  \/(pi)
```

```
          calculate the function with any accuracy if |x|   > -------.
                                                              (delta)

          For large positive x, the relative error amplification is


                         (epsilon)   / 3
          considerable:  ---------~\/ x . However, very large arguments are
                          (delta)
          not possible due to the danger of overflow. Thus in practice the
          actual amplification that occurs is limited.

          8. Further Comments

          None.

          9. Example

          The example program reads values of the argument x from a file,
          evaluates the function at each value of x and prints the results.

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Values for the Bessel functions $Y_{\nu+n}(z)$

— nags.ht —

```
\begin{page}{manpageXXs17dcf}{NAG Documentation: s17dcf}
\beginscroll
\begin{verbatim}



     S17DCF(3NAG)      Foundation Library (12/10/92)      S17DCF(3NAG)



          S17 -- Approximations of Special Functions                 S17DCF
                  S17DCF -- NAG Foundation Library Routine Document
```

Note: Before using this routine, please read the Users' Note for your implementation to check implementation-dependent details. The symbol (*) after a NAG routine name denotes a routine that is not included in the Foundation Library.

1. Purpose

S17DCF returns a sequence of values for the Bessel functions $Y_{(nu)+n}(z)$ for complex z, non-negative (nu) and n=0,1,...,N-1, with an option for exponential scaling.

2. Specification

```
SUBROUTINE S17DCF (FNU, Z, N, SCALE, CY, NZ, CWRK, IFAIL)
INTEGER              N, NZ, IFAIL
DOUBLE PRECISION     FNU
COMPLEX(KIND(1.0D0)) Z, CY(N), CWRK(N)
CHARACTER*1          SCALE
```

3. Description

This subroutine evaluates a sequence of values for the Bessel function $Y_{(nu)}(z)$, where z is complex, -(pi) < arg z <= (pi), and (nu) is the real, non-negative order. The N-member sequence is generated for orders (nu), (nu)+1,...,(nu)+N-1. Optionally, the sequence is scaled by the factor $e^{-|Im z|}$.

Note: although the routine may not be called with (nu) less than zero, for negative orders the formula $Y_{-(nu)}(z)=Y_{(nu)}(z)cos((pi)(nu))+J_{(nu)}(z)sin((pi)(nu))$ may be used (for the Bessel function $J_{(nu)}(z)$, see S17DEF).

The routine is derived from the routine CBESY in Amos [2]. It is based on the relation $Y_{(nu)}(z)=\frac{H^{(1)}_{(nu)}(z)-H^{(2)}_{(nu)}(z)}{2i}$, where $H^{(1)}_{(nu)}(z)$ and $H^{(2)}_{(nu)}(z)$ are the Hankel functions of the first and second kinds respectively (see S17DLF).

When N is greater than 1, extra values of $Y_{(nu)}(z)$ are computed

(nu)
using recurrence relations.

For very large |z| or ((nu)+N-1), argument reduction will cause total loss of accuracy, and so no computation is performed. For slightly smaller |z| or ((nu)+N-1), the computation is performed but results are accurate to less than half of machine precision. If |z| is very small, near the machine underflow threshold, or ((nu)+N-1) is too large, there is a risk of overflow and so no computation is performed. In all the above cases, a warning is given by the routine.

4. References

[1] Abramowitz M and Stegun I A (1968) Handbook of Mathematical Functions. Dover Publications.

[2] Amos D E (1986) Algorithm 644: A Portable Package for Bessel Functions of a Complex Argument and Nonnegative Order. ACM Trans. Math. Softw. 12 265--273.

5. Parameters

1: FNU -- DOUBLE PRECISION Input
On entry: the order, (nu), of the first member of the sequence of functions. Constraint: FNU >= 0.0.

2: Z -- COMPLEX(KIND(1.0D0)) Input
On entry: the argument, z, of the functions. Constraint: Z /= (0.0, 0.0).

3: N -- INTEGER Input
On entry: the number, N, of members required in the sequence $Y_{(nu)}(z)$, $Y_{(nu)+1}(z)$,...,$Y_{(nu)+N-1}(z)$. Constraint: N >= 1.

4: SCALE -- CHARACTER*1 Input
On entry: the scaling option.

If SCALE = 'U', the results are returned unscaled.

If SCALE = 'S', the results are returned scaled by the factor $e^{-|Imz|}$. Constraint: SCALE = 'U' or 'S'.

5: CY(N) -- COMPLEX(KIND(1.0D)) array Output
On exit: the N required function values: CY(i) contains $Y_{(nu)+i-1}(z)$, for i=1,2,...,N.

```
 6:  NZ -- INTEGER                                        Output
     On exit: the number of components of CY that are set to zero
     due to underflow. The positions of such components in the
     array CY are arbitrary.

 7:  CWRK(N) -- COMPLEX(KIND(1.0D)) array              Workspace

 8:  IFAIL -- INTEGER                               Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

IFAIL= 1
     On entry FNU < 0.0,

     or        Z = (0.0, 0.0),

     or        N < 1,

     or        SCALE /= 'U' or 'S'.

IFAIL= 2
     No computation has been performed due to the likelihood of
     overflow, because ABS(Z) is less than a machine-dependent
     threshold value (given in the Users' Note for your
     implementation).

IFAIL= 3
     No computation has been performed due to the likelihood of
     overflow, because FNU + N - 1 is too large - how large
     depends on Z as well as the overflow threshold of the
     machine.

IFAIL= 4
     The computation has been performed, but the errors due to
     argument reduction in elementary functions make it likely
     that the results returned by S17DCF are accurate to less
     than half of machine precision. This error exit may occur if
     either ABS(Z) or FNU + N - 1 is greater than a machine-
     dependent threshold value (given in the Users' Note for
```

your implementation).

IFAIL= 5

No computation has been performed because the errors due to argument reduction in elementary functions mean that all precision in results returned by S17DCF would be lost. This error exit may occur if either ABS(Z) or FNU + N - 1 is greater than a machine-dependent threshold value (given in the Users' Note for your implementation).

IFAIL= 6

No results are returned because the algorithm termination condition has not been met. This may occur because the parameters supplied to S17DCF would have caused overflow or underflow.

7. Accuracy

All constants in subroutine S17DCF are given to approximately 18 digits of precision. Calling the number of digits of precision in the floating-point arithmetic being used t, then clearly the maximum number of correct digits in the results obtained is limited by p=min(t,18). Because of errors in argument reduction when computing elementary functions inside S17DCF, the actual number of correct digits is limited, in general, by p-s, where $s \sim max(1,|log_{10} |z||,|log_{10} (nu)|)$ represents the number of digits lost due to the argument reduction. Thus the larger the values of |z| and (nu), the less the precision in the result. If S17DCF is called with N>1, then computation of function values via recurrence may lead to some further small loss of accuracy.

If function values which should nominally be identical are computed by calls to S17DCF with different base values of (nu) and different N, the computed values may not agree exactly. Empirical tests with modest values of (nu) and z have shown that the discrepancy is limited to the least significant 3-4 digits of precision.

8. Further Comments

The time taken by the routine for a call of S17DCF is approximately proportional to the value of N, plus a constant. In general it is much cheaper to call S17DCF with N greater than 1, rather than to make N separate calls to S17DCF.

Paradoxically, for some values of z and (nu), it is cheaper to call S17DCF with a larger value of N than is required, and then discard the extra function values returned. However, it is not possible to state the precise circumstances in which this is

```
          likely to occur. It is due to the fact that the base value used
          to start recurrence may be calculated in different regions for
          different N, and the costs in each region may differ greatly.

          Note that if the function required is Y (x) or Y (x), i.e., (nu)
                                                 0        1
          = 0.0 or 1.0, where x is real and positive, and only a single
          unscaled function value is required, then it may be much cheaper
          to call S17ACF or S17ADF respectively.

          9. Example

          The example program prints a caption and then proceeds to read
          sets of data from the input data stream. The first datum is a
          value for the order FNU, the second is a complex value for the
          argument, Z, and the third is a value for the parameter SCALE.
          The program calls the routine with N = 2 to evaluate the function
          for orders FNU and FNU + 1, and it prints the results. The
          process is repeated until the end of the input data stream is
          encountered.

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.
\end{verbatim}
\endscroll
\end{page}
```

Values for the Bessel functions $J_{\nu+n}(z)$

— **nags.ht** —

```
\begin{page}{manpageXXs17def}{NAG Documentation: s17def}
\beginscroll
\begin{verbatim}



     S17DEF(3NAG)      Foundation Library (12/10/92)      S17DEF(3NAG)



          S17 -- Approximations of Special Functions                 S17DEF
                  S17DEF -- NAG Foundation Library Routine Document
```

Note: Before using this routine, please read the Users' Note for your implementation to check implementation-dependent details. The symbol (*) after a NAG routine name denotes a routine that is not included in the Foundation Library.

1. Purpose

S17DEF returns a sequence of values for the Bessel functions $J_{(nu)+n}(z)$ for complex z, non-negative (nu) and n=0,1,...,N-1, with an option for exponential scaling.

2. Specification

```
      SUBROUTINE S17DEF (FNU, Z, N, SCALE, CY, NZ, IFAIL)
      INTEGER              N, NZ, IFAIL
      DOUBLE PRECISION     FNU
      COMPLEX(KIND(1.0D0)) Z, CY(N)
      CHARACTER*1          SCALE
```

3. Description

This subroutine evaluates a sequence of values for the Bessel function $J_{(nu)}(z)$, where z is complex, -(pi) < arg z <= (pi), and (nu) is the real, non-negative order. The N-member sequence is generated for orders (nu), (nu)+1,...,(nu)+N-1. Optionally, the sequence is scaled by the factor $e^{-|Im z|}$.

Note: although the routine may not be called with (nu) less than zero, for negative orders the formula $J_{-(nu)}(z)=J_{(nu)}(z)cos((pi)(nu))-Y_{(nu)}(z)sin((pi)(nu))$ may be used (for the Bessel function $Y_{(nu)}(z)$, see S17DCF).

The routine is derived from the routine CBESJ in Amos [2]. It is based on the relations $J_{(nu)}(z)=e^{(nu)(pi)i/2} I_{(nu)}(-iz)$, Im z>=0.0 and $J_{(nu)}(z)=e^{-(nu)(pi)i/2} I_{(nu)}(iz)$, Im z<0.0.

The Bessel function $I_{(nu)}(z)$ is computed using a variety of techniques depending on the region under consideration.

When N is greater than 1, extra values of $J_{(nu)}(z)$ are computed

```
                                        (nu)
using recurrence relations.

For very large |z| or ((nu)+N-1), argument reduction will cause
total loss of accuracy, and so no computation is performed. For
slightly smaller |z| or ((nu)+N-1), the computation is performed
but results are accurate to less than half of machine precision.
If Im z is large, there is a risk of overflow and so no
computation is performed. In all the above cases, a warning is
given by the routine.

4. References
```



```
[1]   Abramowitz M and Stegun I A (1968) Handbook of Mathematical
      Functions. Dover Publications.

[2]   Amos D E (1986) Algorithm 644: A Portable Package for Bessel
      Functions of a Complex Argument and Nonnegative Order. ACM
      Trans. Math. Softw. 12 265--273.
```



```
5. Parameters

 1:  FNU -- DOUBLE PRECISION                                Input
     On entry: the order, (nu), of the first member of the
     sequence of functions. Constraint: FNU >= 0.0.

 2:  Z -- COMPLEX(KIND(1.0D0))                              Input
     On entry: the argument z of the functions.

 3:  N -- INTEGER                                           Input
     On entry: the number, N, of members required in the sequence
     J    (z),J      (z),...,J        (z). Constraint: N >= 1.
      (nu)     (nu)+1         (nu)+N-1

 4:  SCALE -- CHARACTER*1                                   Input
     On entry: the scaling option.
          If SCALE = 'U', the results are returned unscaled.

          If SCALE = 'S', the results are returned scaled by the
                  -|Imz|
          factor e      .
     Constraint: SCALE = 'U' or 'S'.

 5:  CY(N) -- COMPLEX(KIND(1.0D)) array                    Output
     On exit: the N required function values: CY(i) contains
     J        (z), for i=1,2,...,N.
      (nu)+i-1

 6:  NZ -- INTEGER                                         Output
     On exit: the number of components of CY that are set to zero
```

```
     due to underflow. If NZ > 0, then elements CY(N-NZ+1), CY(N-
     NZ+2),...,CY(N) are set to zero.

 7:  IFAIL -- INTEGER                                Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

IFAIL= 1
     On entry FNU < 0.0,

     or       N < 1,

     or       SCALE /= 'U' or 'S'.

IFAIL= 2
     No computation has been performed due to the likelihood of
     overflow, because Im Z is larger than a machine-dependent
     threshold value (given in the Users' Note for your
     implementation). This error exit can only occur when SCALE =
     'U'.

IFAIL= 3
     The computation has been performed, but the errors due to
     argument reduction in elementary functions make it likely
     that the results returned by S17DEF are accurate to less
     than half of machine precision. This error exit may occur if
     either ABS(Z) or FNU + N - 1 is greater than a machine-
     dependent threshold value (given in the Users' Note for
     your implementation).

IFAIL= 4
     No computation has been performed because the errors due to
     argument reduction in elementary functions mean that all
     precision in results returned by S17DEF would be lost. This
     error exit may occur when either ABS(Z) or FNU + N - 1 is
     greater than a machine-dependent threshold value (given in
     the Users' Note for your implementation).

IFAIL= 5
```

No results are returned because the algorithm termination condition has not been met. This may occur because the parameters supplied to S17DEF would have caused overflow or underflow.

7. Accuracy

All constants in subroutine S17DEF are given to approximately 18 digits of precision. Calling the number of digits of precision in the floating-point arithmetic being used t, then clearly the maximum number of correct digits in the results obtained is limited by p=min(t,18). Because of errors in argument reduction when computing elementary functions inside S17DEF, the actual number of correct digits is limited, in general, by p-s, where $s\sim max(1,|log_{10}|(|z|),|log_{10}(nu)|)$ represents the number of digits lost due to the argument reduction. Thus the larger the values of |z| and (nu), the less the precision in the result. If S17DEF is called with N>1, then computation of function values via recurrence may lead to some further small loss of accuracy.

If function values which should nominally be identical are computed by calls to S17DEF with different base values of (nu) and different N, the computed values may not agree exactly. Empirical tests with modest values of (nu) and z have shown that the discrepancy is limited to the least significant 3-4 digits of precision.

8. Further Comments

The time taken by the routine for a call of S17DEF is approximately proportional to the value of N, plus a constant. In general it is much cheaper to call S17DEF with N greater than 1, rather than to make N separate calls to S17DEF.

Paradoxically, for some values of z and (nu), it is cheaper to call S17DEF with a larger value of N than is required, and then discard the extra function values returned. However, it is not possible to state the precise circumstances in which this is likely to occur. It is due to the fact that the base value used to start recurrence may be calculated in different regions for different N, and the costs in each region may differ greatly.

Note that if the function required is $J_0(x)$ or $J_1(x)$, i.e., (nu) = 0.0 or 1.0, where x is real and positive, and only a single unscaled function value is required, then it may be much cheaper to call S17AEF or S17AFF respectively.

9. Example

```
         The example program prints a caption and then proceeds to read
         sets of data from the input data stream. The first datum is a
         value for the order FNU, the second is a complex value for the
         argument, Z, and the third is a value for the parameter SCALE.

         The program calls the routine with N = 2 to evaluate the function
         for orders FNU and FNU + 1, and it prints the results. The
         process is repeated until the end of the input data stream is
         encountered.

         The example program is not reproduced here. The source code for
         all example programs is distributed with the NAG Foundation
         Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Value of the Airy function $Ai(z)$ or derivative $Ai'(z)$

— nags.ht —

```
\begin{page}{manpageXXs17dgf}{NAG Documentation: s17dgf}
\beginscroll
\begin{verbatim}



     S17DGF(3NAG)      Foundation Library (12/10/92)      S17DGF(3NAG)



         S17 -- Approximations of Special Functions                 S17DGF
                 S17DGF -- NAG Foundation Library Routine Document

         Note: Before using this routine, please read the Users' Note for
         your implementation to check implementation-dependent details.
         The symbol (*) after a NAG routine name denotes a routine that is
         not included in the Foundation Library.

         1. Purpose

         S17DGF returns the value of the Airy function Ai(z) or its
         derivative Ai'(z) for complex z, with an option for exponential
```

scaling.

2. Specification

```
SUBROUTINE S17DGF (DERIV, Z, SCALE, AI, NZ, IFAIL)
INTEGER              NZ, IFAIL
COMPLEX(KIND(1.0D0)) Z, AI
CHARACTER*1          DERIV, SCALE
```

3. Description

This subroutine returns a value for the Airy function Ai(z) or its derivative Ai'(z), where z is complex, $-\pi < \arg z \le \pi$.

Optionally, the value is scaled by the factor $e^{2z\sqrt{z}/3}$.

The routine is derived from the routine CAIRY in Amos [2]. It is based on the relations $Ai(z)=\frac{\sqrt{z}K_{1/3}(w)}{\pi\sqrt{3}}$, and $Ai'(z)=\frac{-zK_{2/3}(w)}{\pi\sqrt{3}}$,

where K_{ν} is the modified Bessel function and $w=2z\sqrt{z}/3$.

For very large |z|, argument reduction will cause total loss of accuracy, and so no computation is performed. For slightly smaller |z|, the computation is performed but results are accurate to less than half of machine precision. If Re w is too large, and the unscaled function is required, there is a risk of overflow and so no computation is performed. In all the above cases, a warning is given by the routine.

4. References

[1] Abramowitz M and Stegun I A (1968) Handbook of Mathematical Functions. Dover Publications.

[2] Amos D E (1986) Algorithm 644: A Portable Package for Bessel Functions of a Complex Argument and Nonnegative Order. ACM Trans. Math. Softw. 12 265--273.

5. Parameters

```
 1:  DERIV -- CHARACTER*1                                   Input
     On entry: specifies whether the function or its derivative
     is required.
          If DERIV = 'F', Ai(z) is returned.

          If DERIV = 'D', Ai'(z) is returned.
     Constraint: DERIV = 'F' or 'D'.

 2:  Z -- COMPLEX(KIND(1.0D0))                              Input
     On entry: the argument z of the function.

 3:  SCALE -- CHARACTER*1                                   Input
     On entry: the scaling option.

     If SCALE = 'U', the result is returned unscaled.

     If SCALE = 'S', the result is returned scaled by the factor


      2z\/z/3
     e       . Constraint: SCALE = 'U' or 'S'.

 4:  AI -- COMPLEX(KIND(1.0D0))                            Output
     On exit: the required function or derivative value.

 5:  NZ -- INTEGER                                         Output
     On exit: NZ indicates whether or not AI is set to zero due
     to underflow. This can only occur when SCALE = 'U'.

     If NZ = 0, AI is not set to zero.

     If NZ = 1, AI is set to zero.

 6:  IFAIL -- INTEGER                                Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

IFAIL= 1
```

On entry DERIV /= 'F' or 'D'.

or SCALE /= 'U' or 'S'.

IFAIL= 2

No computation has been performed due to the likelihood of

overflow, because Re w is too large, where w=2Z\/Z/3 -- how large depends on Z and the overflow threshold of the machine. This error exit can only occur when SCALE = 'U'.

IFAIL= 3

The computation has been performed, but the errors due to argument reduction in elementary functions make it likely that the result returned by S17DGF is accurate to less than half of machine precision. This error exit may occur if ABS (Z) is greater than a machine-dependent threshold value (given in the Users' Note for your implementation).

IFAIL= 4

No computation has been performed because the errors due to argument reduction in elementary functions mean that all precision in the result returned by S17DGF would be lost. This error exit may occur if ABS(Z) is greater than a machine-dependent threshold value (given in the Users' Note for your implementation).

IFAIL= 5

No result is returned because the algorithm termination condition has not been met. This may occur because the parameters supplied to S17DGF would have caused overflow or underflow.

7. Accuracy

All constants in subroutine S17DGF are given to approximately 18 digits of precision. Calling the number of digits of precision in the floating-point arithmetic being used t, then clearly the maximum number of correct digits in the results obtained is limited by p=min(t,18). Because of errors in argument reduction when computing elementary functions inside S17DGF, the actual number of correct digits is limited, in general, by p-s, where $s \sim \max(1,|\log_{10}|z||)$ represents the number of digits lost due to the argument reduction. Thus the larger the value of |z|, the less the precision in the result.

Empirical tests with modest values of z, checking relations between Airy functions Ai(z), Ai'(z), Bi(z) and Bi'(z), have

```
          shown errors limited to the least significant 3-4 digits of
          precision.

          8. Further Comments

          Note that if the function is required to operate on a real
          argument only, then it may be much cheaper to call S17AGF or
          S17AJF.

          9. Example

          The example program prints a caption and then proceeds to read
          sets of data from the input data stream. The first datum is a
          value for the parameter DERIV, the second is a complex value for
          the argument, Z, and the third is a value for the parameter
          SCALE. The program calls the routine and prints the results. The
          process is repeated until the end of the input data stream is
          encountered.


          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.
\end{verbatim}
\endscroll
\end{page}
```

—————

Value of the Airy function $Bi(z)$ or derivative $Bi'(z)$

— nags.ht —

```
\begin{page}{manpageXXs17dhf}{NAG Documentation: s17dhf}
\beginscroll
\begin{verbatim}



     S17DHF(3NAG)      Foundation Library (12/10/92)      S17DHF(3NAG)



          S17 -- Approximations of Special Functions                 S17DHF
                  S17DHF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
```

your implementation to check implementation-dependent details. The symbol (*) after a NAG routine name denotes a routine that is not included in the Foundation Library.

1. Purpose

S17DHF returns the value of the Airy function Bi(z) or its derivative Bi'(z) for complex z, with an option for exponential scaling.

2. Specification

```
       SUBROUTINE S17DHF (DERIV, Z, SCALE, BI, IFAIL)
       INTEGER              IFAIL
       COMPLEX(KIND(1.0D0)) Z, BI
       CHARACTER*1          DERIV, SCALE
```

3. Description

This subroutine returns a value for the Airy function Bi(z) or its derivative Bi'(z), where z is complex, $-\pi < \arg z \le \pi$.

Optionally, the value is scaled by the factor $e^{|\mathrm{Re}\,(2z\sqrt{z}/3)|}$.

The routine is derived from the routine CBIRY in Amos [2]. It is based on the relations $Bi(z)=\frac{\sqrt{z}}{\sqrt{3}}(I_{-1/3}(w)+I_{1/3}(w))$, and $Bi'(z)=\frac{z}{\sqrt{3}}(I_{-2/3}(w)+I_{2/3}(w))$, where I_{ν} is the modified Bessel function and $w=2z\sqrt{z}/3$.

For very large |z|, argument reduction will cause total loss of accuracy, and so no computation is performed. For slightly smaller |z|, the computation is performed but results are accurate to less than half of machine precision. If Re z is too large, and the unscaled function is required, there is a risk of overflow and so no computation is performed. In all the above cases, a warning is given by the routine.

4. References

[1] Abramowitz M and Stegun I A (1968) Handbook of Mathematical Functions. Dover Publications.

[2] Hammersley J M and Handscomb D C (1967) Monte-Carlo Methods. Methuen.

5. Parameters

```
 1:  DERIV -- CHARACTER*1                                  Input
     On entry: specifies whether the function or its derivative
     is required.
          If DERIV = 'F', Bi(z) is returned.

          If DERIV = 'D', Bi'(z) is returned.
     Constraint: DERIV = 'F' or 'D'.

 2:  Z -- COMPLEX(KIND(1.0D0))                             Input
     On entry: the argument z of the function.

 3:  SCALE -- CHARACTER*1                                  Input
     On entry: the scaling option.
          If SCALE = 'U', the result is returned unscaled.

          If SCALE = 'S', the result is returned scaled by the


                  |Re(2z\/z/3)|
          factor e             .
     Constraint: SCALE = 'U' or 'S'.

 4:  BI -- COMPLEX(KIND(1.0D0))                           Output
     On exit: the required function or derivative value.

 5:  IFAIL -- INTEGER                               Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).
```

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL= 1

```
     On entry DERIV /= 'F' or 'D'.

     or       SCALE /= 'U' or 'S'.

IFAIL= 2
     No computation has been performed due to the likelihood of
     overflow, because real(Z) is too large - how large depends
     on the overflow threshold of the machine. This error exit
     can only occur when SCALE = 'U'.

IFAIL= 3
     The computation has been performed, but the errors due to
     argument reduction in elementary functions make it likely
     that the result returned by S17DHF is accurate to less than
     half of machine precision. This error exit may occur if ABS
     (Z) is greater than a machine-dependent threshold value
     (given in the Users' Note for your implementation).

IFAIL= 4
     No computation has been performed because the errors due to
     argument reduction in elementary functions mean that all
     precision in the result returned by S17DHF would be lost.
     This error exit may occur if ABS(Z) is greater than a
     machine-dependent threshold value (given in the Users' Note
     for your implementation).

IFAIL= 5
     No result is returned because the algorithm termination
     condition has not been met. This may occur because the
     parameters supplied to S17DHF would have caused overflow or
     underflow.

7. Accuracy

All constants in subroutine S17DHF are given to approximately 18
digits of precision. Calling the number of digits of precision in
the floating-point arithmetic being used t, then clearly the
maximum number of correct digits in the results obtained is
limited by p=min(t,18). Because of errors in argument reduction
when computing elementary functions inside S17DHF, the actual
number of correct digits is limited, in general, by p-s, where
s~max(1,|log  |z||) represents the number of digits lost due to
            10
the argument reduction. Thus the larger the value of |z|, the
less the precision in the result.

Empirical tests with modest values of z, checking relations
between Airy functions Ai(z), Ai'(z), Bi(z) and Bi'(z), have
shown errors limited to the least significant 3-4 digits of
precision.
```

```
          8. Further Comments

          Note that if the function is required to operate on a real
          argument only, then it may be much cheaper to call S17AHF or
          S17AKF.

          9. Example

          The example program prints a caption and then proceeds to read
          sets of data from the input data stream. The first datum is a
          value for the parameter DERIV, the second is a complex value for
          the argument, Z, and the third is a value for the parameter
          SCALE. The program calls the routine and prints the results. The
          process is repeated until the end of the input data stream is
          encountered.


          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.
\end{verbatim}
\endscroll
\end{page}
```

Returns a sequence of values for the Hankel functions

— nags.ht —

```
\begin{page}{manpageXXs17dlf}{NAG Documentation: s17dlf}
\beginscroll
\begin{verbatim}



     S17DLF(3NAG)      Foundation Library (12/10/92)      S17DLF(3NAG)



          S17 -- Approximations of Special Functions                S17DLF
                  S17DLF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
```

not included in the Foundation Library.

1. Purpose

S17DLF returns a sequence of values for the Hankel functions $H^{(1)}_{\nu+n}(z)$ or $H^{(2)}_{\nu+n}(z)$ for complex z, non-negative ν and $n=0,1,...,N-1$, with an option for exponential scaling.

2. Specification

```
SUBROUTINE S17DLF (M, FNU, Z, N, SCALE, CY, NZ, IFAIL)
INTEGER              M, N, NZ, IFAIL
DOUBLE PRECISION     FNU
COMPLEX(KIND(1.0D0)) Z, CY(N)
CHARACTER*1          SCALE
```

3. Description

This subroutine evaluates a sequence of values for the Hankel function $H^{(1)}_{\nu}(z)$ or $H^{(2)}_{\nu}(z)$, where z is complex, $-\pi < \arg z \le \pi$, and ν is the real, non-negative order. The N-member sequence is generated for orders ν, $\nu+1,...,\nu+N-1$. Optionally, the sequence is scaled by the factor e^{-iz} if the function is $H^{(1)}_{\nu}(z)$ or by the factor e^{iz} if the function is $H^{(2)}_{\nu}(z)$.

Note: although the routine may not be called with ν less than zero, for negative orders the formulae $H^{(1)}_{-\nu}(z)=e^{\nu\pi i}H^{(1)}_{\nu}(z)$, and $H^{(2)}_{-\nu}(z)=e^{-\nu\pi i}H^{(2)}_{\nu}(z)$ may be used.

The routine is derived from the routine CBESH in Amos [2]. It is based on the relation

$$H^{(m)}_{\nu}(z)=\frac{1}{p}e^{-p\nu}K_{\nu}(ze^{-p}),$$

(pi) (pi)

where $p=i\,\frac{\text{----}}{2}$ if m=1 and $p=-i\,\frac{\text{----}}{2}$ if m=2, and the Bessel function $K_{(nu)}(z)$ is computed in the right half-plane only. Continuation of $K_{(nu)}(z)$ to the left half-plane is computed in terms of the Bessel function $I_{(nu)}(z)$. These functions are evaluated using a variety of different techniques, depending on the region under consideration.

When N is greater than 1, extra values of $H^{(m)}_{(nu)}(z)$ are computed using recurrence relations.

For very large |z| or ((nu)+N-1), argument reduction will cause total loss of accuracy, and so no computation is performed. For slightly smaller |z| or ((nu)+N-1), the computation is performed but results are accurate to less than half of machine precision. If |z| is very small, near the machine underflow threshold, or ((nu)+N-1) is too large, there is a risk of overflow and so no computation is performed. In all the above cases, a warning is given by the routine.

4. References

[1] Abramowitz M and Stegun I A (1968) Handbook of Mathematical Functions. Dover Publications.

[2] Amos D E (1986) Algorithm 644: A Portable Package for Bessel Functions of a Complex Argument and Nonnegative Order. ACM Trans. Math. Softw. 12 265--273.

5. Parameters

1: M -- INTEGER Input

On entry: the kind of functions required.

If M = 1, the functions are $H^{(1)}_{(nu)}(z)$.

If M = 2, the functions are $H^{(2)}_{(nu)}(z)$.

Constraint: M = 1 or 2.

2: FNU -- DOUBLE PRECISION Input

On entry: the order, (nu), of the first member of the sequence of functions. Constraint: FNU >= 0.0.

```
 3:  Z -- COMPLEX(KIND(1.0D0))                             Input
     On entry: the argument z of the functions. Constraint: Z /=
     (0.0, 0.0).

 4:  N -- INTEGER                                          Input
     On entry: the number, N, of members required in the sequence
      (M)    (M)          (M)
     H    , H      ,..., H        . Constraint: N >= 1.
      (nu)   (nu)+1       (nu)+N-1

 5:  SCALE -- CHARACTER*1                                  Input
     On entry: the scaling option.
          If SCALE = 'U', the results are returned unscaled.

          If SCALE = 'S', the results are returned scaled by the
                  -iz                               iz
          factor e    when M = 1, or by the factor e   when M =
          2.
     Constraint: SCALE = 'U' or 'S'.

 6:  CY(N) -- COMPLEX(KIND(1.0D)) array                   Output
     On exit: the N required function values: CY(i) contains
      (M)
     H        , for i=1,2,...,N.
      (nu)+i-1

 7:  NZ -- INTEGER                                        Output
     On exit: the number of components of CY that are set to zero
     due to underflow. If NZ > 0, then if Imz>0.0 and M = 1, or
     Imz<0.0 and M = 2, elements CY(1), CY(2),...,CY(NZ) are set
     to zero. In the complementary half-planes, NZ simply states
     the number of underflows, and not which elements they are.

 8:  IFAIL -- INTEGER                               Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

IFAIL= 1
```

```
     On entry M /= 1 and M /= 2,

     or       FNU < 0.0,

     or       Z = (0.0, 0.0),

     or       N < 1,

     or       SCALE /= 'U' or 'S'.

IFAIL= 2
     No computation has been performed due to the likelihood of
     overflow, because ABS(Z) is less than a machine-dependent
     threshold value (given in the Users' Note for your
     implementation).

IFAIL= 3
     No computation has been performed due to the likelihood of
     overflow, because FNU + N - 1 is too large - how large
     depends on Z and the overflow threshold of the machine.

IFAIL= 4
     The computation has been performed, but the errors due to
     argument reduction in elementary functions make it likely
     that the results returned by S17DLF are accurate to less
     than half of machine precision. This error exit may occur if
     either ABS(Z) or FNU + N - 1 is greater than a machine-
     dependent threshold value (given in the Users' Note for
     your implementation).

IFAIL= 5
     No computation has been performed because the errors due to
     argument reduction in elementary functions mean that all
     precision in results returned by S17DLF would be lost. This
     error exit may occur when either of ABS(Z) or FNU + N - 1 is
     greater than a machine-dependent threshold value (given in
     the Users' Note for your implementation).

IFAIL= 6
     No results are returned because the algorithm termination
     condition has not been met. This may occur because the
     parameters supplied to S17DLF would have caused overflow or
     underflow.

7. Accuracy

All constants in subroutine S17DLF are given to approximately 18
digits of precision. Calling the number of digits of precision in
the floating-point arithmetic being used t, then clearly the
maximum number of correct digits in the results obtained is
```

```
          limited by p=min(t,18). Because of errors in argument reduction
          when computing elementary functions inside S17DLF, the actual
          number of correct digits is limited, in general, by p-s, where
          s~max(1,|log  |z||,|log  (nu)|) represents the number of digits
                      10         10
          lost due to the argument reduction. Thus the larger the values of
          |z| and (nu), the less the precision in the result. If S17DLF is
          called with N>1, then computation of function values via
          recurrence may lead to some further small loss of accuracy.

          If function values which should nominally be identical are
          computed by calls to S17DLF with different base values of (nu)
          and different N, the computed values may not agree exactly.
          Empirical tests with modest values of (nu) and z have shown that
          the discrepancy is limited to the least significant 3-4 digits of
          precision.

          8. Further Comments

          The time taken by the routine for a call of S17DLF is
          approximately proportional to the value of N, plus a constant. In
          general it is much cheaper to call S17DLF with N greater than 1,
          rather than to make N separate calls to S17DLF.

          Paradoxically, for some values of z and (nu), it is cheaper to
          call S17DLF with a larger value of N than is required, and then
          discard the extra function values returned. However, it is not
          possible to state the precise circumstances in which this is
          likely to occur. It is due to the fact that the base value used
          to start recurrence may be calculated in different regions for
          different N, and the costs in each region may differ greatly.

          9. Example

          The example program prints a caption and then proceeds to read
          sets of data from the input data stream. The first datum is a
          value for the kind of function, M, the second is a value for the
          order FNU, the third is a complex value for the argument, Z, and
          the fourth is a value for the parameter SCALE. The program calls
          the routine with N = 2 to evaluate the function for orders FNU
          and FNU + 1, and it prints the results. The process is repeated
          until the end of the input data stream is encountered.

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.
\end{verbatim}
\endscroll
\end{page}
```

———-

Returns the value of the modified Bessel Function $K_0(x)$

— nags.ht —

```
\begin{page}{manpageXXs18acf}{NAG Documentation: s18acf}
\beginscroll
\begin{verbatim}



     S18ACF(3NAG)      Foundation Library (12/10/92)      S18ACF(3NAG)



          S18 -- Approximations of Special Functions                 S18ACF
                  S18ACF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          S18ACF returns the value of the modified Bessel Function K (x),
                                                                    0
          via the routine name.

          2. Specification

                 DOUBLE PRECISION FUNCTION S18ACF (X, IFAIL)
                 INTEGER          IFAIL
                 DOUBLE PRECISION X

          3. Description

          This routine evaluates an approximation to the modified Bessel
          Function of the second kind K (x).
                                       0

          Note: K (x) is undefined for x<=0 and the routine will fail for
                 0
          such arguments.

          The routine is based on five Chebyshev expansions:
```

For $0<x\le 1$,

$$K_0(x)=-\ln x \sum_{r=0}{}' a_r T_r(t)+\sum_{r=0}{}' b_r T_r(t), \text{ where } t=2x^2-1;$$

For $1<x\le 2$,

$$K_0(x)=e^{-x}\sum_{r=0}{}' c_r T_r(t), \text{ where } t=2x-3;$$

For $2<x\le 4$,

$$K_0(x)=e^{-x}\sum_{r=0}{}' d_r T_r(t), \text{ where } t=x-3;$$

For $x>4$,

$$K_0(x)=\frac{e^{-x}}{\sqrt{x}}\sum_{r=0}{}' e_r T_r(t), \text{ where } t=\frac{9-x}{1+x}.$$

For x near zero, $K_0(x)\simeq -\gamma-\ln\left(\frac{x}{2}\right)$, where γ denotes Euler's constant. This approximation is used when x is sufficiently small for the result to be correct to machine precision.

For large x, where there is a danger of underflow due to the smallness of K_0, the result is set exactly to zero.

4. References

[1] Abramowitz M and Stegun I A (1968) Handbook of Mathematical Functions. Dover Publications.

5. Parameters

1: X -- DOUBLE PRECISION Input
On entry: the argument x of the function. Constraint: X > 0.
0.

```
 2:  IFAIL -- INTEGER                                Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).
```

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1

X <= 0.0, K_0 is undefined. On soft failure the routine returns zero.

7. Accuracy

Let (delta) and (epsilon) be the relative errors in the argument and result respectively.

If (delta) is somewhat larger than the machine precision (i.e., if (delta) is due to data errors etc), then (epsilon) and (delta) are approximately related by:

$$(\epsilon) \simeq \left| \frac{x K_1(x)}{K_0(x)} \right| (\delta).$$

Figure 1 shows the behaviour of the error amplification factor

$$\left| \frac{x K_1(x)}{K_0(x)} \right|.$$

Figure 1
Please see figure in printed Reference Manual

However, if (delta) is of the same order as machine precision, then rounding errors could make (epsilon) slightly larger than the above relation predicts.

For small x, the amplification factor is approximately $\left| \frac{1}{\ln x} \right|$,

```
          which implies strong attenuation of the error, but in general
          (epsilon) can never be less than the machine precision.

          For large x, (epsilon)~=x(delta) and we have strong amplification
          of the relative error. Eventually K , which is asymptotically
                                             0
                     -x
                    e
          given by  ---, becomes so small that it cannot be calculated


                    \/x
          without underflow and hence the routine will return zero. Note
          that for large x the errors will be dominated by those of the
          Fortran intrinsic function EXP.

          8. Further Comments

          For details of the time taken by the routine see the appropriate
          the Users' Note for your implementation.

          9. Example

          The example program reads values of the argument x from a file,
          evaluates the function at each value of x and prints the results.

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.
\end{verbatim}
\endscroll
\end{page}
```

Returns the value of the modified Bessel Function $K_1(x)$

— **nags.ht** —

```
\begin{page}{manpageXXs18adf}{NAG Documentation: s18adf}
\beginscroll
\begin{verbatim}


     S18ADF(3NAG)      Foundation Library (12/10/92)      S18ADF(3NAG)
```

S18 -- Approximations of Special Functions S18ADF

S18ADF -- NAG Foundation Library Routine Document

Note: Before using this routine, please read the Users' Note for your implementation to check implementation-dependent details. The symbol (*) after a NAG routine name denotes a routine that is not included in the Foundation Library.

1. Purpose

S18ADF returns the value of the modified Bessel Function $K_1(x)$, via the routine name.

2. Specification

```
DOUBLE PRECISION FUNCTION S18ADF (X, IFAIL)
INTEGER          IFAIL
DOUBLE PRECISION X
```

3. Description

This routine evaluates an approximation to the modified Bessel Function of the second kind $K_1(x)$.

Note: $K_1(x)$ is undefined for $x<=0$ and the routine will fail for such arguments.

The routine is based on five Chebyshev expansions:

For $0<x<=1$,

$$K_1(x)=\frac{1}{x}+x\ln x\sum_{r=0}{}' a_r T_r(t)-x\sum_{r=0}{}' b_r T_r(t) \text{ , where } t=2x^2-1;$$

For $1<x<=2$,

$$K_1(x)=e^{-x}\sum_{r=0}{}' c_r T_r(t) \text{ , where } t=2x-3;$$

For $2<x<=4$,

$$K_1(x)=e^{-x}\sum_{r=0}' d_r T_r(t)\text{ , where } t=x-3;$$

For x>4,

$$K_1(x)=\frac{e^{-x}}{\sqrt{x}}\sum_{r=0}' e_r T_r(t)\text{ , where } t=\frac{9-x}{1+x}.$$

For x near zero, $K_1(x)\simeq \frac{1}{x}$. This approximation is used when x is sufficiently small for the result to be correct to machine precision. For very small x on some machines, it is impossible to calculate $\frac{1}{x}$ without overflow and the routine must fail.

For large x, where there is a danger of underflow due to the smallness of K_1, the result is set exactly to zero.

4. References

[1] Abramowitz M and Stegun I A (1968) Handbook of Mathematical Functions. Dover Publications.

5. Parameters

1: X -- DOUBLE PRECISION Input
On entry: the argument x of the function. Constraint: X > 0.
0.

2: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1

X <= 0.0, K_1 is undefined. On soft failure the routine returns zero.

IFAIL= 2

X is too small, there is a danger of overflow. On soft failure the routine returns approximately the largest representable value.

7. Accuracy

Let (delta) and (epsilon) be the relative errors in the argument and result respectively.

If (delta) is somewhat larger than the machine precision (i.e., if (delta) is due to data errors etc), then (epsilon) and (delta) are approximately related by:

$$(\epsilon) \simeq \left| \frac{xK_0(x)-K_1(x)}{K_1(x)} \right| (\delta).$$

Figure 1 shows the behaviour of the error amplification factor

$$\left| \frac{xK_0(x)-K_1(x)}{K_1(x)} \right|.$$

Figure 1
Please see figure in printed Reference Manual

However if (delta) is of the same order as the machine precision, then rounding errors could make (epsilon) slightly larger than the above relation predicts.

For small x, (epsilon)~=(delta) and there is no amplification of errors.

For large x, (epsilon)~=x(delta) and we have strong amplification of the relative error. Eventually K_1, which is asymptotically given by $\frac{e^{-x}}{}$, becomes so small that it cannot be calculated

```
         \/x
without underflow and hence the routine will return zero. Note
that for large x the errors will be dominated by those of the
Fortran intrinsic function EXP.

8. Further Comments

For details of the time taken by the routine see the Users' Note
for your implementation.

9. Example

The example program reads values of the argument x from a file,
evaluates the function at each value of x and prints the results.

The example program is not reproduced here. The source code for
all example programs is distributed with the NAG Foundation
Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

———-

Returns the value of the modified Bessel Function $I_0(x)$

— nags.ht —

```
\begin{page}{manpageXXs18aef}{NAG Documentation: s18aef}
\beginscroll
\begin{verbatim}



     S18AEF(3NAG)      Foundation Library (12/10/92)      S18AEF(3NAG)



          S18 -- Approximations of Special Functions                 S18AEF
                  S18AEF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.
```

1. Purpose

S18AEF returns the value of the modified Bessel Function $I_0(x)$, via the routine name.

2. Specification

```
DOUBLE PRECISION FUNCTION S18AEF (X, IFAIL)
INTEGER          IFAIL
DOUBLE PRECISION X
```

3. Description

This routine evaluates an approximation to the modified Bessel Function of the first kind $I_0(x)$.

Note: $I_0(-x)=I_0(x)$, so the approximation need only consider $x>=0$.

The routine is based on three Chebyshev expansions:

For $0<x<=4$,

$$I_0(x)=e^x \sum_{r=0}' a_r T_r(t) , \text{ where } t=2\left(\frac{x}{4}\right)-1;$$

For $4<x<=12$,

$$I_0(x)=e^x \sum_{r=0}' b_r T_r(t) , \text{ where } t=\frac{x-8}{4};$$

For $x>12$,

$$I_0(x)=\frac{e^x}{\sqrt{x}} \sum_{r=0}' c_r T_r(t) , \text{ where } t=2\left(\frac{12}{x}\right)-1.$$

For small x, $I_0(x)\simeq1$. This approximation is used when x is sufficiently small for the result to be correct to machine precision.

For large x, the routine must fail because of the danger of overflow in calculating e^x.

4. References

[1] Abramowitz M and Stegun I A (1968) Handbook of Mathematical Functions. Dover Publications.

5. Parameters

1: X -- DOUBLE PRECISION Input

On entry: the argument x of the function.

2: IFAIL -- INTEGER Input/Output

On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1

X is too large. On soft failure the routine returns the approximate value of $I_0(x)$ at the nearest valid argument.

7. Accuracy

Let (delta) and (epsilon) be the relative errors in the argument and result respectively.

If (delta) is somewhat larger than the machine precision (i.e., if (delta) is due to data errors etc), then (epsilon) and (delta) are approximately related by:

$$\epsilon \simeq \left| \frac{x I_1(x)}{I_0(x)} \right| \delta .$$

Figure 1 shows the behaviour of the error amplification factor

$$\left| x I_1(x) \right.$$

```
                                        | ------|.
                                        | I (x) |
                                        |  0    |

                                        Figure 1
                              Please see figure in printed Reference Manual

          However if (delta) is of the same order as machine precision,
          then rounding errors could make (epsilon) slightly larger than
          the above relation predicts.

                                                                  2
                                                                 x
          For small x the amplification factor is approximately  --, which
                                                                 2
          implies strong attenuation of the error, but in general (epsilon)
          can never be less than the machine precision.

          For large x, (epsilon)~=x(delta) and we have strong amplification
          of errors. However the routine must fail for quite moderate
          values of x, because I (x) would overflow; hence in practice the
                                 0
          loss of accuracy for large x is not excessive. Note that for
          large x the errors will be dominated by those of the Fortran
          intrinsic function EXP.

          8. Further Comments

          For details of the time taken by the routine see the Users' Note
          for your implementation.

          9. Example

          The example program reads values of the argument x from a file,
          evaluates the function at each value of x and prints the results.

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.
\end{verbatim}
\endscroll
\end{page}
```

Returns a value for the modified Bessel Function $I_1(x)$

— nags.ht —

```
\begin{page}{manpageXXs18aff}{NAG Documentation: s18aff}
\beginscroll
\begin{verbatim}



     S18AFF(3NAG)      Foundation Library (12/10/92)      S18AFF(3NAG)



          S18 -- Approximations of Special Functions                 S18AFF
                  S18AFF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          S18AFF returns a value for the modified Bessel Function I (x),
                                                                   1
          via the routine name.

          2. Specification

                 DOUBLE PRECISION FUNCTION S18AFF (X, IFAIL)
                 INTEGER          IFAIL
                 DOUBLE PRECISION X

          3. Description

          This routine evaluates an approximation to the modified Bessel
          Function of the first kind I (x).
                                      1

          Note: I (-x)=-I (x), so the approximation need only consider x>=0
                 1       1

          The routine is based on three Chebyshev expansions:

          For 0<x<=4,

                             --'                    ( x)2
                   I (x)=x >    a T (t) , where t=2( -) -1;
```

```
             1      --   r r              ( 4)
                   r=0
```

For 4<x<=12,

```
                   x --'                     x-8
             I (x)=e  >   b T (t) , where t= ---;
              1       --   r r                4
                     r=0
```

For x>12,

```
                 x
                e   --'                     ( 12)
          I (x)= --- >   c T (t) , where t=2( --)-1.
           1        --   r r                ( x )
               \/x r=0
```

For small x, I (x)~=x. This approximation is used when x is
              1
sufficiently small for the result to be correct to machine precision.

For large x, the routine must fail because I (x) cannot be
                                            1
represented without overflow.

4. References

[1] Abramowitz M and Stegun I A (1968) Handbook of Mathematical Functions. Dover Publications.

5. Parameters

1: X -- DOUBLE PRECISION Input
On entry: the argument x of the function.

2: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1

```
     X is too large. On soft failure the routine returns the
     approximate value of I (x) at the nearest valid argument.
                           1
```

7. Accuracy

Let (delta) and (epsilon) be the relative errors in the argument and result respectively.

If (delta) is somewhat larger than the machine precision (i.e., if (delta) is due to data errors etc), then (epsilon) and (delta) are approximately related by:

```
                          | xI (x)-I (x)|
                          |   0     1   |
               (epsilon)~=| ------------|(delta).
                          |    I (x)    |
                          |     1       |
```

Figure 1 shows the behaviour of the error amplification factor

```
                        | xI (x)-I (x)|
                        |   0     1   |
                        | ------------|,
                        |    I (x)    |
                        |     1       |
```

```
                           Figure 1
         Please see figure in printed Reference Manual
```

However if (delta) is of the same order as machine precision, then rounding errors could make (epsilon) slightly larger than the above relation predicts.

For small x, (epsilon)~=(delta) and there is no amplification of errors.

For large x, (epsilon)~=x(delta) and we have strong amplification of errors. However the routine must fail for quite moderate values of x because I (x) would overflow; hence in practice the

```
                     1
```

loss of accuracy for large x is not excessive. Note that for large x, the errors will be dominated by those of the Fortran intrinsic function EXP.

8. Further Comments

For details of the time taken by the routine see the Users' Note for your implementation.

```
         9. Example

         The example program reads values of the argument x from a file,
         evaluates the function at each value of x and prints the results.

         The example program is not reproduced here. The source code for
         all example programs is distributed with the NAG Foundation
         Library software and should be available on-line.
\end{verbatim}
\endscroll
\end{page}
```

Sequence of values for the modified Bessel $K_{\nu_n}(z)$

— **nags.ht** —

```
\begin{page}{manpageXXs18dcf}{NAG Documentation: s18dcf}
\beginscroll
\begin{verbatim}



     S18DCF(3NAG)      Foundation Library (12/10/92)      S18DCF(3NAG)



         S18 -- Approximations of Special Functions                 S18DCF
                 S18DCF -- NAG Foundation Library Routine Document

         Note: Before using this routine, please read the Users' Note for
         your implementation to check implementation-dependent details.
         The symbol (*) after a NAG routine name denotes a routine that is
         not included in the Foundation Library.

         1. Purpose

         S18DCF returns a sequence of values for the modified Bessel
         functions K      (z) for complex z, non-negative (nu) and
                    (nu)+n
         n=0,1,...,N-1, with an option for exponential scaling.

         2. Specification

                SUBROUTINE S18DCF (FNU, Z, N, SCALE, CY, NZ, IFAIL)
                INTEGER             N, NZ, IFAIL
```

```
DOUBLE PRECISION     FNU
COMPLEX(KIND(1.0D0)) Z, CY(N)
CHARACTER*1          SCALE
```

3. Description

This subroutine evaluates a sequence of values for the modified Bessel function $K_{(nu)}(z)$, where z is complex, -(pi) < arg z <= (pi), and (nu) is the real, non-negative order. The N-member sequence is generated for orders (nu), (nu)+1,...,(nu)+N-1. Optionally, the sequence is scaled by the factor e^{z}.

The routine is derived from the routine CBESK in Amos [2].

Note: although the routine may not be called with (nu) less than zero, for negative orders the formula $K_{-(nu)}(z)=K_{(nu)}(z)$ may be used.

When N is greater than 1, extra values of $K_{(nu)}(z)$ are computed using recurrence relations.

For very large |z| or ((nu)+N-1), argument reduction will cause total loss of accuracy, and so no computation is performed. For slightly smaller |z| or ((nu)+N-1), the computation is performed but results are accurate to less than half of machine precision. If |z| is very small, near the machine underflow threshold, or ((nu)+N-1) is too large, there is a risk of overflow and so no computation is performed. In all the above cases, a warning is given by the routine.

4. References

[1] Abramowitz M and Stegun I A (1968) Handbook of Mathematical Functions. Dover Publications.

[2] Amos D E (1986) Algorithm 644: A Portable Package for Bessel Functions of a Complex Argument and Nonnegative Order. ACM Trans. Math. Softw. 12 265--273.

5. Parameters

1: FNU -- DOUBLE PRECISION Input

On entry: the order, (nu), of the first member of the sequence of functions. Constraint: FNU >= 0.0.

2: Z -- COMPLEX(KIND(1.0D0)) Input

On entry: the argument z of the functions. Constraint: Z /= (0.0, 0.0).

3: N -- INTEGER Input
On entry: the number, N, of members required in the sequence $K_{(nu)}(z)$, $K_{(nu)+1}(z)$,..., $K_{(nu)+N-1}(z)$. Constraint: N >= 1.

4: SCALE -- CHARACTER*1 Input
On entry: the scaling option.

If SCALE = 'U', the results are returned unscaled.

If SCALE = 'S', the results are returned scaled by the factor e^{z}. Constraint: SCALE = 'U' or 'S'.

5: CY(N) -- COMPLEX(KIND(1.0D)) array Output
On exit: the N required function values: CY(i) contains $K_{(nu)+i-1}(z)$, for i=1,2,...,N.

6: NZ -- INTEGER Output
On exit: the number of components of CY that are set to zero due to underflow. If NZ > 0 and Rez>=0.0, elements CY(1),CY(2),...,CY(NZ) are set to zero. If Rez<0.0, NZ simply states the number of underflows, and not which elements they are.

7: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL= 1
On entry FNU < 0.0,

or Z = (0.0, 0.0),

or N < 1,

```
     or       SCALE /= 'U' or 'S'.

IFAIL= 2
     No computation has been performed due to the likelihood of
     overflow, because ABS(Z) is less than a machine-dependent
     threshold value (given in the Users' Note for your
     implementation).

IFAIL= 3
     No computation has been performed due to the likelihood of
     overflow, because FNU + N - 1 is too large - how large
     depends on Z and the overflow threshold of the machine.

IFAIL= 4
     The computation has been performed, but the errors due to
     argument reduction in elementary functions make it likely
     that the results returned by S18DCF are accurate to less
     than half of machine precision. This error exit may occur if
     either ABS(Z) or FNU + N - 1 is greater than a machine-
     dependent threshold value (given in the Users' Note for
     your implementation).

IFAIL= 5
     No computation has been performed because the errors due to
     argument reduction in elementary functions mean that all
     precision in results returned by S18DCF would be lost. This
     error exit may occur when either ABS(Z) or FNU + N - 1 is
     greater than a machine-dependent threshold value (given in
     the Users' Note for your implementation).

IFAIL= 6
     No results are returned because the algorithm termination
     condition has not been met. This may occur because the
     parameters supplied to S18DCF would have caused overflow or
     underflow.

7. Accuracy


All constants in subroutine S18DCF are given to approximately 18
digits of precision. Calling the number of digits of precision in
the floating-point arithmetic being used t, then clearly the
maximum number of correct digits in the results obtained is
limited by p=min(t,18). Because of errors in argument reduction
when computing elementary functions inside S18DCF, the actual
number of correct digits is limited, in general, by p-s, where
s~max(1,|log  |z||,|log  (nu)|) represents the number of digits
            10        10
lost due to the argument reduction. Thus the larger the values of
```

```
|z| and (nu), the less the precision in the result. If S18DCF is
called with N>1, then computation of function values via
recurrence may lead to some further small loss of accuracy.

If function values which should nominally be identical are
computed by calls to S18DCF with different base values of (nu)
and different N, the computed values may not agree exactly.
Empirical tests with modest values of (nu) and z have shown that
the discrepancy is limited to the least significant 3-4 digits of
precision.

8. Further Comments

The time taken by the routine for a call of S18DCF is
approximately proportional to the value of N, plus a constant. In
general it is much cheaper to call S18DCF with N greater than 1,
rather than to make N separate calls to S18DCF.

Paradoxically, for some values of z and (nu), it is cheaper to
call S18DCF with a larger value of N than is required, and then
discard the extra function values returned. However, it is not
possible to state the precise circumstances in which this is
likely to occur. It is due to the fact that the base value used
to start recurrence may be calculated in different regions for
different N, and the costs in each region may differ greatly.

Note that if the function required is K (x) or K (x), i.e.,
                                       0        1
(nu)=0.0 or 1.0, where x is real and positive, and only a single
function value is required, then it may be much cheaper to call
S18ACF, S18ADF, S18CCF(*) or S18CDF(*), depending on whether a
scaled result is required or not.

9. Example

The example program prints a caption and then proceeds to read
sets of data from the input data stream. The first datum is a
value for the order FNU, the second is a complex value for the
argument, Z, and the third is a value for the parameter SCALE.
The program calls the routine with N = 2 to evaluate the function
for orders FNU and FNU + 1, and it prints the results. The
process is repeated until the end of the input data stream is
encountered.

The example program is not reproduced here. The source code for
all example programs is distributed with the NAG Foundation
Library software and should be available on-line.
```

```
\end{verbatim}
\endscroll
```

```
\end{page}
```

———-

Sequence of values for the modified Bessel $I_{\nu+n}$

— nags.ht —

```
\begin{page}{manpageXXs18def}{NAG Documentation: s18def}
\beginscroll
\begin{verbatim}



     S18DEF(3NAG)      Foundation Library (12/10/92)      S18DEF(3NAG)



          S18 -- Approximations of Special Functions                S18DEF
                  S18DEF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          S18DEF returns a sequence of values for the modified Bessel
          functions I      (z) for complex z, non-negative (nu) and
                     (nu)+n
          n=0,1,...,N-1, with an option for exponential scaling.

          2. Specification

                 SUBROUTINE S18DEF (FNU, Z, N, SCALE, CY, NZ, IFAIL)
                 INTEGER              N, NZ, IFAIL
                 DOUBLE PRECISION     FNU
                 COMPLEX(KIND(1.0D0)) Z, CY(N)
                 CHARACTER*1          SCALE

          3. Description

          This subroutine evaluates a sequence of values for the modified
          Bessel function I    (z), where z is complex, -(pi) < argz <=
                           (nu)
          (pi), and (nu) is the real, non-negative order. The N-member
```

sequence is generated for orders (nu), (nu)+1,...,(nu)+N-1.

Optionally, the sequence is scaled by the factor $e^{-|Rez|}$.

The routine is derived from the routine CBESI in Amos [2].

Note: although the routine may not be called with (nu) less than zero, for negative orders the formula

$$I_{-(nu)}(z)=I_{(nu)}(z)+\frac{2}{(pi)}\sin((pi)(nu))K_{(nu)}(z)$$ may be used (for the Bessel function $K_{(nu)}(z)$, see S18DCF).

When N is greater than 1, extra values of $I_{(nu)}(z)$ are computed using recurrence relations.

For very large |z| or ((nu)+N-1), argument reduction will cause total loss of accuracy, and so no computation is performed. For slightly smaller |z| or ((nu)+N-1), the computation is performed but results are accurate to less than half of machine precision. If Re(z) is too large and the unscaled function is required, there is a risk of overflow and so no computation is performed. In all the above cases, a warning is given by the routine.

4. References

[1] Abramowitz M and Stegun I A (1968) Handbook of Mathematical Functions. Dover Publications.

[2] Amos D E (1986) Algorithm 644: A Portable Package for Bessel Functions of a Complex Argument and Nonnegative Order. ACM Trans. Math. Softw. 12 265--273.

5. Parameters

1: FNU -- DOUBLE PRECISION Input

On entry: the order, (nu), of the first member of the sequence of functions. Constraint: FNU >= 0.0.

2: Z -- COMPLEX(KIND(1.0D0)) Input

On entry: the argument z of the functions.

3: N -- INTEGER Input

On entry: the number, N, of members required in the sequence $I_{(nu)}(z),I_{(nu)+1}(z),...,I_{(nu)+N-1}(z)$. Constraint: N >= 1.

4: SCALE -- CHARACTER*1 Input

```
     On entry: the scaling option.
          If SCALE = 'U', the results are returned unscaled.

          If SCALE = 'S', the results are returned scaled by the
                  -|Rez|
          factor e      .
     Constraint: SCALE = 'U' or 'S'.

 5:  CY(N) -- COMPLEX(KIND(1.0D)) array                    Output
     On exit: the N required function values: CY(i) contains
     I        (z), for i=1,2,...,N.
      (nu)+i-1

 6:  NZ -- INTEGER                                         Output
     On exit: the number of components of CY that are set to zero
     due to underflow.

     If NZ > 0, then elements CY(N-NZ+1),CY(N-NZ+2),...,CY(N) are
     set to zero.

 7:  IFAIL -- INTEGER                                Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

IFAIL= 1
     On entry FNU < 0.0,

     or       N < 1,

     or       SCALE /= 'U' or 'S'.

IFAIL= 2
     No computation has been performed due to the likelihood of
     overflow, because real(Z) is greater than a machine-
     dependent threshold value (given in the Users' Note for
     your implementation). This error exit can only occur when
     SCALE = 'U'.

IFAIL= 3
```

The computation has been performed, but the errors due to argument reduction in elementary functions make it likely that the results returned by S18DEF are accurate to less than half of machine precision. This error exit may occur when either ABS(Z) or FNU + N - 1 is greater than a machine-dependent threshold value (given in the Users' Note for your implementation).

IFAIL= 4

No computation has been performed because the errors due to argument reduction in elementary functions mean that all precision in results returned by S18DEF would be lost. This error exit may occur when either ABS(Z) or FNU + N - 1 is greater than a machine-dependent threshold value (given in the Users' Note for your implementation).

IFAIL= 5

No results are returned because the algorithm termination condition has not been met. This may occur because the parameters supplied to S18DEF would have caused overflow or underflow.

7. Accuracy

All constants in subroutine S18DEF are given to approximately 18 digits of precision. Calling the number of digits of precision in the floating-point arithmetic being used t, then clearly the maximum number of correct digits in the results obtained is limited by p=min(t,18). Because of errors in argument reduction when computing elementary functions inside S18DEF, the actual number of correct digits is limited, in general, by p-s, where s~max(1,|log $_{10}$ |z||,|log $_{10}$ (nu)|) represents the number of digits lost due to the argument reduction. Thus the larger the values of |z| and (nu), the less the precision in the result. If S18DEF is called with N>1, then computation of function values via recurrence may lead to some further small loss of accuracy.

If function values which should nominally be identical are computed by calls to S18DEF with different base values of (nu) and different N, the computed values may not agree exactly. Empirical tests with modest values of (nu) and z have shown that the discrepancy is limited to the least significant 3-4 digits of precision.

8. Further Comments

The time taken by the routine for a call of S18DEF is approximately proportional to the value of N, plus a constant. In general it is much cheaper to call S18DEF with N greater than 1,

```
          rather than to make N separate calls to S18DEF.

          Paradoxically, for some values of z and (nu), it is cheaper to
          call S18DEF with a larger value of N than is required, and then
          discard the extra function values returned. However, it is not
          possible to state the precise circumstances in which this is
          likely to occur. It is due to the fact that the base value used
          to start recurrence may be calculated in different regions for
          different N, and the costs in each region may differ greatly.

          Note that if the function required is I (x) or I (x), i.e.,
                                                 0       1
          (nu)=0.0 or 1.0, where x is real and positive, and only a single
          function value is required, then it may be much cheaper to call
          S18AEF, S18AFF, S18CEF(*) or S18CFF(*), depending on whether a
          scaled result is required or not.

          9. Example

          The example program prints a caption and then proceeds to read
          sets of data from the input data stream. The first datum is a
          value for the order FNU, the second is a complex value for the
          argument, Z, and the third is a value for the parameter SCALE.
          The program calls the routine with N = 2 to evaluate the function
          for orders FNU and FNU + 1, and it prints the results. The
          process is repeated until the end of the input data stream is
          encountered.

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Returns a value for the Kelvin function ber x

— nags.ht —

```
\begin{page}{manpageXXs19aaf}{NAG Documentation: s19aaf}
\beginscroll
\begin{verbatim}
```

```
S19AAF(3NAG)      Foundation Library (12/10/92)      S19AAF(3NAG)



     S19 -- Approximations of Special Functions                 S19AAF
             S19AAF -- NAG Foundation Library Routine Document

     Note: Before using this routine, please read the Users' Note for
     your implementation to check implementation-dependent details.
     The symbol (*) after a NAG routine name denotes a routine that is
     not included in the Foundation Library.

     1. Purpose

     S19AAF returns a value for the Kelvin function ber x via the
     routine name.

     2. Specification

            DOUBLE PRECISION FUNCTION S19AAF (X, IFAIL)
            INTEGER          IFAIL
            DOUBLE PRECISION X

     3. Description

     This routine evaluates an approximation to the Kelvin function
     berx.

     Note: ber(-x)=berx, so the approximation need only consider
     x>=0.0.

     The routine is based on several Chebyshev expansions:

     For 0<=x<=5,

                           --'                 ( x)4
                     berx= >   a T (t) with t=2( -) -1;
                           --   r r            ( 5)
                           r=0

     For x>5,



                  x/\/2
                 e      [(   1    )            1              ]
           berx= --------[(1+ -a(t))cos(alpha)+ -b(t)sin(alpha)]
                        [(   x    )            x              ]
                 \/2(pi)x
```

```
              -x/\/2
             e      [(   1    )           1             ]
          + --------[(1+ -c(t))sin(beta)+ -d(t)cos(beta)]
                    [(   x    )           x             ]
            \/2(pi)x

                 x   (pi)           x   (pi)
where (alpha)= ---- ----, (beta)= ---+ ----,
                      8                  8
                \/2               /2
```

and a(t), b(t), c(t), and d(t) are expansions in the variable

```
   10
t= ---1.
   x
```

When x is sufficiently close to zero, the result is set directly to ber 0=1.0.

For large x, there is a danger of the result being totally inaccurate, as the error amplification factor grows in an essentially exponential manner; therefore the routine must fail.

4. References

[1] Abramowitz M and Stegun I A (1968) Handbook of Mathematical Functions. Dover Publications.

5. Parameters

1: X -- DOUBLE PRECISION Input
On entry: the argument x of the function.

2: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1
On entry ABS(X) is too large for an accurate result to be

```
     returned. On soft failure, the routine returns zero.

7. Accuracy

Since the function is oscillatory, the absolute error rather than
the relative error is important. Let E be the absolute error in
the result and (delta) be the relative error in the argument. If
(delta) is somewhat larger than the machine precision, then we
have:

                   |  x             |
                E~=| ---(ber x+bei x)|(delta)
                   |        1     1  |
                   | \/2             |

(provided E is within machine bounds).

For small x the error amplification is insignificant and thus the
absolute error is effectively bounded by the machine precision.

For medium and large x, the error behaviour is oscillatory and


                          /   x   x/\/2
its amplitude grows like   /  -----e    . Therefore it is not
                         \/   2(pi)

possible to calculate the function with any accuracy when


    x/\/2  /2(pi)
\/xe     > -------. Note that this value of x is much smaller than
           (delta)

the minimum value of x for which the function overflows.

8. Further Comments

For details of the time taken by the routine see the Users' Note
for your implementation.

9. Example

The example program reads values of the argument x from a file,
evaluates the function at each value of x and prints the results.

The example program is not reproduced here. The source code for
all example programs is distributed with the NAG Foundation
Library software and should be available on-line.
```
\end{verbatim}

```
\endscroll
\end{page}
```

———-

Returns a value for the Kelvin function bei x

— nags.ht —

```
\begin{page}{manpageXXs19abf}{NAG Documentation: s19abf}
\beginscroll
\begin{verbatim}



     S19ABF(3NAG)      Foundation Library (12/10/92)      S19ABF(3NAG)



          S19 -- Approximations of Special Functions                 S19ABF
                  S19ABF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          S19ABF returns a value for the Kelvin function bei x via the
          routine name.

          2. Specification

                 DOUBLE PRECISION FUNCTION S19ABF (X, IFAIL)
                 INTEGER          IFAIL
                 DOUBLE PRECISION X

          3. Description

          This routine evaluates an approximation to the Kelvin function
          beix.

          Note: bei(-x)=beix, so the approximation need only consider
          x>=0.0.

          The routine is based on several Chebyshev expansions:
```

For $0<=x<=5$,

$$\mathrm{bei}\,x=\frac{x^2}{4}\sum_{r=0}^{\prime} a_r T_r(t), \quad \text{with} \quad t=2\left(\frac{x}{5}\right)^4-1;$$

For $x>5$,

$$\mathrm{bei}\,x=\frac{e^{x/\sqrt{2}}}{\sqrt{2(\pi)x}}\left[\left(1+\frac{1}{x}a(t)\right)\sin(\alpha)-\frac{1}{x}b(t)\cos(\alpha)\right]$$

$$+\frac{e^{x/\sqrt{2}}}{\sqrt{2(\pi)x}}\left[\left(1+\frac{1}{x}c(t)\right)\cos(\beta)-\frac{1}{x}d(t)\sin(\beta)\right]$$

where $(\alpha)=\frac{x}{\sqrt{2}}-\frac{(\pi)}{8}$, $(\beta)=\frac{x}{\sqrt{2}}+\frac{(\pi)}{8}$,

and $a(t)$, $b(t)$, $c(t)$, and $d(t)$ are expansions in the variable $t=\frac{10}{x}-1$.

When x is sufficiently close to zero, the result is computed as $\mathrm{bei}\,x=\frac{x^2}{4}$. If this result would underflow, the result returned is $\mathrm{bei}x=0.0$.

For large x, there is a danger of the result being totally inaccurate, as the error amplification factor grows in an essentially exponential manner; therefore the routine must fail.

4. References

[1] Abramowitz M and Stegun I A (1968) Handbook of Mathematical Functions. Dover Publications.

5. Parameters

1: X -- DOUBLE PRECISION Input
On entry: the argument x of the function.

2: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1
On entry ABS(X) is too large for an accurate result to be returned. On soft failure, the routine returns zero.

7. Accuracy

Since the function is oscillatory, the absolute error rather than the relative error is important. Let E be the absolute error in the function, and (delta) be the relative error in the argument. If (delta) is somewhat larger than the machine precision, then we have:

```
    |  x              |
 E~=| ---(-ber x+bei x)|(delta)
    |        1     1  |
    | \/2             |
```

(provided E is within machine bounds).

For small x the error amplification is insignificant and thus the absolute error is effectively bounded by the machine precision.

For medium and large x, the error behaviour is oscillatory and

```
                          /   x   x\/2
its amplitude grows like   /  -----e    . Therefore it is
                         \/   2(pi)
```

impossible to calculate the functions with any accuracy when

```
             x/\/2 /2(pi)
        \/xe      > -------. Note that this value of x is much smaller than
                    (delta)

        the minimum value of x for which the function overflows.

        8. Further Comments

        For details of the time taken by the routine see the Users' Note
        for your implementation.

        9. Example

        The example program reads values of the argument x from a file,
        evaluates the function at each value of x and prints the results.

        The example program is not reproduced here. The source code for
        all example programs is distributed with the NAG Foundation
        Library software and should be available on-line.
\end{verbatim}
\endscroll
\end{page}
```

Returns a value for the Kelvin function ker x

— nags.ht —

```
\begin{page}{manpageXXs19acf}{NAG Documentation: s19acf}
\beginscroll
\begin{verbatim}



     S19ACF(3NAG)      Foundation Library (12/10/92)      S19ACF(3NAG)



        S19 -- Approximations of Special Functions                 S19ACF
                S19ACF -- NAG Foundation Library Routine Document

        Note: Before using this routine, please read the Users' Note for
        your implementation to check implementation-dependent details.
        The symbol (*) after a NAG routine name denotes a routine that is
```

not included in the Foundation Library.

1. Purpose

S19ACF returns a value for the Kelvin function ker x, via the routine name.

2. Specification

```
       DOUBLE PRECISION FUNCTION S19ACF (X, IFAIL)
       INTEGER          IFAIL
       DOUBLE PRECISION X
```

3. Description

This routine evaluates an approximation to the Kelvin function ker x.

Note: for $x<0$ the function is undefined and at $x=0$ it is infinite so we need only consider $x>0$.

The routine is based on several Chebyshev expansions:

For $0<x<=1$,

$$\mathrm{ker}\, x=-f(t)\log x+\frac{\pi}{16}x^2g(t)+y(t)$$

where $f(t)$, $g(t)$ and $y(t)$ are expansions in the variable $t=2x^4-1$;

For $1<x<=3$,

$$\mathrm{ker}\, x=\exp\left(-\frac{11}{16}x\right)q(t)$$

where $q(t)$ is an expansion in the variable $t=x-2$;

For $x>3$,

$$\mathrm{ker}\, x=\sqrt{\frac{\pi}{2x}}e^{-x/\sqrt{2}}\left[\left(1+\frac{1}{x}c(t)\right)\cos(\beta)-\frac{1}{x}d(t)\sin(\beta)\right]$$

where $\beta=\frac{x}{\sqrt{2}}+\frac{\pi}{8}$, and $c(t)$ and $d(t)$ are expansions in the variable $t=\frac{6}{x}-1$.

When x is sufficiently close to zero, the result is computed as

$$\mathrm{ker}\,x=-\gamma-\log\left(\frac{x}{2}\right)+\left(\pi-\frac{3}{8}x^2\right)\frac{x^2}{16}$$

and when x is even closer to zero, simply as

$$\mathrm{ker}\,x=-\gamma-\log\left(\frac{x}{2}\right).$$

For large x, ker x is asymptotically given by $\sqrt{\frac{\pi}{2x}}e^{-x/\sqrt{2}}$ and this becomes so small that it cannot be computed without underflow and the routine fails.

4. References

[1] Abramowitz M and Stegun I A (1968) Handbook of Mathematical Functions. Dover Publications.

5. Parameters

1: X -- DOUBLE PRECISION Input
On entry: the argument x of the function. Constraint: X > 0.

2: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1
: On entry X is too large, the result underflows. On soft failure, the routine returns zero.

IFAIL= 2
: On entry X <= 0, the function is undefined. On soft failure the routine returns zero.

7. Accuracy

Let E be the absolute error in the result, (epsilon) be the relative error in the result and (delta) be the relative error in the argument. If (delta) is somewhat larger than the machine precision, then we have:

$$E \simeq \left| \frac{x}{\sqrt{2}} (\mathrm{ker}_1 x + \mathrm{kei}_1 x) \right| (\delta),$$

$$(\epsilon) \simeq \left| \frac{x}{\sqrt{2}} \frac{\mathrm{ker}_1 x + \mathrm{kei}_1 x}{\mathrm{ker}\, x} \right| (\delta).$$

For very small x, the relative error amplification factor is approximately given by $\frac{1}{|\log x|}$, which implies a strong attenuation of relative error. However, (epsilon) in general cannot be less than the machine precision.

For small x, errors are damped by the function and hence are limited by the machine precision.

For medium and large x, the error behaviour, like the function itself, is oscillatory, and hence only the absolute accuracy for the function can be maintained. For this range of x, the

amplitude of the absolute error decays like $\sqrt{\frac{(\pi)x}{2}}e^{-x/\sqrt{2}}$

which implies a strong attenuation of error. Eventually, ker x,

/ (pi) -x/\/2

```
          which asymptotically behaves like   /  ----e     , becomes so
                                            \/    2x

          small that it cannot be calculated without causing underflow, and
          the routine returns zero. Note that for large x the errors are
          dominated by those of the Fortran intrinsic function EXP.

          8. Further Comments

          Underflow may occur for a few values of x close to the zeros of
          ker x, below the limit which causes a failure with IFAIL = 1.

          9. Example

          The example program reads values of the argument x from a file,
          evaluates the function at each value of x and prints the results.

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.
\end{verbatim}
\endscroll
\end{page}
```

Returns a value for the Kelvin function keix

— nags.ht —

```
\begin{page}{manpageXXs19adf}{NAG Documentation: s19adf}
\beginscroll
\begin{verbatim}



     S19ADF(3NAG)      Foundation Library (12/10/92)      S19ADF(3NAG)



          S19 -- Approximations of Special Functions                S19ADF
                  S19ADF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.
```

1. Purpose

S19ADF returns a value for the Kelvin function keix via the routine name.

2. Specification

```
DOUBLE PRECISION FUNCTION S19ADF (X, IFAIL)
INTEGER          IFAIL
DOUBLE PRECISION X
```

3. Description

This routine evaluates an approximation to the Kelvin function keix.

Note: for $x<0$ the function is undefined, so we need only consider $x>=0$.

The routine is based on several Chebyshev expansions:

For $0<=x<=1$,

$$keix=-\frac{\pi}{4}f(t)+\frac{x^2}{4}[-g(t)\log x+v(t)]$$

where $f(t)$, $g(t)$ and $v(t)$ are expansions in the variable $t=2x^4-1$;

For $1<x<=3$,

$$keix=\exp\left(-\frac{9}{8}x\right)u(t)$$

where $u(t)$ is an expansion in the variable $t=x-2$;

For $x>3$,

$$keix=\sqrt{\frac{\pi}{2x}}e^{-x/\sqrt{2}}\left[\left(1+\frac{1}{x}\right)c(t)\sin(\beta)+\frac{1}{x}d(t)\cos(\beta)\right]$$

x (pi)

```
where (beta)= ---+ ----, and c(t) and d(t) are expansions in the
                    8
              \/2
```

variable $t=\frac{6}{x}-1$.

For $x<0$, the function is undefined, and hence the routine fails and returns zero.

When x is sufficiently close to zero, the result is computed as

$$keix=-\frac{\pi}{4}+\left(1-\gamma-\log\left(\frac{x}{2}\right)\right)\frac{x^2}{4}$$

and when x is even closer to zero simply as

$$keix=-\frac{\pi}{4}.$$

For large x, keix is asymptotically given by $\sqrt{\frac{\pi}{2x}}e^{-x/\sqrt{2}}$ and this becomes so small that it cannot be computed without underflow and the routine fails.

4. References

[1] Abramowitz M and Stegun I A (1968) Handbook of Mathematical Functions. Dover Publications.

5. Parameters

1: X -- DOUBLE PRECISION Input
On entry: the argument x of the function. Constraint: X >= 0.

2: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL= 1
     On entry X is too large, the result underflows. On soft
     failure, the routine returns zero.

IFAIL= 2
     On entry X < 0, the function is undefined. On soft failure
     the routine returns zero.

7. Accuracy

Let E be the absolute error in the result, and (delta) be the
relative error in the argument. If (delta) is somewhat larger
than the machine representation error, then we have:

```
                   |  x               |
                E~=| ---(-ker  x+kei x)|(delta).
                   |          1     1  |
                   | \/2               |
```

For small x, errors are attenuated by the function and hence are
limited by the machine precision.

For medium and large x, the error behaviour, like the function
itself, is oscillatory and hence only absolute accuracy of the
function can be maintained. For this range of x, the amplitude of

```
                                  / (pi)x -x/\/2
the absolute error decays like   /  -----e      , which implies a
                               \/     2
```

strong attenuation of error. Eventually, keix, which is

```
                           / (pi) -x/\/2
asymptotically given by   /  ----e      , becomes so small that it
                        \/    2x
```

cannot be calculated without causing underflow and therefore the
routine returns zero. Note that for large x, the errors are
dominated by those of the Fortran intrinsic function EXP.

8. Further Comments

Underflow may occur for a few values of x close to the zeros of

```
         keix, below the limit which causes a failure with IFAIL = 1.

         9. Example

         The example program reads values of the argument x from a file,
         evaluates the function at each value of x and prints the results.

         The example program is not reproduced here. The source code for
         all example programs is distributed with the NAG Foundation
         Library software and should be available on-line.
\end{verbatim}
\endscroll
\end{page}
```

—————

Returns a value for the Fresnel Integral $S(x)$

— nags.ht —

```
\begin{page}{manpageXXs20acf}{NAG Documentation: s20acf}
\beginscroll
\begin{verbatim}



     S20ACF(3NAG)      Foundation Library (12/10/92)      S20ACF(3NAG)



         S20 -- Approximations of Special Functions                 S20ACF
                 S20ACF -- NAG Foundation Library Routine Document

         Note: Before using this routine, please read the Users' Note for
         your implementation to check implementation-dependent details.
         The symbol (*) after a NAG routine name denotes a routine that is
         not included in the Foundation Library.

         1. Purpose

         S20ACF returns a value for the Fresnel Integral S(x), via the
         routine name.

         2. Specification

                DOUBLE PRECISION FUNCTION S20ACF (X, IFAIL)
                INTEGER          IFAIL
```

```
      DOUBLE PRECISION X
```

3. Description

This routine evaluates an approximation to the Fresnel Integral

$$S(x)=\int_0^x \sin\left(\frac{\pi}{2}t^2\right)dt.$$

Note: $S(x)=-S(-x)$, so the approximation need only consider $x>=0.0$

The routine is based on three Chebyshev expansions:

For $0<x<=3$,

$$S(x)=x^3 {\sum_{r=0}}' a_r T_r(t) \text{ , with } t=2\left(\frac{x}{3}\right)^4-1;$$

For $x>3$,

$$S(x)=\frac{1}{2}-\frac{f(x)}{x}\cos\left(\frac{\pi}{2}x^2\right)-\frac{g(x)}{x^3}\sin\left(\frac{\pi}{2}x^2\right),$$

where $f(x) = {\sum_{r=0}}' b_r T_r(t)$,

and $g(x) = {\sum_{r=0}}' c_r T_r(t)$, with $t=2\left(\frac{3}{x}\right)^4-1$.

For small x, $S(x)\sim=\frac{\pi}{6}x^3$. This approximation is used when x is sufficiently small for the result to be correct to machine precision. For very small x, this approximation would underflow; the result is then set exactly to zero.

For large x, $f(x)\sim=\frac{1}{\pi}$ and $g(x)\sim=\frac{1}{2}$. Therefore for moderately large x, when $\frac{1}{\pi^2 x^3}$ is negligible compared with $\frac{1}{2}$, the second term in the approximation for $x>3$ may be dropped. For very large x, when $\frac{1}{\pi x}$ becomes negligible, $S(x) \simeq \frac{1}{2}$. However there will be considerable difficulties in calculating $\cos\left(\frac{\pi}{2}x^2\right)$ accurately before this final limiting value can be used. Since $\cos\left(\frac{\pi}{2}x^2\right)$ is periodic, its value is essentially determined by the fractional part of x^2. If $x^2=N+\theta$ where N is an integer and $0\le\theta<1$, then $\cos\left(\frac{\pi}{2}x^2\right)$ depends on θ and on N modulo 4. By exploiting this fact, it is possible to retain significance in the calculation of $\cos\left(\frac{\pi}{2}x^2\right)$ either all the way to the very large x limit, or at least until the integer part of $\frac{x}{2}$ is equal to the maximum integer allowed on the machine.

4. References

[1] Abramowitz M and Stegun I A (1968) Handbook of Mathematical Functions. Dover Publications.

5. Parameters

1: X -- DOUBLE PRECISION Input
On entry: the argument x of the function.

2: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

There are no failure exits from this routine. The parameter IFAIL has been included for consistency with other routines in this chapter.

7. Accuracy

Let (delta) and (epsilon) be the relative errors in the argument and result respectively.

If (delta) is somewhat larger than the machine precision (i.e., if (delta) is due to data errors etc), then (epsilon) and (delta) are approximately related by:

$$\epsilon \simeq \left| \frac{x\sin\left(\frac{\pi}{2}x^2\right)}{S(x)} \right| \delta .$$

Figure 1 shows the behaviour of the error amplification factor

$$\left| \frac{x\sin\left(\frac{\pi}{2}x^2\right)}{S(x)} \right| .$$

Figure 1
Please see figure in printed Reference Manual

However if (delta) is of the same order as the machine precision, then rounding errors could make (epsilon) slightly larger than the above relation predicts.

For small x, (epsilon)~=3(delta) and hence there is only moderate amplification of relative error. Of course for very small x where the correct result would underflow and exact zero is returned, relative error-control is lost.

For moderately large values of x,

$$|\epsilon| \simeq \left|2x\sin\left(\frac{\pi}{2}x^2\right)\right| |\delta|$$

and the result will be subject to increasingly large amplification of errors. However the above relation breaks down

```
          for large values of x (i.e., when  -- is of the order of the
                                              2
                                             x
          machine precision in this region the relative error in the result
                                        2
          is essentially bounded by  -----).
                                     (pi)x

          Hence the effects of error amplification are limited and at worst
          the relative error loss should not exceed half the possible
          number of significant figures.

          8. Further Comments

          None.

          9. Example

          The example program reads values of the argument x from a file,
          evaluates the function at each value of x and prints the results.

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

———-

Returns a value for the Fresnel Integral $C(x)$

— nags.ht —

```
\begin{page}{manpageXXs20adf}{NAG Documentation: s20adf}
\beginscroll
\begin{verbatim}



     S20ADF(3NAG)      Foundation Library (12/10/92)      S20ADF(3NAG)



          S20 -- Approximations of Special Functions                 S20ADF
                  S20ADF -- NAG Foundation Library Routine Document
```

Note: Before using this routine, please read the Users' Note for your implementation to check implementation-dependent details. The symbol (*) after a NAG routine name denotes a routine that is not included in the Foundation Library.

1. Purpose

S20ADF returns a value for the Fresnel Integral C(x), via the routine name.

2. Specification

```
       DOUBLE PRECISION FUNCTION S20ADF (X, IFAIL)
       INTEGER          IFAIL
       DOUBLE PRECISION X
```

3. Description

This routine evaluates an approximation to the Fresnel Integral

$$C(x)=\int_0^x \cos\left(\frac{\pi}{2}t^2\right)dt.$$

Note: $C(x)=-C(-x)$, so the approximation need only consider $x>=0.0$

The routine is based on three Chebyshev expansions:

For $0<x<=3$,

$$C(x)=x\sum_{r=0}{}' a_r T_r(t) \text{ , with } t=2\left(\frac{x}{3}\right)^4-1;$$

For $x>3$,

$$C(x)=\frac{1}{2}+\frac{f(x)}{x}\sin\left(\frac{\pi}{2}x^2\right)-\frac{g(x)}{x^3}\cos\left(\frac{\pi}{2}x^2\right),$$

where $f(x) = \sum_{r=0}{}' b_r T_r(t)$, and $g(x) = \sum_{r=0}' c_r T_r(t)$, with $t=2\left(\frac{3}{x}\right)^4-1$.

For small x, $C(x) \simeq x$. This approximation is used when x is sufficiently small for the result to be correct to machine precision.

For large x, $f(x) \simeq \frac{1}{\pi x}$ and $g(x) \simeq \frac{1}{\pi^2 x^3}$. Therefore for moderately large x, when $\frac{1}{\pi^2 x^3}$ is negligible compared with $\frac{1}{2}$, the second term in the approximation for x>3 may be dropped. For very large x, when $\frac{1}{\pi x}$ becomes negligible, $C(x) \simeq \frac{1}{2}$. However there will be considerable difficulties in calculating $\sin\left(\frac{\pi}{2}x^2\right)$ accurately before this final limiting value can be used. Since $\sin\left(\frac{\pi}{2}x^2\right)$ is periodic, its value is essentially determined by the fractional part of x^2. If $x^2=N+\theta$, where N is an integer and $0 \le \theta < 1$, then $\sin\left(\frac{\pi}{2}x^2\right)$ depends on θ and on N modulo 4. By exploiting this fact, it is possible to retain some significance in the calculation of $\sin\left(\frac{\pi}{2}x^2\right)$ either all the way to the very large x limit, or at least until the integer part of $\frac{x}{2}$ is equal to the maximum integer allowed on the machine.

4. References

[1] Abramowitz M and Stegun I A (1968) Handbook of Mathematical Functions. Dover Publications.

5. Parameters

```
 1:  X -- DOUBLE PRECISION                              Input
     On entry: the argument x of the function.

 2:  IFAIL -- INTEGER                            Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).
```

6. Error Indicators and Warnings

There are no failure exits from this routine. The parameter IFAIL has been included for consistency with other routines in this chapter.

7. Accuracy

Let (delta) and (epsilon) be the relative errors in the argument and result respectively.

If (delta) is somewhat larger than the machine precision (i.e if (delta) is due to data errors etc), then (epsilon) and (delta) are approximately related by:

```
                          |     ( (pi) 2)|
                          | xcos( ----x )|
                          |     (  2    )|
               (epsilon)~=| -------------|(delta).
                          |     C(x)     |
```

Figure 1 shows the behaviour of the error amplification factor

```
|     ( (pi) 2)|
| xcos( ----x )|
|     (  2    )|
| -------------|.
|     C(x)     |
```

Figure 1
Please see figure in printed Reference Manual

However if (delta) is of the same order as the machine precision, then rounding errors could make (epsilon) slightly larger than the above relation predicts.

For small x, (epsilon)~=(delta) and there is no amplification of relative error.

```
          For moderately large values of x,

                                |     ( (pi) 2)|
                       |(epsilon)|~=|2xcos( ----x )||(delta)|
                                |     (  2    )|

          and the result will be subject to increasingly large
          amplification of errors. However the above relation breaks down
                                            1
          for large values of x (i.e., when  -- is of the order of the
                                             2
                                            x
          machine precision); in this region the relative error in the
                                             2
          result is essentially bounded by  -----).
                                           (pi)x

          Hence the effects of error amplification are limited and at worst
          the relative error loss should not exceed half the possible
          number of significant figures.

          8. Further Comments

          None.

          9. Example

          The example program reads values of the argument x from a file,
          evaluates the function at each value of x and prints the results.

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.
\end{verbatim}
\endscroll
\end{page}
```

———-

Returns a value of an elementary integral

— nags.ht —

```
\begin{page}{manpageXXs21baf}{NAG Documentation: s21baf}
\beginscroll
\begin{verbatim}
```

```
S21BAF(3NAG)      Foundation Library (12/10/92)      S21BAF(3NAG)



     S21 -- Approximations of Special Functions                S21BAF
             S21BAF -- NAG Foundation Library Routine Document

     Note: Before using this routine, please read the Users' Note for
     your implementation to check implementation-dependent details.
     The symbol (*) after a NAG routine name denotes a routine that is
     not included in the Foundation Library.

     1. Purpose

     S21BAF returns a value of an elementary integral, which occurs as
     a degenerate case of an elliptic integral of the first kind, via
     the routine name.

     2. Specification

            DOUBLE PRECISION FUNCTION S21BAF (X, Y, IFAIL)
            INTEGER          IFAIL
            DOUBLE PRECISION X, Y

     3. Description

     This routine calculates an approximate value for the integral

                                 infty
                               1 /         dt
                       R (x,y)= - |      ----------
                        C      2 /
                                 0     \/t+x(t+y)

     where x>=0 and y/=0.

     This function, which is related to the logarithm or inverse
     hyperbolic functions for y<x and to inverse circular functions if
     x<y, arises as a degenerate form of the elliptic integral of the
     first kind. If y<0, the result computed is the Cauchy principal
     value of the integral.

     The basic algorithm, which is due to Carlson [2] and [3], is to
     reduce the arguments recursively towards their mean by the
     system:

         x =x,
```

$_0$

$$y_0 = y$$

$$\mu_n = (x_n + 2y_n)/3,$$

$$S_n = (y_n - x_n)/3\mu_n$$

$$\lambda_n = y_n + 2\sqrt{x_n y_n}$$

$$x_{n+1} = (x_n + \lambda_n)/4,$$

$$y_{n+1} = (y_n + \lambda_n)/4.$$

The quantity $|S_n|$ for $n=0,1,2,3,\ldots$ decreases with increasing n, eventually $|S_n| \sim 1/4^n$. For small enough S_n the required function value can be approximated by the first few terms of the Taylor series about the mean. That is

$$R_C(x,y) = \left(1 + \frac{3S_n^2}{10} + \frac{S_n^3}{7} + \frac{3S_n^4}{8} + \frac{9S_n^5}{22}\right) / \sqrt{\mu_n}.$$

The truncation error involved in using this approximation is bounded by $16|S_n|^6/(1-2|S_n|)$ and the recursive process is stopped when S_n is small enough for this truncation error to be negligible compared to the machine precision.

Within the domain of definition, the function value is itself representable for all representable values of its arguments. However, for values of the arguments near the extremes the above algorithm must be modified so as to avoid causing underflows or overflows in intermediate steps. In extreme regions arguments are pre-scaled away from the extremes and compensating scaling of the result is done before returning to the calling program.

4. References

[1] Abramowitz M and Stegun I A (1968) Handbook of Mathematical Functions. Dover Publications.

[2] Carlson B C (1978) Computing Elliptic Integrals by Duplication. (Preprint) Department of Physics, Iowa State University.

[3] Carlson B C (1988) A Table of Elliptic Integrals of the Third Kind. Math. Comput. 51 267--280.

5. Parameters

1: X -- DOUBLE PRECISION Input

2: Y -- DOUBLE PRECISION Input
On entry: the arguments x and y of the function, respectively. Constraint: X >= 0.0 and Y /= 0.0.

3: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL= 1
On entry X < 0.0; the function is undefined.

IFAIL= 2
On entryY = 0.0; the function is undefined.

On soft failure the routine returns zero.

7. Accuracy

In principle the routine is capable of producing full machine precision. However round-off errors in internal arithmetic will result in slight loss of accuracy. This loss should never be

```
          excessive as the algorithm does not involve any significant
          amplification of round-off error. It is reasonable to assume that
          the result is accurate to within a small multiple of the machine
          precision.

          8. Further Comments

          Users should consult the Chapter Introduction which shows the
          relationship of this function to the classical definitions of the
          elliptic integrals.

          9. Example

          This example program simply generates a small set of non-extreme
          arguments which are used with the routine to produce the table of
          low accuracy results.

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Value of the symmetrised elliptic integral of first kind

— nags.ht —

```
\begin{page}{manpageXXs21bbf}{NAG Documentation: s21bbf}
\beginscroll
\begin{verbatim}



     S21BBF(3NAG)      Foundation Library (12/10/92)      S21BBF(3NAG)



          S21 -- Approximations of Special Functions                S21BBF
                  S21BBF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
```

not included in the Foundation Library.

1. Purpose

S21BBF returns a value of the symmetrised elliptic integral of the first kind, via the routine name.

2. Specification

```
DOUBLE PRECISION FUNCTION S21BBF (X, Y, Z, IFAIL)
INTEGER          IFAIL
DOUBLE PRECISION X, Y, Z
```

3. Description

This routine calculates an approximation to the integral

$$R_F(x,y,z) = \frac{1}{2}\int_0^{\infty} \frac{dt}{\sqrt{(t+x)(t+y)(t+z)}}$$

where x, y, z>=0 and at most one is zero.

The basic algorithm, which is due to Carlson [2] and [3], is to reduce the arguments recursively towards their mean by the rule:

$x_0 = \min(x,y,z), z_0 = \max(x,y,z)$,

y_0 = remaining third intermediate value argument.

(This ordering, which is possible because of the symmetry of the function, is done for technical reasons related to the avoidance of overflow and underflow.)

$$(mu)_n = (x_n + y_n + 3z_n)/3$$

$$X_n = (1-x_n)/(mu)_n$$

$$Y_n = (1-y_n)/(mu)_n$$

$$Z_n = (1-z_n)/(mu)_n$$

$$\lambda_n = \sqrt{x_n y_n} + \sqrt{y_n z_n} + \sqrt{z_n x_n}$$

$$x_{n+1} = (x_n + \lambda_n)/4$$

$$y_{n+1} = (y_n + \lambda_n)/4$$

$$z_{n+1} = (z_n + \lambda_n)/4$$

$\epsilon_n = \max(|X_n|,|Y_n|,|Z_n|)$ and the function may be approximated adequately by a 5th order power series:

$$R_F(x,y,z) = \frac{1}{\sqrt{\mu_n}}\left(1 - \frac{E_2}{10} + \frac{E_2^2}{24} - \frac{3E_2E_3}{44} + \frac{E_3}{14}\right)$$

where $E_2 = X_nY_n + Y_nZ_n + Z_nX_n$, $E_3 = X_nY_nZ_n$.

The truncation error involved in using this approximation is bounded by $\epsilon_n^6/4(1-\epsilon_n)$ and the recursive process is stopped when this truncation error is negligible compared with the machine precision.

Within the domain of definition, the function value is itself representable for all representable values of its arguments. However, for values of the arguments near the extremes the above algorithm must be modified so as to avoid causing underflows or overflows in intermediate steps. In extreme regions arguments are pre-scaled away from the extremes and compensating scaling of the result is done before returning to the calling program.

4. References

[1] Abramowitz M and Stegun I A (1968) Handbook of Mathematical Functions. Dover Publications.

[2] Carlson B C (1978) Computing Elliptic Integrals by

Duplication. (Preprint) Department of Physics, Iowa State University.

[3] Carlson B C (1988) A Table of Elliptic Integrals of the Third Kind. Math. Comput. 51 267--280.

5. Parameters

1: X -- DOUBLE PRECISION Input

2: Y -- DOUBLE PRECISION Input

3: Z -- DOUBLE PRECISION Input
On entry: the arguments x, y and z of the function.
Constraint: X, Y, Z >= 0.0 and only one of X, Y and Z may be zero.

4: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL= 1
On entry one or more of X, Y and Z is negative; the function is undefined.

IFAIL= 2
On entry two or more of X, Y and Z are zero; the function is undefined.

On soft failure, the routine returns zero.

7. Accuracy

In principle the routine is capable of producing full machine precision. However round-off errors in internal arithmetic will result in slight loss of accuracy. This loss should never be excessive as the algorithm does not involve any significant amplification of round-off error. It is reasonable to assume that the result is accurate to within a small multiple of the machine

```
         precision.

         8. Further Comments

         Users should consult the Chapter Introduction which shows the
         relationship of this function to the classical definitions of the
         elliptic integrals.

         If two arguments are equal, the function reduces to the
         elementary integral R , computed by S21BAF.
                               C

         9. Example

         This example program simply generates a small set of non-extreme
         arguments which are used with the routine to produce the table of
         low accuracy results.

         The example program is not reproduced here. The source code for
         all example programs is distributed with the NAG Foundation
         Library software and should be available on-line.
\end{verbatim}
\endscroll
\end{page}
```

Value of the symmetrised elliptic integral of second kind

— nags.ht —

```
\begin{page}{manpageXXs21bcf}{NAG Documentation: s21bcf}
\beginscroll
\begin{verbatim}



     S21BCF(3NAG)      Foundation Library (12/10/92)      S21BCF(3NAG)



         S21 -- Approximations of Special Functions                S21BCF
                 S21BCF -- NAG Foundation Library Routine Document

         Note: Before using this routine, please read the Users' Note for
         your implementation to check implementation-dependent details.
         The symbol (*) after a NAG routine name denotes a routine that is
```

not included in the Foundation Library.

1. Purpose

S21BCF returns a value of the symmetrised elliptic integral of the second kind, via the routine name.

2. Specification

```
       DOUBLE PRECISION FUNCTION S21BCF (X, Y, Z, IFAIL)
       INTEGER          IFAIL
       DOUBLE PRECISION X, Y, Z
```

3. Description

This routine calculates an approximate value for the integral

```
                          infty
                        3 /            dt
             R (x,y,z)= - |     ------------------
              D         2 /
                          0      /              3
                               \/ (t+x)(t+y)(t+z)
```

where x, y>=0, at most one of x and y is zero, and z>0.

The basic algorithm, which is due to Carlson [2] and [3], is to reduce the arguments recursively towards their mean by the rule:

```
    x    =   x
     0

    y    =   y
     0

    z    =   z
     0

    (mu) =   (x +y +3z )/5
        n      n  n   n

    X    =   (1-x )/(mu)
     n           n     n

    Y    =   (1-y )/(mu)
     n           n     n

    Z    =   (1-z )/(mu)
     n           n     n
```

$$\lambda_n = \sqrt{x_n}\sqrt{y_n} + \sqrt{y_n}\sqrt{z_n} + \sqrt{z_n}\sqrt{x_n}$$

$$x_{n+1} = (x_n + \lambda_n)/4$$

$$y_{n+1} = (y_n + \lambda_n)/4$$

$$z_{n+1} = (z_n + \lambda_n)/4$$

For n sufficiently large,

$$\epsilon_n = \max(|X_n|,|Y_n|,|Z_n|) \sim \left(\frac{1}{4}\right)^n$$

and the function may be approximated adequately by a 5th order power series $R_D(x,y,z) = 3\sum_{m=0}^{n-1} \frac{4^{-m}}{(z_m+\lambda_n)\sqrt{z_m}} +$

$$\frac{4^{-n}}{\sqrt{\mu_n^3}}\left[1 + \frac{3}{7}S_n^{(2)} + \frac{1}{3}S_n^{(3)} + \frac{3}{22}\left(S_n^{(2)}\right)^2 + \frac{3}{11}S_n^{(4)} + \frac{3}{13}S_n^{(2)}S_n^{(3)} + \frac{3}{13}S_n^{(5)}\right]$$

where

$$S_n^{(m)} = \left(X_n^m + Y_n^m + 3Z_n^m\right)/2m.$$

The truncation error in this expansion is bounded by

$\dfrac{3\epsilon_n^6}{\quad}$ and the recursive process is terminated when

```
    /              3
   / (1-(epsilon) )
 \/              n
```

this quantity is negligible compared with the machine precision.

The routine may fail either because it has been called with arguments outside the domain of definition, or with arguments so extreme that there is an unavoidable danger of setting underflow or overflow.

```
                -3/2
Note: R (x,x,x)=x    , so there exists a region of extreme
       D
```

arguments for which the function value is not representable.

4. References

[1] Abramowitz M and Stegun I A (1968) Handbook of Mathematical Functions. Dover Publications.

[2] Carlson B C (1978) Computing Elliptic Integrals by Duplication. (Preprint) Department of Physics, Iowa State University.

[3] Carlson B C (1988) A Table of Elliptic Integrals of the Third Kind. Math. Comput. 51 267--280.

5. Parameters

1: X -- DOUBLE PRECISION Input

2: Y -- DOUBLE PRECISION Input

3: Z -- DOUBLE PRECISION Input
On entry: the arguments x, y and z of the function.
Constraint: X, Y >= 0.0, Z > 0.0 and only one of X and Y may be zero.

4: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

```
          If on entry IFAIL = 0 or -1, explanatory error messages are
          output on the current error message unit (as defined by X04AAF).

          IFAIL= 1
               On entry either X or Y is negative, or both X and Y are
               zero; the function is undefined.

          IFAIL= 2
               On entry Z <= 0.0; the function is undefined.

          IFAIL= 3
               On entry either Z is too close to zero or both X and Y are
               too close to zero: there is a danger of setting overflow.

          IFAIL= 4
               On entry at least one of X, Y and Z is too large: there is a
               danger of setting underflow.

               On soft failure the routine returns zero.

          7. Accuracy

          In principle the routine is capable of producing full machine
          precision. However round-off errors in internal arithmetic will
          result in slight loss of accuracy. This loss should never be
          excessive as the algorithm does not involve any significant
          amplification of round-off error. It is reasonable to assume that
          the result is accurate to within a small multiple of the machine
          precision.

          8. Further Comments

          Users should consult the Chapter Introduction which shows the
          relationship of this function to the classical definitions of the
          elliptic integrals.

          9. Example

          This example program simply generates a small set of non-extreme
          arguments which are used with the routine to produce the table of
          low accuracy results.

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.
\end{verbatim}
\endscroll
\end{page}
```

——–

Value of the symmetrised elliptic integral of third kind

— nags.ht —

```
\begin{page}{manpageXXs21bdf}{NAG Documentation: s21bdf}
\beginscroll
\begin{verbatim}



     S21BDF(3NAG)      Foundation Library (12/10/92)      S21BDF(3NAG)



          S21 -- Approximations of Special Functions                S21BDF
                  S21BDF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          S21BDF returns a value of the symmetrised elliptic integral of
          the third kind, via the routine name.

          2. Specification

                 DOUBLE PRECISION FUNCTION S21BDF (X, Y, Z, R, IFAIL)
                 INTEGER          IFAIL
                 DOUBLE PRECISION X, Y, Z, R

          3. Description

          This routine calculates an approximation to the integral

                                   infty
                                 3 /                 dt
                 R (x,y,z,(rho))= - |     --------------------------
                  J              2 /
                                   0     (t+(rho))\/(t+x)(t+y)(t+z)

          where x, y, z>=0, (rho)/=0 and at most one of x, y and z is zero.

          If p<0, the result computed is the Cauchy principal value of the
```

integral.

The basic algorithm, which is due to Carlson [2] and [3], is to reduce the arguments recursively towards their mean by the rule:

$$x_0 = x$$

$$y_0 = y$$

$$z_0 = z$$

$$\rho_0 = \rho$$

$$\mu_n = (x_n + y_n + z_n + 2\rho_n)/5$$

$$X_n = (1 - x_n)/\mu_n$$

$$Y_n = (1 - y_n)/\mu_n$$

$$Z_n = (1 - z_n)/\mu_n$$

$$P_n = (1 - \rho_n)/\mu_n$$

$$\lambda_n = \sqrt{x_n y_n} + \sqrt{y_n z_n} + \sqrt{z_n x_n}$$

$$x_{n+1} = (x_n + \lambda_n)/4$$

$$y_{n+1} = (y_n + \lambda_n)/4$$

$$z_{n+1} = (z_n + \lambda_n)/4$$

$$\rho_{n+1} = (\rho_n + \lambda_n)/4$$

$$[\qquad\qquad]^2$$

```
    (alpha) =[(rho) (  /x +  /y +  /z )+  /x y z ]
           n [     n \/  n /  n /  n  /  n n n]

                                       2
    (beta) = (rho) ((rho) +(lambda) )
          n       n      n         n

For n sufficiently large,

                                                     1
             (epsilon) =max(|X |,|Y |,|Z |,|P |)~ --
                      n       n    n    n    n     n
                                                     4

and the function may be approximated by a 5th order power series
                  n-1
                  --  -m
R (x,y,z,(rho))=3 >  4  R ((alpha) ,(beta) )
 J                --     C        m       m
                  m=0


      -n
     4     [   3 (2)  1 (3)  3   (2) 2  3  (4)  3  (2) (3)  3  (5)]
+ ---------[1+ -S   + -S   + --(S   ) + --S   + --S   S   + --S   ]
           [   7 n    3 n    22  n      11 n    13 n   n    13 n  ]
     /   3
    / (mu)
  \/      n

       (m)     m  m  m   m
where S    = (X +Y +Z +2P )/2m.
       n       n  n  n   n

The truncation error in this expansion is bounded by


          6    /             3
3(epsilon) /  / (1-(epsilon) )  and the recursion process is
          n \/               n
terminated when this quantity is negligible compared with the
machine precision. The routine may fail either because it has
been called with arguments outside the domain of definition or
with arguments so extreme that there is an unavoidable danger of
setting underflow or overflow.

                    3
                  - -
                    2
```

Note: $R_J(x,x,x,x)=x^{-3/2}$, so there exists a region of extreme arguments for which the function value is not representable.

4. References

[1] Abramowitz M and Stegun I A (1968) Handbook of Mathematical Functions. Dover Publications.

[2] Carlson B C (1978) Computing Elliptic Integrals by Duplication. (Preprint) Department of Physics, Iowa State University.

[3] Carlson B C (1988) A Table of Elliptic Integrals of the Third Kind. Math. Comput. 51 267--280.

5. Parameters

1: X -- DOUBLE PRECISION Input

2: Y -- DOUBLE PRECISION Input

3: Z -- DOUBLE PRECISION Input

4: R -- DOUBLE PRECISION Input
On entry: the arguments x, y, z and (rho) of the function.
Constraint: X, Y, Z >= 0.0, R /= 0.0 and at most one of X, Y and Z may be zero.

5: IFAIL -- INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL= 1
On entry at least one of X, Y and Z is negative, or at least two of them are zero; the function is undefined.

IFAIL= 2
On entry R = 0.0; the function is undefined.

```
          IFAIL= 3
               On entry either R is too close to zero, or any two of X, Y
               and Z are too close to zero; there is a danger of setting
               overflow.

          IFAIL= 4
               On entry at least one of X, Y, Z and R is too large; there
               is a danger of setting underflow.

          IFAIL= 5
               An error has occurred in a call to S21BAF. Any such
               occurrence should be reported to NAG.

               On soft failure, the routine returns zero.

          7. Accuracy

          In principle the routine is capable of producing full machine
          precision. However round-off errors in internal arithmetic will
          result in slight loss of accuracy. This loss should never be
          excessive as the algorithm does not involve any significant
          amplification of round-off error. It is reasonable to assume that
          the result is accurate to within a small multiple of the machine
          precision.

          8. Further Comments

          Users should consult the Chapter Introduction which shows the
          relationship of this function to the classical definitions of the
          elliptic integrals.

          If the argument R is equal to any of the other arguments, the
          function reduces to the integral R , computed by S21BCF.
                                            D

          9. Example

          This example program simply generates a small set of non-extreme
          arguments which are used with the routine to produce the table of
          low accuracy results.

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.
\end{verbatim}
\endscroll
\end{page}
```

———-

22.7 nagx.ht

Mathematical Constants

— nagx.ht —

```
\begin{page}{manpageXXx01}{NAG Documentation: x01}
\beginscroll
\begin{verbatim}



     X01(3NAG)         Foundation Library (12/10/92)         X01(3NAG)



          X01 -- Mathematical Constants                 Introduction -- X01
                                    Chapter X01
                              Mathematical Constants

          1. Scope of the Chapter

          This chapter is concerned with the provision of mathematical
          constants required by other routines within the Library.

          It should be noted that because of the trivial nature of the
          routines individual routine documents are not provided.

          2. Background to the Problems

          Some Library routines require mathematical constants to maximum
          machine precision. These routines call Chapter X01 and thus
          lessen the number of changes that have to be made between
          different implementations of the Library.

          3. Recommendations on Choice and Use of Routines

          Although these routines are primarily intended for use by other
          routines they may be accessed directly by the user:

          Constant           Fortran Specification

          (pi)               DOUBLE PRECISION FUNCTION X01AAF(X)
                             DOUBLE PRECISION X
```

```
         (gamma)            DOUBLE PRECISION FUNCTION X01ABF(X)
         (Euler constant)   DOUBLE PRECISION X

         The argument X of these routines is a dummy argument.


         X01 -- Mathematical Constants                    Contents -- X01
         Chapter X01

         Mathematical Constants

         X01AAF  (pi)

         X01ABF  Euler's constant, (gamma)

\end{verbatim}
\endscroll
\end{page}
```

Machine Constants

— nagx.ht —

```
\begin{page}{manpageXXx02}{NAG Documentation: x02}
\beginscroll
\begin{verbatim}



     X02(3NAG)        Foundation Library (12/10/92)        X02(3NAG)



         X02 -- Machine Constants                     Introduction -- X02
                                    Chapter X02
                                 Machine Constants

         1. Scope of the Chapter

         This chapter is concerned with parameters which characterise
         certain aspects of the computing environment in which the NAG
         Foundation Library is implemented. They relate primarily to
         floating-point arithmetic, but also to integer arithmetic and the
         elementary functions. The values of the parameters vary from one
         implementation of the Library to another, but within the context
```

of a single implementation they are constants.

The parameters are intended for use primarily by other routines in the Library, but users of the Library may sometimes need to refer to them directly.

Each parameter-value is returned by a separate Fortran function. Because of the trivial nature of the functions, individual routine documents are not provided; the necessary details are given in Section 3 of this Introduction.

2. Background to the Problems

2.1. Floating-Point Arithmetic

2.1.1. A model of floating-point arithmetic

In order to characterise the important properties of floating-point arithmetic by means of a small number of parameters, NAG uses a simplified model of floating-point arithmetic. The parameters of the model can be chosen to provide a sufficiently close description of the behaviour of actual implementations of floating-point arithmetic, but not, in general, an exact description; actual implementations vary too much in the details of how numbers are represented or arithmetic operations are performed.

The model is based on that developed by Brown [1], but differs in some respects. The essential features are summarised here.

The model is characterised by four integer parameters and one logical parameter. The four integer parameters are:

- b : the base
- p : the precision (i.e. the number of significant base-B digits)
- e_{min} : the minimum exponent
- e_{max} : the maximum exponent

These parameters define a set of numerical values of the form:

$$f*b^{e}$$

where the exponent e must lie in the range [e_{min},e_{max}], and the min max

fraction f (also called the mantissa or significand) lies in the range [1/b,1), and may be written:

$$f=0.f_1 f_2 ...f_p$$

Thus f is a p-digit fraction to the base b; the f_i are the base-b digits of the fraction: they are integers in the range 0 to b-1, and the leading digit f_1 must not be zero.

The set of values so defined (together with zero) are called model numbers. For example, if b=10, p=5, e_{min}=-99 and e_{max}=+99, then a typical model number is $0.12345*10^{67}$.

The model numbers must obey certain rules for the computed results of the following basic arithmetic operations: addition, subtraction, multiplication, negation, absolute value, and comparisons. The rules depend on the value of the logical parameter ROUNDS.

If ROUNDS is true, then the computed result must be the nearest model number to the exact result (assuming that overflow or underflow does not occur); if the exact result is midway between two model numbers, then it may be rounded either way.

If ROUNDS is false, then: if the exact result is a model number, the computed result must be equal to the exact result; otherwise, the computed result may be either of the adjacent model numbers on either side of the exact result.

For division and square root, this latter rule is further relaxed (regardless of the value of ROUNDS): the computed result may also be one of the next adjacent model numbers on either side of the permitted values just stated.

On some machines, the full set of representable floating-point numbers conforms to the rules of the model with appropriate values of b, p, e_{min}, e_{max} and ROUNDS. For example, for machines with IEEE arithmetic, in double precision:

b = 2

p = 53

$e_{min}=-1021$

$e_{max}=1024$ and ROUNDS is true.

For other machines, values of the model parameters must be chosen which define a large subset of the representable numbers; typically it may be necessary to decrease p by 1 (in which case ROUNDS is always set to false), or to increase e_{min} or decrease e_{max} by a little bit. There are additional rules to ensure that arithmetic operations on those representable numbers which are not model numbers, are consistent with arithmetic on model numbers.

(Note: the model used here differs from that described in Brown [1] in the following respects: square-root is treated, like division, as a weakly supported operator; and the logical parameter ROUNDS has been introduced to take account of machines with good rounding.)

2.1.2. Derived parameters of floating-point arithmetic

Most numerical algorithms require access, not to the basic parameters of the model, but to certain derived values, of which the most important are:

the machine precision (epsilon): $= \left(\frac{1}{2}\right)*b^{1-p}$ if ROUNDS is true,

$= b^{1-p}$ otherwise (but see Note below).

the smallest positive model number: $= b^{e_{min}-1}$

the largest positive model number: $= (1-b^{-p})*b^{e_{max}}$

Note: this value is increased very slightly in some implementations to ensure that the computed result of 1+(epsilon) or 1-(epsilon) differs from 1. For example in IEEE binary single -24 -47

```
precision arithmetic the value is set to 2   +2   .

Two additional derived values are used in the NAG Foundation
Library. Their definitions depend not only on the properties of
the basic arithmetic operations just considered, but also on
properties of some of the elementary functions. We define the
safe range parameter to be the smallest positive model number z
such that for any x in the range [z,1/z] the following can be
computed without undue loss of accuracy, overflow, underflow or
other error:

    -x

    1/x

    -1/x

    SQRT(x)

    LOG(x)

    EXP(LOG(x))

    y**(LOG(x)/LOG(y)) for any y

In a similar fashion we define the safe range parameter for
complex arithmetic as the smallest positive model number z such
that for any x in the range [z,1/z] the following can be computed
without any undue loss of accuracy, overflow, underflow or other
error:

    -w

    1/w

    -1/w

    SQRT(w)

    LOG(w)

    EXP(LOG(w))

    y**(LOG(w)/LOG(y)) for any y

    ABS(w)

where w is any of x, ix, x+ix, 1/x, i/x, 1/x+i/x, and i is the
square root of -1.
```

This parameter was introduced to take account of the quality of complex arithmetic on the machine. On machines with well implemented complex arithmetic, its value will differ from that of the real safe range parameter by a small multiplying factor less than 10. For poorly implemented complex arithmetic this factor may be larger by many orders of magnitude.

2.2. Other Aspects of the Computing Environment

No attempt has been made to characterise comprehensively any other aspects of the computing environment. The other functions in this chapter provide specific information that is occasionally required by routines in the Library.

2.3. References

[1] Brown W S (1981) A Simple but Realistic Model of Floating-point Computation. ACM Trans. Math. Softw. 7 445--480.

3. Recommendations on Choice and Use of Routines

3.1. Parameters of Floating-point Arithmetic

```
DOUBLE PRECISION FUNCTION returns the machine precision i.e.
X02AJF()                  (1)  1-p                        1-p
                          (-)*b    if ROUNDS is true or b
                          (2)
                          otherwise (or a value very slightly
                          larger than this, see Section 2.1.2)

DOUBLE PRECISION FUNCTION returns the smallest positive model
X02AKF()                              e   -1
                                       min
                          number i.e. b

DOUBLE PRECISION FUNCTION returns the largest positive model
X02ALF()                                      e
                                         -p    max
                          number i.e. (1-b  )*b


DOUBLE PRECISION FUNCTION returns the safe range parameter as
X02AMF()                  defined in Section 2.1.2

DOUBLE PRECISION FUNCTION returns the safe range parameter for
X02ANF()                  complex arithmetic as defined in
                          Section 2.1.2

INTEGER FUNCTION X02BHF() returns the model parameter b
```

INTEGER FUNCTION X02BJF() returns the model parameter p

INTEGER FUNCTION X02BKF() returns the model parameter e_{min}

INTEGER FUNCTION X02BLF() returns the model parameter e_{max}

LOGICAL FUNCTION X02DJF() returns the model parameter ROUNDS

3.2. Parameters of Other Aspects of the Computing Environment

DOUBLE PRECISION FUNCTION X02AHF(X) DOUBLE PRECISION X	returns the largest positive real argument for which the SIN and COS routines return a result with some meaningful accuracy
INTEGER FUNCTION X02BBF (X) DOUBLE PRECISION X	returns the largest positive integer value
INTEGER FUNCTION X02BEF (X) DOUBLE PRECISION X	returns the maximum number of decimal digits which can be accurately represented over the whole range of floating-point numbers

The argument X of these routines is a dummy argument.

4. Example Program Text

The example program simply prints the values of all the functions in Chapter X02. Obviously the results will vary from one implementation of the Library to another.

X02 -- Machine Constants Contents -- X02
Chapter X02

Machine Constants

```
          X02ANF  Safe range of complex floating-point arithmetic

          X02BBF  Largest representable integer

          X02BEF  Maximum number of decimal digits that can be represented

          X02BHF  Parameter of floating-point arithmetic model, b

          X02BJF  Parameter of floating-point arithmetic model, p

          X02BKF  Parameter of floating-point arithmetic model, e
                                                                 min

          X02BLF  Parameter of floating-point arithmetic model, e
                                                                 max

          X02DJF  Parameter of floating-point arithmetic model, ROUNDS

\end{verbatim}
\endscroll
\end{page}
```

Input/Output Utilities

— nagx.ht —

```
\begin{page}{manpageXXx04}{NAG Documentation: x04}
\beginscroll
\begin{verbatim}



     X04(3NAG)         Foundation Library (12/10/92)         X04(3NAG)



          X04 -- Input/Output Utilities                 Introduction -- X04
                                      Chapter X04
                                Input/Output Utilities

          1. Scope of the Chapter

          This chapter contains utility routines concerned with input and
          output to or from an external file.
```

2. Background to the Problems

2.1. Output from NAG Foundation Library Routines

Output from NAG Foundation Library routines to an external file falls into two categories:

(a) Error messages which are always associated with an error exit from a routine, that is, with a non-zero value of IFAIL as specified in Section 6 of the routine document.

(b) Advisory messages which include output of final results, output of intermediate results to monitor the course of a computation, and various warning or informative messages.

Each category of output is written to its own Fortran output unit - the error message unit or the advisory message unit. In practice these may be the same unit number. Default unit numbers are provided for each implementation of the Library (see the Users' Note); they may be changed by users. Output of error messages may be controlled by the setting of IFAIL (see the Essential Introduction). Output of advisory messages may usually be controlled by the setting of some other parameter (e.g. MSGLVL) (or in some routines also by IFAIL). An alternative mechanism for completely suppressing output is to set the relevant unit number < 0.

For further information about error and advisory messages, see Chapter P01.

2.2. Matrix Printing Routines

Routines are provided to allow formatted output of general rectangular or triangular matrices stored in a two-dimensional array (real and complex data types).

All output is directed to the unit number for output of advisory messages, which may be altered by a call to X04ABF.

3. Recommendations on Choice and Use of Routines

Apart from the obvious utility of the matrix printing routines, users of the Library may need to call routines in Chapter X04 for the following purposes:

if the default unit number for error messages (given in the Users' Note for your implementation) is not satisfactory, it may be changed to a new value NERR by the statement

```
CALL X04AAF(1,NERR)
```

```
     Similarly the unit number for advisory messages may be
     changed to a new value NADV by the statement

          CALL X04ABF(1,NADV)

4. Index

Accessing unit number:
     of advisory message unit                               X04ABF
     of error message unit                                  X04AAF
Printing matrices:
     general complex matrix                                 X04DAF
     general real matrix                                    X04CAF



X04 -- Input/Output Utilities                        Contents -- X04
Chapter X04

Input/Output Utilities

X04AAF  Return or set unit number for error messages

X04ABF  Return or set unit number for advisory messages

X04CAF  Print a real general matrix

X04DAF  Print a complex general matrix

\end{verbatim}
\endscroll
\end{page}
```

———-

Value of the current error message unit number

— nagx.ht —

```
\begin{page}{manpageXXx04aaf}{NAG Documentation: x04aaf}
\beginscroll
\begin{verbatim}


     X04AAF(3NAG)      Foundation Library (12/10/92)      X04AAF(3NAG)
```

```
X04 -- Input/Output Utilities                               X04AAF
        X04AAF -- NAG Foundation Library Routine Document

Note: Before using this routine, please read the Users' Note for
your implementation to check implementation-dependent details.
The symbol (*) after a NAG routine name denotes a routine that is
not included in the Foundation Library.

1. Purpose

X04AAF returns the value of the current error message unit
number, or sets the current error message unit number to a new
value.

2. Specification

       SUBROUTINE X04AAF (IFLAG, NERR)
       INTEGER          IFLAG, NERR

3. Description

This routine enables those library routines which output error
messages, to determine the number of the output unit to which the
error messages are to be sent; in this case X04AAF is called with
IFLAG = 0. X04AAF may also be called with IFLAG = 1 to set the
unit number to a specified value. Otherwise a default value
(stated in the Users' Note for your implementation) is returned.

Records written to this output unit by other library routines are
at most 80 characters long (including a line-printer carriage
control character).

Note that if the unit number is set < 0, no messages will be
output.

4. References

None.

5. Parameters

 1:  IFLAG -- INTEGER                                      Input
     On entry: the action to be taken (see NERR). Constraint:
     IFLAG = 0 or 1.

 2:  NERR -- INTEGER                                Input/Output
     On entry:
```

```
                    if IFLAG = 0, NERR need not be set;

                    if IFLAG = 1, NERR must specify the new error message
                    unit number.
               On exit:
                    if IFLAG = 0, NERR is set to the current error message
                    unit number,

                    if IFLAG = 1, NERR is unchanged.
               Note that Fortran unit numbers must be positive or zero. If
               NERR is set < 0, output of error messages is totally
               suppressed.

          6. Error Indicators and Warnings

          None.

          7. Accuracy

          Not applicable.

          8. Further Comments

          The time taken by the routine is negligible.

          9. Example

          In this example X04AAF is called by the user's main program to
          make the error message from the routine DUMMY appear on the same
          unit as the rest of the output (unit 6). Normally a NAG
          Foundation Library routine with an IFAIL parameter (see Essential
          Introduction) would take the place of DUMMY.

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.
\end{verbatim}
\endscroll
\end{page}
```

———

Value of the current advisory message unit number

— nagx.ht —

```
\begin{page}{manpageXXx04abf}{NAG Documentation: x04abf}
```

```
\beginscroll
\begin{verbatim}


     X04ABF(3NAG)      Foundation Library (12/10/92)      X04ABF(3NAG)



          X04 -- Input/Output Utilities                              X04ABF
                  X04ABF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          X04ABF returns the value of the current advisory message unit
          number, or sets the current advisory message unit number to a new
          value.

          2. Specification

                 SUBROUTINE X04ABF (IFLAG, NADV)
                 INTEGER          IFLAG, NADV

          3. Description

          This routine enables those library routines which output advisory
          messages, to determine the number of the output unit to which the
          advisory messages are to be sent; in this case X04ABF is called
          with IFLAG = 0. X04ABF may also be called with IFLAG = 1 to set
          the unit number to a specified value. Otherwise a default value
          (stated in the User's Note for your implementation) is returned.

          Records written to this output unit by other library routines are
          at most 120 characters long (including a line-printer carriage
          control character), unless those library routines allow users to
          specify longer records.

          Note that if the unit number is set < 0, no messages will be
          output.

          4. References

          None.

          5. Parameters
```

```
 1:  IFLAG -- INTEGER                                       Input
     On entry: the action to be taken (see NADV). Constraint:
     IFLAG = 0 or 1.

 2:  NADV -- INTEGER                                 Input/Output
     On entry:
          if IFLAG = 0, NADV need not be set;

          if IFLAG = 1, NADV must specify the new advisory
          message unit number.
     On exit:
          if IFLAG = 0, NADV is set to the current advisory
          message unit number;

          if IFLAG = 1, NADV is unchanged.
     Note that Fortran unit numbers must be positive or zero. If
     NADV is set < 0, output of advisory messages is totally
     suppressed.

6. Error Indicators and Warnings

None.

7. Accuracy

Not applicable.

8. Further Comments

The time taken by this routine is negligible.

9. Example

In this example X04ABF is called by the user's main program to
make the advisory message from the routine DUMMY appear on the
same unit as the rest of the output (unit 6). Normally a NAG
Foundation Library routine with an IFAIL parameter (see Essential
Introduction) would take the place of DUMMY.

The example program is not reproduced here. The source code for
all example programs is distributed with the NAG Foundation
Library software and should be available on-line.
\end{verbatim}
\endscroll
\end{page}
```

Print a real matrix stored in a two-dimensional array

— nagx.ht —

```
\begin{page}{manpageXXx04caf}{NAG Documentation: x04caf}
\beginscroll
\begin{verbatim}



     X04CAF(3NAG)      Foundation Library (12/10/92)      X04CAF(3NAG)



          X04 -- Input/Output Utilities                              X04CAF
                  X04CAF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          X04CAF is an easy-to-use routine to print a real matrix stored in
          a two-dimensional array.

          2. Specification

                 SUBROUTINE X04CAF (MATRIX, DIAG, M, N, A, LDA, TITLE,
                1                   IFAIL)
                 INTEGER          M, N, LDA, IFAIL
                 DOUBLE PRECISION A(LDA,*)
                 CHARACTER*1      MATRIX, DIAG
                 CHARACTER*(*)    TITLE

          3. Description

          X04CAF prints a real matrix. It is an easy-to-use driver for
          X04CBF(*). The routine uses default values for the format in
          which numbers are printed, for labelling the rows and columns,
          and for output record length.

          X04CAF will choose a format code such that numbers will be
          printed with either an F8.4, F11.4 or a 1PE13.4 format. The F8.4
          code is chosen if the sizes of all the matrix elements to be
          printed lie between 0.001 and 1.0. The F11.4 code is chosen if
          the sizes of all the matrix elements to be printed lie between 0.
          001 and 9999.9999. Otherwise the 1PE13.4 code is chosen.
```

```
The matrix is printed with integer row and column labels, and
with a maximum record length of 80.

The matrix is output to the unit defined by X04ABF.

4. References

None.

5. Parameters

 1:  MATRIX -- CHARACTER*1                                 Input
     On entry: indicates the part of the matrix to be printed, as
     follows:

     MATRIX = 'G' (General), the whole of the rectangular matrix.

     MATRIX = 'L' (Lower), the lower triangle of the matrix, or
     the lower trapezium if the matrix has more rows than
     columns.

     MATRIX = 'U' (Upper), the upper triangle of the matrix, or
     the upper trapezium if the matrix has more columns than
     rows. Constraint: MATRIX must be one of 'G', 'L' or 'U'.

 2:  DIAG -- CHARACTER*1                                   Input
     On entry: unless MATRIX = 'G', DIAG must specify whether the
     diagonal elements of the matrix are to be printed, as
     follows:

     DIAG = 'B' (Blank), the diagonal elements of the matrix are
     not referenced and not printed.

     DIAG = 'U' (Unit diagonal), the diagonal elements of the
     matrix are not referenced, but are assumed all to be unity,
     and are printed as such.

     DIAG = 'N' (Non-unit diagonal), the diagonal elements of the
     matrix are referenced and printed.

     If MATRIX = 'G', then DIAG need not be set. Constraint: If
     MATRIX /= 'G', then DIAG must be one of 'B', 'U' or 'N'.

 3:  M -- INTEGER                                          Input

 4:  N -- INTEGER                                          Input
     On entry: the number of rows and columns of the matrix,
     respectively, to be printed.
```

```
     If either of M or N is less than 1, X04CAF will exit
     immediately after printing TITLE; no row or column labels
     are printed.

 5:  A(LDA,*) -- DOUBLE PRECISION array                     Input
     Note: the second dimension of the array A must be at least
     max(1,N).
     On entry: the matrix to be printed. Only the elements that
     will be referred to, as specified by parameters MATRIX and
     DIAG, need be set.

 6:  LDA -- INTEGER                                         Input
     On entry:
     the first dimension of the array A as declared in the
     (sub)program from which X04CAF is called.
     Constraint: LDA >= M.

 7:  TITLE -- CHARACTER*(*)                                 Input
     On entry: a title to be printed above the matrix. If TITLE =
     ' ', no title (and no blank line) will be printed.

     If TITLE contains more than 80 characters, the contents of
     TITLE will be wrapped onto more than one line, with the
     break after 80 characters.

     Any trailing blank characters in TITLE are ignored.

 8:  IFAIL -- INTEGER                                Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

IFAIL= 1
     On entry MATRIX /= 'G', 'L' or 'U'.

IFAIL= 2
     On entry MATRIX = 'L' or 'U', but DIAG /= 'N', 'U' or 'B'.

IFAIL= 3
     On entry LDA < M.
```

```
          7. Accuracy

          Not applicable.

          8. Further Comments

          A call to X04CAF is equivalent to a call to X04CBF(*) with the
          following argument values:


                NCOLS = 80
                INDENT = 0
                LABROW = 'I'
                LABCOL = 'I'
                FORMAT = ' '


          9. Example

          This example program calls X04CAF twice, first to print a 3 by 5
          rectangular matrix, and then to print a 5 by 5 lower triangular
          matrix.

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

Print a complex matrix stored in a 2D array

— nagx.ht —

```
\begin{page}{manpageXXx04daf}{NAG Documentation: x04daf}
\beginscroll
\begin{verbatim}


     X04DAF(3NAG)      Foundation Library (12/10/92)      X04DAF(3NAG)
```

```
X04 -- Input/Output Utilities                              X04DAF
        X04DAF -- NAG Foundation Library Routine Document

Note: Before using this routine, please read the Users' Note for
your implementation to check implementation-dependent details.
The symbol (*) after a NAG routine name denotes a routine that is
not included in the Foundation Library.

1. Purpose

X04DAF is an easy-to-use routine to print a complex matrix stored
in a two-dimensional array.

2. Specification

       SUBROUTINE X04DAF (MATRIX, DIAG, M, N, A, LDA, TITLE,
      1                   IFAIL)
       INTEGER              M, N, LDA, IFAIL
       COMPLEX(KIND(1.0D0)) A(LDA,*)
       CHARACTER*1          MATRIX, DIAG
       CHARACTER*(*)        TITLE

3. Description

X04DAF prints a complex matrix. It is an easy-to-use driver for
X04DBF(*). The routine uses default values for the format in
which numbers are printed, for labelling the rows and columns,
and for output record length.

X04DAF will choose a format code such that numbers will be
printed with either an F8.4, F11.4 or a 1PE13.4 format. The F8.4
code is chosen if the sizes of all the matrix elements to be
printed lie between 0.001 and 1.0. The F11.4 code is chosen if
the sizes of all the matrix elements to be printed lie between 0.
001 and 9999.9999. Otherwise the 1PE13.4 code is chosen. The
chosen code is used to print each complex element of the matrix
with the real part above the imaginary part.

The matrix is printed with integer row and column labels, and
with a maximum record length of 80.

The matrix is output to the unit defined by X04ABF.

4. References

None.

5. Parameters
```

```
1:  MATRIX -- CHARACTER*1                                  Input
    On entry: indicates the part of the matrix to be printed, as
    follows:

    MATRIX = 'G' (General), the whole of the rectangular matrix.

    MATRIX = 'L' (Lower), the lower triangle of the matrix, or
    the lower trapezium if the matrix has more rows than
    columns.

    MATRIX = 'U' (Upper), the upper triangle of the matrix, or
    the upper trapezium if the matrix has more columns than
    rows. Constraint: MATRIX must be one of 'G', 'L' or 'U'.

2:  DIAG -- CHARACTER*1                                    Input
    On entry: unless MATRIX = 'G', DIAG must specify whether the
    diagonal elements of the matrix are to be printed, as
    follows:

    DIAG = 'B' (Blank), the diagonal elements of the matrix are
    not referenced and not printed.

    DIAG = 'U' (Unit diagonal), the diagonal elements of the
    matrix are not referenced, but are assumed all to be unity,
    and are printed as such.

    DIAG = 'N' (Non-unit diagonal), the diagonal elements of the
    matrix are referenced and printed.

    If MATRIX = 'G', then DIAG need not be set. Constraint: If
    MATRIX /= 'G', then DIAG must be one of 'B', 'U' or 'N'.

3:  M -- INTEGER                                           Input

4:  N -- INTEGER                                           Input
    On entry: the number of rows and columns of the matrix,
    respectively, to be printed.

    If either of M or N is less than 1, X04DAF will exit
    immediately after printing TITLE; no row or column labels
    are printed.

5:  A(LDA,*) -- COMPLEX(KIND(1.0D)) array                  Input
    Note: the second dimension of the array A must be at least
    max(1,N).
    On entry: the matrix to be printed. Only the elements that
    will be referred to, as specified by parameters MATRIX and
    DIAG, need be set.

6:  LDA -- INTEGER                                         Input
```

```
     On entry:
     the first dimension of the array A as declared in the
     (sub)program from which X04DAF is called.
     Constraint: LDA >= M.

 7:  TITLE -- CHARACTER*(*)                                Input

     On entry: a title to be printed above the matrix. If TITLE =
     ' ', no title (and no blank line) will be printed.

     If TITLE contains more than 80 characters, the contents of
     TITLE will be wrapped onto more than one line, with the
     break after 80 characters.

     Any trailing blank characters in TITLE are ignored.

 8:  IFAIL -- INTEGER                               Input/Output
     On entry: IFAIL must be set to 0, -1 or 1. For users not
     familiar with this parameter (described in the Essential
     Introduction) the recommended value is 0.

     On exit: IFAIL = 0 unless the routine detects an error (see
     Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or -1, explanatory error messages are
output on the current error message unit (as defined by X04AAF).

IFAIL= 1
     On entry MATRIX /= 'G', 'L' or 'U'.

IFAIL= 2
     On entry MATRIX = 'L' or 'U', but DIAG /= 'N', 'U' or 'B'.

IFAIL= 3
     On entry LDA < M.

7. Accuracy

Not applicable.

8. Further Comments

A call to X04DAF is equivalent to a call to X04DBF(*) with the
following argument values:
```

```
          NCOLS = 80
          INDENT = 0
          LABROW = 'I'
          LABCOL = 'I'
          FORMAT = ' '
          USEFRM = 'A'


     9. Example

     This example program calls X04DAF twice, first to print a 4 by 3
     rectangular matrix, and then to print a 4 by 4 lower triangular
     matrix.

     The example program is not reproduced here. The source code for
     all example programs is distributed with the NAG Foundation
     Library software and should be available on-line.


\end{verbatim}
\endscroll
\end{page}
```

———-

Date and Time Utilities

— nagx.ht —

```
\begin{page}{manpageXXx05}{NAG Documentation: x05}
\beginscroll
\begin{verbatim}



     X05(3NAG)         Foundation Library (12/10/92)         X05(3NAG)



     X05 -- Date and Time Utilities                 Introduction -- X05
                                   Chapter X05
                             Date and Time Utilities

     1. Scope of the Chapter

     This chapter provides routines to obtain the current real time,
     and the amount of processor time used.
```

```
          2. Background to the Problems

          2.1. Real Time

          Routines are provided to obtain the current time in two different
          formats, and to compare two such times.

          2.2. Processor Time

          A routine is provided to return the current amount of processor
          time used. This allows the timing of a particular routine or
          section of code.

          3. Recommendations on Choice and Use of Routines

          X05AAF     returns the current date/time in integer format.

          X05ABF     converts from integer to character string date/time.

          X05ACF     compares two date/time character strings.

          X05BAF     returns the amount of processor time used.


          X05 -- Date and Time Utilities                    Contents -- X05
          Chapter X05

          Date and Time Utilities

          X05AAF  Return date and time as an array of integers

          X05ABF  Convert array of integers representing date and time to
                  character string

          X05ACF  Compare two character strings representing date and time

          X05BAF  Return the CPU time

\end{verbatim}
\endscroll
\end{page}
```

Returns the current date and time

— nagx.ht —

```
\begin{page}{manpageXXx05aaf}{NAG Documentation: x05aaf}
\beginscroll
\begin{verbatim}



     X05AAF(3NAG)      Foundation Library (12/10/92)      X05AAF(3NAG)



          X05 -- Date and Time Utilities                             X05AAF
                  X05AAF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          X05AAF returns the current date and time.

          2. Specification

                 SUBROUTINE X05AAF (ITIME)
                 INTEGER          ITIME(7)

          3. Description

          X05AAF returns the current date and time as a set of seven
          integers.

          4. References

          None.

          5. Parameters

           1:  ITIME(7) -- INTEGER array                             Output
               On exit: the current date and time, as follows:

               ITIME(1) contains the current year.

               ITIME(2) contains the current month, in the range 1--12.
```

```
          ITIME(3) contains the current day, in the range 1--31.

          ITIME(4) contains the current hour, in the range 0--23.

          ITIME(5) contains the current minute, in the range 0--59.

          ITIME(6) contains the current second, in the range 0--59.

          ITIME(7) contains the current millisecond, in the range 0--
          999.

     6. Error Indicators and Warnings

     None.

     7. Accuracy

     The accuracy of this routine depends on the accuracy of the host
     machine. In particular, on some machines it may not be possible
     to return a value for the current millisecond, for example. In
     this case, the value returned will be zero.

     8. Further Comments

     None.

     9. Example

     This program prints out the vector ITIME after a call to X05AAF.

     The example program is not reproduced here. The source code for
     all example programs is distributed with the NAG Foundation
     Library software and should be available on-line.
\end{verbatim}
\endscroll
\end{page}
```

———-

From seven-integer format time and date to character string

— nagx.ht —

```
\begin{page}{manpageXXx05abf}{NAG Documentation: x05abf}
\beginscroll
\begin{verbatim}
```

X05 -- Date and Time Utilities X05ABF

X05ABF -- NAG Foundation Library Routine Document

Note: Before using this routine, please read the Users' Note for your implementation to check implementation-dependent details. The symbol (*) after a NAG routine name denotes a routine that is not included in the Foundation Library.

1. Purpose

X05ABF converts from a seven-integer format time and date, as returned by X05AAF, into a character string, returned via the routine name.

2. Specification

```
CHARACTER*30 FUNCTION X05ABF (ITIME)
INTEGER          ITIME(7)
```

3. Description

X05ABF returns a character string of length 30 which contains the date and time as supplied in argument ITIME. On exit, the character string has the following format:

'DAY XXTH MTH YEAR HR:MN:SC.MIL',

where DAY is one of 'Sun', 'Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat',

XX is an integer denoting the day of the month,

TH is one of 'st', 'nd', 'rd', 'th',

MTH is one of 'Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec',

YEAR is the year as a four digit integer,

HR is the hour,

MN is the minute,

```
        SC    is the second,

        MIL   is the millisecond.

If on entry the date in ITIME is invalid, the string returned is

4. References

None.

5. Parameters

 1:  ITIME(7) -- INTEGER array                              Input
     On entry: a date and time in the format returned by X05AAF,
     as follows:
         ITIME   must contain the year as a positive integer.
         (1)

         ITIME   must contain the month, in the range 1-12.
         (2)

         ITIME   must contain the day, in the range 1 to p, where
         (3)     p = 28, 29, 30 or 31, depending on the month and
                 year.

         ITIME   must contain the hour, in the range 0-23.
         (4)

         ITIME   must contain the minute, in the range 0-59.
         (5)

         ITIME   must contain the second, in the range 0-59.
         (6)

         ITIME   must contain the millisecond, in the range 0-
         (7)     999.

6. Error Indicators and Warnings

None.

7. Accuracy

The day name included as part of the character string returned by
this routine is calculated assuming that the date is part of the
Gregorian calendar. This calendar has been in operation in Europe
since October the 15th 1582, and in Great Britain since September
the 14th 1752. Entry to this routine with a date earlier than
these will therefore not return a day name that is historically
```

```
          accurate.

          8. Further Comments

          Two dates stored in character string format, as returned by this
          routine, may be compared by X05ACF.

          9. Example

          This program initialises a time in ITIME, and converts it to
          character format by a call to X05ABF.

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.
\end{verbatim}
\endscroll
\end{page}
```

Compares two date/time character strings

— nagx.ht —

```
\begin{page}{manpageXXx05acf}{NAG Documentation: x05acf}
\beginscroll
\begin{verbatim}



     X05ACF(3NAG)      Foundation Library (12/10/92)      X05ACF(3NAG)



          X05 -- Date and Time Utilities                             X05ACF
                  X05ACF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          X05ACF compares two date/time character strings, each stored in
          the format returned by X05ABF.
```

2. Specification

```
INTEGER FUNCTION X05ACF (CTIME1, CTIME2)
CHARACTER*(*)    CTIME1, CTIME2
```

3. Description

X05ACF compares two date/time character strings, and returns an integer that specifies which one is the earliest. The result is an integer returned through the routine name, with meaning as follows:

X05ACF = -1: the first date/time string is earlier than the second.

X05ACF = 0: the two date/time strings are equivalent.

X05ACF = 1: the first date/time string is later than the second.

4. References

None.

5. Parameters

1: CTIME1 -- CHARACTER*(*) Input

2: CTIME2 -- CHARACTER*(*) Input
On entry: the date/time strings to be compared. These are expected be in the format returned by X05ABF, although X05ACF will still attempt to interpret the strings if they vary slightly from this format. See Section 8 for further details.

6. Error Indicators and Warnings

None.

7. Accuracy

Not applicable.

8. Further Comments

For flexibility, X05ACF will accept various formats for the two date/time strings CTIME1 and CTIME2.

The strings do not have to be the same length. It is permissible,

```
for example, to enter with one or both of the strings truncated
to a smaller length, in which case missing fields are treated as
zero.

Each character string may be of any length, but everything after
character 80 is ignored.

Each string may or may not include an alphabetic day name, such
as 'Wednesday', at its start. These day names are ignored, and no
check is made that the day name corresponds correctly to the rest
of the date.

The month name may contain any number of characters provided it
uniquely identifies the month, however all characters that are
supplied are significant.

Each field in the character string must be separated by one or
more spaces.

The case of all alphabetic characters is insignificant.

Any field in a date time string that is indecipherable according
to the above rules will be converted to a zero value internally.
Thus two strings that are completely indecipherable will compare
equal.

According to these rules, all the following date/time strings are
equivalent:

    'Thursday 10th July 1958 12:43:17.320'

    'THU 10th JULY 1958 12:43:17.320'

    '10th Jul 1958 12:43:17.320'

9. Example

This program initialises two date/time strings, and compares them
by a call to X05ACF.


The example program is not reproduced here. The source code for
all example programs is distributed with the NAG Foundation
Library software and should be available on-line.

\end{verbatim}
\endscroll
\end{page}
```

——-

Amount of processor time used

— nagx.ht —

```
\begin{page}{manpageXXx05baf}{NAG Documentation: x05baf}
\beginscroll
\begin{verbatim}



     X05BAF(3NAG)      Foundation Library (12/10/92)      X05BAF(3NAG)



          X05 -- Date and Time Utilities                            X05BAF
                  X05BAF -- NAG Foundation Library Routine Document

          Note: Before using this routine, please read the Users' Note for
          your implementation to check implementation-dependent details.
          The symbol (*) after a NAG routine name denotes a routine that is
          not included in the Foundation Library.

          1. Purpose

          X05BAF returns the amount of processor time used since an
          unspecified previous time, via the routine name.

          2. Specification

                 DOUBLE PRECISION FUNCTION X05BAF ()

          3. Description

          X05BAF returns the number of seconds of processor time used since
          some previous time. The previous time is system dependent, but
          may be, for example, the time the current job or the current
          program started running.

          If the system clock of the host machine is inaccessible for any
          reason, X05BAF returns the value zero.

          4. References

          None.

          5. Parameters
```

```
          None.

          6. Error Indicators and Warnings

          None.

          7. Accuracy

          The accuracy of the value returned depends on the accuracy of the
          system clock on the host machine.

          8. Further Comments

          Since the value returned by X05BAF is the amount of processor
          time since some unspecified earlier time, no significance should
          be placed on the value other than as a marker to be compared with
          some later figure returned by X05BAF. The amount of processor
          time that has elapsed between two calls of X05BAF can be simply
          calculated as the earlier value subtracted from the later value.

          9. Example

          This program makes a call to X05BAF, performs some computations,
          makes another call to X05BAF, and gives the time used by the
          computations as the difference between the two returned values.

          The example program is not reproduced here. The source code for
          all example programs is distributed with the NAG Foundation
          Library software and should be available on-line.
\end{verbatim}
\endscroll
\end{page}
```

Chapter 23

NAG ASP Example Code

23.1 aspex.ht

Asp1 Example Code

— aspex.ht —

```
\begin{page}{Asp1ExampleCode}{Asp1 Example Code}
\begin{verbatim}
      DOUBLE PRECISION FUNCTION F(X)
      DOUBLE PRECISION X
      F=DSIN(X)
      RETURN
      END
\end{verbatim}
\end{page}
```

———-

Asp10 Example Code

— aspex.ht —

```
\begin{page}{Asp10ExampleCode}{Asp10 Example Code}
\begin{verbatim}
      SUBROUTINE COEFFN(P,Q,DQDL,X,ELAM,JINT)
      DOUBLE PRECISION ELAM,P,Q,X,DQDL
      INTEGER JINT
```

```
      P=1.0D0
      Q=((-1.0D0*X**3)+ELAM*X*X-2.0D0)/(X*X)
      DQDL=1.0D0
      RETURN
      END
\end{verbatim}
\end{page}
```

————

Asp12 Example Code

— **aspex.ht** —

```
\begin{page}{Asp12ExampleCode}{Asp12 Example Code}
\begin{verbatim}
      SUBROUTINE MONIT (MAXIT,IFLAG,ELAM,FINFO)
      DOUBLE PRECISION ELAM,FINFO(15)
      INTEGER MAXIT,IFLAG
      IF(MAXIT.EQ.-1)THEN
        PRINT*,"Output from Monit"
      ENDIF
      PRINT*,MAXIT,IFLAG,ELAM,(FINFO(I),I=1,4)
      RETURN
      END
\end{verbatim}
\end{page}
```

————

Asp19 Example Code

— **aspex.ht** —

```
\begin{page}{Asp19ExampleCode}{Asp19 Example Code}
\begin{verbatim}
      SUBROUTINE LSFUN2(M,N,XC,FVECC,FJACC,LJC)
      DOUBLE PRECISION FVECC(M),FJACC(LJC,N),XC(N)
      INTEGER M,N,LJC
      INTEGER I,J
      DO 25003 I=1,LJC
        DO 25004 J=1,N
```

```
          FJACC(I,J)=0.0D0
25004   CONTINUE
25003 CONTINUE
       FVECC(1)=((XC(1)-0.14D0)*XC(3)+(15.0D0*XC(1)-2.1D0)*XC(2)+1.0D0)/(
      &XC(3)+15.0D0*XC(2))
       FVECC(2)=((XC(1)-0.18D0)*XC(3)+(7.0D0*XC(1)-1.26D0)*XC(2)+1.0D0)/(
      &XC(3)+7.0D0*XC(2))
       FVECC(3)=((XC(1)-0.22D0)*XC(3)+(4.333333333333333D0*XC(1)-0.953333
      &3333333333D0)*XC(2)+1.0D0)/(XC(3)+4.333333333333333D0*XC(2))
       FVECC(4)=((XC(1)-0.25D0)*XC(3)+(3.0D0*XC(1)-0.75D0)*XC(2)+1.0D0)/(
      &XC(3)+3.0D0*XC(2))
       FVECC(5)=((XC(1)-0.29D0)*XC(3)+(2.2D0*XC(1)-0.6379999999999999D0)*
      &XC(2)+1.0D0)/(XC(3)+2.2D0*XC(2))
       FVECC(6)=((XC(1)-0.32D0)*XC(3)+(1.666666666666667D0*XC(1)-0.533333
      &3333333333D0)*XC(2)+1.0D0)/(XC(3)+1.666666666666667D0*XC(2))
       FVECC(7)=((XC(1)-0.35D0)*XC(3)+(1.285714285714286D0*XC(1)-0.45D0)*
      &XC(2)+1.0D0)/(XC(3)+1.285714285714286D0*XC(2))
       FVECC(8)=((XC(1)-0.39D0)*XC(3)+(XC(1)-0.39D0)*XC(2)+1.0D0)/(XC(3)+
      &XC(2))
       FVECC(9)=((XC(1)-0.37D0)*XC(3)+(XC(1)-0.37D0)*XC(2)+1.285714285714
      &286D0)/(XC(3)+XC(2))
       FVECC(10)=((XC(1)-0.58D0)*XC(3)+(XC(1)-0.58D0)*XC(2)+1.66666666666
      &6667D0)/(XC(3)+XC(2))
       FVECC(11)=((XC(1)-0.73D0)*XC(3)+(XC(1)-0.73D0)*XC(2)+2.2D0)/(XC(3)
      &+XC(2))
       FVECC(12)=((XC(1)-0.96D0)*XC(3)+(XC(1)-0.96D0)*XC(2)+3.0D0)/(XC(3)
      &+XC(2))
       FVECC(13)=((XC(1)-1.34D0)*XC(3)+(XC(1)-1.34D0)*XC(2)+4.33333333333
      &3333D0)/(XC(3)+XC(2))
       FVECC(14)=((XC(1)-2.1D0)*XC(3)+(XC(1)-2.1D0)*XC(2)+7.0D0)/(XC(3)+X
      &C(2))
       FVECC(15)=((XC(1)-4.39D0)*XC(3)+(XC(1)-4.39D0)*XC(2)+15.0D0)/(XC(3
      &)+XC(2))
       FJACC(1,1)=1.0D0
       FJACC(1,2)=-15.0D0/(XC(3)**2+30.0D0*XC(2)*XC(3)+225.0D0*XC(2)**2)
       FJACC(1,3)=-1.0D0/(XC(3)**2+30.0D0*XC(2)*XC(3)+225.0D0*XC(2)**2)
       FJACC(2,1)=1.0D0
       FJACC(2,2)=-7.0D0/(XC(3)**2+14.0D0*XC(2)*XC(3)+49.0D0*XC(2)**2)
       FJACC(2,3)=-1.0D0/(XC(3)**2+14.0D0*XC(2)*XC(3)+49.0D0*XC(2)**2)
       FJACC(3,1)=1.0D0
       FJACC(3,2)=((-0.1110223024625157D-15*XC(3))-4.333333333333333D0)/(
      &XC(3)**2+8.666666666666666D0*XC(2)*XC(3)+18.77777777777778D0*XC(2)
      &**2)
       FJACC(3,3)=(0.1110223024625157D-15*XC(2)-1.0D0)/(XC(3)**2+8.666666
      &666666666D0*XC(2)*XC(3)+18.77777777777778D0*XC(2)**2)
       FJACC(4,1)=1.0D0
       FJACC(4,2)=-3.0D0/(XC(3)**2+6.0D0*XC(2)*XC(3)+9.0D0*XC(2)**2)
       FJACC(4,3)=-1.0D0/(XC(3)**2+6.0D0*XC(2)*XC(3)+9.0D0*XC(2)**2)
       FJACC(5,1)=1.0D0
       FJACC(5,2)=((-0.1110223024625157D-15*XC(3))-2.2D0)/(XC(3)**2+4.399
```

```
     &999999999999D0*XC(2)*XC(3)+4.839999999999998D0*XC(2)**2)
      FJACC(5,3)=(0.1110223024625157D-15*XC(2)-1.0D0)/(XC(3)**2+4.399999
     &999999999D0*XC(2)*XC(3)+4.839999999999998D0*XC(2)**2)
      FJACC(6,1)=1.0D0
      FJACC(6,2)=((-0.2220446049250313D-15*XC(3))-1.666666666666667D0)/(
     &XC(3)**2+3.333333333333333D0*XC(2)*XC(3)+2.777777777777777D0*XC(2)
     &**2)
      FJACC(6,3)=(0.2220446049250313D-15*XC(2)-1.0D0)/(XC(3)**2+3.333333
     &333333333D0*XC(2)*XC(3)+2.777777777777777D0*XC(2)**2)
      FJACC(7,1)=1.0D0
      FJACC(7,2)=((-0.5551115123125783D-16*XC(3))-1.285714285714286D0)/(
     &XC(3)**2+2.571428571428571D0*XC(2)*XC(3)+1.653061224489796D0*XC(2)
     &**2)
      FJACC(7,3)=(0.5551115123125783D-16*XC(2)-1.0D0)/(XC(3)**2+2.571428
     &571428571D0*XC(2)*XC(3)+1.653061224489796D0*XC(2)**2)
      FJACC(8,1)=1.0D0
      FJACC(8,2)=-1.0D0/(XC(3)**2+2.0D0*XC(2)*XC(3)+XC(2)**2)
      FJACC(8,3)=-1.0D0/(XC(3)**2+2.0D0*XC(2)*XC(3)+XC(2)**2)
      FJACC(9,1)=1.0D0
      FJACC(9,2)=-1.285714285714286D0/(XC(3)**2+2.0D0*XC(2)*XC(3)+XC(2)*
     &*2)
      FJACC(9,3)=-1.285714285714286D0/(XC(3)**2+2.0D0*XC(2)*XC(3)+XC(2)*
     &*2)
      FJACC(10,1)=1.0D0
      FJACC(10,2)=-1.666666666666667D0/(XC(3)**2+2.0D0*XC(2)*XC(3)+XC(2)
     &**2)
      FJACC(10,3)=-1.666666666666667D0/(XC(3)**2+2.0D0*XC(2)*XC(3)+XC(2)
     &**2)
      FJACC(11,1)=1.0D0
      FJACC(11,2)=-2.2D0/(XC(3)**2+2.0D0*XC(2)*XC(3)+XC(2)**2)
      FJACC(11,3)=-2.2D0/(XC(3)**2+2.0D0*XC(2)*XC(3)+XC(2)**2)
      FJACC(12,1)=1.0D0
      FJACC(12,2)=-3.0D0/(XC(3)**2+2.0D0*XC(2)*XC(3)+XC(2)**2)
      FJACC(12,3)=-3.0D0/(XC(3)**2+2.0D0*XC(2)*XC(3)+XC(2)**2)
      FJACC(13,1)=1.0D0
      FJACC(13,2)=-4.333333333333333D0/(XC(3)**2+2.0D0*XC(2)*XC(3)+XC(2)
     &**2)
      FJACC(13,3)=-4.333333333333333D0/(XC(3)**2+2.0D0*XC(2)*XC(3)+XC(2)
     &**2)
      FJACC(14,1)=1.0D0
      FJACC(14,2)=-7.0D0/(XC(3)**2+2.0D0*XC(2)*XC(3)+XC(2)**2)
      FJACC(14,3)=-7.0D0/(XC(3)**2+2.0D0*XC(2)*XC(3)+XC(2)**2)
      FJACC(15,1)=1.0D0
      FJACC(15,2)=-15.0D0/(XC(3)**2+2.0D0*XC(2)*XC(3)+XC(2)**2)
      FJACC(15,3)=-15.0D0/(XC(3)**2+2.0D0*XC(2)*XC(3)+XC(2)**2)
      RETURN
      END
\end{verbatim}
\end{page}
```

———-

Asp20 Example Code

— aspex.ht —

```
\begin{page}{Asp20ExampleCode}{Asp20 Example Code}
\begin{verbatim}
      SUBROUTINE QPHESS(N,NROWH,NCOLH,JTHCOL,HESS,X,HX)
      DOUBLE PRECISION HX(N),X(N),HESS(NROWH,NCOLH)
      INTEGER JTHCOL,N,NROWH,NCOLH
      HX(1)=2.0D0*X(1)
      HX(2)=2.0D0*X(2)
      HX(3)=2.0D0*X(4)+2.0D0*X(3)
      HX(4)=2.0D0*X(4)+2.0D0*X(3)
      HX(5)=2.0D0*X(5)
      HX(6)=(-2.0D0*X(7))+(-2.0D0*X(6))
      HX(7)=(-2.0D0*X(7))+(-2.0D0*X(6))
      RETURN
      END
\end{verbatim}
\end{page}
```

———-

Asp24 Example Code

— aspex.ht —

```
\begin{page}{Asp24ExampleCode}{Asp24 Example Code}
\begin{verbatim}
      SUBROUTINE FUNCT1(N,XC,FC)
      DOUBLE PRECISION FC,XC(N)
      INTEGER N
      FC=10.0D0*XC(4)**4+(-40.0D0*XC(1)*XC(4)**3)+(60.0D0*XC(1)**2+5
     &.0D0)*XC(4)**2+((-10.0D0*XC(3))+(-40.0D0*XC(1)**3))*XC(4)+16.0D0*X
     &C(3)**4+(-32.0D0*XC(2)*XC(3)**3)+(24.0D0*XC(2)**2+5.0D0)*XC(3)**2+
     &(-8.0D0*XC(2)**3*XC(3))+XC(2)**4+100.0D0*XC(2)**2+20.0D0*XC(1)*XC(
     &2)+10.0D0*XC(1)**4+XC(1)**2
      RETURN
      END
\end{verbatim}
\end{page}
```

Asp27 Example Code

— aspex.ht —

```
\begin{page}{Asp27ExampleCode}{Asp27 Example Code}
\begin{verbatim}
      FUNCTION DOT(IFLAG,N,Z,W,RWORK,LRWORK,IWORK,LIWORK)
      DOUBLE PRECISION W(N),Z(N),RWORK(LRWORK)
      INTEGER N,LIWORK,IFLAG,LRWORK,IWORK(LIWORK)
      DOT=(W(16)+(-0.5D0*W(15)))*Z(16)+((-0.5D0*W(16))+W(15)+(-0.5D0*W(1
     &4)))*Z(15)+((-0.5D0*W(15))+W(14)+(-0.5D0*W(13)))*Z(14)+((-0.5D0*W(
     &14))+W(13)+(-0.5D0*W(12)))*Z(13)+((-0.5D0*W(13))+W(12)+(-0.5D0*W(1
     &1)))*Z(12)+((-0.5D0*W(12))+W(11)+(-0.5D0*W(10)))*Z(11)+((-0.5D0*W(
     &11))+W(10)+(-0.5D0*W(9)))*Z(10)+((-0.5D0*W(10))+W(9)+(-0.5D0*W(8))
     &)*Z(9)+((-0.5D0*W(9))+W(8)+(-0.5D0*W(7)))*Z(8)+((-0.5D0*W(8))+W(7)
     &+(-0.5D0*W(6)))*Z(7)+((-0.5D0*W(7))+W(6)+(-0.5D0*W(5)))*Z(6)+((-0.
     &5D0*W(6))+W(5)+(-0.5D0*W(4)))*Z(5)+((-0.5D0*W(5))+W(4)+(-0.5D0*W(3
     &)))*Z(4)+((-0.5D0*W(4))+W(3)+(-0.5D0*W(2)))*Z(3)+((-0.5D0*W(3))+W(
     &2)+(-0.5D0*W(1)))*Z(2)+((-0.5D0*W(2))+W(1))*Z(1)
      RETURN
      END
\end{verbatim}
\end{page}
```

Asp28 Example Code

— aspex.ht —

```
\begin{page}{Asp28ExampleCode}{Asp28 Example Code}
\begin{verbatim}
      SUBROUTINE IMAGE(IFLAG,N,Z,W,RWORK,LRWORK,IWORK,LIWORK)
      DOUBLE PRECISION Z(N),W(N),IWORK(LRWORK),RWORK(LRWORK)
      INTEGER N,LIWORK,IFLAG,LRWORK
      W(1)=0.01707454969713436D0*Z(16)+0.001747395874954051D0*Z(15)+0.00
     &2106973900813502D0*Z(14)+0.002957434991769087D0*Z(13)+(-0.00700554
     &0882865317D0*Z(12))+(-0.01219194009813166D0*Z(11))+0.0037230647365
     &3087D0*Z(10)+0.04932374658377151D0*Z(9)+(-0.03586220812223305D0*Z(
     &8))+(-0.04723268012114625D0*Z(7))+(-0.02434652144032987D0*Z(6))+0.
     &2264766947290192D0*Z(5)+(-0.1385343580686922D0*Z(4))+(-0.116530050
     &8238904D0*Z(3))+(-0.2803531651057233D0*Z(2))+1.019463911841327D0*Z
```

```
&(1)
 W(2)=0.0227345011107737D0*Z(16)+0.00881232119739807 2D0*Z(15)+0.010
&94012210519586D0*Z(14)+(-0.0176407246399974 4D0*Z(13))+(-0.01357136
&72105995D0*Z(12))+0.0015746615736227 2D0*Z(11)+0.0525888918633828 2D
&0*Z(10)+(-0.0198153238824337 9D0*Z(9))+(-0.0609539068867969 7D0*Z(8)
&)+(-0.0415311995556905 1D0*Z(7))+0.217656107657146 5D0*Z(6)+(-0.0532
&5555586632358D0*Z(5))+(-0.168897736898464 1D0*Z(4))+(-0.32440166056
&67343D0*Z(3))+0.912822294187217 3D0*Z(2)+(-0.241965270341542 9D0*Z(1
&))
 W(3)=0.0337119819719030 2D0*Z(16)+0.0202160315012226 5D0*Z(15)+(-0.0
&0660730553468970 2D0*Z(14))+(-0.0303239223896817 9D0*Z(13))+0.002033
&30523102494 8D0*Z(12)+0.0537594495676772 8D0*Z(11)+(-0.0163213312502
&9967D0*Z(10))+(-0.0548318656203551 2D0*Z(9))+(-0.0490142882257987 2D
&0*Z(8))+0.209109792788761 2D0*Z(7)+(-0.0576056034138311 3D0*Z(6))+(-
&0.123667920615640 3D0*Z(5))+(-0.352368385302625 9D0*Z(4))+0.88929961
&32269974D0*Z(3)+(-0.299542954578145 7D0*Z(2))+(-0.0298658281257491 7
&D0*Z(1))
 W(4)=0.0514156371366011 9D0*Z(16)+0.00523916596077929 9D0*Z(15)+(-0.
&0162342773577969 9D0*Z(14))+(-0.0196580974604037 1D0*Z(13))+0.054688
&97337339577D0*Z(12)+(-0.014224695935687D0*Z(11))+(-0.0505181779315
&6355D0*Z(10))+(-0.0435307420607649 1D0*Z(9))+0.201223049753072 6D0*Z
&(8)+(-0.0663087451453595 2D0*Z(7))+(-0.128082996372005 3D0*Z(6))+(-0
&.305169742604165D0*Z(5))+0.860042712845019 1D0*Z(4)+(-0.32415033802
&68184D0*Z(3))+(-0.0903353198069331 4D0*Z(2))+0.0908920551710911 1D0*
&Z(1)
 W(5)=0.0455636976777637 5D0*Z(16)+(-0.00182273769758186 9D0*Z(15))+(
&-0.00251222650194185 6D0*Z(14))+0.0294704646070737 9D0*Z(13)+(-0.014
&4507963208617 7D0*Z(12))+(-0.0503424219661493 7D0*Z(11))+(-0.0376966
&3291725935D0*Z(10))+0.217110310217519 8D0*Z(9)+(-0.0824949256021352
&4D0*Z(8))+(-0.147399520928894 5D0*Z(7))+(-0.315042193418466D0*Z(6))
&+0.959162334782400 2D0*Z(5)+(-0.385239695376304 5D0*Z(4))+(-0.141718
&5427288274D0*Z(3))+(-0.0342349546101104 3D0*Z(2))+0.319820917706851
&6D0*Z(1)
 W(6)=0.0401514727740574 4D0*Z(16)+0.0132858574134155 9D0*Z(15)+0.048
&2608200546596 5D0*Z(14)+(-0.0431964111620770 6D0*Z(13))+(-0.04931323
&319055762D0*Z(12))+(-0.0352688631750547 4D0*Z(11))+0.22295383396730
&01D0*Z(10)+(-0.0737531764931515 5D0*Z(9))+(-0.158939131199156 1D0*Z(
&8))+(-0.328001910890377D0*Z(7))+0.952576555482747D0*Z(6)+(-0.31583
&0997578673 1D0*Z(5))+(-0.184688204222538 3D0*Z(4))+(-0.0703762046700
&4427D0*Z(3))+0.231185296432738 2D0*Z(2)+0.0425408349182502 5D0*Z(1)
 W(7)=0.0606977896402371 8D0*Z(16)+0.0668126388467132 2D0*Z(15)+(-0.0
&2113506688615768D0*Z(14))+(-0.083996867458326D0*Z(13))+(-0.0329843
&8523869648D0*Z(12))+0.227687832632773 4D0*Z(11)+(-0.067356038933017
&95D0*Z(10))+(-0.155981396538221 8D0*Z(9))+(-0.336326295769470 5D0*Z(
&8))+0.944279115856094 8D0*Z(7)+(-0.319995524940465 7D0*Z(6))+(-0.136
&2463839920727D0*Z(5))+(-0.100618517157058 6D0*Z(4))+0.2057504515015
&423D0*Z(3)+(-0.0206587926928670 7D0*Z(2))+0.0316099026674551 3D0*Z(1
&)
 W(8)=0.126386868896738D0*Z(16)+0.0025633700394764 18D0*Z(15)+(-0.05
&58175773945564 1D0*Z(14))+(-0.0777789320590068 5D0*Z(13))+0.23117338
```

```
&45834199D0*Z(12)+(-0.06031581134427592D0*Z(11))+(-0.14805474755869
&52D0*Z(10))+(-0.3364014128402243D0*Z(9))+0.9364014128402244D0*Z(8)
&+(-0.3269452524413048D0*Z(7))+(-0.1396841886557241D0*Z(6))+(-0.056
&1733845834199D0*Z(5))+0.1777789320590069D0*Z(4)+(-0.04418242260544
&359D0*Z(3))+(-0.02756337003947642D0*Z(2))+0.07361313110326199D0*Z(
&1)
 W(9)=0.07361313110326199D0*Z(16)+(-0.02756337003947642D0*Z(15))+(-
&0.04418242260544359D0*Z(14))+0.1777789320590069D0*Z(13)+(-0.056173
&3845834199D0*Z(12))+(-0.1396841886557241D0*Z(11))+(-0.326945252441
&3048D0*Z(10))+0.9364014128402244D0*Z(9)+(-0.3364014128402243D0*Z(8
&))+(-0.1480547475586952D0*Z(7))+(-0.06031581134427592D0*Z(6))+0.23
&11733845834199D0*Z(5)+(-0.07777893205900685D0*Z(4))+(-0.0558175773
&9455641D0*Z(3))+0.002563370039476418D0*Z(2)+0.126386868896738D0*Z(
&1)
 W(10)=0.03160990266745513D0*Z(16)+(-0.02065879269286707D0*Z(15))+0
&.2057504515015423D0*Z(14)+(-0.1006185171570586D0*Z(13))+(-0.136246
&3839920727D0*Z(12))+(-0.3199955249404657D0*Z(11))+0.94427911585609
&48D0*Z(10)+(-0.3363262957694705D0*Z(9))+(-0.1559813965382218D0*Z(8
&))+(-0.06735603893301795D0*Z(7))+0.2276878326327734D0*Z(6)+(-0.032
&98438523869648D0*Z(5))+(-0.083996867458326D0*Z(4))+(-0.02113506688
&615768D0*Z(3))+0.06681263884671322D0*Z(2)+0.06069778964023718D0*Z(
&1)
 W(11)=0.04254083491825025D0*Z(16)+0.2311852964327382D0*Z(15)+(-0.0
&7037620467004427D0*Z(14))+(-0.1846882042225383D0*Z(13))+(-0.315830
&9975786731D0*Z(12))+0.952576555482747D0*Z(11)+(-0.328001910890377D
&0*Z(10))+(-0.1589391311991561D0*Z(9))+(-0.07375317649315155D0*Z(8)
&)+0.2229538339673001D0*Z(7)+(-0.03526886317505474D0*Z(6))+(-0.0493
&1323319055762D0*Z(5))+(-0.04319641116207706D0*Z(4))+0.048260820054
&65965D0*Z(3)+0.01328585741341559D0*Z(2)+0.04015147277405744D0*Z(1)
 W(12)=0.3198209177068516D0*Z(16)+(-0.03423495461011043D0*Z(15))+(-
&0.1417185427288274D0*Z(14))+(-0.3852396953763045D0*Z(13))+0.959162
&3347824002D0*Z(12)+(-0.315042193418466D0*Z(11))+(-0.14739952092889
&45D0*Z(10))+(-0.08249492560213524D0*Z(9))+0.2171103102175198D0*Z(8
&)+(-0.03769663291725935D0*Z(7))+(-0.05034242196614937D0*Z(6))+(-0.
&01445079632086177D0*Z(5))+0.02947046460707379D0*Z(4)+(-0.002512226
&501941856D0*Z(3))+(-0.001822737697581869D0*Z(2))+0.045563697677763
&75D0*Z(1)
 W(13)=0.09089205517109111D0*Z(16)+(-0.09033531980693314D0*Z(15))+(
&-0.3241503380268184D0*Z(14))+0.8600427128450191D0*Z(13)+(-0.305169
&742604165D0*Z(12))+(-0.1280829963720053D0*Z(11))+(-0.0663087451453
&5952D0*Z(10))+0.2012230497530726D0*Z(9)+(-0.04353074206076491D0*Z(
&8))+(-0.05051817793156355D0*Z(7))+(-0.014224695935687D0*Z(6))+0.05
&468897337339577D0*Z(5)+(-0.01965809746040371D0*Z(4))+(-0.016234277
&35779699D0*Z(3))+0.005239165960779299D0*Z(2)+0.05141563713660119D0
&*Z(1)
 W(14)=(-0.02986582812574917D0*Z(16))+(-0.2995429545781457D0*Z(15))
&+0.8892996132269974D0*Z(14)+(-0.3523683853026259D0*Z(13))+(-0.1236
&679206156403D0*Z(12))+(-0.05760560341383113D0*Z(11))+0.20910979278
&87612D0*Z(10)+(-0.04901428822579872D0*Z(9))+(-0.05483186562035512D
&0*Z(8))+(-0.01632133125029967D0*Z(7))+0.05375944956767728D0*Z(6)+0
```

```
     &.002033305231024948D0*Z(5)+(-0.03032392238968179D0*Z(4))+(-0.00660
     &7305534689702D0*Z(3))+0.0202160315012265D0*Z(2)+0.033711981971903
     &02D0*Z(1)
      W(15)=(-0.2419652703415429D0*Z(16))+0.9128222941872173D0*Z(15)+(-0
     &.3244016605667343D0*Z(14))+(-0.1688977368984641D0*Z(13))+(-0.05325
     &555586632358D0*Z(12))+0.2176561076571465D0*Z(11)+(-0.0415311995556
     &9051D0*Z(10))+(-0.06095390688679697D0*Z(9))+(-0.01981532388243379D
     &0*Z(8))+0.05258889186338282D0*Z(7)+0.0015746615736227 2D0*Z(6)+(-0.
     &0135713672105995D0*Z(5))+(-0.0176407246399974 4D0*Z(4))+0.010940122
     &10519586D0*Z(3)+0.008812321197398072D0*Z(2)+0.0227345011107737D0*Z
     &(1)
      W(16)=1.019463911841327D0*Z(16)+(-0.2803531651057233D0*Z(15))+(-0.
     &1165300508238904D0*Z(14))+(-0.1385343580686922D0*Z(13))+0.22647669
     &47290192D0*Z(12)+(-0.02434652144032987D0*Z(11))+(-0.04723268012114
     &625D0*Z(10))+(-0.03586220812223305D0*Z(9))+0.04932374658377151D0*Z
     &(8)+0.00372306473653087D0*Z(7)+(-0.01219194009813166D0*Z(6))+(-0.0
     &07005540882865317D0*Z(5))+0.002957434991769087D0*Z(4)+0.0021069739
     &00813502D0*Z(3)+0.001747395874954051D0*Z(2)+0.01707454969713436D0*
     &Z(1)
      RETURN
      END
\end{verbatim}
\end{page}
```

———-

Asp29 Example Code

— **aspex.ht** —

```
\begin{page}{Asp29ExampleCode}{Asp29 Example Code}
\begin{verbatim}
      SUBROUTINE MONIT(ISTATE,NEXTIT,NEVALS,NEVECS,K,F,D)
      DOUBLE PRECISION D(K),F(K)
      INTEGER K,NEXTIT,NEVALS,NVECS,ISTATE
      CALL F02FJZ(ISTATE,NEXTIT,NEVALS,NEVECS,K,F,D)
      RETURN
      END
\end{verbatim}
\end{page}
```

———-

Asp30 Example Code

— **aspex.ht** —

```
\begin{page}{Asp30ExampleCode}{Asp30 Example Code}
\begin{verbatim}
      SUBROUTINE APROD(MODE,M,N,X,Y,RWORK,LRWORK,IWORK,LIWORK)
      DOUBLE PRECISION X(N),Y(M),RWORK(LRWORK)
      INTEGER M,N,LIWORK,IFAIL,LRWORK,IWORK(LIWORK),MODE
      DOUBLE PRECISION A(5,5)
      EXTERNAL F06PAF
      A(1,1)=1.0D0
      A(1,2)=0.0D0
      A(1,3)=0.0D0
      A(1,4)=-1.0D0
      A(1,5)=0.0D0
      A(2,1)=0.0D0
      A(2,2)=1.0D0
      A(2,3)=0.0D0
      A(2,4)=0.0D0
      A(2,5)=-1.0D0
      A(3,1)=0.0D0
      A(3,2)=0.0D0
      A(3,3)=1.0D0
      A(3,4)=-1.0D0
      A(3,5)=0.0D0
      A(4,1)=-1.0D0
      A(4,2)=0.0D0
      A(4,3)=-1.0D0
      A(4,4)=4.0D0
      A(4,5)=-1.0D0
      A(5,1)=0.0D0
      A(5,2)=-1.0D0
      A(5,3)=0.0D0
      A(5,4)=-1.0D0
      A(5,5)=4.0D0
      IF(MODE.EQ.1)THEN
        CALL F06PAF('N',M,N,1.0D0,A,M,X,1,1.0D0,Y,1)
      ELSEIF(MODE.EQ.2)THEN
        CALL F06PAF('T',M,N,1.0D0,A,M,Y,1,1.0D0,X,1)
      ENDIF
      RETURN
      END
\end{verbatim}
\end{page}
```

————

Asp31 Example Code

— aspex.ht —

```
\begin{page}{Asp31ExampleCode}{Asp31 Example Code}
\begin{verbatim}
      SUBROUTINE PEDERV(X,Y,PW)
      DOUBLE PRECISION X,Y(*)
      DOUBLE PRECISION PW(3,3)
      PW(1,1)=-0.03999999999999999D0
      PW(1,2)=10000.0D0*Y(3)
      PW(1,3)=10000.0D0*Y(2)
      PW(2,1)=0.03999999999999999D0
      PW(2,2)=(-10000.0D0*Y(3))+(-60000000.0D0*Y(2))
      PW(2,3)=-10000.0D0*Y(2)
      PW(3,1)=0.0D0
      PW(3,2)=60000000.0D0*Y(2)
      PW(3,3)=0.0D0
      RETURN
      END
\end{verbatim}
\end{page}
```

————

Asp33 Example Code

— aspex.ht —

```
\begin{page}{Asp33ExampleCode}{Asp33 Example Code}
\begin{verbatim}
      SUBROUTINE REPORT(X,V,JINT)
      DOUBLE PRECISION V(3),X
      INTEGER JINT
      RETURN
      END
\end{verbatim}
\end{page}
```

————

Asp34 Example Code

— **aspex.ht** —

```
\begin{page}{Asp34ExampleCode}{Asp34 Example Code}
\begin{verbatim}
      SUBROUTINE MSOLVE(IFLAG,N,X,Y,RWORK,LRWORK,IWORK,LIWORK)
      DOUBLE PRECISION RWORK(LRWORK),X(N),Y(N)
      INTEGER I,J,N,LIWORK,IFLAG,LRWORK,IWORK(LIWORK)
      DOUBLE PRECISION W1(3),W2(3),MS(3,3)
      IFLAG=-1
      MS(1,1)=2.0D0
      MS(1,2)=1.0D0
      MS(1,3)=0.0D0
      MS(2,1)=1.0D0
      MS(2,2)=2.0D0
      MS(2,3)=1.0D0
      MS(3,1)=0.0D0
      MS(3,2)=1.0D0
      MS(3,3)=2.0D0
      CALL F04ASF(MS,N,X,N,Y,W1,W2,IFLAG)
      IFLAG=-IFLAG
      RETURN
      END
\end{verbatim}
\end{page}
```

———

Asp35 Example Code

— **aspex.ht** —

```
\begin{page}{Asp35ExampleCode}{Asp35 Example Code}
\begin{verbatim}
      SUBROUTINE FCN(N,X,FVEC,FJAC,LDFJAC,IFLAG)
      DOUBLE PRECISION X(N),FVEC(N),FJAC(LDFJAC,N)
      INTEGER LDFJAC,N,IFLAG
      IF(IFLAG.EQ.1)THEN
        FVEC(1)=(-1.0D0*X(2))+X(1)
        FVEC(2)=(-1.0D0*X(3))+2.0D0*X(2)
        FVEC(3)=3.0D0*X(3)
      ELSEIF(IFLAG.EQ.2)THEN
        FJAC(1,1)=1.0D0
        FJAC(1,2)=-1.0D0
```

```
        FJAC(1,3)=0.0D0
        FJAC(2,1)=0.0D0
        FJAC(2,2)=2.0D0
        FJAC(2,3)=-1.0D0
        FJAC(3,1)=0.0D0
        FJAC(3,2)=0.0D0
        FJAC(3,3)=3.0D0
      ENDIF
      END
\end{verbatim}
\end{page}
```

——-

Asp4 Example Code

— aspex.ht —

```
\begin{page}{Asp4ExampleCode}{Asp4 Example Code}
\begin{verbatim}
      DOUBLE PRECISION FUNCTION FUNCTN(NDIM,X)
      DOUBLE PRECISION X(NDIM)
      INTEGER NDIM
      FUNCTN=(4.0D0*X(1)*X(3)**2*DEXP(2.0D0*X(1)*X(3)))/(X(4)**2+(2.0D0*
     &X(2)+2.0D0)*X(4)+X(2)**2+2.0D0*X(2)+1.0D0)
      RETURN
      END
\end{verbatim}
\end{page}
```

——-

Asp41 Example Code

— aspex.ht —

```
\begin{page}{Asp41ExampleCode}{Asp41 Example Code}
\begin{verbatim}
      SUBROUTINE FCN(X,EPS,Y,F,N)
      DOUBLE PRECISION EPS,F(N),X,Y(N)
      INTEGER N
      F(1)=Y(2)
      F(2)=Y(3)
```

```
      F(3)=(-1.0D0*Y(1)*Y(3))+2.0D0*EPS*Y(2)**2+(-2.0D0*EPS)
      RETURN
      END
      SUBROUTINE JACOBF(X,EPS,Y,F,N)
      DOUBLE PRECISION EPS,F(N,N),X,Y(N)
      INTEGER N
      F(1,1)=0.0D0
      F(1,2)=1.0D0
      F(1,3)=0.0D0
      F(2,1)=0.0D0
      F(2,2)=0.0D0
      F(2,3)=1.0D0
      F(3,1)=-1.0D0*Y(3)
      F(3,2)=4.0D0*EPS*Y(2)
      F(3,3)=-1.0D0*Y(1)
      RETURN
      END
      SUBROUTINE JACEPS(X,EPS,Y,F,N)
      DOUBLE PRECISION EPS,F(N),X,Y(N)
      INTEGER N
      F(1)=0.0D0
      F(2)=0.0D0
      F(3)=2.0D0*Y(2)**2-2.0D0
      RETURN
      END
\end{verbatim}
\end{page}
```

———-

Asp42 Example Code

— **aspex.ht** —

```
\begin{page}{Asp42ExampleCode}{Asp42 Example Code}
\begin{verbatim}
      SUBROUTINE G(EPS,YA,YB,BC,N)
      DOUBLE PRECISION EPS,YA(N),YB(N),BC(N)
      INTEGER N
      BC(1)=YA(1)
      BC(2)=YA(2)
      BC(3)=YB(2)-1.0D0
      RETURN
      END
      SUBROUTINE JACOBG(EPS,YA,YB,AJ,BJ,N)
      DOUBLE PRECISION EPS,YA(N),AJ(N,N),BJ(N,N),YB(N)
      INTEGER N
```

```
      AJ(1,1)=1.0D0
      AJ(1,2)=0.0D0
      AJ(1,3)=0.0D0
      AJ(2,1)=0.0D0
      AJ(2,2)=1.0D0
      AJ(2,3)=0.0D0
      AJ(3,1)=0.0D0
      AJ(3,2)=0.0D0
      AJ(3,3)=0.0D0
      BJ(1,1)=0.0D0
      BJ(1,2)=0.0D0
      BJ(1,3)=0.0D0
      BJ(2,1)=0.0D0
      BJ(2,2)=0.0D0
      BJ(2,3)=0.0D0
      BJ(3,1)=0.0D0
      BJ(3,2)=1.0D0
      BJ(3,3)=0.0D0
      RETURN
      END
      SUBROUTINE JACGEP(EPS,YA,YB,BCEP,N)
      DOUBLE PRECISION EPS,YA(N),YB(N),BCEP(N)
      INTEGER N
      BCEP(1)=0.0D0
      BCEP(2)=0.0D0
      BCEP(3)=0.0D0
      RETURN
      END
\end{verbatim}
\end{page}
```

————-

Asp49 Example Code

— aspex.ht —

```
\begin{page}{Asp49ExampleCode}{Asp49 Example Code}
\begin{verbatim}
      SUBROUTINE OBJFUN(MODE,N,X,OBJF,OBJGRD,NSTATE,IUSER,USER)
      DOUBLE PRECISION X(N),OBJF,OBJGRD(N),USER(*)
      INTEGER N,IUSER(*),MODE,NSTATE
      OBJF=X(4)*X(9)+((-1.0D0*X(5))+X(3))*X(8)+((-1.0D0*X(3))+X(1))*X(7)
     &+(-1.0D0*X(2)*X(6))
      OBJGRD(1)=X(7)
      OBJGRD(2)=-1.0D0*X(6)
      OBJGRD(3)=X(8)+(-1.0D0*X(7))
```

```
      OBJGRD(4)=X(9)
      OBJGRD(5)=-1.0D0*X(8)
      OBJGRD(6)=-1.0D0*X(2)
      OBJGRD(7)=(-1.0D0*X(3))+X(1)
      OBJGRD(8)=(-1.0D0*X(5))+X(3)
      OBJGRD(9)=X(4)
      RETURN
      END
\end{verbatim}
\end{page}
```

———-

Asp50 Example Code

— aspex.ht —

```
\begin{page}{Asp50ExampleCode}{Asp50 Example Code}
\begin{verbatim}
      SUBROUTINE LSFUN1(M,N,XC,FVECC)
      DOUBLE PRECISION FVECC(M),XC(N)
      INTEGER I,M,N
      FVECC(1)=((XC(1)-2.4D0)*XC(3)+(15.0D0*XC(1)-36.0D0)*XC(2)+1.0D0)/(
     &XC(3)+15.0D0*XC(2))
      FVECC(2)=((XC(1)-2.8D0)*XC(3)+(7.0D0*XC(1)-19.6D0)*XC(2)+1.0D0)/(X
     &C(3)+7.0D0*XC(2))
      FVECC(3)=((XC(1)-3.2D0)*XC(3)+(4.333333333333333D0*XC(1)-13.866666
     &66666667D0)*XC(2)+1.0D0)/(XC(3)+4.333333333333333D0*XC(2))
      FVECC(4)=((XC(1)-3.5D0)*XC(3)+(3.0D0*XC(1)-10.5D0)*XC(2)+1.0D0)/(X
     &C(3)+3.0D0*XC(2))
      FVECC(5)=((XC(1)-3.9D0)*XC(3)+(2.2D0*XC(1)-8.579999999999998D0)*XC
     &(2)+1.0D0)/(XC(3)+2.2D0*XC(2))
      FVECC(6)=((XC(1)-4.199999999999999D0)*XC(3)+(1.666666666666667D0*X
     &C(1)-7.0D0)*XC(2)+1.0D0)/(XC(3)+1.666666666666667D0*XC(2))
      FVECC(7)=((XC(1)-4.5D0)*XC(3)+(1.285714285714286D0*XC(1)-5.7857142
     &85714286D0)*XC(2)+1.0D0)/(XC(3)+1.285714285714286D0*XC(2))
      FVECC(8)=((XC(1)-4.899999999999999D0)*XC(3)+(XC(1)-4.8999999999999
     &99D0)*XC(2)+1.0D0)/(XC(3)+XC(2))
      FVECC(9)=((XC(1)-4.699999999999999D0)*XC(3)+(XC(1)-4.6999999999999
     &99D0)*XC(2)+1.285714285714286D0)/(XC(3)+XC(2))
      FVECC(10)=((XC(1)-6.8D0)*XC(3)+(XC(1)-6.8D0)*XC(2)+1.6666666666666
     &67D0)/(XC(3)+XC(2))
      FVECC(11)=((XC(1)-8.299999999999999D0)*XC(3)+(XC(1)-8.299999999999
     &999D0)*XC(2)+2.2D0)/(XC(3)+XC(2))
      FVECC(12)=((XC(1)-10.6D0)*XC(3)+(XC(1)-10.6D0)*XC(2)+3.0D0)/(XC(3)
     &+XC(2))
      FVECC(13)=((XC(1)-1.34D0)*XC(3)+(XC(1)-1.34D0)*XC(2)+4.33333333333
```

```
     &3333D0)/(XC(3)+XC(2))
      FVECC(14)=((XC(1)-2.1D0)*XC(3)+(XC(1)-2.1D0)*XC(2)+7.0D0)/(XC(3)+X
     &C(2))
      FVECC(15)=((XC(1)-4.39D0)*XC(3)+(XC(1)-4.39D0)*XC(2)+15.0D0)/(XC(3
     &)+XC(2))
      END
\end{verbatim}
\end{page}
```

———-

Asp55 Example Code

— aspex.ht —

```
\begin{page}{Asp55ExampleCode}{Asp55 Example Code}
\begin{verbatim}
      SUBROUTINE CONFUN(MODE,NCNLN,N,NROWJ,NEEDC,X,C,CJAC,NSTATE,IUSER
     &,USER)
      DOUBLE PRECISION C(NCNLN),X(N),CJAC(NROWJ,N),USER(*)
      INTEGER N,IUSER(*),NEEDC(NCNLN),NROWJ,MODE,NCNLN,NSTATE
      IF(NEEDC(1).GT.0)THEN
        C(1)=X(6)**2+X(1)**2
        CJAC(1,1)=2.0D0*X(1)
        CJAC(1,2)=0.0D0
        CJAC(1,3)=0.0D0
        CJAC(1,4)=0.0D0
        CJAC(1,5)=0.0D0
        CJAC(1,6)=2.0D0*X(6)
      ENDIF
      IF(NEEDC(2).GT.0)THEN
        C(2)=X(2)**2+(-2.0D0*X(1)*X(2))+X(1)**2
        CJAC(2,1)=(-2.0D0*X(2))+2.0D0*X(1)
        CJAC(2,2)=2.0D0*X(2)+(-2.0D0*X(1))
        CJAC(2,3)=0.0D0
        CJAC(2,4)=0.0D0
        CJAC(2,5)=0.0D0
        CJAC(2,6)=0.0D0
      ENDIF
      IF(NEEDC(3).GT.0)THEN
        C(3)=X(3)**2+(-2.0D0*X(1)*X(3))+X(2)**2+X(1)**2
        CJAC(3,1)=(-2.0D0*X(3))+2.0D0*X(1)
        CJAC(3,2)=2.0D0*X(2)
        CJAC(3,3)=2.0D0*X(3)+(-2.0D0*X(1))
        CJAC(3,4)=0.0D0
        CJAC(3,5)=0.0D0
        CJAC(3,6)=0.0D0
```

```
      ENDIF
      RETURN
      END
\end{verbatim}
\end{page}
```

———-

Asp6 Example Code

— aspex.ht —

```
\begin{page}{Asp6ExampleCode}{Asp6 Example Code}
\begin{verbatim}
      SUBROUTINE FCN(N,X,FVEC,IFLAG)
      DOUBLE PRECISION X(N),FVEC(N)
      INTEGER N,IFLAG
      FVEC(1)=(-2.0D0*X(2))+(-2.0D0*X(1)**2)+3.0D0*X(1)+1.0D0
      FVEC(2)=(-2.0D0*X(3))+(-2.0D0*X(2)**2)+3.0D0*X(2)+(-1.0D0*X(1))+1.
     &0D0
      FVEC(3)=(-2.0D0*X(4))+(-2.0D0*X(3)**2)+3.0D0*X(3)+(-1.0D0*X(2))+1.
     &0D0
      FVEC(4)=(-2.0D0*X(5))+(-2.0D0*X(4)**2)+3.0D0*X(4)+(-1.0D0*X(3))+1.
     &0D0
      FVEC(5)=(-2.0D0*X(6))+(-2.0D0*X(5)**2)+3.0D0*X(5)+(-1.0D0*X(4))+1.
     &0D0
      FVEC(6)=(-2.0D0*X(7))+(-2.0D0*X(6)**2)+3.0D0*X(6)+(-1.0D0*X(5))+1.
     &0D0
      FVEC(7)=(-2.0D0*X(8))+(-2.0D0*X(7)**2)+3.0D0*X(7)+(-1.0D0*X(6))+1.
     &0D0
      FVEC(8)=(-2.0D0*X(9))+(-2.0D0*X(8)**2)+3.0D0*X(8)+(-1.0D0*X(7))+1.
     &0D0
      FVEC(9)=(-2.0D0*X(9)**2)+3.0D0*X(9)+(-1.0D0*X(8))+1.0D0
      RETURN
      END
\end{verbatim}
\end{page}
```

———-

Asp7 Example Code

— aspex.ht —

```
\begin{page}{Asp7ExampleCode}{Asp7 Example Code}
\begin{verbatim}
      SUBROUTINE FCN(X,Z,F)
      DOUBLE PRECISION F(*),X,Z(*)
      F(1)=DTAN(Z(3))
      F(2)=((-0.03199999999999999D0*DCOS(Z(3))*DTAN(Z(3)))+(-0.02D0*Z(2)
     &**2))/(Z(2)*DCOS(Z(3)))
      F(3)=-0.03199999999999999D0/(X*Z(2)**2)
      RETURN
      END
\end{verbatim}
\end{page}
```

————-

Asp73 Example Code

— **aspex.ht** —

```
\begin{page}{Asp73ExampleCode}{Asp73 Example Code}
\begin{verbatim}
      SUBROUTINE PDEF(X,Y,ALPHA,BETA,GAMMA,DELTA,EPSOLN,PHI,PSI)
      DOUBLE PRECISION ALPHA,EPSOLN,PHI,X,Y,BETA,DELTA,GAMMA,PSI
      ALPHA=DSIN(X)
      BETA=Y
      GAMMA=X*Y
      DELTA=DCOS(X)*DSIN(Y)
      EPSOLN=Y+X
      PHI=X
      PSI=Y
      RETURN
      END
\end{verbatim}
\end{page}
```

————-

Asp74 Example Code

— **aspex.ht** —

```
\begin{page}{Asp74ExampleCode}{Asp74 Example Code}
\begin{verbatim}
```

```
      SUBROUTINE BNDY(X,Y,A,B,C,IBND)
      DOUBLE PRECISION A,B,C,X,Y
      INTEGER IBND
      IF(IBND.EQ.0)THEN
        A=0.0D0
        B=1.0D0
        C=-1.0D0*DSIN(X)
      ELSEIF(IBND.EQ.1)THEN
        A=1.0D0
        B=0.0D0
        C=DSIN(X)*DSIN(Y)
      ELSEIF(IBND.EQ.2)THEN
        A=1.0D0
        B=0.0D0
        C=DSIN(X)*DSIN(Y)
      ELSEIF(IBND.EQ.3)THEN
        A=0.0D0
        B=1.0D0
        C=-1.0D0*DSIN(Y)
      ENDIF
      END
\end{verbatim}
\end{page}
```

———

Asp77 Example Code

— aspex.ht —

```
\begin{page}{Asp77ExampleCode}{Asp77 Example Code}
\begin{verbatim}
      SUBROUTINE FCNF(X,F)
      DOUBLE PRECISION X
      DOUBLE PRECISION F(2,2)
      F(1,1)=0.0D0
      F(1,2)=1.0D0
      F(2,1)=0.0D0
      F(2,2)=-10.0D0
      RETURN
      END
\end{verbatim}
\end{page}
```

———

Asp78 Example Code

— **aspex.ht** —

```
\begin{page}{Asp78ExampleCode}{Asp78 Example Code}
\begin{verbatim}
      SUBROUTINE FCNG(X,G)
      DOUBLE PRECISION G(*),X
      G(1)=0.0D0
      G(2)=0.0D0
      END
\end{verbatim}
\end{page}
```

————

Asp8 Example Code

— **aspex.ht** —

```
\begin{page}{Asp8ExampleCode}{Asp8 Example Code}
\begin{verbatim}
      SUBROUTINE OUTPUT(XSOL,Y,COUNT,M,N,RESULT,FORWRD)
      DOUBLE PRECISION Y(N),RESULT(M,N),XSOL
      INTEGER M,N,COUNT
      LOGICAL FORWRD
      DOUBLE PRECISION X02ALF,POINTS(8)
      EXTERNAL X02ALF
      INTEGER I
      POINTS(1)=1.0D0
      POINTS(2)=2.0D0
      POINTS(3)=3.0D0
      POINTS(4)=4.0D0
      POINTS(5)=5.0D0
      POINTS(6)=6.0D0
      POINTS(7)=7.0D0
      POINTS(8)=8.0D0
      COUNT=COUNT+1
      DO 25001 I=1,N
        RESULT(COUNT,I)=Y(I)
25001 CONTINUE
      IF(COUNT.EQ.M)THEN
        IF(FORWRD)THEN
          XSOL=X02ALF()
        ELSE
```

```
          XSOL=-X02ALF()
        ENDIF
      ELSE
        XSOL=POINTS(COUNT)
      ENDIF
      END
\end{verbatim}
\end{page}
```

———-

Asp80 Example Code

— aspex.ht —

```
\begin{page}{Asp80ExampleCode}{Asp80 Example Code}
\begin{verbatim}
      SUBROUTINE BDYVAL(XL,XR,ELAM,YL,YR)
      DOUBLE PRECISION ELAM,XL,YL(3),XR,YR(3)
      YL(1)=XL
      YL(2)=2.0D0
      YR(1)=1.0D0
      YR(2)=-1.0D0*DSQRT(XR+(-1.0D0*ELAM))
      RETURN
      END
\end{verbatim}
\end{page}
```

———-

Asp9 Example Code

— aspex.ht —

```
\begin{page}{Asp9ExampleCode}{Asp9 Example Code}
\begin{verbatim}
      DOUBLE PRECISION FUNCTION G(X,Y)
      DOUBLE PRECISION X,Y(*)
      G=X+Y(1)
      RETURN
      END
\end{verbatim}
\end{page}
```

Chapter 24

NAG ANNA Expert System

24.1 annaex.ht

Axiom/NAG Expert System

— annaex.ht —

```
\begin{page}{UXANNA}{Axiom/NAG Expert System}
\centerline{\tt{\inputbitmap{\htbmdir{}/anna_logo.xbm}}\rm}
\newline
\centerline{This expert system chooses, and uses, NAG numerical routines.}
\begin{scroll}
\blankline
\indent{2}
\beginmenu
\menumemolink{Integration}{UXANNAInt}
\blankline
\menumemolink{Ordinary Differential Equations}{UXANNAOde}
\blankline
\menumemolink{Partial Differential Equations}{UXANNAPde}
\blankline
\menumemolink{Optimization}{UXANNAOpt}
\vspace{10}
\menumemolink{About the Axiom/NAG Expert System}{UXANNATxt}
%\unixcommand{(Postscript)}{ghostview \ \htbmdir{}/anna.ps}
%\blankline
%\menumemolink{How to use the NAGLINK}{nagLinkIntroPage}
%\blankline
%\menumemolink{Tutorial}{tutorialIntroPage}
%\blankline
%\item \menulispdownlink{Interpolation}{(|interp1|}
```

```
\endmenu
\indent{0}
\end{scroll}
\autobutt{HelpContents}
\end{page}
```

———-

Integration

— **annaex.ht** —

```
\begin{page}{UXANNAInt}{Integration}
\pageto{Integration}{LispFunctions}
\pageto{Multiple Integration}{LispFunctions}
Welcome to the Integration section of {\tt
\inputbitmap{\htbmdir{}/anna.xbm.tiny}}, the {\em Axiom/NAG Expert
System}.  This system chooses, and uses, NAG numerical routines.
\begin{scroll}
\blankline
\indent{2}
\beginmenu
\item \menulispdownlink{Integration}{(|annaInt|)}\space{}\newline
\indent{5} Integrating a function over a finite or infinite range.
\blankline
\item \menulispdownlink{Multiple Integration}{(|annaMInt|)}\space{}
\newline
\indent{5} Integrating a multivariate function over a finite space.
The dimensions of the space need to be 2 <= n <= 15.
\blankline
\menudownlink{Examples}{UXANNAIntEx}\space{}\newline
\indent{5} Examples of integration.  These examples cover all of the major
methods.  Parameters can be changed to investigate the effect
on the choice of method.
\endmenu
\indent{0}
\end{scroll}
\autobutt{MainHelp}
\end{page}
```

———-

Ordinary Differential Equations

⇒ "Ordinary Differential Equations" (LispFunctions) 3.71 on page 952

— annaex.ht —

```
\begin{page}{UXANNAOde}{Ordinary Differential Equations}
Welcome to the Ordinary Differential Equations section of {\tt
\inputbitmap{\htbmdir{}/anna.xbm.tiny}}, the
{\em Axiom/NAG Expert System}.
This system chooses, and uses, NAG numerical routines.
\begin{scroll}
\blankline
\blankline
\indent{2}
\beginmenu
\item \menulispdownlink{Ordinary Differential Equations}{(|annaOde|)}
\space{}\newline
\indent{5} Finding a solution to an Initial Value Problem of a set
of Ordinary Differential Equations.
\blankline
\menudownlink{Examples}{UXANNAOdeEx}\newline
\indent{5} Examples of ODE problems with various features using both
stiff and non-stiff methods.  Parameters can be changed to investigate
the effect on the choice of method.
\autobutt{MainHelp}
\endmenu
\indent{0}
\end{scroll}
\end{page}
```

———-

Optimization

⇒ "Optimization of a set of observations of a data set" (LispFunctions) 3.71 on page 952

— annaex.ht —

```
\begin{page}{UXANNAOpt}{Optimization}
Welcome to the Optimization section of {\tt
\inputbitmap{\htbmdir{}/anna.xbm.tiny}}, the {\em Axiom/NAG Expert System}.
This system chooses, and uses, NAG numerical routines.
\begin{scroll}
\blankline
\indent{2}
\beginmenu
```

```
\item \menulispdownlink{Optimization of a Single Multivariate Function}
{(|annaOpt|)}\space{}\newline
\indent{6} Finding the minimum of a function in n variables.
\newline
\indent{6} Linear Programming and Quadratic Programming problems.
\blankline
\indent{4}
\beginmenu
\menudownlink{Examples}{UXANNAOptEx}\newline
\indent{8}
Examples of optimization problems with various constraint features.
\endmenu
\blankline
\item \menulispdownlink
{Optimization of a set of observations of a data set}
{(|annaOpt2|)}\space{}\newline
\indent{6} Least-squares problems.
\newline
\indent{6} Checking the goodness of fit of a least-squares model.
\blankline
\indent{4}
\beginmenu
\menudownlink{Examples}{UXANNAOpt2Ex}\newline
\indent{8} Examples of least squares problems.
\endmenu
\endmenu
\indent{0}
\end{scroll}
\autobutt{MainHelp}
\end{page}
```

———-

Partial Differential Equations

⇒ "Second Order Elliptic Partial Differential Equation" (LispFunctions) 3.71 on page 952

— **annaex.ht** —

```
\begin{page}{UXANNAPde}{Partial Differential Equations}
Welcome to the Partial Differential Equations section of {\tt
\inputbitmap{\htbmdir{}/anna.xbm.tiny}}, the
{\em Axiom/NAG Expert System}.
\begin{scroll}
\indent{2}
\beginmenu
\menulispdownlink{Second Order Elliptic Partial Differential Equation}
{(|annaPDESolve|)}
```

```
\newline
\indent{4} Discretizing the PDE:
\newline
\centerline{\inputbitmap{\htbmdir{}/d03eef.xbm}}
defined on a rectangular region with boundary conditions of the form
\centerline{\inputbitmap{\htbmdir{}/d03eef1.bitmap}}
and solving the resulting
seven-diagonal finite difference equations using a multigrid technique.
\blankline
%\menulispdownlink{Helmholtz Equation in 3-D, Cartesian Coordinates}
%{(|d03fafVars|)}
%\newline
%\indent{4} Descretizing the PDE:
%\newline
%\centerline{\inputbitmap{\htbmdir{}/d03faf.xbm}}
%and solving the resulting
%seven-diagonal finite difference equations using a method based on
%the Fast Fourier Transform.
\endmenu
\end{scroll}
\autobutt{MainHelp}
\end{page}
```

———-

Examples Using the Axiom/NAG Expert System

— annaex.ht —

```
\begin{page}{UXANNAOptEx}{Examples Using the Axiom/NAG Expert System}
\pageto{Example 1}{LispFunctions}
\pageto{Example 2}{LispFunctions}
\pageto{Example 3}{LispFunctions}
\pageto{Example 4}{LispFunctions}
\pageto{Example 5}{LispFunctions}
\begin{scroll}
Select any of these examples and you will be presented with a page which
contains active areas for the function and its parameters.
\blankline
These parameters can be altered by selecting the area and replacing the
default parameters by the new values.
\blankline
\beginmenu
\item \menulispdownlink{Example 1: \newline
\indent{2} Minimize the function:
\centerline{\inputbitmap{\htbmdir{}/opt1.xbm}}}
{(|annaOptDefaultSolve1|)}\space{}
```

```
\blankline
\item \menulispdownlink{Example 2: \newline
\indent{2} Minimize the function:
\centerline{\inputbitmap{\htbmdir{}/opt2.xbm}}}
{(|annaOptDefaultSolve2|)}\space{}
\newline \indent{3} With conditions:
\centerline{\inputbitmap{\htbmdir{}/opt2c.xbm}}
\blankline
\item \menulispdownlink{Example 3: \newline
\indent{2} Minimize the function:
\centerline{\inputbitmap{\htbmdir{}/opt3.xbm}}}
{(|annaOptDefaultSolve3|)}\space{}
\newline \indent{3} With conditions:
\centerline{\inputbitmap{\htbmdir{}/opt3c1.xbm}}
\centerline{\inputbitmap{\htbmdir{}/opt3c2.xbm}}
\blankline
\item \menulispdownlink{Example 4: \newline
\indent{2} Minimize the function:
\centerline{\inputbitmap{\htbmdir{}/opt4.xbm}}
}{(|annaOptDefaultSolve4|)}\space{}
\newline \indent{3} With conditions:
\centerline{\inputbitmap{\htbmdir{}/opt4c1.xbm}}
\centerline{\inputbitmap{\htbmdir{}/opt4c2.xbm}}
\centerline{\inputbitmap{\htbmdir{}/opt4c3.xbm}}
\blankline
\item \menulispdownlink{Example 5: \newline
\indent{2} Minimize the function:
\centerline{\inputbitmap{\htbmdir{}/opt5.xbm}}
}{(|annaOptDefaultSolve5|)}\space{}
\newline \indent{3} With conditions:
\centerline{\inputbitmap{\htbmdir{}/opt4c1.xbm}}
\centerline{\inputbitmap{\htbmdir{}/opt4c2.xbm}}
\centerline{\inputbitmap{\htbmdir{}/opt4c3.xbm}}
\blankline
\endmenu
\end{scroll}
\autobutt{Mainhelp}
\end{page}
```

———-

Examples Using the Axiom/NAG Expert System

⇒ "Example 1" (LispFunctions) 3.71 on page 952

— **annaex.ht** —

```
\begin{page}{UXANNAOpt2Ex}{Examples Using the Axiom/NAG Expert System}
```

```
\begin{scroll}
Select this example and you will be presented with a page which
contains active areas for the function and its parameters.
\blankline
These parameters can be altered by selecting the area and replacing the
default parameters by the new values.
\blankline
\beginmenu
\blankline
\item \menulispdownlink{Example 1: \newline
\indent{2} Calculate a least-squares
minimization of the following functions:
\centerline{\inputbitmap{\htbmdir{}/opt61.xbm}}
\centerline{\inputbitmap{\htbmdir{}/opt62.xbm}}
\centerline{\inputbitmap{\htbmdir{}/opt63.xbm}}}
{(|annaOpt2DefaultSolve|)}\space{}
\endmenu
\end{scroll}
\autobutt{MainHelp}
\end{page}
```

———-

Examples Using the Axiom/NAG Expert System

⇒ "Example 1" (LispFunctions) 3.71 on page 952
⇒ "Example 2" (LispFunctions) 3.71 on page 952
⇒ "Example 3" (LispFunctions) 3.71 on page 952
⇒ "Example 4" (LispFunctions) 3.71 on page 952
⇒ "Example 5" (LispFunctions) 3.71 on page 952
⇒ "Example 6" (LispFunctions) 3.71 on page 952
⇒ "Example 7" (LispFunctions) 3.71 on page 952
⇒ "Example 8" (LispFunctions) 3.71 on page 952
⇒ "Example 9" (LispFunctions) 3.71 on page 952
⇒ "Example 10" (LispFunctions) 3.71 on page 952
⇒ "Example 11" (LispFunctions) 3.71 on page 952
⇒ "Example 12" (LispFunctions) 3.71 on page 952

— **annaex.ht** —

```
\begin{page}{UXANNAIntEx}{Examples Using the Axiom/NAG Expert System}
\begin{scroll}
Select any of these examples and you will be presented with a page which
contains active areas for the function and its parameters.
\blankline
These parameters can be altered by selecting the area and replacing the
```

```
default parameters by the new values.  In this way you can investigate the
effect of the new parameters on the choice of method.
\blankline
\beginmenu
\item \menulispdownlink{Example 1: \newline
\centerline{\inputbitmap{\htbmdir{}/int1.xbm}}}{(|annaFoo|)}\space{}
\blankline
\item \menulispdownlink{Example 2: \newline
\centerline{\inputbitmap{\htbmdir{}/int2.xbm}}}{(|annaBar|)}\space{}
\blankline
\item \menulispdownlink{Example 3: \newline
\centerline{\inputbitmap{\htbmdir{}/int3.xbm}}}{(|annaJoe|)}\space{}
\blankline
\item \menulispdownlink{Example 4: \newline
\centerline{\inputbitmap{\htbmdir{}/int4.xbm}}}{(|annaSue|)}\space{}
\blankline
\item \menulispdownlink{Example 5: \newline
\centerline{\inputbitmap{\htbmdir{}/int5.xbm}}}{(|annaAnn|)}\space{}
\blankline
\item \menulispdownlink{Example 6: \newline
\centerline{\inputbitmap{\htbmdir{}/int6.xbm}}}{(|annaBab|)}\space{}
\blankline
\item \menulispdownlink{Example 7: \newline
\centerline{\inputbitmap{\htbmdir{}/int7.xbm}}}{(|annaFnar|)}\space{}
\blankline
\item \menulispdownlink{Example 8: \newline
\centerline{\inputbitmap{\htbmdir{}/int8.xbm}}}{(|annaDan|)}\space{}
\blankline
\item \menulispdownlink{Example 9: \newline
\centerline{\inputbitmap{\htbmdir{}/int9.xbm}}}{(|annaBlah|)}\space{}
\blankline
\item \menulispdownlink{Example 10: \newline
\centerline{\inputbitmap{\htbmdir{}/int10.xbm}}}{(|annaTub|)}\space{}
\blankline
\item \menulispdownlink{Example 11: \newline
\centerline{\inputbitmap{\htbmdir{}/int13.xbm}}}{(|annaRats|)}\space{}
\blankline
\item \menulispdownlink{Example 12: \newline
\centerline{\inputbitmap{\htbmdir{}/int11.xbm}}}{(|annaMInt|)}\space{}
\endmenu
\end{scroll}
\autobutt{MainHelp}
\end{page}
```

Examples Using the Axiom/NAG Expert System

⇒ "annaOdeDefaultSolve1" (LispFunctions) 3.71 on page 952
⇒ "annaOdeDefaultSolve2" (LispFunctions) 3.71 on page 952

— annaex.ht —

```
\begin{page}{UXANNAOdeEx}{Examples Using the Axiom/NAG Expert System}
\begin{scroll}
Analyses the function for various attributes, chooses and
then uses a suitable ODE solver to provide a
solution to the system of n ODEs \center{\htbitmap{d02gaf},}
for i = 1,2,...,n.
\blankline
Select either of these examples and you will be presented with a page
which contains active areas for the function and its parameters.
\blankline
These parameters can be altered by selecting the area and replacing
the default parameters by the new values.  In this way you can
investigate the effect of the new parameters on the choice of method.
\blankline
\beginmenu
\item \menulispdownlink{Example 1: \tab{12}
\inputbitmap{\htbmdir{}/ode1.xbm}}{(|annaOdeDefaultSolve1|)}
\blankline with initial conditions: \newline
\tab{12}\inputbitmap{\htbmdir{}/y1.xbm} \space{1} and \space{1}
\inputbitmap{\htbmdir{}/x1.xbm}
\blankline
\blankline
\blankline
\item \menulispdownlink{Example 2: \tab{12}
\inputbitmap{\htbmdir{}/ode2.xbm}}{(|annaOdeDefaultSolve2|)}
\blankline with initial conditions: \newline
\tab{12}\inputbitmap{\htbmdir{}/y2.xbm} \space{1} and \space{1}
\inputbitmap{\htbmdir{}/x1.xbm}
\blankline
\blankline
\blankline
\endmenu
\end{scroll}
\autobutt{MainHelp}
\end{page}
```

———-

About the Axiom/NAG Expert System

⇒ "notitle" (UXANNAEx) 24.1 on page 3984
⇒ "notitle" (UXANNAIntro) 24.1 on page 3983
⇒ "notitle" (UXANNADec) 24.1 on page 3991
⇒ "notitle" (UXANNAInfer) 24.1 on page 3992
⇒ "notitle" (UXANNAMeas) 24.1 on page 3994

— **annaex.ht** —

```
\begin{page}{UXANNATxt}{About the Axiom/NAG Expert System}
\begin{scroll}
\centerline{\tt\inputbitmap{\htbmdir{}/anna_logo.xbm}\rm}
\vspace{-30}\horizontalline
In applied mathematics, electronic and chemical engineering, the
modelling process can produce a number of mathematical problems which
require numerical solutions for which symbolic methods are either not
possible or not obvious.  With the plethora of numerical library
routines for the solution of these problems often the numerical
analyst has to answer the question {\em Which routine to choose?}
\blankline
Some analysis needs to be carried out before the appropriate routine
can be identified i.e. {\em How stiff is this ODE?} and {\em Is this
function continuous?}  It may well be the case that more than one
routine is applicable to the problem.  So the question may become {\em
Which is likely to be the best?}  Such a choice may be critical for
both accuracy and efficiency.
\blankline
An expert system is thus required to make this choice based on the
result of its own analysis of the problem, call the routine and act on
the outcome.  This may be to put the answer in a relevant form or
react to an apparent failure of the chosen routine and thus choose and
call an alternative.  It should also have sufficient explanation
mechanisms to inform on the choice of routine and the reasons for that
choice.
\blankline
\end{scroll}
\downlink{ Examples }{UXANNAEx}
\downlink{ Introduction }{UXANNAIntro}
\downlink{ Decision Agents }{UXANNADec}
\downlink{ Inference Mechanisms }{UXANNAInfer}
\downlink{ Measure Functions }{UXANNAMeas}
\end{page}
```

————

Introduction to the Axiom/NAG Expert System

⇒ "notitle" (UXANNADec) 24.1 on page 3991
⇒ "notitle" (UXANNAInfer) 24.1 on page 3992
⇒ "notitle" (UXANNAMeth) 24.1 on page 3993
⇒ "notitle" (UXANNAMeas) 24.1 on page 3994

— **annaex.ht** —

```
\begin{page}{UXANNAIntro}{Introduction to the Axiom/NAG Expert System}
\begin{scroll}
\centerline{\tt\inputbitmap{\htbmdir{}/anna_logo.xbm}\rm}
\vspace{-30}\horizontalline
Deciding amongst, and then implementing, several possible approaches
to solving a numerical problem can be daunting for a novice user, or
tedious for an expert.  Different attributes of the problem need to be
identified and their possible interactions weighed up before a final
decision about which method to use can be made.
\blankline
The implementation is then largely an automatic, if laborious, process
of writing, compiling and linking usually Fortran code.  The aim is to
build an expert system which will use computer algebra to analyse such
features of a problem, inference mechanisms and a knowledge base to
choose a numerical method appropriate to the solution of a given
problem.
\blankline
Any interactive system is constrained by the need to provide a
reasonable response time for the user.  Given the complexity of some
of the analysis our system will need to do, it is clear that we should
only aim to select a good method, rather than try to identify the best
one available.  The overall goal is to provide a ``black-box''
interface to numerical software which allows non-experts access to its
full potential.  It will also provide explanation mechanisms
commensurate with its role as a teaching aid.
\blankline
Given, say, an integration to perform (which may or may not be able to be
handled symbolically), the system should choose and apply an appropriate
method, thus mirroring as closely as possible the way that an experienced
numerical analyst would think so, for example, given an integration to
perform:\newline
{\it \centerline{\inputbitmap{\htbmdir{}/int1.xbm}}}
\newline
the experienced analyst would see that the integral is semi-infinite and
transform it by splitting the range and transforming the integral over {\it
[1,\inputbitmap{\htbmdir{}/infty.xbm}]} into an integral over
{\it [0,1] } using the transformation {\it x -> 1/t}.
A different numerical routine might be used over each
sub-region and the results added to give the final answer.
\blankline
```

```
It then requires
the translation of the problem into Fortran code which may be extensive.
Even with this simple example, the process is quite involved.
\blankline
\end{scroll}
\autobuttons
\downlink{ Decision Agents }{UXANNADec}
\downlink{ Inference Mechanisms }{UXANNAInfer}
\downlink{ Method Domains }{UXANNAMeth}
\downlink{ Measure Functions }{UXANNAMeas}
\end{page}
```

———-

Example using the Axiom/NAG Expert System

⇒ "notitle" (UXANNAEx2) 24.1 on page 3989

— **annaex.ht** —

```
\begin{page}{UXANNAEx}{Example using the Axiom/NAG Expert System}
\begin{scroll}
\xtc{
{\bf Example 1}: The integral
{\centerline{\inputbitmap{\htbmdir{}/int1.xbm}}}
\newline
is performed as follows:
\blankline
}{}
\xtc{
}{
\spadpaste{ans :=
integrate((exp(-X^3)+exp(-3*X^2))/sqrt(X),0.0..\%plusInfinity)
\bound{ans} }
}
\blankline
\xtc{
It creates a composite structure for which the field containing
the result can be
expanded as required.\blankline
}{
\spadpaste{ans . 'result\free{ans}}
}
\blankline
\xtc{
}{
\spadpaste{ans . 'abserr\free{ans}}
}
```

```
\blankline
This system has performed the analysis described above, done the
necessary problem transformation, written any necessary Fortran,
called two different numerical routines, and amalgamated their
results.  This whole process was transparent to the user.
\end{scroll}
\autobuttons
\downlink{Example 2}{UXANNAEx2}
%\downlink{Decision Agents}{UXANNADec}
\end{page}

\begin{patch}{UXANNAExPatch1}
\begin{paste}{UXANNAExFull1}{UXANNAExEmpty1}
\pastebutton{UXANNAExFull1}{\hidepaste}
\tab{5}\spadcommand{ans := integrate((exp(-X^3)+exp(-3*X^2))/sqrt(X),0.0..\%plusInfinity)\bound{ans }}
\indentrel{3}\begin{verbatim}
   (1)
   [d01ajfAnnaTypeAnswer: Result, abserr: DoubleFloat,
    method: Result, result: DoubleFloat,
    explanations: List(String), attributes: List(Any),
    d01apfAnnaTypeAnswer: Result]
                                          Type: Result
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UXANNAExEmpty1}
\begin{paste}{UXANNAExEmpty1}{UXANNAExPatch1}
\pastebutton{UXANNAExEmpty1}{\showpaste}
\tab{5}\spadcommand{ans := integrate((exp(-X^3)+exp(-3*X^2))/sqrt(X),0.0..\%plusInfinity)\bound{ans }}
\end{paste}\end{patch}

\begin{patch}{UXANNAExPatch2}
\begin{paste}{UXANNAExFull2}{UXANNAExEmpty2}
\pastebutton{UXANNAExFull2}{\hidepaste}
\tab{5}\spadcommand{ans . 'result\free{ans }}
\indentrel{3}\begin{verbatim}
   (2)  3.23287256251958
                                     Type: DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UXANNAExEmpty2}
\begin{paste}{UXANNAExEmpty2}{UXANNAExPatch2}
\pastebutton{UXANNAExEmpty2}{\showpaste}
\tab{5}\spadcommand{ans . 'result\free{ans }}
\end{paste}\end{patch}

\begin{patch}{UXANNAExPatch3}
\begin{paste}{UXANNAExFull3}{UXANNAExEmpty3}
\pastebutton{UXANNAExFull3}{\hidepaste}
```

```
\tab{5}\spadcommand{ans . 'abserr\free{ans }}
\indentrel{3}\begin{verbatim}
   (3)  2.69960156338737e-08
                                      Type: DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UXANNAExEmpty3}
\begin{paste}{UXANNAExEmpty3}{UXANNAExPatch3}
\pastebutton{UXANNAExEmpty3}{\showpaste}
\tab{5}\spadcommand{ans . 'abserr\free{ans }}
\end{paste}\end{patch}

\begin{patch}{UXANNAEx2Patch1}
\begin{paste}{UXANNAEx2Full1}{UXANNAEx2Empty1}
\pastebutton{UXANNAEx2Full1}{\hidepaste}
\tab{5}\spadcommand{ans2 := solve([Y[2],-1001*Y[2]-1000*Y[1]], 0.0, 10.0, [1.0,-1.0], [2,4,6,8],
\indentrel{3}\begin{verbatim}
   (1)
   [ifail: Integer, intensityFunctions: List(String),
    tol: DoubleFloat, result: Matrix(DoubleFloat),
    y: Matrix(DoubleFloat), method: Result,
    explanations: List(String), x: DoubleFloat]
                                           Type: Result
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UXANNAEx2Empty1}
\begin{paste}{UXANNAEx2Empty1}{UXANNAEx2Patch1}
\pastebutton{UXANNAEx2Empty1}{\showpaste}
\tab{5}\spadcommand{ans2 := solve([Y[2],-1001*Y[2]-1000*Y[1]], 0.0, 10.0, [1.0,-1.0], [2,4,6,8],
\end{paste}\end{patch}

\begin{patch}{UXANNAEx2Patch2}
\begin{paste}{UXANNAEx2Full2}{UXANNAEx2Empty2}
\pastebutton{UXANNAEx2Full2}{\hidepaste}
\tab{5}\spadcommand{ans2 . 'result\free{ans2 }}
\indentrel{3}\begin{verbatim}
                1.0                   - 1.0

         0.135329965475963     - 0.135329965475963

   (2)   0.018315370486087     - 0.0183153704860871

        0.00247869592278824   - 0.00247869592278825

        0.000335497710871914  - 0.000335497710871913
                               Type: Matrix DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}
```

```
\begin{patch}{UXANNAEx2Empty2}
\begin{paste}{UXANNAEx2Empty2}{UXANNAEx2Patch2}
\pastebutton{UXANNAEx2Empty2}{\showpaste}
\tab{5}\spadcommand{ans2 . 'result\free{ans2 }}
\end{paste}\end{patch}

\begin{patch}{UXANNAEx2Patch3}
\begin{paste}{UXANNAEx2Full3}{UXANNAEx2Empty3}
\pastebutton{UXANNAEx2Full3}{\hidepaste}
\tab{5}\spadcommand{ans2 . 'y\free{ans2 }}
\indentrel{3}\begin{verbatim}
   (3)  [4.54002176643708e-05  - 4.54002176643708e-05]
                              Type: Matrix DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UXANNAEx2Empty3}
\begin{paste}{UXANNAEx2Empty3}{UXANNAEx2Patch3}
\pastebutton{UXANNAEx2Empty3}{\showpaste}
\tab{5}\spadcommand{ans2 . 'y\free{ans2 }}
\end{paste}\end{patch}

\begin{patch}{UXANNAEx3Patch1}
\begin{paste}{UXANNAEx3Full1}{UXANNAEx3Empty1}
\pastebutton{UXANNAEx3Full1}{\hidepaste}
\tab{5}\spadcommand{ans3 := optimize((X[1]+10*X[2])**2 + 5*(X[3]-X[4])**2 + (X[2]-2*X[3])**4 + 10*(X[1]-X[4]
\indentrel{3}\begin{verbatim}
   (1)
   [ifail: Integer, bl: Matrix(DoubleFloat),
    bu: Matrix(DoubleFloat), method: Result,
    attributes: List(String), explanations: List(String),
    x: Matrix(DoubleFloat), objf: DoubleFloat]
                                          Type: Result
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UXANNAEx3Empty1}
\begin{paste}{UXANNAEx3Empty1}{UXANNAEx3Patch1}
\pastebutton{UXANNAEx3Empty1}{\showpaste}
\tab{5}\spadcommand{ans3 := optimize((X[1]+10*X[2])**2 + 5*(X[3]-X[4])**2 + (X[2]-2*X[3])**4 + 10*(X[1]-X[4]
\end{paste}\end{patch}

\begin{patch}{UXANNAEx3Patch2}
\begin{paste}{UXANNAEx3Full2}{UXANNAEx3Empty2}
\pastebutton{UXANNAEx3Full2}{\hidepaste}
\tab{5}\spadcommand{ans3 . objf\free{ans3 }}
\indentrel{3}\begin{verbatim}
   (2)  2.43378751212073
                                     Type: DoubleFloat
```

```
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UXANNAEx3Empty2}
\begin{paste}{UXANNAEx3Empty2}{UXANNAEx3Patch2}
\pastebutton{UXANNAEx3Empty2}{\showpaste}
\tab{5}\spadcommand{ans3 . objf\free{ans3 }}
\end{paste}\end{patch}

\begin{patch}{UXANNAEx3Patch3}
\begin{paste}{UXANNAEx3Full3}{UXANNAEx3Empty3}
\pastebutton{UXANNAEx3Full3}{\hidepaste}
\tab{5}\spadcommand{ans3 . x\free{ans3 }}
\indentrel{3}\begin{verbatim}
   (3)
   [1.0  - 0.0852325900999037  0.409303588204477  1.0]
                               Type: Matrix DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UXANNAEx3Empty3}
\begin{paste}{UXANNAEx3Empty3}{UXANNAEx3Patch3}
\pastebutton{UXANNAEx3Empty3}{\showpaste}
\tab{5}\spadcommand{ans3 . x\free{ans3 }}
\end{paste}\end{patch}

\begin{patch}{UXANNAEx3Patch4}
\begin{paste}{UXANNAEx3Full4}{UXANNAEx3Empty4}
\pastebutton{UXANNAEx3Full4}{\hidepaste}
\tab{5}\spadcommand{ans3 . attributes\free{ans3 }}
\indentrel{3}\begin{verbatim}
   (4)
   ["The object function is non-linear",
    "There are simple bounds on the variables",
    "There are no constraint functions"]
                                      Type: List String
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UXANNAEx3Empty4}
\begin{paste}{UXANNAEx3Empty4}{UXANNAEx3Patch4}
\pastebutton{UXANNAEx3Empty4}{\showpaste}
\tab{5}\spadcommand{ans3 . attributes\free{ans3 }}
\end{paste}\end{patch}

\begin{patch}{UXANNAAgentPatch1}
\begin{paste}{UXANNAAgentFull1}{UXANNAAgentEmpty1}
\pastebutton{UXANNAAgentFull1}{\hidepaste}
\tab{5}\spadcommand{s := singularitiesOf(tan x,[x],0..12*\%pi)$ESCONT\free{lib3 }}
\indentrel{3}\begin{verbatim}
```

```
   (1)
   [1.5707963267948966, 4.7123889803846897,
    7.8539816339744828, 10.995574287564276,
    14.137166941154069, 17.27875959474386,
    20.420352248333657, 23.561944901923447,
    26.703537555513243, 29.845130209103033, ...]
                                  Type: Stream DoubleFloat
\end{verbatim}
\indentrel{-3}\end{paste}\end{patch}

\begin{patch}{UXANNAAgentEmpty1}
\begin{paste}{UXANNAAgentEmpty1}{UXANNAAgentPatch1}
\pastebutton{UXANNAAgentEmpty1}{\showpaste}
\tab{5}\spadcommand{s := singularitiesOf(tan x,[x],0..12*\%pi)$ESCONT\free{lib3 }}
\end{paste}\end{patch}
```

———-

Example using the Axiom/NAG Expert System

⇒ "notitle" (UXANNAEx3) 24.1 on page 3990

— annaex.ht —

```
\begin{page}{UXANNAEx2}{Example using the Axiom/NAG Expert System}
\begin{scroll}
\xtc{
{\bf Example 2}: The ODE
{\centerline{\inputbitmap{\htbmdir{}/ode3.xbm}\space{1}with
\space{1}
{\inputbitmap{\htbmdir{}/y3.xbm}}}}
\newline
could be solved as follows:
\blankline
}{}
\xtc{
}{
\spadpaste{ans2 := solve([Y[2],-1001*Y[2]-1000*Y[1]], 0.0, 10.0,
[1.0,-1.0], [2,4,6,8], 1.0e-4)\bound{ans2} }
}
\blankline
\xtc{
It creates a composite structure for which the field containing the
result can be expanded as required.
\blankline
}{
\spadpaste{ans2 . 'result\free{ans2}}
```

```
}
\blankline
\xtc{
}{
\spadpaste{ans2 . 'y\free{ans2}}
}
\blankline
\end{scroll}
\autobuttons
\downlink{Example 3}{UXANNAEx3}
%\downlink{Decision Agents}{UXANNADec}
\end{page}
```

———-

Example using the Axiom/NAG Expert System

⇒ "notitle" (UXANNADec) 24.1 on page 3991

— annaex.ht —

```
\begin{page}{UXANNAEx3}{Example using the Axiom/NAG Expert System}
\begin{scroll}
\xtc{
{\bf Example 3}: The function
{\centerline{\inputbitmap{\htbmdir{}/opt2.xbm}}}
with simple bounds
{\centerline{\inputbitmap{\htbmdir{}/opt2c.xbm}}}
\newline
could be minimized as follows:
\blankline
}{}
\xtc{
}{
\spadpaste{ans3 := optimize((X[1]+10*X[2])**2 + 5*(X[3]-X[4])**2 +
(X[2]-2*X[3])**4 + 10*(X[1]-X[4])**4, [3,-1,0,1], [1,-2,\%minusInfinity,1],
[3,0,\%plusInfinity,3])\bound{ans3} }
}
\blankline
\xtc{
It creates a composite structure for which the field containing the
minimum can be expanded as required.
\blankline
}{
\spadpaste{ans3 . objf\free{ans3}}
}
\blankline
\xtc{
```

```
}{
\spadpaste{ans3 . x\free{ans3}}
}
\blankline
\xtc{
}{
\spadpaste{ans3 . attributes\free{ans3}}
}
\blankline
\end{scroll}
\autobuttons
\downlink{Decision Agents}{UXANNADec}
\end{page}
```

———-

Decision Agents

⇒ "notitle" (UXANNAInfer) 24.1 on page 3992
⇒ "notitle" (UXANNAMeth) 24.1 on page 3993
⇒ "notitle" (UXANNAMeas) 24.1 on page 3994
⇒ "notitle" (UXANNAAgent) 24.1 on page 3995

— **annaex.ht** —

```
\begin{page}{UXANNADec}{Decision Agents}
\begin{scroll}
\blankline
Some features are either present or absent in a problem.  Examples of
such binary decisions include {\em is a matrix symmetric?} and {\em is
a function continuous?}  However in practice many questions are about
the {\em degree} to which a problem exhibits a property: {\em how much
does a function oscillate?}, or {\em how stiff are these differential
equations?}
\blankline
We have therefore created decision agents of two types, reflecting
their property --- {\em Binary Agents} are Boolean functions returning
either true or false and {\em Intensity Functions} are quantitative
and return a range of different values, either numerical or structured
types. The framework we are developing is able to deal with both these
forms of information.
\blankline

In any given problem area (for example solving ordinary differential
equations, optimization etc.) we have a selection of {\em methods}.
These might be to use a particular NAG routine, or they might involve
employing a higher-level strategy such as transforming the problem
```

```
into an equivalent, but easier to solve, form.
\blankline
Associated with every method we define a {\em measure function} which
assesses the suitability of that method to a particular problem.  Each
measure function has access to a range of symbolic {\em agents} which
can answer questions about the various properties of the problem in
hand.
\blankline
\end{scroll}
\downlink{ Inference Mechanisms }{UXANNAInfer}
\downlink{ Method Domains }{UXANNAMeth}
\downlink{ Measure Functions }{UXANNAMeas}
\downlink{ Computational Agents }{UXANNAAgent}

\end{page}
```

———-

Inference Mechanisms

⇒ "notitle" (UXANNAMeth) 24.1 on page 3993
⇒ "notitle" (UXANNAMeas) 24.1 on page 3994
⇒ "notitle" (UXANNAAgent) 24.1 on page 3995
⇒ "notitle" (UXANNAEx) 24.1 on page 3984

— **annaex.ht** —

```
\begin{page}{UXANNAInfer}{Inference Mechanisms}
\begin{scroll}
\blankline
The inference machine will take the problem description as provided by
the user and perform an initial analysis to verify its validity.  It
will consider, in turn, all of the available methods within its
knowledge base which might solve that problem.  In doing so it
analyses the input problem to find out about any attributes that could
affect the ability of the methods under consideration to perform
effectively.
\blankline
Some of these measures may use lazy evaluation in the sense that, if a
method already assessed is believed to be a good candidate, and if
evaluating the current measure will be relatively expensive, then that
measure will not be evaluated unless later evidence shows that the
selected method is not, in fact, a successful strategy, for example if
it has failed.
\end{scroll}
\downlink{ Method Domains }{UXANNAMeth}
\downlink{ Measure Functions }{UXANNAMeas}
```

```
\downlink{ Computational Agents }{UXANNAAgent}
\downlink{ Examples }{UXANNAEx}
\end{page}
```

———-

Method Domains

⇒ "notitle" (UXANNAMeas) 24.1 on page 3994
⇒ "notitle" (UXANNAAgent) 24.1 on page 3995
⇒ "notitle" (UXANNAEx) 24.1 on page 3984

— **annaex.ht** —

```
\begin{page}{UXANNAMeth}{Method Domains}
\begin{scroll}
\blankline
An Axiom {\em domain} has been created for each method or strategy for
solving the problem.  These method domains each implement two
functions with a uniform (method independant) interface.  \blankline
{\bf measure:} A function which calculates an estimate of suitability
of this particular method to the problem if there is a possibility
that the method under consideration is more appropriate than one
already investigated.
\blankline
If it may be possible to improve on the current favourite method, the
function will call computational agents to analyse the problem for
specific features and calculate the measure from the results these
agents return, using a variation on the Lucks/Gladwell intensity and
compatibility model if conflict between attributes, as investigated by
these computational agents, may be present.
\blankline
{\bf implementation:} A function which may be one of two distinct
kinds.  The first kind uses the interface to the NAG Library to call a
particular routine with the required parameters.  Some of the
parameters may need to be calculated from the data provided before the
external function call.
\blankline
The other kind will apply a ``high level'' strategy to try to solve
the problem e.g.~a transformation of an expression from one that is
difficult to solve to one which is easier, or a splitting of the
problem into several more easily solvable parts.  For example, for a
solution of the equation above, since the integral is semi-infinite we
might wish to transform the range by, say, using the mapping
{\it y -> 1/x}
on the section {\it 1 < x < \inputbitmap{\htbmdir{}/infty.xbm}})
and adding the result to the unmapped section {\it 0 < x < 1}.
```

```
\blankline
\end{scroll}
\downlink{ Measure Functions }{UXANNAMeas}
\downlink{ Computational Agents }{UXANNAAgent}
\downlink{ Examples }{UXANNAEx}
\end{page}
```

———-

Measure Functions

⇒ "notitle" (UXANNAAgent) 24.1 on page 3995
⇒ "notitle" (UXANNAEx) 24.1 on page 3984

— **annaex.ht** —

```
\begin{page}{UXANNAMeas}{Measure Functions}
\begin{scroll}
\blankline
Each measure function will estimate the ability of a particular method to
solve a problem.  It will consult whichever agents are needed to perform
analysis on the problem in order to calculate the measure.  There is a
parameter which would contain the best compatibility
value found so far.
\blankline
However, the interpretation we give to the results of some tests is
not always clear-cut.  If a set of tests give conflicting advice as to
the appropriateness of a particular method, it becomes important to
decide not only {\it whether} certain properties are present but also
their {\it degree}.  This gives us a basis for estimating the
compatibility of each property.
\blankline
We have taken for our model the system recommended by Lucks and
Gladwell which uses a system of measurement of compatibility allowing
for interaction and conflict between a number of attributes.  All of
these processes may not be required if the choice is clear-cut e.g. we
have an integral to calculate which has a particular singularity
structure for which one particular method has been specifically
constructed.  However, for more difficult cases a composite picture
should be built up to calculate a true measurement.
\blankline
How the compatibility functions interpret the measurements of various
attributes is up to them and may vary between differing methods.  It
is this area that takes as its basis the {\it judgement} of Numerical
Analysis `experts' whether that be from the documentation (which may
be deficient in certain respects) or from alternative sources.
However, its assessment of the suitability or otherwise of a
```

```
particular method is reflected in a single normalised value
facilitating the direct comparison of the suitability of a number of
possible methods.
\blankline
\end{scroll}
\downlink{ Computational Agents }{UXANNAAgent}
\downlink{ Examples }{UXANNAEx}
\end{page}
```

———-

Computational Agents

⇒ "notitle" (UXANNAEx) 24.1 on page 3984

— **annaex.ht** —

```
\begin{page}{UXANNAAgent}{Computational Agents}
\begin{scroll}
\blankline
Computational Agents are those program segments which investigate the
attributes of the input function or functions, such as {\bf
stiffnessAndStabilityOfODEIF} (the {\em IF} indicates that it is an
{\em Intensity Function} i.e. one that returns a normalised real
number or a set of normalised real numbers).  They are usually
functions or programs written completely in the Axiom language and
implemented using computer algebra.
\blankline
Some agents will be common to more than one problem domain whereas
others will be specific to a single domain.  They also vary greatly in
their complexity.  It is a fairly simple task to return details about
the range of a function since this information will have been included
in the problem specification.  It is a different order of complexity
to return details of its singularity structure.
\blankline
\xtc{
As an example, here is a call to the computational agent {\bf
singularitiesOf} to obtain the list of singularities of the function
{\it tan x} which are in the range
{\it 0..12\inputbitmap{\htbmdir{}/pi.xbm}}:
\blankline
}{
}
\xtc{
}{
\spadpaste{s := singularitiesOf(tan x,[x],0..12*\%pi)$ESCONT \free{lib3} }
}
\blankline
```

```
Each of these computational agents which may be called by a number of
method domains retain their output in a dynamic hash-table, so
speeding the process and retaining efficiency.
\end{scroll}
\downlink{ Examples }{UXANNAEx}
\end{page}
```

Chapter 25

ANNA Algebra Code

```
)co annacat.spad
)co cont.spad
)co d01Package.spad
)co d01agents.spad
)co d01routine.spad
)co d01transform.spad
)co d01weights.spad
)co d02Package.spad
)co d02agents.spad
)co d02routine.spad
)co d03Package.spad
)co d03agents.spad
)co d03routine.spad
)co e04Package.spad
)co e04agents.spad
)co e04routine.spad
)co functions.spad
)co routines.spad
)co tools.spad
```

Chapter 26

Page hierarchy layout

This is the forest of pages in hyperdoc. Any page at the leftmost margin is a root page. Notice that the roots are disconnnected. The only reachable pages are those from the "RootPage" page below. All others exist but cannot be reached.

Indentation implies that a page can be reached by the less indented page.

Pages which have vertical bars or parens around them are lisp calls.

```
RootPage
  BasicCommand
    Calculus
      |bcDifferentiate|
      |bcIndefiniteIntegrate|
      |bcDefiniteIntegrate|
      |bcLimit|
      |bcSum|
      |bcProduct|
    |bcMatrix|
    |bcExpand|
    |bcDraw|
    |bcSeries|
    |bcSolve|
  TopReferencePage
    YouTriedIt
    ugWhatsNewTwoTwoPage
      ugTwoTwoPolynomialsPage
      ugTwoTwoHyperdocPage
      ugTwoTwoNAGLinkPage
      ugTwoTwoCCLPage
    FoundationLibraryDocPage
      manpageXXintro
      manpageXXc02
      manpageXXc02aff
```

manpageXXc02agf
manpageXXc05
manpageXXc05adf
manpageXXc05nbf
manpageXXc05pbf
manpageXXc05zaf
manpageXXc06
manpageXXc06aef
manpageXXc06ebf
manpageXXc06ecf
manpageXXc06ekf
manpageXXc06fpf
manpageXXc06fqf
manpageXXc06frf
manpageXXc06fuf
manpageXXc06gbf
manpageXXc06gcf
manpageXXc06gqf
manpageXXc06gsf
manpageXXd01
manpageXXd01ajf
manpageXXd01akf
manpageXXd01alf
manpageXXd01amf
manpageXXd01anf
manpageXXd01apf
manpageXXd01aqf
manpageXXd01asf
manpageXXd01bbf
manpageXXd01fcf
manpageXXd01gaf
manpageXXd01gbf
manpageXXd02
manpageXXd02bbf
manpageXXd02bhf
manpageXXd02cjf
manpageXXd02ejf
manpageXXd02gaf
manpageXXd02gbf
manpageXXd02kef
manpageXXd02raf
manpageXXd03
manpageXXd03edf
manpageXXd03eef
manpageXXd03faf
manpageXXe01
manpageXXe01baf
manpageXXe01bef
manpageXXe01bff
manpageXXe01bgf

manpageXXf02
manpageXXf02aaf
manpageXXf02abf
manpageXXf02adf
manpageXXf02aef
manpageXXf02aff
manpageXXf02agf
manpageXXf02ajf
manpageXXf02akf
manpageXXf02awf
manpageXXf02axf
manpageXXf02bbf
manpageXXf02bjf
manpageXXf02fjf
manpageXXf02wef
manpageXXf02xef
manpageXXf04
manpageXXf04adf
manpageXXf04arf
manpageXXf04asf
manpageXXf04atf
manpageXXf04axf
manpageXXf04faf
manpageXXf04jgf
manpageXXf04maf
manpageXXf04mbf
manpageXXf04mcf
manpageXXf04qaf
manpageXXf07
manpageXXf07adf
manpageXXf07aef
manpageXXf07fdf
manpageXXf07fef
manpageXXs
manpageXXs01eaf
manpageXXs13aaf
manpageXXs13acf
manpageXXs13adf
manpageXXs14aaf
manpageXXs14abf
manpageXXs14baf
manpageXXs15adf
manpageXXs15aef
manpageXXs17acf
manpageXXs17adf
manpageXXs17aef
manpageXXs17aff
manpageXXs17agf
manpageXXs17ahf
manpageXXs17ajf

```
  manpageXXs17akf
  manpageXXs17dcf
  manpageXXs17def
  manpageXXs17dgf
  manpageXXs17dhf
  manpageXXs17dlf
  manpageXXs18acf
  manpageXXs18adf
  manpageXXs18aef
  manpageXXs18aff
  manpageXXs18dcf
  manpageXXs18def
  manpageXXs19aaf
  manpageXXs19abf
  manpageXXs19acf
  manpageXXs19adf
  manpageXXs20acf
  manpageXXs20adf
  manpageXXs21baf
  manpageXXs21bbf
  manpageXXs21bcf
  manpageXXs21bdf
  manpageXXonline
  manpageXXkwic
  manpageXXsummary
  manpageXXconvert
TopicPage
  |htTutorialSearch|
  NumberPage
    IntegerPage
      IntegerXmpPage
      ugxIntegerPrimesPage
      IntNumberTheoryFnsXmpPage
      IntegerExamplePage
        IntegerExampleProofPage
      IntegerProblemPage
        IntegerProblemProofPage
        IntegerProblemAnswerPage1
        IntegerProblemAnswerPage2
    FractionPage
      RationalNumberPage
      FractionXmpPage
    DoubleFloatXmpPage
      ugGraphPage
      ugProblemNumericPage
      FloatXmpPage
    FloatXmpPage
      ugGraphPage
      ugProblemNumericPage
      DoubleFloatXmpPage
```

```
      ugxFloatIntroPage
      ugxFloatConvertPage
      ugxFloatOutputPage
      ugxFloatHilbertPage
    ComplexXmpPage
      ugProblemNumericPage
      ugTypesConvertPage
    ugProblemFinitePage
    ugProblemNumericPage
    CardinalNumberXmpPage
    SingleIntegerXmpPage
    RomanNumeralXmpPage
    ContinuedFractionXmpPage
    PartialFractionXmpPage
    QuaternionXmpPage
    OctonionXmpPage
    DecimalExpansionXmpPage
      BinaryExpansionXmpPage
      HexExpansionXmpPage
      RadixExpansionXmpPage
    BinaryExpansionXmpPage
      DecimalExpansionXmpPage
      RadixExpansionXmpPage
    HexExpansionXmpPage
    RadixExpansionXmpPage
    PAdicPage
  PolynomialPage
    PolynomialBasicPage
    PolynomialSubstitutionPage
    ugProblemFactorPage
    PolynomialGCDPage
    PolynomialRootPage
    PolynomialTypesPage
  FunctionPage
    RationalFunctionPage
    AlgebraicFunctionPage
    ElementaryFunctionPage
    FunctionSimplificationPage
  EquationPage
    ugxProblemLinSysPage
    ugxProblemOnePolPage
    ugxProblemPolSysPage
    ugProblemDEQPage
  CalculusPage
    ugProblemLimitsPage
    ugIntroCalcDerivPage
    ugIntroIntegratePage
    ugProblemIntegrationPage
    ugProblemLaplacePage
    ugProblemSeriesPage
```

```
      ugProblemDEQPage
    LinAlgPage
      ugIntroTwoDimPage
      ugxMatrixCreatePage
      ugxMatrixOpsPage
      ugProblemEigenPage
      ugxFloatHilbertPage
      PermanentXmpPage
      VectorXmpPage
      SqMatrixXmpPage
      OneDimensionalArrayXmpPage
      TwoDimensionalArrayXmpPage
      ugTypesConvertPage
    GraphicsPage
    AlgebraPage
      NumberTheoryPage
      GroupTheoryPage
        InfoGroupTheoryPage
        InfoRepTheoryPage
        RepA6Page
  ugLangPage
  ExamplesExposedPage
  ugSysCmdPage
  GlossaryPage
  HTXTopPage
    HTXIntroTopPage
      HTXIntroPage1
        HTXIntroPage2
        ugHyperPage
      HTXIntroPage2
        HTXIntroPage3
      HTXIntroPage3
        HTXLinkPage6
        HTXTryPage
        HTXFormatTopPage
    HTXFormatTopPage
      HTXFormatPage1
      HTXFormatPage2
      HTXFormatPage3
      HTXFormatPage4
      HTXFormatPage5
      HTXFormatPage6
      HTXFormatPage7
      HTXFormatPage8
    HTXLinkTopPage
      HTXLinkPage1
        HTXLinkTopPage
        TestHelpPage
        HTXLinkPage2
      HTXLinkPage2
```

```
        HTXLinkPage6
        SpadNotConnectedPage
        UnknownPage
        ErrorPage
        ProtectedQuitPage
        HTXLinkPage3
      HTXLinkPage3
        HTXLinkPage4
      HTXLinkPage4
        HTXLinkPage5
      HTXLinkPage5
        HTXLinkPage6
      HTXLinkPage6
        HTXLinkTopPage
    HTXAdvTopPage
      HTXAdvPage1
        HTXAdvPage2
      HTXAdvPage2
        HTXAdvPage3
      HTXAdvPage3
        HTXAdvPage4
      HTXAdvPage4
        HTXAdvPage5
      HTXAdvPage5
        HTXAdvPage6
      HTXAdvPage6
        HTXAdvTopPage
    HTXTryPage
  RefSearchPage
TopicPage
Man0Page
  |kSearch|
  |oSearch|
  |aSearch|
  |aokSearch|
  |docSearch|
  |genSearch|
  |detailedSearch|
  htsearch
  ugSysCmdPage
TopExamplePage
  GraphicsExamplePage
    AssortedGraphicsExamplePage
    ThreeDimensionalGraphicsExamplePage
      TwoVariableGraphicsPage
      SpaceCurveGraphicsPage
      ParametricTubeGraphicsPage
      ParametricSurfaceGraphicsPage
      ugGraphThreeDBuildPage
    OneVariableGraphicsExamplePage
```

```
  PolarGraphicsExamplePage
  ParametricCurveGraphicsExamplePage
  ImplicitCurveGraphicsExamplePage
  ListPointsGraphicsExamplePage
  ComplexFunctionGraphicsExamplePage
ExamplesExposedPage
  AssociationListXmpPage
    TableXmpPage
    ListXmpPage
  BalancedBinaryTreeXmpPage
  BasicOperatorXmpPage
  BinaryExpansionXmpPage
    HexExpansionXmpPage
    DecimalExpansion
    RadixExpansion
  BinarySearchTreeXmpPage
  CardinalNumberXmpPage
  CartesianTensorXmpPage
  CharacterXmpPage
    CharacterClassXmpPage
    StringXmpPage
  CharacterClassXmpPage
  CliffordAlgebraXmpPage
    ugxCliffordComplexPage
    ugxCliffordQuaternPage
    ugxCliffordExteriorPage
    ugxCliffordDiracPage
  ComplexXmpPage
  ContinuedFractionXmpPage
    StreamXmpPage
  CycleIndicatorsXmpPage
  DeRhamComplexXmpPage
  DecimalExpansionXmpPage
    HexExpansionXmpPage
    BinaryExpansion
    RadixExpansion
  DistributedMultivariatePolyXmpPage
    ugIntroVariablesPage
    ugTypesConvertPage
    PolynomialXmpPage
    UnivariatePolyXmpPage
    MultivariatePolyXmpPage
  DoubleFloatXmpPage
  EqTableXmpPage
    TableXmpPage
  EquationXmpPage
  ExitXmpPage
  ExpressionXmpPage
    KernelXmpPage
    ugIntroCalcDerivPage
```

```
    ugIntroCalLimitsPage
    ugIntroSeriesPage
    ugProblemDEQPage
    ugProblemIntegrationPage
    ugUserRulesPage
  FactoredXmpPage
    ugxFactoredDecompPage
    ugxFactoredExpandPage
    ugxFactoredArithPage
    ugxFactoredNewPage
      FactoredFnsTwoXmpPage
    ugxFactoredVarPage
  FactoredFnsTwoXmpPage
    FactoredXmpPage
    ugProblemGaloisPage
  FileXmpPage
    TextFileXmpPage
    KeyedAccessFileXmpPage
    LibraryXmpPage
    FileNameXmpPage
  FileNameXmpPage
  FlexibleArrayXmpPage
    OneDimensionalArrayXmpPage
    VectorXmpPage
  FloatXmpPage
  FractionXmpPage
    ContinuedFractionXmpPage
    PartialFractionXmpPage
  FullPartialFracExpansionXmpPage
    PartialFractionXmpPage
  GeneralSparseTableXmpPage
    TableXmpPage
  GroebnerFactorizationPkgXmpPage
  HeapXmpPage
    FlexibleArray
  HexExpansionXmpPage
    DecimalExpansion
    BinaryExpansion
    RadixExpansion
  IntegerXmpPage
    ugIntroNumbersPage
    IntNumberTheoryFnsXmpPage
    DecimalExpansionXmpPage
    BinaryExpansionXmpPage
    HexExpansionXmpPage
    RadixExpansionXmpPage
    ugxIntegerBasicPage
      FractionXmpPage
      ugTypesUnionsPage
      ugTypesRecordsPage
```

```
  ugxIntegerPrimesPage
    FactoredXmpPage
    ComplexXmpPage
  ugxIntegerNTPage
    IntNumberTheoryFnsXmpPage
IntegerLinearDependenceXmpPage
IntNumberTheoryFnsXmpPage
KernelXmpPage
  BasicOperatorXmpPage
  ExpressionXmpPage
KeyedAccessFileXmpPage
  FileXmpPage
  TextFileXmpPage
  LibraryXmpPage
LexTriangularPkgXmpPage
LazardSetSolvingPackageXmpPage
LibraryXmpPage
  FileXmpPage
  TextFileXmpPage
  KeyedAccessFileXmpPage
LieExponentialsXmpPage
LiePolynomialXmpPage
LinearOrdinaryDifferentialOperatorXmpPage
  ugxLinearOrdinaryDifferentialOperatorSeriesPage
LinearOrdinaryDifferentialOperatorOneXmpPage
  ugxLinearOrdinaryDifferentialOperatorOneRatPage
LinearODEOperatorTwoXmpPage
  ugxLinearODEOperatorTwoConstPage
  ugxLinearODEOperatorTwoMatrixPage
ListXmpPage
  ugxListCreatePage
  ugxListAccessPage
  ugxListChangePage
  ugxListOtherPage
  ugxListDotPage
LyndonWordXmpPage
MagmaXmpPage
MakeFunctionXmpPage
  ugUserMakePage
MappingPackageOneXmpPage
MatrixXmpPage
  ugxMatrixCreatePage
  ugxMatrixOpsPage
    ugIntroTwoDimPage
    ugProblemEigenPage
    ugxFloatHilbertPage
    PermanentXmpPage
    VectorXmpPage
    OneDimensionalArrayXmpPage
    TwoDimensionalArrayXmpPage
```

```
MultiSetXmpPage
MultivariatePolyXmpPage
  PolynomialXmpPage
  UnivariatePolyXmpPage
  DistributedMultivariatePolyXmpPage
NoneXmpPage
OctonionXmpPage
  QuaternionXmpPage
OneDimensionalArrayXmpPage
  VectorXmpPage
  FlexibleArrayXmpPage
OperatorXmpPage
OrderedVariableListXmpPage
OrderlyDifferentialPolyXmpPage
PartialFractionXmpPage
  FullPartialFracExpansionXmpPage
PermanentXmpPage
PolynomialXmpPage
  DistributedMultivariatePolyXmpPage
  MultivariatePolyXmpPage
  UnivariatePolyXmpPage
  FactoredXmpPage
  ugProblemFactorPage
QuaternionXmpPage
RadixExpansionXmpPage
  HexExpansionXmpPage
  DecimalExpansion
  BinaryExpansion
RealClosureXmpPage
RegularTriangularSetXmpPage
RomanNumeralXmpPage
SegmentXmpPage
SegmentBindingXmpPage
  SegmentXmpPage
  UniversalSegmentXmpPage
SetXmpPage
  ListXmpPage
SingleIntegerXmpPage
  ugTypesDeclare
  ugTypesPkgCallPage
  ugBrowsePage
SparseTableXmpPage
  TableXmpPage
  GeneralSparseTableXmpPage
SqMatrixXmpPage
  MatrxiXmpPage
  ugTypesWritingModesPage
  ugTypesExposePage
SqFreeRegTriangSetXmpPage
StreamXmpPage
```

```
  ugLangItsPage
  ugProblemSeriesPage
  ContinuedFractionXmpPage
  ListXmpPage
StringXmpPage
  CharacterXmpPage
  CharacterClassXmpPage
StringTableXmpPage
  TableXmpPage
SymbolXmpPage
TableXmpPage
  AssociationList
  EqTableXmpPage
  StringTableXmpPage
  SparseTableXmpPage
  KeyedAccessFileXmpPage
TextFileXmpPage
  FileXmpPage
  KeyedAccessFileXmpPage
  LibraryXmpPage
TwoDimensionalArrayXmpPage
  ugTypesAnyNonePage
  MatrixXmpPage
  OneDimensionalArrayXmpPage
UnivariatePolyXmpPage
  ugProblemFactorPage
  ugIntroVariablesPage
  ugTypesConvertPage
  PolynomialXmpPage
  MultivariatePolyXmpPage
  DistributedMultivariatePolyXmpPage
UnivariateSkewPolynomialPage
UniversalSegmentXmpPage
  SegmentXmpPage
  SegmentBindingXmpPage
  ListXmpPage
  StreamXmpPage
VectorXmpPage
  OneDimensionalArrayXmpPage
  ListXmpPage
  MatrixXmpPage
  OneDimensionalArrayXmpPage
  SetXmpPage
  TableXmpPage
  TwoDimensionalArrayXmpPage
VoidXmpPage
WuWenTsunTriangularSetXmpPage
XPBWPolynomialXmpPage
XPolynomialXmpPage
XPolynomialRingXmpPage
```

```
    ZeroDimSolvePkgXmpPage
  ExampleCoverPage
    Menuexdiff
      ExDiffBasic
      ExDiffSeveralVariables
      ExDiffHigherOrder
      ExDiffMultipleI
      ExDiffMultipleII
      ExDiffFormalIntegral
    Menuexint
      ExIntRationalFunction
      ExIntRationalWithRealParameter
      ExIntRationalWithComplexParameter
      ExIntTwoSimilarIntegrands
      ExIntNoSolution
      ExIntTrig
      ExIntAlgebraicRelation
        ExIntAlgebraicRelationExplain
      ExIntRadicalOfTranscendental
      ExIntNonElementary
    Menuexlap
      ExLapSimplePole
      ExLapTrigTrigh
      ExLapDefInt
      ExLapExpExp
      ExLapSpecial1
      ExLapSpecial2
    Menuexlimit
      ExLimitBasic
        ExLimitTwoSided
        ExLimitOneSided
      ExLimitParameter
      ExLimitOneSided
      ExLimitTwoSided
      ExLimitInfinite
      ExLimitRealComplex
      ExLimitComplexInfinite
    Menuexmatrix
      ExMatrixBasicFunction
      ExConstructMatrix
      ExTraceMatrix
      ExDeterminantMatrix
      ExInverseMatrix
      ExRankMatrix
    Menuexplot2d
      ExPlot2DFunctions
      ExPlot2DParametric
      ExPlot2DPolar
      ExPlot2DAlgebraic
    Menuexplot3d
```

```
        ExPlot3DFunctions
        ExPlot3DParametricSurface
        ExPlot3DParametricCurve
      Menuexseries
        ExSeriesConvert
        ExSeriesManipulate
        ExSeriesFunctions
        ExSEriesSubstitution
      Menuexsum
        ExSumListEntriesI
        ExSumListEntriesII
        ExSumApproximateE
        ExSumClosedForm
        ExSumCubes
        ExSumPolynomial
        ExSumGeneralFunction
        ExSumInfinite
  TopSettingsPage
    ugSysCmdPage
  RootPageLogo
  releaseNotes
  ugHyperPage


htxl
  nagLinkIntroPage
  htxl1
    c02
      manpageXXc02
      (|kSearch| "NagPolynomialRootsPackage")
      (|c02aff|)
      (|c02agf|)
    c05
      manpageXXc05
      (|kSearch| "NagRootFindingPackage")
      (|c05adf|)
      (|c05nbf|)
      (|c05pbf|)
    c06
      manpageXXc06
      (|kSearch| "NagSeriesSummationPackage")
      (|c06eaf|)
      (|c06ebf|)
      (|c06ecf|)
      (|c06ekf|)
      (|c06fpf|)
      (|c06fqf|)
      (|c06frf|)
      (|c06fuf|)
      (|c06gbf|)
```

```
  (|c06gcf|)
  (|c06gqf|)
  (|c06gsf|)
d01
  manpageXXd01
  (|kSearch| "NagIntegrationPackage")
  (|d01ajf|)
  (|d01akf|)
  (|d01alf|)
  (|d01amf|)
  (|d01anf|)
  (|d01apf|)
  (|d01aqf|)
  (|d01asf|)
  (|d01bbf|)
  (|d01fcf|)
  (|d01gaf|)
  (|d01gbf|)
d02
  manpageXXd02
  (|kSearch| "NagOrdinaryDifferentialEquationsPackage")
  (|d02bbf|)
  (|d02bhf|)
  (|d02cjf|)
  (|d02ejf|)
  (|d02gaf|)
  (|d02gbf|)
  (|d02kef|)
  (|d02raf|)
d03
  manpageXXd03
  (|kSearch| "NagPartialDifferentialEquationsPackage")
  (|d03edf|)
  (|d03eef|)
  (|d03faf|)
e01
  manpageXXe01
  (|kSearch| "NagInterpolationPackage")
  (|e01baf|)
  (|e01bef|)
  (|e01bff|)
  (|e01bgf|)
  (|e01bhf|)
  (|e01daf|)
  (|e01saf|)
  (|e01sef|)
e02
  manpageXXe02
  (|kSearch| "NagFittingPackage")
  (|e02adf|)
```

```
  (|e02aef|)
  (|e02agf|)
  (|e02ahf|)
  (|e02ajf|)
  (|e02akf|)
  (|e02baf|)
  (|e02bbf|)
  (|e02bcf|)
  (|e02bdf|)
  (|e02bef|)
  (|e02daf|)
  (|e02dcf|)
  (|e02ddf|)
  (|e02def|)
  (|e02dff|)
  (|e02gaf|)
  (|e02zaf|)
e04
  manpageXXe04
  (|kSearch| "NagOptimisationPackage")
  (|e04dgf|)
  (|e04fdf|)
  (|e04gcf|)
  (|e04jaf|)
  (|e04mbf|)
  (|e04naf|)
  (|e04ucf|)
  (|e04ycf|)
f01
  manpageXXf
  manpageXXf01
  (|kSearch| "NagMatrixOperationsPackage")
  (|f01brf|)
  (|f01bsf|)
  (|f01maf|)
  (|f01mcf|)
  (|f01qcf|)
  (|f01qdf|)
  (|f01qef|)
  (|f01rcf|)
  (|f01rdf|)
  (|f01ref|)
f02
  manpageXXf
  manpageXXf02
  (|kSearch| "NagEigenPackage")
  (|f02aaf|)
  (|f02abf|)
  (|f02adf|)
  (|f02aef|)
```

```
    (|f02aff|)
    (|f02agf|)
    (|f02ajf|)
    (|f02akf|)
    (|f02awf|)
    (|f02axf|)
    (|f02bbf|)
    (|f02bjf|)
    (|f02fjf|)
    (|f02wef|)
    (|f02xef|)
  f04
    manpageXXf
    manpageXXf04
    (|kSearch| "NagLinearEquationSolvingPackage")
    (|f04adf|)
    (|f04arf|)
    (|f04asf|)
    (|f04atf|)
    (|f04axf|)
    (|f04faf|)
    (|f04jgf|)
    (|f04maf|)
    (|f04mbf|)
    (|f04mcf|)
    (|f04qaf|)
  f07
    manpageXXf
    manpageXXf07
    (|kSearch| "NagLapack")
    (|f07adf|)
    (|f07aef|)
    (|f07fdf|)
    (|f07fef|)
  s
    manpageXXs
    (|kSearch| "NagSpecialFunctionsPackage")
    (|s01eaf|)
    (|s13aaf|)
    (|s13acf|)
    (|s13adf|)
    (|s14aaf|)
    (|s14abf|)
    (|s14baf|)
    (|s15adf|)
    (|s15aef|)
    (|s17acf|)
    (|s17adf|)
    (|s17aef|)
    (|s17aff|)
```

```
      (|s17agf|)
      (|s17ahf|)
      (|s17ajf|)
      (|s17akf|)
      (|s17dcf|)
      (|s17def|)
      (|s17dgf|)
      (|s17dhf|)
      (|s17dlf|)
      (|s18acf|)
      (|s18adf|)
      (|s18aef|)
      (|s18aff|)
      (|s18dcf|)
      (|s18def|)
      (|s19aaf|)
      (|s19abf|)
      (|s19acf|)
      (|s19adf|)
      (|s20acf|)
      (|s20adf|)
      (|s21baf|)
      (|s21bbf|)
      (|s21bcf|)
      (|s21bdf|)
  (|kSearch| "Nag*")
  FoundationLibraryDocPage
  nagDocumentationPage
    manpageXXintro
    manpageXXonline
    FoundationLibraryDocPage
    aspSectionPage
  nagLinkUsagePage
  aspSectionPage
  generalFortranPage
  nagTechnicalPage

UXANNA
UXANNAInt
UXANNAOde
UXANNAOpt
UXANNAPde
UXANNAOptEx
UXANNAOpt2Ex
UXANNAIntEx
UXANNAOdeEx
UXANNATxt
  UXANNAEx
  UXANNAIntro
    UXANNADec
```

```
    UXANNAInfer
    UXANNAMeth
    UXANNAMeas
  UXANNADec
  UXANNAInfer
  UXANNAMeas
UXANNAIntro
  UXANNADec
  UXANNAInfer
  UXANNAMeth
  UXANNAMeas
UXANNAEx
  UXANNAEx2
UXANNAEx2
  UXANNAEx3
UXANNAEx3
  UXANNADec
UXANNADec
  UXANNAInfer
  UXANNAMeth
  UXANNAMeas
  UXANNAAgent
UXANNAInfer
  UXANNAMeth
  UXANNAMeas
  UXANNAAgent
  UXANNAEx
UXANNAMeth
  UXANNAMeas
  UXANNAAgent
  UXANNAEx
UXANNAMeas
  UXANNAAgent
  UXANNAEx
UXANNAAgent
  UXANNAEx

Asp1ExampleCode
Asp10ExampleCode
Asp12ExampleCode
Asp19ExampleCode
Asp20ExampleCode
Asp24ExampleCode
Asp27ExampleCode
Asp28ExampleCode
Asp29ExampleCode
Asp30ExampleCode
Asp31ExampleCode
Asp33ExampleCode
Asp34ExampleCode
```

```
Asp35ExampleCode
Asp4ExampleCode
Asp41ExampleCode
Asp42ExampleCode
Asp49ExampleCode
Asp50ExampleCode
Asp55ExampleCode
Asp6ExampleCode
Asp7ExampleCode
Asp73ExampleCode
Asp74ExampleCode
Asp77ExampleCode
Asp78ExampleCode
Asp8ExampleCode
Asp80ExampleCode
Asp9ExampleCode

NoMoreHelpPage

hyperdoc
  ViewportPage
  BitMaps
  CPHelp

PrefixEval
InfixEval

helpExpose
ExposureSystem
ExposureDef
ExposureDetails

DomainMapping
  (|dbSpecialDescription| '|Mapping|)
  (|dbSpecialOperations| '|Mapping|)
MappingDescription

DomainRecord
  (|dbSpecialDescription| '|Record|)
  (|dbSpecialOperations| '|Record|)
RecordDescription

CategoryType

DomainUnion
  (|dbSpecialDescription| '|Union|)
  (|dbSpecialOperations| '|Union|)
UnionDescription
UntaggedUnion
UTUnionDescription
```

```
UsersGuideExposedPage
  ugWhatsNewTwoTwoPage
  ugIntroPage
    ugIntroTypoPage
    ugIntroStartPage
      ugHyperPage
      ugSysCmdPage
      ugAvailCLEFPage
    ugIntroTypoPage
    ugIntroExpressionsPage
      ugIntroArithmeticPage
      ugIntroPreviousPage
      ugIntroTypesPage
        ugTypesPage
      ugIntroAssignPage
        ugLangAssignPage
      ugIntroConversionPage
        ugTypesConvertPage
      ugIntroCallFunPage
      ugIntroMacrosPage
        ugUserMacrosPage
      ugIntroLongPage
        ugInOutInPage
        ugLangBlocksPage
      ugIntroCommentsPage
    ugIntroGraphicsPage
      ugProblemNumericPage
      ugAppGraphicsPage
      ugGraphPage
    ugIntroNumbersPage
      FloatXmpPage
      DoubleFloatXmpPage
      ComplexXmpPage
      DecimalExpansionXmpPage
      ContinuedFractionXmpPage
      PartialFractionXmpPage
      RadixExpansionXmpPage
      ugxProblemFinitePrimePage
    ugIntroCollectPage
      ListXmpPage
      ugLangItsPage
      StreamXmpPage
      ugLangItsPage
      OneDimensionalArrayXmpPage
      FlexibleArrayXmpPage
      HeapXmpPage
      BinarySearchTreeXmpPage
      BalancedBinaryTreeXmpPage
      SetXmpPage
```

```
    MultiSetXmpPage
    AssociationListXmpPage
    KeyedAccessFileXmpPage
    LibraryXmpPage
    SparseTableXmpPage
    StringTableXmpPage
    TableXmpPage
    ugTypesRecordsPage
    ugTypesUnionsPage
    ugDomainsPage
  ugIntroTwoDimPage
    TwoDimensionalArrayXmpPage
    MatrixXmpPage
    PermanentXmpPage
    SqMatrixXmpPage
    VectorXmpPage
    ugProblemEigenPage
    ugProblemLinPolEqnPage
    ugLangItsPage
  ugIntroYouPage
    ugUserPage
    ugInOutInPage
  ugIntroVariablesPage
  ugIntroCalcLimitsPage
    ugProblemLimitsPage
  ugIntroSeriesPage
    ugProblemSeriesPage
  ugIntroCalcDerivPage
  ugIntroIntegratePage
    ugProblemIntegrationPage
  ugIntroDiffEqnsPage
  ugIntroSolutionPage
  ugIntroSysCmmandsPage
    ugSysCmdPage
ugTypesPage
  ugTypesBasicPage
    ugTypesBasicDomainConsPage
      ugCategoriesPage
      ugTypesConvertPage
  ugTypesWritingPage
    ugTypesBasicPage
    ugTypesDeclarePage
    ugTypesConvertPage
    ugTypesPkgCallPage
    ugTypesWritingZeroPage
    ugTypesWritingOnePage
      ugTypesPkgCallPage
    ugTypesWritingMorePage
    ugTypesWritingModesPage
      ugTypesDeclarePage
```

```
        ugTypesConvertPage
      ugTypesWritingAbbrPage
        ugSysCmdwhatPage
    ugTypesDeclarePage
      ugLangAssignPage
      ugUserDeclarePage
      ugTypesConvertPageugIntroAssignPage
    ugTypesRecordsPage
    ugTypesUnionsPage
      ugTypesUnionsWOSelPage
      ugTypesUnionsWSelPage
        ugTypesRecordsPage
        ugTypesUnionsWOSelPage
    ugTypesAnyNonePage
    ugTypesConvertPage
    ugTypesSubdomainsPage
    ugTypesPkgCallPage
      ugTypesDeclarePage
      ugUserUsePage
    ugTypesResolvePage
    ugTypesExposePage
      ugTypesPkgCallPage
      ugUserTrianglePage
      ugSysCmdframePage
    ugAvailSnoopPage
      ugBrowsePage
      ComplexXmpPage
      ugUserDeclarePage
  ugHyperPage
    YouTriedIt
    ugHyperHeadingsPage
    ugHyperKeysPage
      ugHyperScrollPage
      ugHyperInputPage
    ugHyperScrollPage
      ugHyperInputPage
    ugHyperInputPage
    ugHyperButtonsPage
    ugHyperSearchPage
      ugLogicalSearchesPage
    ugHyperExamplePage
    ugHyperResourcesPage
    ugInOutPage
      ugInOutInPage
        ugLangBlocksPage
      ugInOutSpadprofPage
      ugInOutOutPage
      ugInOutAlgebraPage
      ugInOutTeXPage
      ugInOutScriptPage
```

```
    ugInOutFortranPage
ugLangPage
  ugLangAssignPage
    ugUserDelayPage
  ugLangBlocksPage
    ugLangIfPage
      ugTypesResolvePage
      ugTypesPkgCallPage
  ugLangIfPage
    ugTypesResolvePage
    ugTypesPkgCallPage
  ugLangLoopsPage
    ugLangLoopsCompIntPage
      ugUserCompIntPage
    ugLangLoopsReturnPage
      ugUserBlocksPage
    ugLangLoopsBreakPage
      ugLangLoopsReturnPage
    ugLangLoopsBreakVsPage
    ugLangLoopsBreakMorePage
      ugLangLoopsForInPage
    ugLangLoopsIteratePage
    ugLangLoopsWhilePage
    ugLangLoopsForInPage
    ugLangLoopsForInNMPage
      SegmentXmpPage
    ugLangLoopsForInNMSPage
    ugLangLoopsForInNPage
    ugLangLoopsForInXLPage
    ugLangLoopsForInPredPage
    ugLangLoopsParPage
      ugLangLoopsForInPredPage
  ugLangItsPage
    ugLangLoopsPage
    ListXmpPage
    StreamXmpPage
  ugLangStreamsPrimesPage
ugUserPage
  ugUserFunMacPage
    ugUserAnonPage
    ExitXmpPage
    VoidXmpPage
  ugUserMacrosPage
  ugUserIntroPage
    ugTypesPkgCallPage
    ugUserAnonPage
  ugUserDeclarePage
    ugTypesDeclarePage
  ugUserOnePage
  ugUserDecUndecPage
```

```
    ugCategoriesPage
  ugUserDecOpersPage
    MappingPackageOneXmpPage
    ugPackagesAbstractPage
    ugPackagesPage
    ugCategoriesPage
  ugUserDelayPage
    ugLangAssignPage
  ugUserUsePage
    ugTypesPkgCallPage
    ugTypesResolvePage
  ugUserCompIntPage
    ugTypesSubdomainsPage
  ugUserPiecePage
    ugUserPieceBasicPage
    ugUserPiecePickingPage
      ugUserPieceBasicPage
    ugUserPiecePredPage
      ugUserPieceBasicPage
  ugUserCachePage
    ugUserFreeLocalPage
  ugUserRecurPage
    ugUserFreeLocalPage
    ugUserCachePage
  ugUserMakePage
    MakeFunctionXmpPage
  ugUserBlocksPage
    ugLangBlocksPage
  ugUserFreeLocalPage
    ugUserCachePage
    ugUserRecurPage
  ugUserAnonPage
    ugUserAnonExampPage
    ugUserAnonDeclarePage
  ugUserDatabasePage
  ugUserTrianglePage
    ugTypesExposePage
  ugUserPalPage
    ugUserTrianglePage
  ugUserRulesPage
ugGraphPage
  ugGraphTwoDPage
    ugGraphTwoDPlotPage
      ugGraphTwoDOptionsPage
    ugGraphTwoDParPage
      ugGraphThreeDOptionsPage
    ugGraphTwoDPlanePage
      ugGraphTwoDOptionsPage
    ugGraphTwoDOptionsPage
      ugGraphColorPage
```

```
      ugGraphColorPalettePage
      ugGraphColorPage
      ugGraphColorPalettePage
    ugGraphColorPage
    ugGraphColorPalettePage
    ugGraphTwoDControlPage
    ugGraphTwoDopsPage
    ugGraphTwoDbuildPage
    ugGraphTwoDappendPage
  ugGraphThreeDPage
    ugGraphThreeDPlotPage
      ugGraphThreeDOptionsPage
    ugGraphThreeDParmPage
      ugGraphThreeDOptionsPage
    ugGraphThreeDParPage
      ugGraphThreeDOptionsPage
      ugGraphCoordPage
    ugGraphThreeDOptionsPage
      ugGraphCoordPage
    ugGraphMakeObjectPage
    ugGraphThreeDBuildPage
    ugGraphCoordPage
    ugGraphClipPage
    ugGraphThreeDControlPage
    ugGraphThreeDopsPage
    ugXdefaultsPage
ugProblemPage
  ugProblemNumericPage
    ugIntroPage
    FloatXmpPage
    DoubleFloatXmpPage
  ugProblemFactorPage
    ugProblemFactorIntRatPage
    ugProblemFactorFFPage
      ugProblemFinitePage
    ugProblemFactorAlgPage
    ugProblemFactorRatFunPage
  ugProblemSymRootPage
    ugxProblemOnePolPage
    ugxProblemPolSysPage
    ugxProblemSymRootOnePage
    ugxProblemSymRootAllPage
      ugxProblemOnePolPage
  ugProblemEigenPage
  ugProblemLinPolEqnPage
    ugProblemDEQPage
    ugxProblemLinSysPage
    ugxProblemOnePolPage
    ugxProblemPolSysPage
      ugxProblemOnePolPage
```

```
  ugProblemLimitsPage
  ugProblemLaplacePage
  ugProblemIntegrationPage
    ugxProblemSymRootAllPage
  ugProblemSeriesPage
    ugxProblemDEQSeriesPage
      ugxProblemSeriesConversionsPage
      ugTypesDeclarePage
      ugxProblemSeriesFunctionsPage
      ugxProblemSeriesFormulaPage
    ugxProblemSeriesCreatePage
    ugxProblemSeriesCoefficientsPage
    ugxProblemSeriesArithmeticPage
    ugxProblemSeriesFunctionsPage
    ugxProblemSeriesConversionsPage
    ugxProblemSeriesFormulaPage
      ugxProblemSeriesConversionsPage
      ugUserAnonPage
      ugxProblemSeriesConversionsPage
    ugxProblemSeriesSubstitutePage
    ugxProblemSeriesBernoulliPage
  ugProblemDEQPage
    ugProblemLinPolEqnPage
    ugxProblemLDEQClosedPage
    ugxProblemNLDEQClosedPage
    ugxProblemDEQSeriesPage
  ugProblemFinitePage
    ugxProblemFinitePrimePage
    ugxProblemFinitePrimePage
    ugxProblemFiniteExtensionFinitePage
    ugxProblemFiniteModulusPage
      ugxProblemFiniteExtensionFinitePage
    ugxProblemFiniteCyclicPage
      ugxProblemFiniteUtilityPage
    ugxProblemFiniteNormalPage
      ugxProblemFiniteUtilityPage
    ugxProblemFiniteConversionPage
      ugxProblemFiniteExtensionFinitePage
    ugxProblemFiniteUtilityPage
  ugProblemIdealPage
  ugProblemGaloisPage
    FactoredXmpPage
    ugAvailCLEFPage
  ugProblemGeneticPage
OctonionXmpPage
ExamplesExposedPage
ugIntProgPage
  ugAppGraphicsPage
  ugIntProgDrawingPage
    ugGraphPage
```

```
  ugIntProgRibbonPage
    ugUserPage
    ugLangBlocksPage
  ugIntProgColorPage
  ugIntProgPLCPage
  ugIntProgColorArrPage
  ugIntProgVecFieldsPage
  ugIntProgCompFunsPage
    ugGraphPage
  ugIntProgFunctionsPage
    ugUserMakePage
  ugIntProgNewtonPage
    MappingPackageOneXmpPage
ugPackagesPage
  ugIntProgPage
  ugIntProgPage
  ugPackagesNamesPage
    ugIntProgPage
    ugTypesWritingAbbrPage
  ugPackagesSyntaxPage
  ugPackagesAbstractPage
    ugIntProgCompFunsPage
  ugPackagesCapsulesPage
  ugPackagesInputFilesPage
  ugPackagesPackagesPage
  ugPackagesParametersPage
    ugTypesPage
    ugUserBlocksPage
    ugCategoriesAttributesPage
  ugPackagesCondsPage
    ugUserBlocksPage
  ugPackagesCompilingPage
    EqTableXmpPage
  ugPackagesHowPage
    ugCategoriesHierPage
ugCategoriesPage
  ugTypesBasicDomainConsPage
  ugCategoriesDefsPage
    ugTypesPage
  ugCategoriesExportsPage
  ugCategoriesDocPage
  ugCategoriesHierPage
  ugCategoriesMembershipPage
  ugCategoriesDefaultsPage
    ugCategoriesHierPage
    ugPackagesPage
  ugCategoriesAxiomsPage
    ugCategoriesDefaultsPage
  ugCategoriesCorrectnessPage
  ugCategoriesAttributesPage
```

```
    ugCategoriesParametersPage
    ugCategoriesConditionalsPage
      ugPackagesCondsPage
    ugCategoriesAndPackagesPage
      ugPackagesAbstractPage
  ugDomainsPage
    ugPackagesDomsPage
      ugPackagesPage
    ugDomainsDefsPage
    ugDomainsAssertionsPage
      ugCategoriesCorrectnessPage
      ugCategoriesConditionalsPage
    ugDomainsDemoPage
    ugDomainsBrowsePage
    ugDomainsRepPage
      ugDomainsDemoPage
    ugDomainsMultipleRepsPage
      ugTypesUnionsPage
    ugDomainsAddDomainPage
      ugDomainsDemoPage
    ugDomainsDefaultsPage
      ugPackagesPage
      ugCategoriesDefaultsPage
    ugDomainsOriginsPage
    ugDomainsShortFormsPage
    ugDomainsCliffordPage
      CliffordAlgebraXmpPage
    ugDomsinsDatabasePage
      ugDomainsQueryLanguagePage
      ugDomainsDatabaseConstructorPage
      ugDomainsQueryEquationsPage
      ugDomainsDataListsPage
      ugDomainsDatabasePage
      ugDomainsCreatingPage
      ugDomainsPuttingPage
      ugDomainsExamplesPage
  ugBrowsePage
    ugBrowseStartPage
      ugBrowseCapitalizationConventionPage
      ugBrowseDomainPage
      ugBrowseViewsOfConstructorsPage
      ugBrowseViewsOfOperationsPage
    ugBrowseDomainPage
    ugBrowseMiscellaneousFeaturesPage
  ugWhatsNewPage
    ugWhatsNewLanguagePage
      ugLangLoopsBreakPage
      ugLangBlocksPage
    ugWhatsNewLibraryPage
      FullPartialFracExpansionXmpPage
```

```
  ugWhatsNewHyperDocPage
    ugHyperKeysPage
  ugWhatsNewDocumentationPage
    ugGraphTwoDbuildPage
    ugGraphTwoDappendPage
    ugIntroCallFunPage
    ugUserRulesPage
ugSysCmdPage
  ugSysCmdOverviewPage
    ugSysCmdsetPage
    ugSysCmdcompilePage
  ugSysCmdabbreviationPage
  ugSysCmdbootPage
    ugSysCmdfinPage
    ugSysCmdlispPage
    ugSysCmdsetPage
    ugSysCmdsystemPage
  ugSysCmdcdPage
    ugSysCmdcompilePage
    ugSysCmdeditPage
    ugSysCmdhistoryPage
    ugSysCmdlibraryPage
    ugSysCmdreadPage
    ugSysCmdspoolPage
  ugSysCmdclosePage
    ugSysCmdquitPage
  ugSysCmdclearPage
    ugSysCmddisplayPage
    ugSysCmdhistoryPage
    ugSysCmdundoPage
  ugSysCmdcompilePage
    ugSysCmdcdPage
    ugSysCmdtracePage
    ugSysCmdabbreviationPage
    ugSysCmdeditPage
    ugSysCmdlibraryPage
  ugSysCmddisplayPage
    ugSysCmdclearPage
    ugSysCmdhistoryPage
    ugSysCmdsetPage
    ugSysCmdshowPage
    ugSysCmdwhatPage
  ugSysCmdeditPage
    ugSysCmdsystemPage
    ugSysCmdcompilePage
    ugSysCmdreadPage
  ugSysCmdfinPage
    ugSysCmdpquitPage
  ugSysCmdframePage
    ugSysCmdhistoryPage
```

```
  ugSysCmdsetPage
ugSysCmdhelpPage
ugSysCmdhistoryPage
  ugSysCmdframePage
  ugSysCmdcdPage
  ugSysCmdreadPage
  ugSysCmdsetPage
  ugSysCmdundoPage
ugSysCmdlibraryPage
ugSysCmdlispPage
  ugSysCmdsystemPage
  ugSysCmdbootPage
  ugSysCmdfinPage
ugSysCmdloadPage
ugSysCmdltracePage
  ugSysCmdbootPage
  ugSysCmdlispPage
  ugSysCmdtracePage
ugSysCmdpquitPage
  ugSysCmdfinPage
  ugSysCmdhistoryPage
  ugSysCmdclosePage
  ugSysCmdquitPage
  ugSysCmdsystemPage
ugSysCmdquitPage
  ugSysCmdfinPage
  ugSysCmdhistoryPage
  ugSysCmdclosePage
  ugSysCmdpquitPage
  ugSysCmdsystemPage
ugSysCmdreadPage
  ugInOutInPage
  ugSysCmdcompilePage
  ugSysCmdeditPage
  ugSysCmdhistoryPage
ugSysCmdsetPage
  ugSysCmdquitPage
ugSysCmdshowPage
  ugSysCmddisplayPage
  ugSysCmdsetPage
  ugSysCmdwhatPage
ugSysCmdspoolPage
  ugSysCmdcdPage
ugSysCmdsynonymPage
  ugSysCmdsetPage
  ugSysCmdwhatPage
ugSysCmdsystemPage
  ugSysCmdbootPage
  ugSysCmdfinPage
  ugSysCmdlispPage
```

```
    ugSysCmdpquitPage
    ugSysCmdquitPage
  ugSysCmdtracePage
    ugSysCmdcompilePage
    ugSysCmdbootPage
    ugSysCmdlispPage
    ugSysCmdltracePage
  ugSysCmdundoPage
    ugSysCmdhistoryPage
  ugSysCmdwhatPage
    ugSysCmddisplayPage
    ugSysCmdsetPage
    ugSysCmdshowPage
    ugAppGraphicsPage


ugFimagesOnePage
ugFimagesTwoPage
ugFimagesThreePage
ugFimagesFivePage
ugFimagesSixPage
ugFimagesSevenPage
ugFimagesEightPage
ugFconformalPage
ugFtknotPage
ugFntubePage
  ugFimagesFivePage
ugFdhtriPage
ugFtetraPage
ugFantoinePage
ugFscherkPage

GraphicsPage
  GraphicsExamplePage
  TwoDimensionalGraphicsPage
    OneVariableGraphicsPage
    ParametricCurveGraphicsPage
    PolarGraphicsPage
    ImplicitCurveGraphicsPage
    ListPointsGraphicsPage
  ThreeDimensionalGraphicsPage
    TwoVariableGraphicsPage
    SpaceCurveGraphicsPage
    ParametricTubeGraphicsPage
    ParametricSurfaceGraphicsPage
    ugGraphThreeDBuildPage
  ViewportPage

3DObjectGraphicsPage
```

```
BROWSEhelp
  ugBrowseStartPage
  ugBrowseDomainPage
  ugBrowseMiscellaneousFeaturesPage

nagm.ht
  manpageXXm01
  manpageXXm01caf
  manpageXXm01daf
  manpageXXm01def
  manpageXXm01djf
  manpageXXm01eaf
  manpageXXm01zaf

nagx.ht
  manpageXXx01
  manpageXXx02
  manpageXXx04
  manpageXXx04aaf
  manpageXXx04abf
  manpageXXx04caf
  manpageXXx04daf
  manpageXXx05
  manpageXXx05aaf
  manpageXXx05abf
  manpageXXx05acf
  manpageXXx05baf
```

Chapter 27

Makefile

— * —

```
BOOK=${SPD}/books/bookvol7.1.pamphlet
WORK=${OBJ}/${SYS}/hyper/pages
IN=${SPD}/books

# These files are not reachable from RootPage
IGNORE=annaex

PHT= alist array1 array2 bbtree binary bop bstree card carten \
     cclass char clif complex contfrac coverex cycles decimal derham \
     dfloat dmp eq eqtbl evalex exdiff exint exit exlap exlimit \
     exmatrix explot2d explot3d expr exseries exsum farray file float \
     fname fparfrac fr2 frac fr function gbf graphics grpthry gstbl \
     heap hexadec intheory int kafile kernel lazm3pk lexp lextripk \
     lib list lodo1 lodo2 lodo lpoly lword magma mappkg1 matrix \
     mkfunc mpoly mset none numbers oct odpol op ovar perman pfr poly1 \
     poly quat radix reclos regset roman segbind seg set sint sqmatrix \
     sregset stbl stream string strtbl symbol table textfile ug01 ug02 \
     ug03 ug04 ug06 ug07 ug08 ug10 ug11 ug12 ug13 ug15 uniseg \
     up vector void wutset xpbwpoly xpoly xpr zdsolve zlindep

PAGELIST= ${PHT} algebra aspex basic  bmcat cphelp expose gloss  \
    htxadvpage1 htxadvpage2 htxadvpage3 htxadvpage4 htxadvpage5 \
    htxadvpage6 htxadvtoppage htxformatpage1 htxformatpage2 \
    htxformatpage3 htxformatpage4 htxformatpage5 htxformatpage6 \
    htxformatpage7 htxformatpage8 htxformattoppage htxintropage1 \
    htxintropage2 htxintropage3 htxintrotoppage htxlinkpage1 \
    htxlinkpage2 htxlinkpage3 htxlinkpage3.ht htxlinkpage4 \
    htxlinkpage5 htxlinkpage6 htxlinktoppage htxtoppage htxtrypage \
    hyperdoc link man0 mapping nagaux nagc nagd nage nagf nagm nags \
```

```
    nagx newuser record releasenotes rootpage topics type ug00 ug05 \
    ug14 ug16 ug21 ug union util xmpexp
```

```
HYPER=${MNT}/${SYS}/doc
PAGEFILE=${HYPER}/bookvol7.1.pamphlet
SMAN=${MNT}/${SYS}/bin/sman
HTADD=${MNT}/${SYS}/bin/htadd

all: dir ${PAGEFILE} ${HYPER}/ht.db
@echo 3 finished ${BOOK}

dir:
@echo 0 making hypertex directories
@mkdir -p ${WORK}
@mkdir -p ${HYPER}
```

——–

Due to the awesome programming skills of Scott Morrison the htadd program will use the original source file in literate form to build the ht.db

— * —

```
${PAGEFILE}: ${BOOK}
@echo 1 making ${PAGEFILE} from ${BOOK}
@cp ${BOOK} ${PAGEFILE}

%:
@echo 1 making ${HYPER}/$*.ht from ${BOOK}
@ echo '(tangle "${BOOK}" "$*.ht" "${HYPER}/$*.ht")' | \
  ${OBJ}/${SYS}/bin/lisp
```

——–

The patch/paste files are now included directly in bookvol7.1 Add new patch/paste files is a manual process when editing the book.

— * —

```
${HYPER}/ht.db: ${BOOK}
@echo 2 making ${HYPER} from ${BOOK}
@ (cd ${HYPER} ; \
           rm -f ht.db ; \
   ${HTADD} ${PAGEFILE} )
@ cp -pr ${IN}/bitmaps ${HYPER}
@ cp -pr ${IN}/viewports ${HYPER}
@ (cd ${HYPER}/viewports ; \
```

```
for i in `find . -name "*.Z"` ; do gunzip $$i ; done )
```

Index