Matlab, Introduction

Resources:
3. Matlab on Athena (MIT computer services web page).
4. www.mathworks.com, Matlab online documentation: answers most of the questions.
Main Features of Matlab

¥ Matlab = matrix laboratory, matrix oriented.
¥ Any variable is an array by default, thus almost no declarations. All variables are by default double.
¥ High level language: (i) quick and easy coding
  (ii) lots of tools (Spectral Analysis, Image Processing, Signal Processing, Financial, Symbolic Math etc.)
  (iii) relatively slow
¥ All Matlab functions are precompiled.
¥ One may add extra functions by creating M-files.

01 November 2001
Main Features of Matlab

¥ Translator - interpreter: line after line, no .exe files, does not reevaluate old variables (example)

\[
\begin{align*}
>> a &= 2 \\
a &= 2 \\
>> b &= 3 \times a \\
b &= 6 \\
>> a &= 4 \\
a &= 4 \\
>> b &= b \\
b &= 6
\end{align*}
\]

a has been changed, but b has not been reevaluated!
Comparison with C.

¥ Syntax is similar

¥ Language structure is similar to C:
  —MATLAB supports variables, arrays, structures, subroutines, files
  —MATLAB does NOT support pointers and does not require variable declarations
Matlab, Getting Started

1. Accessing Matlab on Athena:
add matlab
matlab &
2. Log out: quit or exit

Useful hints and commands:
¥ input: variable_name ->
   output: variable_value
¥ semicolon at the end will suppress the output
Useful Hints & Commands

¥ command history: upper & lower arrows, also command name guess:
   (i) type abc
   (ii) hit upper arrow key -> get the last command starting from abc

¥ format compact - no blank lines in the output
   format loose - back to default

¥ help commandname - info on commandname
Workspace Maintenance

¥ clear all - clears all the memory (workspace)

clear xyz - removes xyz from the memory

¥ who - lists all the variables from the workspace

¥ whos - also gives the details

>> who
Your variables are:
ans        c1          c2

>> whos
Name  Size     Bytes   Class
ans    1x1      8      double array
c1     1x1     16      double array(complex)
c2     2x2     64      double array(complex)
Workspace Maintenance

¥ save saves all workspace variables on disk in file matlab.mat

¥ save filename x y z - x, y, z are saved in file filename.mat

¥ load filename - loads contents of the filename.mat to the workspace

¥ load filename x y z - loads only x, y, z from filename.mat to the workspace

¥ Each array requires a continuous chunk of memory; use pack for memory defragmentation.
Dealing with Matrices

Entering matrices by explicit list of elements:

\[
A = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \\
A = \\
1 & 2 & 3
\]

\[
A = \begin{bmatrix} 1; & 2; & 3 \end{bmatrix} \\
A = \\
1 & 2 & 3
\]

\[
A = \begin{bmatrix} 1 & 2 & 3; & 4 & 5 & 6; & 7 & 8 & 9 \end{bmatrix} \\
\text{or} \\
A = \begin{bmatrix} 1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9 \end{bmatrix}
\]

Spaces separate the elements, semicolons and new line symbols separate the rows.
Dealing with Matrices

Complex matrices:
either $A = \begin{bmatrix} 1 & 2; 3 & 4 \end{bmatrix} + i \begin{bmatrix} 5 & 6; 7 & 8 \end{bmatrix}$
or $A = \begin{bmatrix} 1+5i & 2+6i; 3+7i & 4+8i \end{bmatrix}$

No blank spaces, i or j stands for imaginary one.

Matrix and array operations.

\[
\begin{align*}
+ & \quad \{ \text{element-wise (array operations)} \\
- & \quad \{ \text{array or matrix operations} \\
* & \quad \{ \text{conjugate transpose} \\
\text{\backslash} & \quad \{ \text{left division} \\
\text{/} & \quad \{ \text{right division} \\
\end{align*}
\]

only matrix operations
Dealing with Matrices, Examples

\[
\text{>> } C = A + B; \\
C(k,l) = A(k,l) + B(k,l)
\]

\[
\text{>> } C = A*B; \\
C(k,l) = A(k,m) * B(m,l)
\]

Matrix multiplication, summation over the repeating index is implied.

\[
\text{>> } C = A.*B \\
C(k,l) = A(k,l)*B(k,l)
\]

Element-wise (array) operation

\[
\text{>> } C = A^\text{alpha}; \\
\text{>> } C = A.^\text{alpha}; \\
C(k,l) = A(k,l)^{\text{alpha}}
\]
Dealing with Matrices

Conjugate transpose: swaps the indices and changes the sign of imaginary part of each element.
\[ C = A \]
\[ C(i,j) = \text{real}(A(j,i)) - i * \text{imag}(A(j,i)) \]

\[ x = A \backslash b \quad \text{(left)} \quad A * x = b \quad \text{A-square matrix, b -column vector} \]
\[ x = b / A \quad \text{(right)} \quad x * A = b \]

*Colon notation*: used to construct vectors of equally spaced elements:
\[ >> a = 1:6 \]
\[ a = \]
\[ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \]
\[ >> b = 1:2:7 \]
\[ b = \]
\[ 1 \ 3 \ 5 \ 7 \]
Dealing with Matrices

Submatrices:
\[ A(1:4, 3) \] - column vector, first 4 elements of the 3-d column of \( A \).
\[ A(:, 3) \] - the 3-d column of \( A \).
\[ A(:, [2 4]) \] - 2 columns of \( A \): 2-d & 4-th.

*Standard math. functions of matrices* operate in array sense:
\[ \exp(A), \sin(A), \sqrt{A} = A^{0.5} \]

\[
\text{>> } B = \exp(A) \\
B(i,j) = \exp(A(i,j))
\]
Relational & Logical Operators & Functions

True: non-zero, false: zero.
Relational: <, <=, >, >=, ==, ~=.
Operate on matrices in elementwise fashion:

```matlab
>> A = 1:9, B = 9 - A
A = 1 2 3 4 5 6 7 8 9
B = 8 7 6 5 4 3 2 1 0
>> tf = A > 4
tf = 0 0 0 0 1 1 1 1 1
>> tf = (A==B)
0 0 0 0 0 0 0 0 0
```
Relational & Logical Operators & Functions

Logical: & AND; | OR; ~ NOT.

>> tf = ~(A>4)
tf = 1 1 1 1 0 0 0 0 0

>> tf = (A>2) & (A<6)
tf = 0 0 1 1 1 0 0 0 0

Functions: xor(x,y) - exclusive OR, true if either x or y is non-zero, false of both are true or false.
isempty - true for empty matrix
isreal, isequal, isfinite,...
Flow of Control

*For loops.* Syntax:

```plaintext
for x = array
    (commands)
end
```

Example:

```plaintext
>> for n = 1:10
    x(n) = sin(n*pi/10);
end
```
Flow of Control

Nested loops, decrement loop.

```>> for n = 1:5
    for m = 5:-1:1
        A(n,m) = n^2 + m^2;
    end
end
```

Alternative: *vectorized* solution, much faster: assigns memory for x only once.

```>> n = 1:10;
>> x = sin(n*pi/10)
```
Flow of Control

**While loops.** Syntax:

```plaintext
while expression
    (commands)
end
```

(commands) will be executed as long as all the elements of expression are true.

Example: search for the smallest number EPS which if added to 1 will give the result greater than 1.
Flow of Control

>> num = 0; EPS = 1;
>> while (1+EPS)>1
    EPS = EPS/2;
    num = num+1;
end
>> num
num = 53
>> EPS = 2*EPS
EPS = 2.2204e-16
Flow of Control

**If-Else-End constructions. Syntax:**

if expression1
  (commands1: if expr-n1 is true)
elseif expression2
  (commands2: if expr-n2 is true)
elseif expression3
  (commands3: if expr-n3 is true)
  . . . . . . . . . . . . . . . . . . .
else
  (commands: if 1,2,...,n are false)
end
Flow of Control

Breaking out of the loop:

```plaintext
>> EPS = 1;
>> for num = 1:1000
    EPS = EPS/2;
    if (1+EPS)<+1
        EPS = EPS*2
        break
    end
end
EPS = 2.2204e-16
```
M-files

Script files & Function files

*Script files:* contain a set of Matlab commands - programs. To execute the file: enter the file name.

```matlab
% script M-file example.m
erasers = 4; pads = 6; tape = 2;
items = erasers + pads + tape
cost = erasers*25 + pads*52 + tape*99
average_cost = cost/items
```

```matlab
>>example
items =  12
cost = 610
average_cost = 50.833
```
M-files

Interpreter actions while processing example statement:
1. Is example a current Matlab variable?
2. Is example a built-in Matlab command?
3. Is Example an M-file?
4. Opens the file and evaluates commands as if they were entered from the command line.

Thus: (i) all workspace variables are accessible to the commands form the M-file.
(ii) all variables created by M-file become a part of the work space.
M-files

Function files
¥ Analogous to functions in C.
¥ Communicate with the workspace only through variables passed to it and the output variables it creates. All internal variables are invisible to the workspace.
¥ M-files name = functions name.
¥ The first line - function-declaration line

```
function  s=area(a,b,alpha)
```

01 November 2001
function s=area(a,b,alpha)
% AREA calculates triangles area given 2 side s & angle between them
% AREA reads in two sides of the triangle and the angle between them
% (in radians) and returns the area of the triangle.

if a < 0 | b<0
    error(a and b can not be negative.)
end
s = a*b*sin(alpha)/2;
Function M-files

¥ Function M-files may call script files, the script file being evaluated in the workspace.

¥ F. M-files can have zero input and output arguments.

¥ Functions may share variables. The variable must be declared as `global` in each desired workspace.

¥