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.
Main Features of Matlab

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

```
>> a = 2
a =
   2
>> b = 3 * a
b =
   6
```

```
>> a = 4
a =
   4
>> b
b =
   6
```

*a has been changed, but b has not been reevaluated!*

01 November 2001
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:

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

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 $A = [1 \ 2; \ 3 \ 4] + i*[5 \ 6; \ 7 \ 8]$  
or $A = [1+5i \ 2+6i; \ 3+7i \ 4+8i]$

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

Matrix and array operations.

+ - } element-wise (array operations)
* ^ } array or matrix operations
\ conjugate transpose
\ left division
/ right division } only matrix operations
Dealing with Matrices, Examples

>> C = A + B;
C(k,l) = A(k,l) + B(k,l)

>> C = A*B;
C(k,l) = A(k,m) * B(m,l)

Matrix multiplication, summation over the repeating index is implied.

>> C = A.*B
C(k,l) = A(k,l)*B(k,l)

Element-wise (array) operation

>> C = A^alpha;
>> C = A.^alpha;
C(k,l) = A(k,l)^alpha

01 November 2001
Dealing with Matrices

Conjugate transpose: swaps the indices and changes the sign of imaginary part of each element.

\[ C = A' \]

\[ C(i,j) = \text{real}(A(j,i)) - i \times \text{imag}(A(j,i)) \]

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

\[ x = b / A \quad \text{(right)} \quad x \times A = b \]

*Colon notation*: used to construct vectors of equally spaced elements:

\[ \gg a = 1:6 \]

\[ a = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \end{bmatrix} \]

\[ \gg b = 1:2:7 \]

\[ b = \begin{bmatrix} 1 & 3 & 5 & 7 \end{bmatrix} \]
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 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:

for x = array
    (commands)
end

Example:

>> for n = 1:10
    x(n) = sin(n*pi/10);
end
Nested loops, decrement loop.

```matlab
>> 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.

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

**While loops.** Syntax:

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.

01 November 2001
Flow of Control

```matlab
>> 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:*

```plaintext
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
```
Intro to Matlab II

1. Flow of control.
2. M-files.
3. Structures and cell arrays.
3. Graphics: 2-d & 3-d

01 November 2001
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
```
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

>>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 will become a part of the workspace if declared global.
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-file’s name = function’s name.
• The first line - *function-declaration line*

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

01 November 2001
function s = area(a, b, alpha)
%AREA calculates triangle’s area given 2 sides & 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;

searched and displayed by the lookfor command
searched and displayed by the help command
Function M-files

- Function M-files may call script files, the script file being evaluated in the workspace.
- Function M-files may have zero input and output arguments.
- Functions may share variables. The variable must be declared as `global` in each desired workspace.
Structures & Cell Arrays

- Allow to group dissimilar, but related arrays into a single variable.
- Analogous to P/O boxes. Each box stores some array.
- One should address contents, not a structure or an array to perform mathematical operations.
- Structures: mail boxes have names, cell arrays: mail boxes have numbers.
Structures & Cell Arrays

- Often used inside Matlab, say, when solving differential equations.
- Similar to classes in C++.
- Handy for object oriented programming and database programming.
- Accessed via conventional matrix indexing.
Structures & Cell Arrays

Creating a structure circle:

```matlab
>> circle.radius = 2.5;
>> circle.center = [0 1];
>> circle.linestyle = '--';
>> circle.color = 'red'
```

circle =

```matlab
    radius: 2.5
    center: [0 1]
    linestyle: '--'
    color: 'red'
```

Structure circle has 4 fields and 1 element
Adding a new element:

```matlab
>> circle(2).radius = 3.2;
>> circle(2).center = [0 3];
>> circle(2).linestyle = '-';
>> circle(2).color = 'blue'
circle =
1x2 struct array with fields:
    radius
    center
    linestyle
    color
```

Structure circle has 4 fields and 2 elements
Structures & Cell Arrays

Accessing the elements of the structure:

```matlab
>>circle.radius
ans = 2.5
ans = 3.3
>>circle(2).radius = '3cm';
>>area1 = pi*circle(1).radius^2;
```
Structures & Cell Arrays

Cell array - array with cells as elements. Cells may contain any data of any type. The assignment is made one cell at a time:

Cell indexing: ( ) identify cells without looking at their content.

```
>>A(1,1) = {[1 2 3; 4 5 6; 7 8 9]};
>>A(1,2) = {7+8i}, A(2,2) = {1:100};
>>A(2,1) = {'Some kind of a text.'};
```

Content addressing: { } access the content of the cells.

```
>>A{1,1} = [1 2 3; 4 5 6; 7 8 9];
>>A{1,2} = 7+8i, A{2,2} = 1:100;
>>A{2,1} = 'Some kind of a text.';
```
Structures & Cell Arrays

\[ \text{>>A\{1,2\} \quad \% content addressing} \]
\[ \text{ans = 7 + 8i} \]

\[ \text{>>A(1,2) \quad \% cell indexing} \]
\[ \text{ans = [1x1 double]} \]
Two-Dimensional Graphics:

• “join-the-dots” x-y plot

\[
\begin{align*}
\text{>> } & x = [1.2 \ 2.3 \ 3.7 \ 4.1 \ 5.0 \ 7.3]; \\
\text{>> } & y = [2.6 \ 2.8 \ 3.7 \ 4.0 \ 4.3 \ 5.6]; \\
\text{>> } & \text{plot}(x,y)
\end{align*}
\]

Syntax: \texttt{plot(x,y,string)}.
String stands for color, marker and plot style.
Example: ‘r*-’ -red, asterisk at each data point, dashed line. Colors: \texttt{r, g, b, c, m, y, k, w}.
Line styles: - solid, -- dashed, : dotted, -. dash-dot.
Plotting many curves:
plot(x,y,'r-',a,b,'g--',....)

Some other control strings:
'LineWidth',2,'MarkerSize',7,'MarkeFaceColor','r',...

plot( ) -> loglog( )
changes lin-lin plot to log-log one.
Graphics

Labels and title:
xlabel('concentration')
ylabel('viscosity')
title('C(η) plot, PEO - H₂O solution.')

Axes:
axis([xmin xmax ymin ymax]),
xlim([xmin xmax]),axis tight, grid on,
axis square, ......

Also go to “edit” option of the plot window.
Graphics

Multiple plots in a Figure:

```matlab
subplot(k,m,1), plot(......)
subplot(k,m,2), plot(......)
```

```

```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
Graphics

Three-dimensional graphics:

(i) linear plots

\texttt{plot3(x1,y1,z1,S1,x2,y2,z2,S2,...)}

3 coordinates, control string, 3 coordinates...

Example; \(\sin(t), \cos(t), t\).
Graphics

(ii) 3-d, scalar functions of 2 variables, mesh plots:

\[ z = f(x,y) \]

Plot of \( f(x,y) \) - surface in 3-d.

1. Create a mesh in \( x-y \) plane:

\[ x = x0:x1, \quad y = y0:y1 \]

\[ [X, Y] = \text{meshgrid}(x,y) \]

\( x \) has \( m \) and \( y \) has \( n \) elements, \( X \) & \( Y \) - matrices nxm, \( X \) consists of \( n \) row vectors \( x \), \( Y \) of \( m \) column vectors \( y \). Each pair \( X(i,j) \) & \( Y(i,j) \) gives coordinates in \( x-y \) space.
Graphics

X & Y may be treated as matrices or arrays. If \( z = f(x,y) = 3(x^2+y)^3 \):

```matlab
>> Z = 3 * (X.^2 + Y).^3  \% Matrix Z is created
>> mesh(X, Y, Z) \% Draws mesh plot of f(x,y)
```

`meshc` - draws the underlying contour plot

`meshz` - meshplot with zero plane

`surf(X, Y, Z)` - surface plot: surface between the mesh points is filled in.
Contour plots:

```matlab
>>contour(X,Y,Z,20)  %Draws contour plot of f(x,y) with 20 contour lines.
>>pcolor(X,Y,Z)  %Uses colors, not lines
>>contourf(X,Y,Z,10)  %Filled contour plot with 10 contour lines.
```