Tutorial on MATLAB

MATH-600: Optimization and Simulation

Winter 2012

Bilal Farooq

February 24, 2012

1. Introduction

MATLAB is a specialized tool developed by MathWorks for performing numerical computations using matrices and vectors. It also has the visualization capabilities to display the results and data graphically. Cleve Moler wrote the first full version of MATLAB in 1970s and since then it has been commercialized and evolved into a leading tool for numerical computations. Due to its specialized nature, various numerical functionalities are available built-in, which makes the development of code in MATLAB faster thus one can focus more on the thinking, thoroughness, and experimentation. The graphical capabilities provided in MATLAB are very useful and can be outputted as pictures (TIFF, JPEG, or EPS), which you can use in MS Word and $\LaTeX$. For advance users, the interfaces to native languages (including: Fortran, C, C++, Java, and .NET framework) are provided.

2. Documentation and Resources

Due to its wide usage by the research community and industry for a long time, there is a wealth of resources available for MATLAB on Internet. The prime source for documentation is recommended to be the official tutorials on the MathWorks' student center (http://www.mathworks.com/academia/student_center/tutorials/launchpad.html). MATLAB Central (http://www.mathworks.com/MATLABcentral/) is also a good place for newsgroup, usergroups, examples, and other support.
EPFL provides latest version of MATLAB to its researchers via their Gasper account. The installer can be downloaded from Distrilog for Windows, MacOS X and Linux (After logging in, search for MATLAB on http://distrilog.epfl.ch/search.aspx). There are two options: Standalone and Network license. The license for the standalone version has to be updated every February, so it is recommended to install the network version. In case

you use MATLAB with network license from home, you will have to logon to EPFL VPN first.

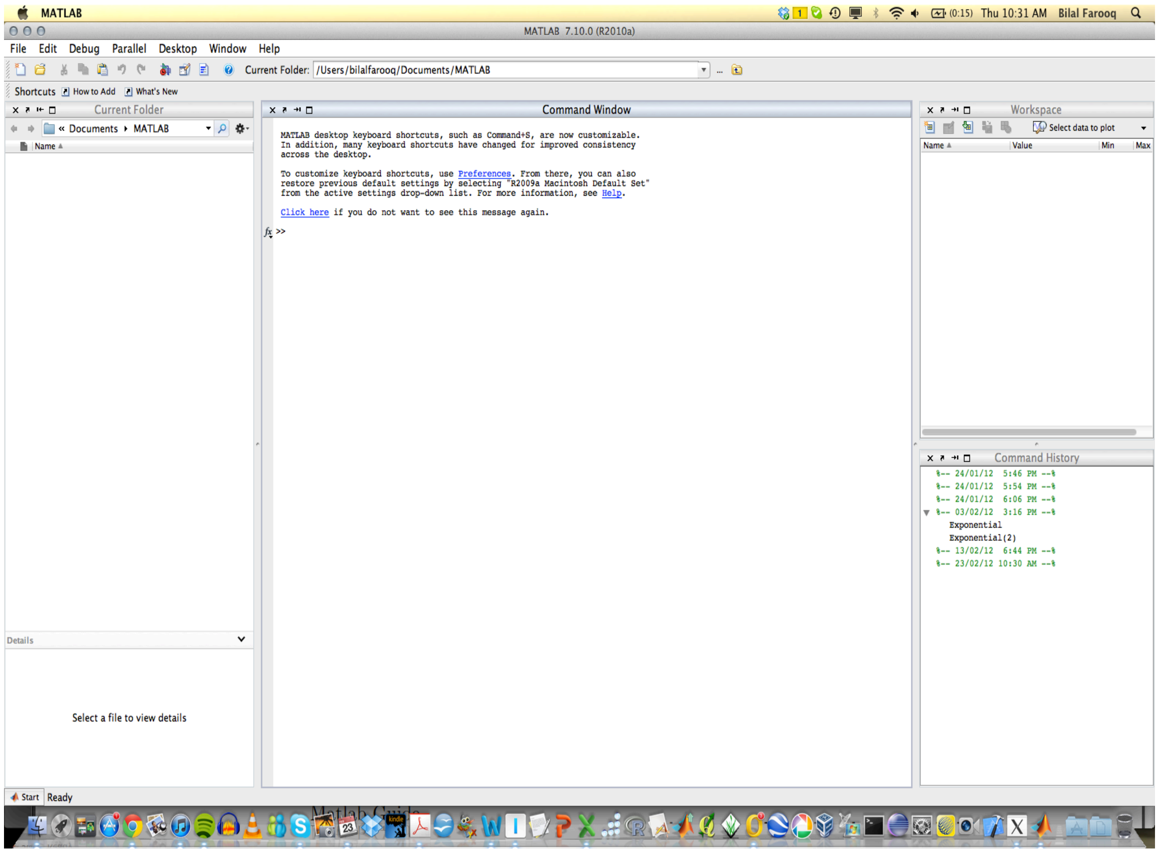
3. Getting Started

After launching the MATLAB application, the main window appears, which is shown in Figure 1. It has various sub frames showing different information.

3.1 The Command Window

As clear from the name, it is the central point of the MATLAB interface (figure 1) from where you can execute commands, functions, and various numerical operations. In the command window, write **clc** and press **Enter/Retrun** key. This will clear the screen from any past text. As a start, you can use it as a calculator by writing the calculations (try 1+1…or may be a bit less trivial than that!) and pressing **Enter/Return**.

The window on bottom right (Command History) keeps the history of commands being executed in current and past sessions. You can browse through them and double click on them to execute any previous command.



Current Folder: /Users/bilalfarooq/Documents/MATLAB

```
MATLAB desktop keyboard shortcuts, such as Command+S, are now customizable.
In addition, many keyboard shortcuts have changed for improved consistency
across the desktop.

To customize keyboard shortcuts, use Preferences. From there, you can also
restore previous default settings by selecting "R2009a Macintosh Default Set"
from the active settings drop-down list. For more information, see Help.

Click here if you do not want to see this message again.

>>
```

```
  %-- 24/01/12  5:46 PM --%
  %-- 24/01/12  5:54 PM --%
  %-- 24/01/12  6:06 PM --%
▼ %-- 03/02/12  3:16 PM --%
    Exponential
    Exponential(2)
  %-- 13/02/12  6:44 PM --%
  %-- 23/02/12 10:30 AM --%
```

Figure 1: MATLAB Interface

Few examples: `exp(1), (log(2)+sin(60))^2, sin(pi/4),1/ -1.3412e+03`

```
format long
```

Tip: Use up arrow few to repeat the command

3.2 Command Line Help

Use **help** command to list the complete help topics. For more focused topics use:

help *matlab/general*

General purpose commands.

help *matlab/lang*

Programming language constructs

help *matlab/elfun*

Elementary math functions.

For help on a keyword use

help *function*

4. Important Tools

In this section we will discuss basic, but important dimensions of the MATLAB scripting language.

4.1 Classes (Data Types)

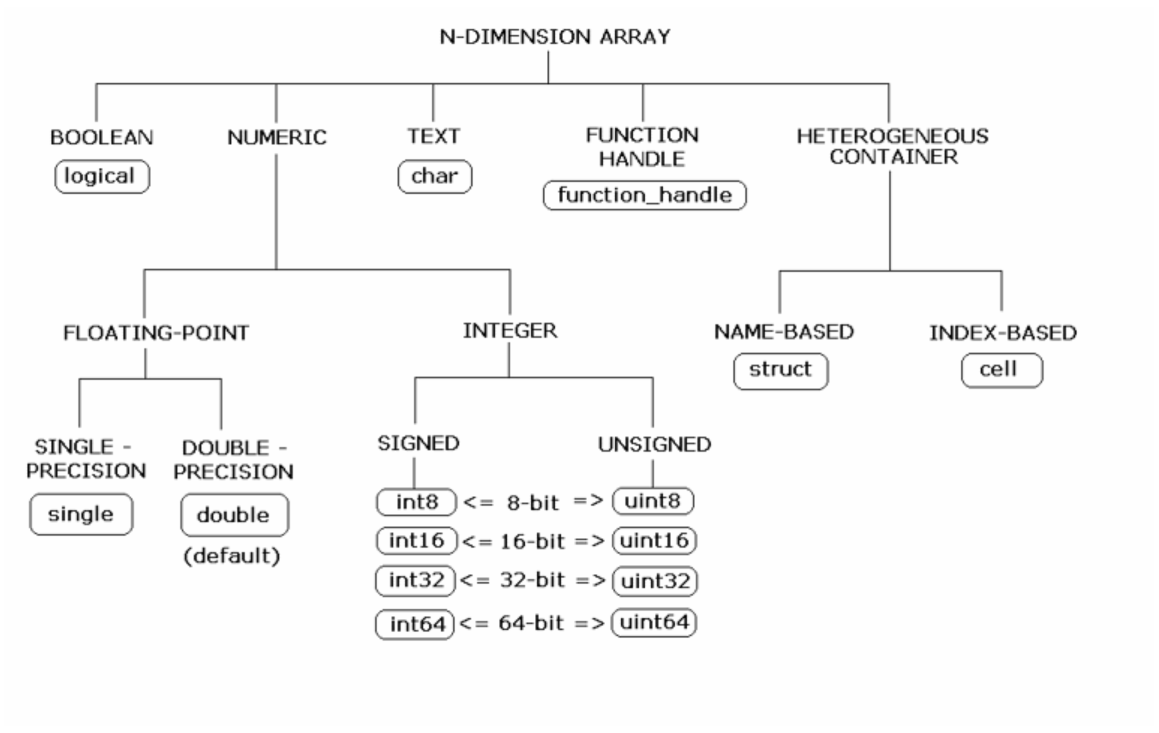


Figure 2: Basic Data Types in MATLAB

(Source: http://www.mathworks.com/help/techdoc/matlab_prog/f2-47534.html)

MATLAB recognizes Complex numbers (3.43–5.4*i*), infinity (*Inf*), and Not a Number (NaN 0/0)

4.2 Variables

A MATLAB variable is essentially a tag that you assign to a value while that value remains in memory. The tag gives you a way to reference the value in memory so that your programs can read it, operate on it with other data, and save it back to memory. MATLAB variable names must begin with a letter, which may be followed by any combination of letters, digits, and underscores. MATLAB distinguishes between uppercase and lowercase characters, so *A* and *a* are not the same variable.

```
>> N=30;
>> N
N =
    30
>>N=N-20
N =
    10
>> isvarname 3_(#@#*$
ans =
    0
```

MATLAB provides three basic types of variables:

4.1.1 Local Variable

Variable defined within function has a local level scope. Variables defined in a function do not remain in memory from one function call to the next, unless they are defined as Global or Persistent. Suppose, for example, you want to study the effect of the interaction coefficients, *α* and *β*, in the Lotka-Volterra predator-prey model.

$$\dot{y}_1 = y_1 - \alpha y_1 y_2$$
$$\dot{y}_2 = -y_2 + \beta y_1 y_2$$

Create a program file, lotka.m

```
    function yp = lotka(y)
```

```
%LOTKA   Lotka-Volterra predator-prey model.
  ALPHA = 0.01
  BETA = 0.02
  yp = [y(1) - ALPHA*y(1)*y(2); -y(2) + BETA*y(1)*y(2)];
end
```

Then using command prompt enter the statements

```
lotka([1; 1]);
```

4.1.2 Global Variable

If the same variable in a base workspace can be used in various functions, it can be defined as global scope.

Create a program file, lotka.m

```
function yp = lotka(y)
%LOTKA   Lotka-Volterra predator-prey model.
  global ALPHA BETA
  yp = [y(1) - ALPHA*y(1)*y(2); -y(2) + BETA*y(1)*y(2)];
end
```

Then using command prompt enter the statements

```
global ALPHA BETA
ALPHA = 0.01
BETA = 0.02
lotka([1; 1]);
```

4.1.3 Persistent Variable

MATLAB does not clear them from memory when the function exits, so their value is retained from one function call to the next.

```
function findSum(inputvalue)
  persistent SUM_X

  if isempty(SUM_X)
    SUM_X = 0;
  end
  SUM_X = SUM_X + inputvalue
end
```

Then using command prompt enter the statements

```
>> findSum(20);
```

```
SUM_X =

    20

>> findSum(20);

SUM_X =

    40

>> findSum(20);

SUM_X =

    60
```

4.2 Vectors

There are two types of vectors: row vector and column vector.

In row vector values is separated by commas or spaces

```
>> v = [3 1];
>> v
v =
   3     1
>> a = [1:3,6]
a =
   1     2     3     6
>> b = [1:0.5:3]
b =
   1.0000    1.5000    2.0000    2.5000    3.0000
```

Accessing an element of a vector

```
>> b(1)
ans =
   1.0000
```

In column vectors values are separated by semicolon

```
>> v = [3;1];
v =
     3
     1
```

Transposing a vector

```
>> v = [3;1]'
v =
     3     1
```

Basic operations on vectors

Vector addition/subtraction

```
>> v(1:3)-v(2:4)
ans =
      -2    -2    -2
```

Vector multiplication

```
>> a =[1;5]
a =
     1
     5
>> v*a
ans =
     8
```

Multiplication/division by constant

```
>> v/3
ans =
    1.0000    0.3333
```

More arithmetic

```
>> -2*a'+v/3
ans =
   -1.0000   -9.6667

>> -2*a+v/3
??? Error using ==> plus
Matrix dimensions must agree.
```

4.3 Matrices

Vectors are special case of matrices, so we can define a matrix in the same way by combing the operation.

```
>> D = [1,2,2;3,2,1]

D =

     1     2     2

     3     2     1
```

Zero matrix

```
>> Z = zeros(3,3)

Z =

     0     0     0

     0     0     0

     0     0     0
```

One matrix

```
>> Z = ones(3,3)

Z =

     1     1     1

     1     1     1

     1     1     1
```

Identity matrix

```
>> I = eye(3),3*I
```

```
I =
     1     0     0
     0     1     0
     0     0     1
ans =
     3     0     0
     0     3     0
     0     0     3
```

Diagonal matrix

```
>> D = [-3 0 0; 0 4 0; 0 0 2]
D =
    -3     0     0
     0     4     0
     0     0     2
```

or

```
>> d = [-3 4 2], D = diag(d)
d =
    -3     4     2
D =
    -3     0     0
     0     4     0
     0     0     2
```

Inverse of a matrix

```
>> inv(D)
ans =
   -0.3333         0         0
         0    0.2500         0
         0         0    0.5000
```

Size of a matrix

```
>> D = [1,2,2;3,2,1]
D =
     1     2     2
     3     2     1
>> size(D)
ans =
     2     3
```

Dot product (.)*

```
>> A = [1,2,3;-1,2,-2;3,5,-9];

>> B = [-2  3 1

        -1  2 2

         3 -2 2];

>> A.*B

ans =

    -2     6     3

     1     4    -4

     9   -10   -18
```

For more details on matrix operations, please see Griffiths, 2005 and http://www.mathworks.com/help/techdoc/learn_matlab/f2-644.html .

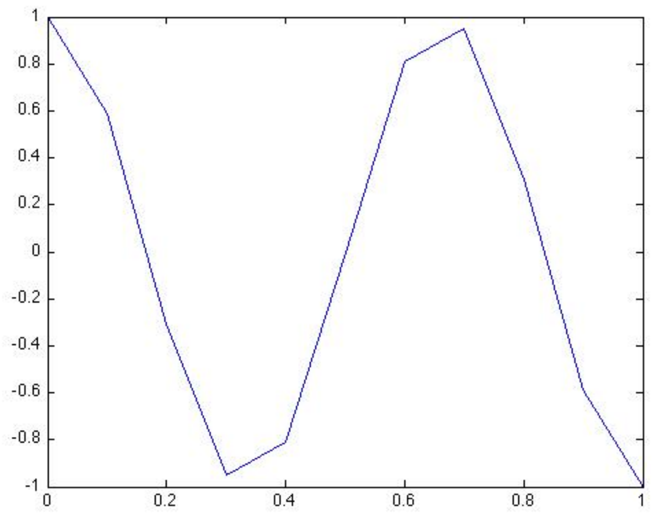
4.4 Plotting (Griffiths, 2005**)**

Suppose we wish to plot a graph of $y = cos(3\pi\, x) for\ 0 \leq x \leq 1$. We do this by sampling the function at a sufficiently large number of points and then joining up the points (x, y) by straight line segments. Suppose we take $N + 1$ points equally spaced a distance h apart:

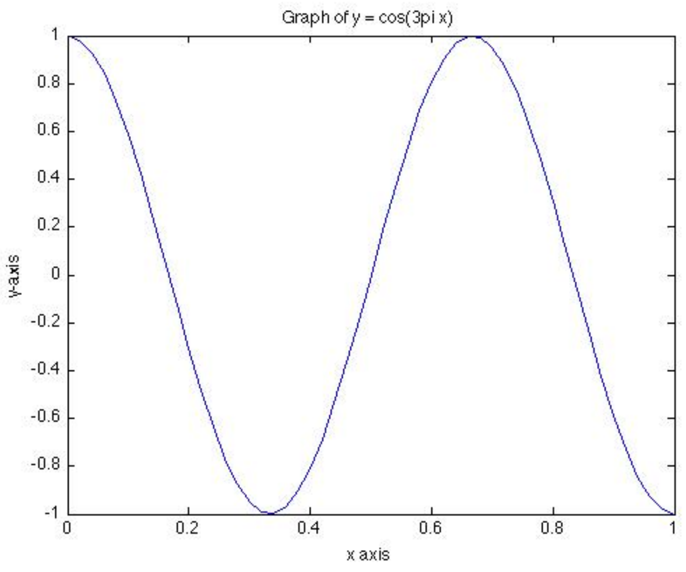
```
>> N=10;h=1/N; x= 0:h:1;
```

This will define set of points $x = 0, h, 2h, \ldots, 1 - h, 1$. Same can be achieved by ***linspace****(a,b,n)*, which generated $n + 1$ equidistant points between a and b, inclusive

```
>> x=linspace(0,1,11);
>> y = cos(3*pi*x);
>> plot(x,y)
```



```
>> x=linspace(0,1,51);
>> y = cos(3*pi*x);
>> plot(x,y)
title('Graph of y = cos(3pi x)')
xlabel('x axis')
ylabel('y-axis')
```



These plots can be saved as jpeg or other picture formats. For details on graphics in MATLAB, please see: http://www.mathworks.com/help/techdoc/ref/f16-8602.html

4.5 .m File format

It is the text based file format in which you can write functions and scripts that can then be executed in MATLAB environment.

4.6 Functions, Scripts and Loops

You can enter commands one at a time at the MATLAB command line, but many times it makes more sense to write a series of commands to a file (.m file) that you then execute as you would any MATLAB function. These functions can then be called like any other MATLAB function or command. There are two kinds of program files:

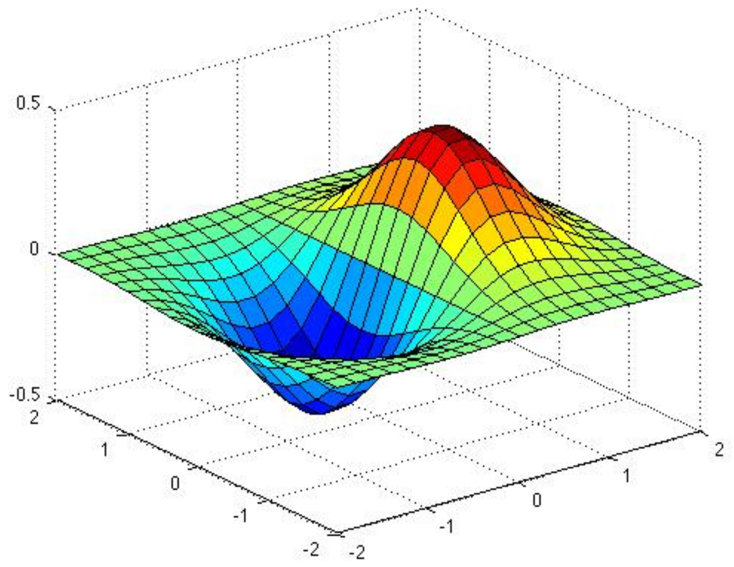
- Scripts, which do not accept input arguments or return output arguments. They operate on data in the workspace
- Functions, which can accept input arguments and return output arguments. Internal variables are local to the function, but you can define global or persistent variables

4.6.1 Script

Create a file name `surface3D.m` that has following code:

```
[a,b] = meshgrid(-2:.2:2, -2:.2:2);
c = a.* exp(-a.^2 - b.^2);
surf(a,b,c)
```

Save it, and type `surface3D` on the command line



*4.6.2 Functions (*Griffiths, 2005*)*

Save the following code in `area.m`

```
function [A] = area(a,b,c)
```

```
% Compute the area of a triangle whose
% sides have length a, b and c.
% Inputs:
% a,b,c: Lengths of sides
% Output:
% A: area of triangle
% Usage:
% Area = area(2,3,4);
% Written by dfg, Oct 14, 1996.
s = (a+b+c)/2;
A = sqrt(s*(s-a)*(s-b)*(s-c));
%%%%%%%%% end of area %%%%%%%%%%%
```

Save and on command line write:

```
>> Area = area(10,15,20)
Area =
72.6184
```

4.6.3 Looping

For loop

```
>> a = [ 0.8 0.1; 0.2 0.9 ]
>> x = [ 1; 0 ]
>> for i = 1:20, x = a*x, end
```

In cases where the interval is not 1, use:

```
>> for i = 1:2:20, x = a*x, end
```

While loop:

```
>> S = 1; n = 1;
>> while S+(n+1)^2 < 100
n = n+1; S = S + n^2;
end
>> [n, S]
ans =
6 91
```

More on programming: http://www.mathworks.com/help/techdoc/learn_matlab/f4-8955.html

5. References

- Griffiths D.F. (2005) An Introduction to MATLAB (v2.3), http://www.maths.dundee.ac.uk/~ftp/na-reports/MATLABNotes.pdf
- MATLAB: Getting started http://www.mathworks.com/help/pdf_doc/MATLAB/getstart.pdf
- http://www.math.utah.edu/lab/ms/MATLAB/MATLAB.html
- MATLAB tutorials and Learning Resources http://www.mathworks.com/academia/student_center/tutorials/launchpad.html
- http://www.math.utah.edu/lab/ms/matlab/matlab.html