

MATLAB Tutorial EE351M DSP

Created: Thursday Jan 25, 2007
Rayyan Jaber

Modified by: Kitaek Bae



Outline

- Part I: Introduction and Overview
- Part II: Matrix manipulations and common functions
- Part III: Plots in MATLAB
- Part IV: M Files



What is MATLAB?

- MATLAB® (MATrix LABoratory) is a high-performance language for technical computing. It is an interactive system whose basic data element is an array **that does not require dimensioning**. Allows to solve many technical computing problems, examples of which include:

- Matrix manipulation
- Finding the roots of polynomials
- Digital signal processing applications (toolbox)
- Plotting: x-y and polar, 3D graphics

Particularly helpful for:

- Algorithm development,
- Modeling, simulation, and prototyping,
- Data acquisition
- Data analysis, exploration, and visualization,
- Application development, including graphical user interface building.

Access to Matlab

- Access to MATLAB
 - MATLAB versions for public use are available in the ECE Learning Resource Centers in ENS 317, and ACA1.102. A student version of Matlab for the PC or Mac may be purchased at the bookstore for roughly \$100.
<http://computerstore.utexas.edu/new/?group=students&category=software:misc>
 - Excellent MATLAB tutorials are available on the UT Austin Web site:
 - MATLAB 7: <https://www.utexas.edu/its/rc/tutorials/matlab/>
 - ITS provides free consulting for UT-Austin student members using MATLAB for education and research purposes. However, direct assistance with homework, class assignments, or projects of a commercial nature is not available.
 - <http://www.utexas.edu/cc/amicus/index.html>
 - Remote access via SSH client such as PuTTY :
 - linux01.ece.utexas.edu through linux24.ece.utexas.edu
 - sun01.ece.utexas.edu through sun08.ece.utexas.edu
 - login as your ECE ID with password
 - Type "module load matlab", then run "matlab -nodisplay"

MATLAB System

- **MATLAB system consists of five main parts**
 - **Development Environment**
 - Set of tools and facilities that help you use MATLAB functions and files.
 - **MATLAB Mathematical Function Library**
 - Collection of functions like sum, sine, cosine, and complex arithmetic, matrix inverse, matrix eigenvalues, and fast Fourier transforms.
 - **The MATLAB Language**
 - High-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features.
 - **Graphics**
 - Provides extensive facilities for displaying vectors and matrices as graphs, as well as annotating and printing these graphs.
 - **Application Program Interface (API)**

5



Starting and Quitting MATLAB

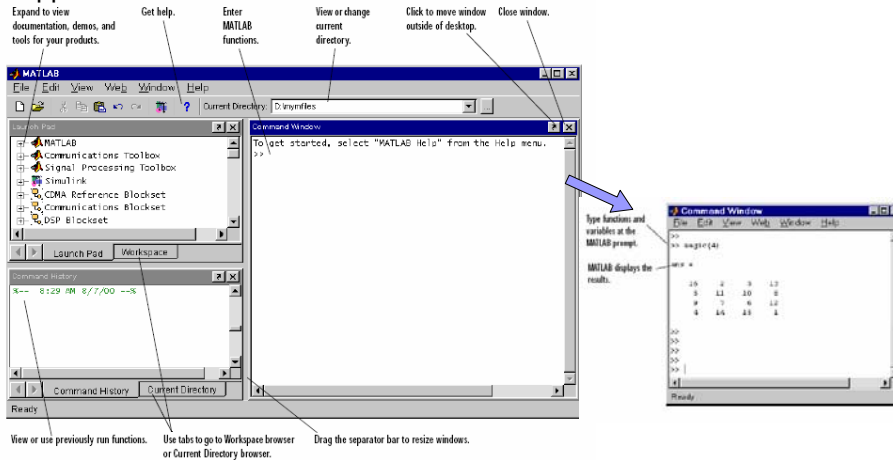
- **Starting MATLAB**
 - On Windows platforms, to start MATLAB, double-click the MATLAB shortcut icon on your Windows desktop.
 - On the Unix machines in the Learning Resource Center, MATLAB is installed in the directory `/usr/local/packages/matlab`. The MATLAB executable is installed in the `/usr/local/bin` directory, which should already be on your path. To run MATLAB, type `matlab`
- **Quitting MATLAB**
 - To end your MATLAB session, select Exit MATLAB from the File menu in the desktop, or type `quit` in the Command Window.

6



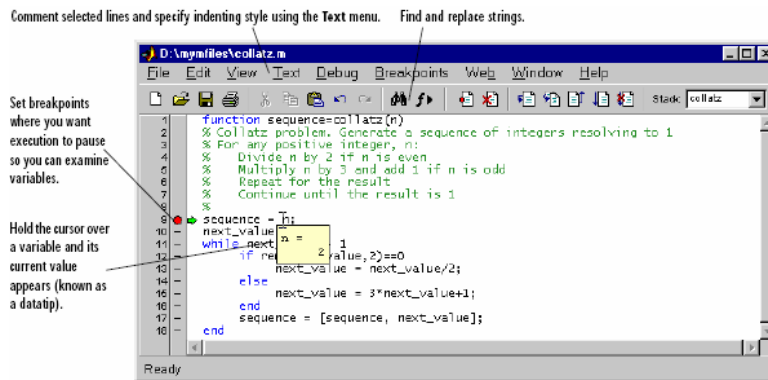
MATLAB Desktop

When you start MATLAB, MATLAB desktop appears, containing tools (graphical user interfaces) for managing files, variables, and applications associated with MATLAB.



Editor/Debugger

Use the Editor/Debugger to create and debug M-files, which are programs you write to run MATLAB functions. The Editor/Debugger provides a graphical user interface for basic text editing, as well as for M-file debugging.



Help in MATLAB

- `>> help <functionname>`
Shows help document for a give function
Example: `help mean`
- `>> lookfor <keyword>`
Searches all the help documents for a given keyword
Example: `lookfor average`
- `>> demo`
- `>> cd` : change default directory
- `>> pwd` : list current directory
- `>> dir/ls` : list contents of default



MATLAB Command Window

- `>>` Matlab command prompt
- `>>` [Enter] to execute and get new prompt
- `>> <command>; <command>` multiple commands on one line separated by “;”
- `>> <variable>` will list variable or matrix
- `>> % <comment>` delimiter for comments
blank lines are ignored
- `>> <expression>;` “;” suppresses listing of expression evaluation
- `>> <long command line> . . .` long command lines can be continued on
- `>> <continuation of command line>` next line using “. . .”
- `>> [Up or Down Arrow]` scroll through previous commands



Variables: Scalars, Vectors and Matrices

Real Scalars

```
>> x = 5  
x = 5
```

Complex Scalars

```
>> x = 5+10j    %5+10i works, as does 5+10*j  
x =  
5.0000 +10.0000i
```

Row Vector (1 x 3)

```
>> x = [ 1 2 3 ]  
x =  
1 2 3
```

Column Vector (3 x 1)

```
>> x = [ 1 ; 2 ; 3 ];    %";" suppresses output  
>> x  
x =  
1  
2  
3
```

Matrix (3 x 3)

```
>> x = [ 1 2 3 ; 4 5 6 ; 7 8 9 ]  
x =  
1 2 3  
4 5 6  
7 8 9
```

Note: Variable Names are case sensitive



Generating Vectors and the Colon Operator

(Helpful for generating time vectors.)

```
>> x = [ 0 : 0.2 : 1 ]    % 0 to 1 in increments of 0.2
```

```
x =  
0 0.20 0.40 0.60 0.80 1.00
```

```
>> x = linspace(0, 1, 6)    % 6 points from 0 to 1 on a linear scale
```

```
x =  
0 0.20 0.40 0.60 0.80 1.00
```

```
>> x = logspace(0,1,6)    % 6 points from 10^0 to 10^1 on a log scale
```

```
x =  
1.0000 1.5849 2.5119 3.9811 6.3096 10.0000
```



Generating Matrices

```
>> B = [ 1 2 ; 8 9 ]
ans =
     1     2
     8     9

>> ones(2,2) % generates an all ones 2 x 2 matrix
ans =
     1     1
     1     1

>> zeros(2,3) % generates an all zero 2 x 3 matrix
ans =
     0     0     0
     0     0     0

>> rand(3,3) % generates a random 3 x 3 matrix
ans =
    0.4447    0.9218    0.4057
    0.6154    0.7382    0.9355
    0.7919    0.1763    0.9169

>> eye(2) % generates the 2 x 2 identity matrix
ans =
     1     0
     0     1
```



Accessing Matrix Elements

- NOTE: Matlab indexes sequences beginning with 1. 0 or negative indices are not allowed!

```
>>A(m,n) % mth row, nth column

>>A(n) % nth sequence

>>A(:,n) % nth column

>>A(m,:) % mth row

>>A(1:K,n) % elements 1-K of the nth column

>>A(2*K+1,n) % expressions allowed as indices
```



Accessing Matrix Elements

```
>> A = [ 1 2 3 ; 4 5 6 ; 7 8 9];
```

```
>> x = A ( 1, 3)   %A(<row>,<column>)
```

```
x =  
    3
```

```
>> y = A ( 2, :)   % selects the 2nd row
```

```
y =  
    4    5    6
```

```
A =  
    1    2    3  
    4    5    6  
    7    8    9
```

```
>> z = A ( 1:2, 1:3) % selects sub-matrix
```

```
z =  
    1    2    3  
    4    5    6
```

How do I select the first column?



Concatenating, Appending, ...

```
>> R = [ 1 2 3 ];
```

```
>> S = [ 10 20 30 ];
```

```
>> T = [ R S ]
```

```
T =  
    1    2    3   10   20   30
```

```
>> Q = [ R ; S ]
```

```
Q =  
    1    2    3  
   10   20   30
```

```
>> Q ( 3, 3) = 100
```

```
Q =  
    1    2    3  
   10   20   30  
    0    0  100
```

if you store a value in an element outside of the matrix, the size increases to accommodate the newcomer.



Workspace and Variables

- Workspace:
 - who** % lists workspace variables in command window
 - whos** % lists full variable information (size, etc.)
 - clear <variable>** % clears single variable from workspace
 - clear or clear all** % clears all variables from current workspace
 - size(A)** % gives matrix size
 - length(x)** % gives vector length
 - disp(x)** % displays variable x
- Variables
 - Matlab automatically allocates appropriate amount of storage when it encounters a new variable name.
 - Names consist of a letter followed by any number of letters, numbers and underscores
Num_samples, value_12, X, x
- NOTE: Matlab is case sensitive.**



Operators

- Mathimatical Operators:
 - + addition
 - subtraction
 - * matrix multiplication
 - .* scalar (point by point) multiplication
 - / matrix division
 - ./ scalar (point by point) division
 - ^ matrix power
 - .^ scalar (point by point) power
 - ' conjugate transpose (Hermitian)
 - .' transpose without conjugation
 - = assignment
- Relational Operators:
 - == relational equality
 - ~= not equal
 - < less than
 - > greater than
 - <= less than or equal
 - >= greater than or equal
- Logical Operators:
 - & and
 - | or
 - ~ not
 - xor



Complex Number Operations

```
>> x = 3+4j

>> abs(x)      %Absolute value.
x = 5

>> angle(x)   %Phase angle (in radians).
x = 0.9273

>> conj(x)    %Complex conjugate.
x = 3-4j

>> imag(x)    %Complex imaginary part.
x = 4

>> real(x)    %Complex real part.
x = 3
```



Some Useful Functions

Some useful math functions:

$\sin(x)$, $\cos(x)$, $\tan(x)$, $\text{atan}(x)$, $\exp(x)$, $\log(x)$, $\log_{10}(x)$, $\text{sqrt}(x)$

```
>> t = [ 0 : 0.01 : 10 ];
```

```
>> x = sin ( 2 * pi * t );
```

Some useful matrix and vector functions:

```
>> size (A)
```

```
ans =
```

```
3 3
```

```
>> length ( t )
```

```
ans =
```

```
1001
```

A =

```
1 2 3
```

```
4 5 6
```

```
7 8 9
```



More Operators and Functions

For vectors, SUM(X) is the sum of the elements of X. For matrices, SUM(X) is a row vector with the sum over each column.

```
>> sum ( A )
ans =
    12    15    18
```

```
>> sum ( ans )    % equivalent to sum(sum(A))
ans =
    45
```

```
A =
    1    2    3
    4    5    6
    7    8    9
```

```
>> A'
ans =
     1     4     7
     2     5     8
     3     6     9
```

% equivalent to transpose(A)

```
>> diag(A)
ans =
     1
     5
     9
```

What is sum(A)' ?

21



Term by Term vs Matrix Operations

```
>> B = [ 1 2 ; 3 4 ];
```

```
B =
     1     2
     3     4
```

$$\begin{matrix} B & B \\ \left(\begin{array}{cc} 1 & 2 \\ 3 & 4 \end{array} \right) & \left(\begin{array}{cc} 1 & 2 \\ 3 & 4 \end{array} \right) \end{matrix}$$

```
>> C = B * B    % or equivalent B^2
```

```
C =
     7    10
    15    22
```

```
>> D = B .* B    % or B.^2 The Dot denotes term by term operations
```

```
D =
     1     4
     9    16
```

What is B / B ? B ./ B ?

22



Roots of Polynomials and PFE

- Find the roots of the polynomial: $13x^3 + 25x^2 + 3x + 4$

```
>> C = [13 25 3 4];
>> r = roots(C)
r =
    -1.8872
   -0.0179 + 0.4034i
   -0.0179 - 0.4034i
```

- Partial Fraction Expansion:

```
>> [R,P,K] = Residue([5,3],[1 3 0 -4])
R =
   -0.8889
    2.3333
    0.8889
P =
   -2.0000
   -2.0000
    1.0000
K = % K is the remainder
    []
```

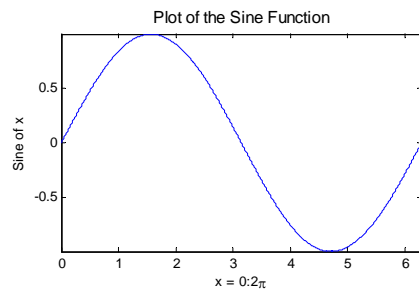
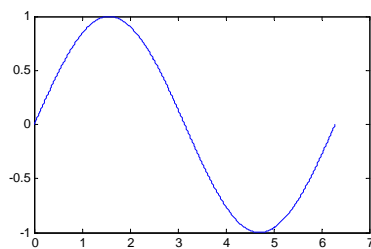
$$\frac{5s+3}{s^3+3s^2-4}$$

$$= \frac{-\frac{8}{9}}{s+2} + \frac{\frac{7}{3}}{(s+2)^2} + \frac{\frac{8}{9}}{s-1}$$



Continuous Time Plots

- The **plot** function has different forms, depending on the input arguments. If y is a vector, $\text{plot}(y)$ produces a piecewise linear graph of the elements of y versus the index its elements. If you specify two vectors as arguments, $\text{plot}(x,y)$ produces a graph of y versus x .



```
>> x = [ 0 : 0.001 : 2*pi ];
>> y = sin ( x );
>> plot (x,y) ;
```

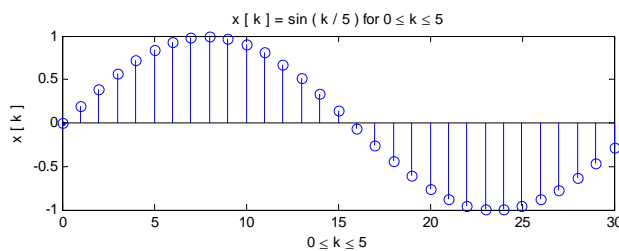
```
>> xlabel('x = 0:2*pi')
>> ylabel('Sine of x')
>> title('Plot of the Sine Function','FontSize',12)
>> axis tight % observe the limits of x-axis
```



Discrete Time Plots – Stems

- Stem function is very similar to plot. It is used to plot discrete time sequences. For more info: [help stem](#)
- Example:

```
>> k = [ 0 : 30 ] ;
>> x = sin ( k / 5 ) ;
>> stem ( k, x)
>> xlabel('0 \leq k \leq 5');
>> ylabel('x [ k ]');
>> title('x [ k ] = sin ( k / 5 ) for 0 \leq k \leq 5');
```

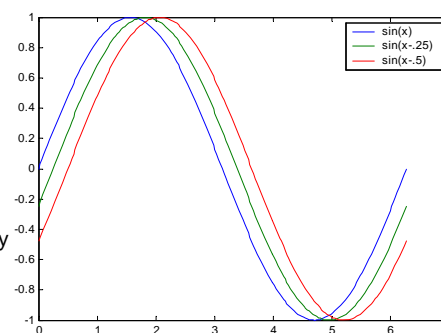


Multiple Data Sets in One Graph

- Multiple x-y pair arguments create multiple graphs with a single call to plot. For example, these statements plot three related functions of x, each curve in a separate distinguishing color.

```
x = 0:pi/100:2*pi;
y = sin(x);
y2 = sin(x-.25);
y3 = sin(x-.5);
plot(x,y,x,y2,x,y3)
legend('sin(x)', 'sin(x-.25)', 'sin(x-.5)')
```

The **legend** command provides an easy way to identify the individual plots.

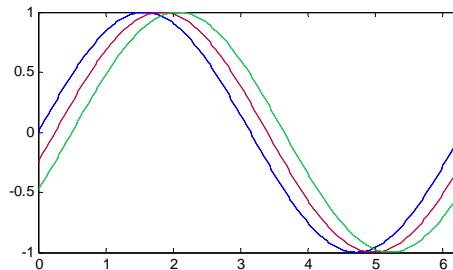


“Hold” Command

- You can also generate multiple overlapping graphs using the hold command as follows (for more info: [help hold](#))

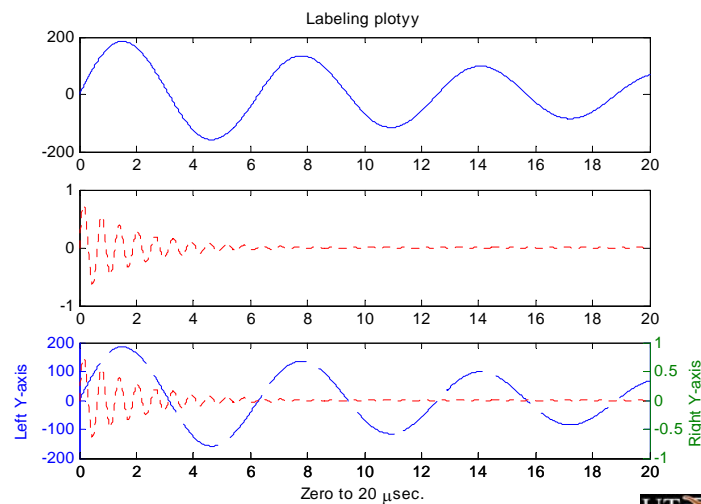
- Example;

```
x = 0 : pi/100 : 2*pi;  
y = sin(x);  
plot(x,y,'b');  
hold on;  
y2 = sin(x - 0.25);  
plot(x,y2,'r');  
y3 = sin(x - 0.5);  
plot(x,y3,'g');  
axis tight;  
hold off;
```



plotyy

- plotyy**(X1,Y1,X2,Y2) plots X1 versus Y1 with y-axis labeling on the left and plots X2 versus Y2 with y-axis labeling on the right.



28



Function

- Syntax
 - `function [out1, out2, ...] = funname(in1, in2, ...)`
- Description
 - `function [out1, out2, ...] = funname(in1, in2, ...)` defines function `funname` that accepts inputs `in1, in2, etc.` and returns outputs `out1, out2, etc.`
- Demo Example



M-files

- M-files are simply text edited files which are sequences of Matlab commands.
- They can be run from the command window or called from another M-file.
- A function is a special kind of M-file which allows parameter passing and local variable scope. (Find help on function)
- A Matlab M-file can be a program using
 - **if**
 - **then**
 - **else**
 - **elseif**
 - **end**
- condition flow as well as **for** loops and **while** loops.
 - See: **if, then, else, elseif, end, for, while, break**



Precious Tip 1

Notice that the array subscripts in Matlab start from 1 and not 0 as is the case in C.

Illustration:

```
>> A = [1 2 3 4];  
>> A(0) % This command will result in syntax error
```

The first index is 1 and not 0.

```
>> A(1)  
ans = 1
```



Precious Tip 2

Tip 2: MATLAB is designed to perform “vector operations”. Long “for-loops” on the other hand are not efficient. You can see that by comparing the following two pieces of code that do the very same thing: filling an array A with 5's.

```
X = 5 * ones(10^7,1);
```

OR

```
Y = zeros(10^7,1);  
for i = 1 : length(A)  
    Y(i) = 5;  
end
```

Try running the above two pieces of codes (red and blue one) in Matlab (Just copy and paste on the command prompt). Compare the time it takes for Matlab to execute them. What do you conclude?

Note: the results of two pieces of code are the same!



Precious Tip 2

- Create a random vector with 1,000,000 elements and compute the sum of the squared elements (i) with a for loop and scalar operations and (ii) only with matrix operations. Measure the time with the commands *tic* and *toc*.

```
clear all;
close all;
tic
y = zeros(1,1);
for i = 1:1000000
    x = rand(1,1);
    y = y + x^2;
end
toc
```

```
clear all;
close all;
tic
x = rand(1,1000000);
y = sum(x.^2);
toc
```



Precious Tip 3

- When you create a function in a Matlab .m file and you want to call it from the workspace, make sure the name of the .m file is the same as the name of the function itself (good programming practice).
- Otherwise, you have to call it through its .m name and not through the function name!



Additional References:

- <http://web.mit.edu/6.003/www/matlab.html>
- http://web.mit.edu/6.003/www/Labs/matlab_6003.pdf
- http://web.mit.edu/6.003/www/Labs/matlab_tut.pdf
- http://www.geo.utexas.edu/courses/325K/MatlabPrimer_brief.pdf

