**Department of Electrical Engineering**

|  |  |
| --- | --- |
| **Faculty Member:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **Dated: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
|  |  |
| **Course/Section:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **Semester: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
|  |  |

**EE-232 Signals and Systems**

**Lab #1 Introduction to Matlab**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Reg. no.** | **Report Marks / 10** | **Viva Marks / 5** | **Total/15** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Lab1: Introduction to MATLAB**

**Objectives**

This Lab experiment has been designed to familiarize students with MATLAB and basic understanding of MATLAB commands

* How to use MATLAB interface
* Familiarization with Mathematical operators in MATLAB
* How to use MATLAB Help
* How to handle Matrices and Vectors in MATLAB
* How to make functions in MATLAB

**Lab Report Instructions**

All questions should be answered precisely to get maximum credit. Lab report must ensure following items:

* Lab objectives
* MATLAB codes
* Results (graphs/tables) duly commented and discussed
* Conclusion

# Familiarize yourself with MATLAB

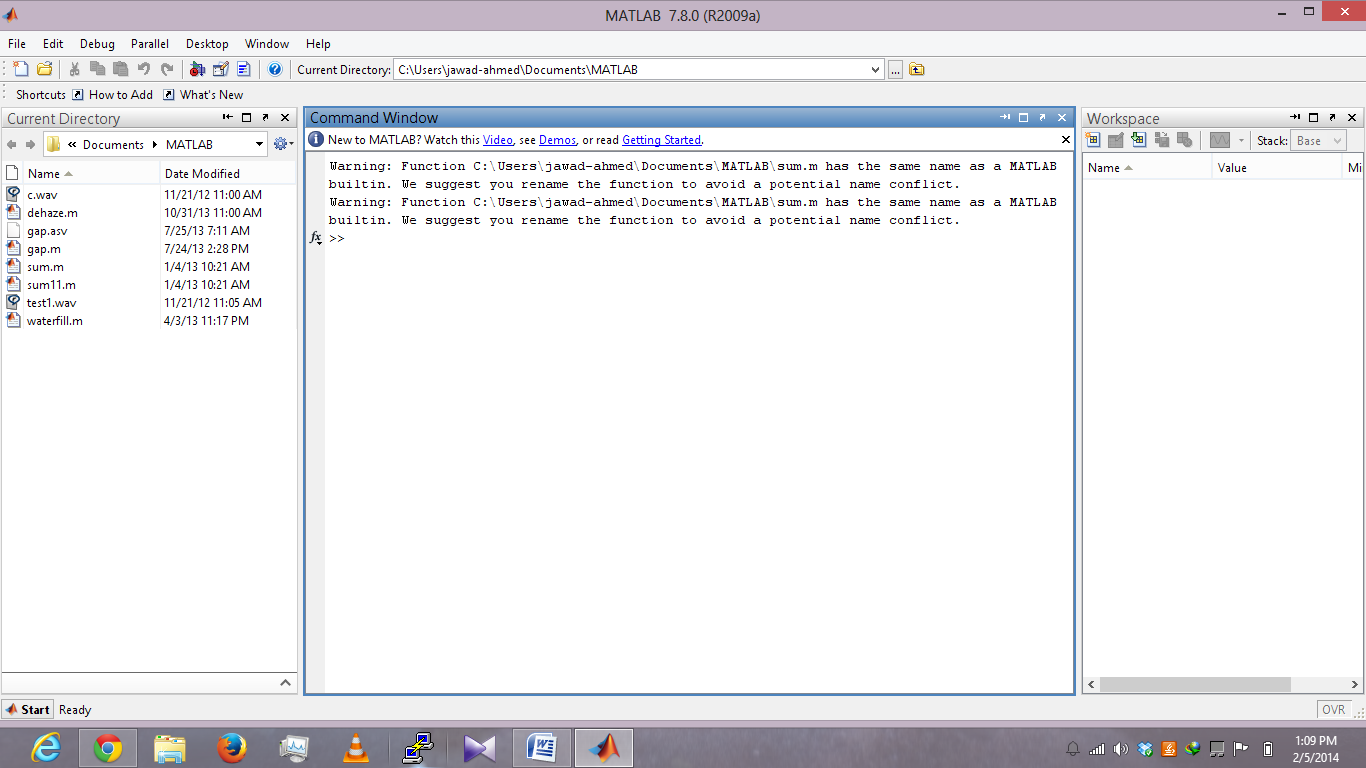
Let us introduce MATLAB in windows environment. Front-end (Graphical user interface) of MATLAB is very easy to use and is user friendly. Further it is compatible with standard windows applications and supports

* File operations
* Multiple windows view.
* help, demos and example (Check whether this component is installed)
* Wizard to create a GUI (graphical user interface)
* Wizard to profile code
* Toolboxes for different components like communication, control systems, data acquisition, curve fitting, fuzzy logic, neural network etc. (These components may or may not be available, depending upon the installation)

When MATLAB application is started, it looks like figure 1.

Figure 1: MATLAB Windows

**Menu Current Directory Workspace window with variable details**



**Command Window**

|  |  |
| --- | --- |
| **WINDOW** | **PURPOSE** |
| Command Window | To enter commands |
| Command History Window | To see previous commands |
| Workspace window | Provide information about the variables that are used |
| Current Directory Window | Shows the files in current directory |
| Launch pad window (Toolbox Menu) | Provide access to tools and demos |

Table 1: Description of common windows

There are other windows of MATLAB, which are equally important. Table 2 describes them in brief and will be introduced later.

|  |  |
| --- | --- |
| **WINDOW** | **PURPOSE** |
| Help window | Provides Help information |
| Editor window | Text editor to write programs |
| Figure window | Output of Graphical commands |

Table 2: Description of other windows

## Pre-Lab Tasks

### The Command window

You can start MATLAB by double clicking on the MATLAB icon that should be on the desktop of your computer. This brings up the window called the **Command Window**. This window allows a user to enter simple commands. To clear the **Command Window** type **clc** and next press the **Enter** or **Return** key. PLEASE NOTE: that MATLAB does not automatically remove variables from the command window if you are working. It is therefore preferable to type clear all to remove previous variables before starting a new sequence of commands

**To** perform simple computations type a command and press the **Enter** or **Return** key. For instance,

**s = 1 + 2**

will result in the output shown below,

s = 3

Here **s** is the variable having the integer value 3.

**fun = sin(pi/4)**

fun = 0.7071

In the second example the trigonometric function sine and the constant π are used. In MATLAB they are named **sin** and **pi**, respectively. Note that the results of these computations are saved in *variables* whose names are chosen by the user. If they will be needed during your current MATLAB session, then you can obtain their values typing their names and pressing the **Enter** or **Return** key. For instance,

**s**

s = 3

Variable name begins with a letter, followed by letters, numbers or underscores. MATLAB Release 2008 recognizes only the first 63 characters of a variable name. To close MATLAB type ***exit*** in the Command Window and press **Enter** or **Return** key. A second way to close your current MATLAB session is to select **File** in the MATLAB's toolbar and next click on **Exit MATLAB** option. All unsaved information residing in the MATLAB **Workspace** will be lost.

### Numbers in MATLAB

There are three kinds of numbers used in MATLAB:

1. integers
2. real numbers
3. complex numbers

Integers are entered without the decimal point

**x\_int = 10**

x\_int = 10

However, the following number

**x\_real = 10.01**

x\_real = 10.0100

is saved as a real number. Please note that x\_int and x\_real are variable names and do not define the type of numbers.

Functions **realmin** and **realmax** return the smallest and the largest positive real numbers in MATLAB. For instance typing realmin returns the minimum number that can be represented or saved in as a real number.

**realmin**

ans = 2.2251e-308

Complex numbers in MATLAB are represented in rectangular coordinate form. The imaginary unit -1 is denoted either by **i** or **j**. **Typing either i or j results in**

**i**

ans = 0 + 1.0000i

In addition to classes of numbers mentioned above, MATLAB has three variables representing the non-numbers:

**-Inf**

**Inf**

**NaN**

The **–Inf** and **Inf** are the IEEE representations for the negative and positive infinity, respectively.

Infinity is generated by overflow or by the operation of dividing by zero. The **NaN** stands for the *not-a-number* and is obtained as a result of the mathematically undefined operations such as 0.0/0.0 or ∞ - ∞

### Workspace in MATLAB

All variables used in the current MATLAB session are saved in the **Workspace**. You can view the content of the Workspace by typing **whos** in the **Command Window**. For instance,

**whos**

Name Size Bytes Class Attributes

ans 1x1 16 double complex

fun 1x1 8 double

s 1x1 8 double

x\_int 1x1 8 double

x\_real 1x1 8 double

shows all variables used in current session.

### MATLAB HELP

One of the nice features of MATLAB is its help system. To learn more about a function, you are to use, say **abs**, type in the **Command Window**

**help abs**

ABS Absolute value.

ABS(X) is the absolute value of the elements of X. When

X is complex, ABS(X) is the complex modulus (magnitude) of the elements of X.

See also sign, angle, unwrap, hypot.

Overloaded methods:

frd/abs

distributed/abs

iddata/abs

sym/abs

Reference page in Help browser

doc abs

If you do not remember the exact name of a function you want to learn more about use command **lookfor** followed by the incomplete name of a function in the **Command Window**. Now in the command window type **help colon**

Make sure you understand how the colon operator works.

To enter a statement that is too long to be typed in one line, use three periods, followed by

**Enter** or **Return**. For instance,

**x = sin(1) - sin(2) + sin(3) - sin(4) + sin(5) -...**

**sin(6) + sin(7) - sin(8) + sin(9) - sin(10)**

x= 0.7744

You can suppress output to the screen by adding a semicolon after the statement

**u = 2 + 3;**

### MATLAB as a calculator

List of basic arithmetic operations in MATLAB include

**Operation Symbol**

addition **+**

subtraction **-**

multiplication **\***

division **/**

exponentiation **^**

Observe the output of the following expressions

pi\*pi – 10

sin(pi/4)

ans ˆ 2 %<--- "ans" holds the last result

x = sin( pi/5 );

cos( pi/5 ) %<--- assigned to what?

y = sqrt( 1 - x\*x )

ans

On complex number, the basic operations are supported. Try the following:

z = 3 + 4i, w = -3 + 4j

real(z), imag(z)

abs([z,w]) %<-- Vector constructor

conj(z+w)

angle(z)

exp( j\*pi )

exp(j\*[ pi/4, 0, -pi/4 ])

**Matrices and Arrays in MATLAB**

*MATLAB* is an abbreviation for "matrix laboratory." While other programming languages mostly work with numbers one at a time, MATLAB is designed to operate primarily on whole matrices and arrays. All MATLAB variables are multidimensional *arrays*, no matter what type of data. A *matrix* is a two-dimensional array often used for linear algebra.

**Array Creation**

To create an array with four elements in a single row, separate the elements with either a comma (,) or a space.

a=[1 2 3 4]

returns

a=1 2 3 4 This type of array is a *row vector*.

To create a matrix that has multiple rows, separate the rows with semicolons.

a=[1 2 3; 4 5 6; 7 8 10]

returns

a= 1 2 3

4 5 6

7 8 10

Another way to create a matrix is to use a function, such as **ones**, **zeros**, or **rand**. For example, create a 5-by-1 column vector of zeros.

z = zeros(5,1)

returns

z=0

0

0

0

0

**Colon Operator:**

The colon alone, without start or end values, specifies all of the elements in that dimension. For example, select all the columns in the third row of a:

a(3,:)

ans = 7 8 10

The colon operator also allows you to create an equally spaced vector of values using the more general form

start:step:end.

A = 0:10:100

A= 0 10 20 30 40 50 60 70 80 90 100

If you omit the middle step,as in start:end, MATLAB uses the default step value of 1.

## Lab Tasks

### Lab Task 1:

(a) Make sure that you understand the **colon** notation. In particular, explain in words what the following MATLAB code will produce

a = 0 : 6

b = 2 : 4 : 17

c = 99 : -1 : 88

d = 2 : (1/9) : 4

e = pi \* [ 0:0.1:2 ];

(b) Extracting and/or inserting numbers into a vector is very easy to do. Consider the following definition of f:

f = [ zeros(1,3), linspace(0,1,5), ones(1,4) ]

f(4:6)

size(f)

length(f)

f(2:2:length(f))

Explain the results echoed from the last four lines of the above code.

(c) Observe the result of the following assignments:

g = f; g(4:6) = pi\*(1:3)

### Lab Task 2:

Now write a statement that will take the vector f defined in part (b) and replace the even indexed elements (i.e., f(2), f(4), etc) with the constant ‘ππ ‘ (pi raised to the power pi) (Try: finding help on ‘^’ operator or the function ‘power’). *Use a vector replacement, not a loop.* Experiment with vectors in MATLAB. Think of the vector as a set of numbers. Try the following:

h = cos( pi\*(0:11)/4 ) %<---comment: compute cosines

Explain how the different values of cosine are stored in the vector h. What is h(1)? Is h(0) defined?

### Lab task 3:

Loops can be written in MATLAB, but they are NOT the most efficient way to get things done. It’s better to **always avoid loops** and use the colon notation instead. The following code has a loop that computes values of the cosine function. (The index of yy() must start at 1.) Rewrite this computation without using the loop (follow the style in the previous part).

g = [ ]; %<--- initialize the yy vector to be empty

for k=-5:5

g(k+6) = cos( k\*pi/3 )

end

g

Explain why it is necessary to write g(k+6). What happens if you use g(k) instead?

**Plotting in MATLAB**

Plotting is easy in MATLAB for both real and complex numbers. The basic plot command will plot a vector y versus a vector x connecting successive points by straight lines. Try the following:

x = [-3 -1 0 1 3 ];

y = x.\*x - 3\*x;

plot( x, y )

z = x + y\*sqrt(-1)

plot( z ) %<---- complex values: plot imag vs. real.

Use help arith to learn how the operation xx.\*xx works when xx is a vector; compare to matrix multiply.

Note: stem() command is used to plot discrete set of data.

### Lab task 4:

Go to File > New > M –file. MATLAB editor will open up. Enter the following code in the editor and then save the file as mylab1.m

clear all;

clc;

t = -1 : 0.01 : 1;

x = cos( 5\*pi\*t );

y = 1.4\*exp(j\*pi/2)\*exp(j\*5\*pi\*t);

plot( t, x, ’b-’, t, real(y), ’r--’ ), grid on

%<--- plot a sinusoid

title(’TEST PLOT of a SINUSOID’)

xlabel(’TIME (sec)’)

Now go to Command Window and type

mylab1 %<---will run the commands in the file

type mylab1 %<---will type out the contents of

% mylab1.m to the screen

Explain why the plot of real(y) is a sinusoid. What is its phase and amplitude? Make a calculation of the phase from a time-shift measured on the plot.

**Functions in MATLAB**

User-defined functions (i.e., those no pre-programmed in Matlab) can be defined by using an m file. These files are simply text files whose name ends with the suffix .m. Those m files that a user may create are typically stored in the work directory. This directory is the default working directory of a MATLAB installation.

Functions defined by m files start with the line

function [assignment\_variable(s)] = function\_name(arguments)

Here, function is a required MATLAB particle to identify the file as a function, assignment\_variable(s) is an optional dummy variable, function\_name is the name given to the function, and arguments are optional values passed on to the function from the main MATLAB interface or from within another function.

### Lab task 5:

Create a function “sigadd” to add two sequences ‘x1’ and ‘x2’.

Function [y,n]=sigadd(x1,n1,x2,n2)

Where ‘x1’ and ‘x2’ are two sequences and ‘n1’ and ‘n2’ are their respective indices vectors. Add values of ‘x1’ and ‘x2’ at corresponding indices, pad zeros if length of two sequences are not same.

Suppose x1= [1 2 3 4 5 6 7 8 9] with index n1= 3:11 and x2= [2 4 6 8 10 12 14 16 18 20 22 24] with index n2=1:12. Here you can observe that the length of both the signals is not same and the indexes of both the signals are not starting from the same point. So you have to pad zeros before adding both the sequences so that the output y will have the index values starting from 1 up to 12.

Hint: You may need the loops and if else checks. Loops syntax is already discussed above and syntax of if else is given below.

**Syntax of if/elseif/else**

Execute statements if condition is true

if *expression*

*statements*

elseif *expression*

*statements*

else

*statements*

end

if *expression*, *statements*, end evaluates an expression, and executes a group of statements when the expression is true.

elseif and else are optional, and execute statements only when previous expressions in the if block are false. An if block can include multiple elseif statements.

An evaluated expression is true when the result is nonempty and contains all nonzero elements (logical or real numeric). Otherwise, the expression is false.

**Best Resource or learning MATLAB: For getting familiar with MATLAB operations, matrices, arrays, loops and plots etc., go to the link given below which contain official documentation on mathworks with name “Getting Started with MATLAB”**

www.mathworks.com/help/pdf\_doc/**matlab**/**getstart**.pdf‎