

Name	Umar Hayyat
Roll no.	2019-EE-360
Marks	

## Lab Manual # 1: Introduction to MATLAB for Control system

### Objectives:

Analysis of the MATLAB and its use for the control systems.

### Overview:

MATLAB is a professional software used for technical computing. MATLAB stands for Matrix Laboratory. In this lab manual we are using some of the basic commands and learn how to use the control system in this software. Previously we have done use this software for data structures and it contains the built-in debugging tools and editing. It is an outstanding tool software for the engineering students.

### Starting MATLAB:

Introduction of the screen of MATLAB consists of the following options as shown in fig. 1.1.

#### ▪ **Command window:**

This is the command prompt window of the MATLAB whenever it is required to write the code and execute the program. Command window is the place where the code or the script is written.

#### ▪ **Workspace:**

This sub-window lists all variables that you have generated so far and shows their type and size. You can do various things with these variables by clicking on a variable and then using the right button on the mouse to select your option.

#### ▪ **Current Directory :**

This is where all the files from the current directory are found. It is used for the file navigation. It also have several options of what to do with a file once you select it.

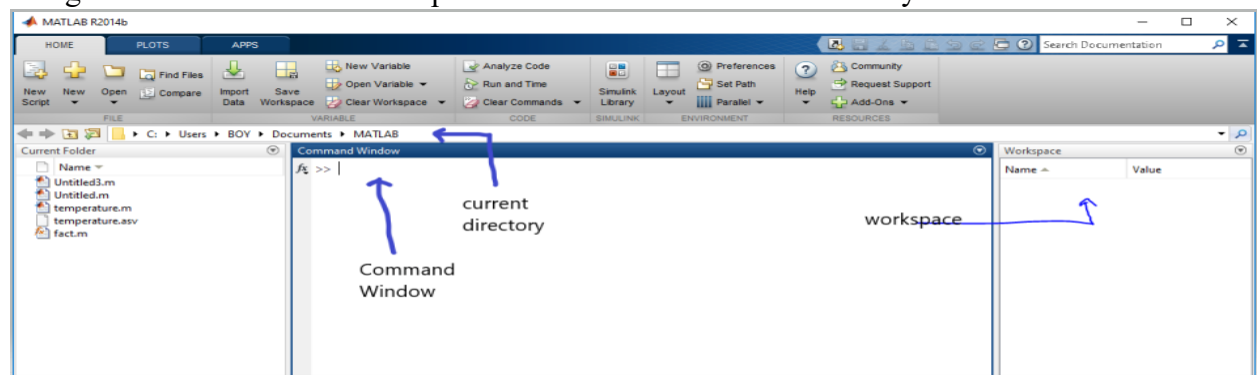
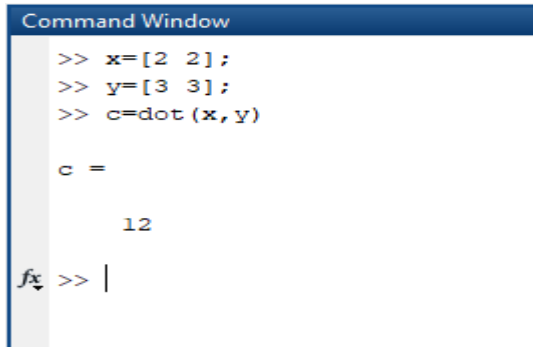


Figure 1.1: The front screen of MATLAB

**MATLAB Basic Commands:**▪ **dot product:**

The MATLAB command `dot(x,y)` returns the scalar dot product of x and y vectors as shown in fig. 1.2 below.



```

Command Window
>> x=[2 2];
>> y=[3 3];
>> c=dot(x,y)

c =

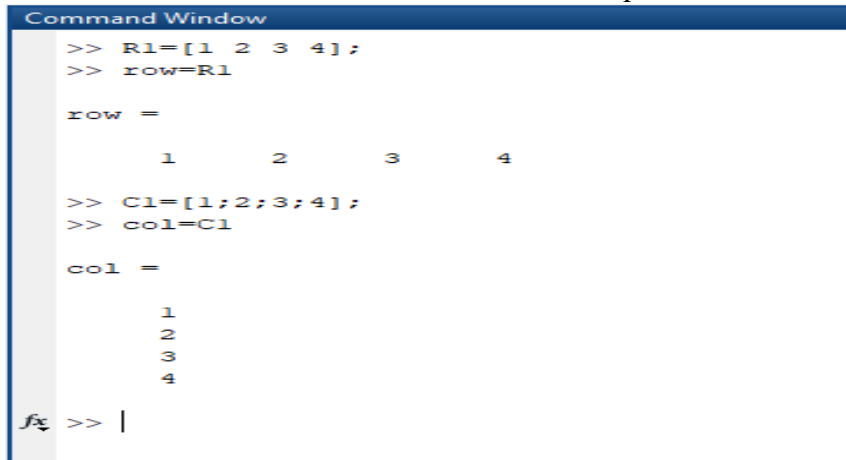
    12

fx >> |
  
```

Fig 1.2: dot product

▪ **Row & Column Vector:**

For row vector we simply put spaces between elements and for column vector we use “;” for next element of column vector enclosed in square brackets as shown in fig 1.3



```

Command Window
>> R1=[1 2 3 4];
>> row=R1

row =

     1     2     3     4

>> C1=[1;2;3;4];
>> col=C1

col =

     1
     2
     3
     4

fx >> |
  
```

Fig 1.3: row and column

▪ **linspace():**

The `linspace` function generates linearly spaced vectors or more specifically row vectors, where a is the starting point and b is the ending point and `linspace` separates a and b in c equal parts. the MATLAB is given in fig.1.4

```
Command Window
>> linspace(1,2,10)

ans =

    Columns 1 through 8

    1.0000    1.1111    1.2222    1.3333    1.4444    1.5556    1.6667    1.7778

    Columns 9 through 10

    1.8889    2.0000

fx >> |
```

Fig 1.4: linspace

#### ▪ **Matrix:**

A matrix is a two-dimensional array of numbers. In MATLAB we create matrix by enclosing numbers in square brackets with space between different numbers and semicolons to mark the end of each row.

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

A =

     1     2     3
     4     5     6
     7     8     9

fx >> |
```

Fig 1.11: Matrix

#### ▪ **rand(n):**

It gives an nxn matrix having elements varying from 0 to 1, as shown below.

```
Command Window
>> rand(3)

ans =

    0.8147    0.9134    0.2785
    0.9058    0.6324    0.5469
    0.1270    0.0975    0.9575

fx >> |
```

Fig 1.12: random function

**Exercise 1:****a) Extract the fourth row of matrix generated by magic(6):**

```

Command Window
>> x=magic(5)

x =

    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9


>> y=x(4,:)

y =

    10    12    19    21     3

fx >> |

```



## Matrices

- A Matrix array is two-dimensional, having both multiple rows and multiple columns, similar to vector arrays:
  - it begins with [, and end with ]
  - spaces or commas are used to separate elements in a row
  - semicolon or enter is used to separate rows.

A is an m x n matrix.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \dots & a_{mn} \end{bmatrix}$$


the main diagonal

•Example:

```

>> f = [ 1 2 3; 4 5 6]
f =
     1     2     3
     4     5     6
>> h = [ 2 4 6
        1 3 5]
h =
     2     4     6
     1     3     5

```



## Matrices (con't...)

- Magic Function**
  - For example you can generate a matrix by entering  
`>> m=magic(4)`  
 It generates a matrix whose elements are such that the sum of all elements in its rows, columns and diagonal elements are same
- Sum Function**
  - You can verify the above magic square by entering  
`>> sum(m)`
  - For rows take the transpose and then take the sum  
`>> sum(m')`
- Diag**
  - You can get the diagonal elements of a matrix by entering  
`>> d=diag(m)`  
`>> sum(d)`

## Matrices (con't...)

- Matrix Addressing:
  - *matrixname(row, column)*
  - **colon** may be used in place of a row or column reference to select the entire row or column.
- Example:
 

```
>> f(2,3)
ans =
6
```

```
>> h(:,1)
ans =
2
1
```

**recall:**

**f =**

1	2	3
4	5	6

**h =**

2	4	6
1	3	5

## Matrices (con't...)

### Some useful commands:

zeros(n)	returns a n x n matrix of zeros
zeros(m,n)	returns a m x n matrix of zeros
ones(n)	returns a n x n matrix of ones
ones(m,n)	returns a m x n matrix of ones
rand(n)	returns a n x n matrix of random number
rand(m,n)	returns a m x n matrix of random number
size (A)	for a m x n matrix A, returns the row vector [m,n] containing the number of rows and columns in matrix.
length(A)	returns the larger of the number of rows or columns in A.



## Array Operations (con't...)

### ■ Element-by-Element Array-Array Mathematics.

<u>Operation</u>	<u>Algebraic Form</u>	<u>MATLAB</u>
Addition	$a + b$	$a + b$
Subtraction	$a - b$	$a - b$
Multiplication	$a \times b$	$a .* b$
Division	$a \div b$	$a ./ b$
Exponentiation	$a^b$	$a .^ b$

### ■ Example:

```
>> x = [ 1 2 3 ];
>> y = [ 4 5 6 ];
>> z = x .* y
z =
     4    10    18
```

Each element in x is multiplied by the corresponding element in y.



## Solutions to Systems of Linear Equations

### ■ Example: a system of 3 linear equations with 3 unknowns ( $x_1, x_2, x_3$ ):

$$3x_1 + 2x_2 - x_3 = 10$$

$$-x_1 + 3x_2 + 2x_3 = 5$$

$$x_1 - x_2 - x_3 = -1$$

Let :

$$A = \begin{bmatrix} 3 & 2 & 1 \\ -1 & 3 & 2 \\ 1 & -1 & -1 \end{bmatrix} \quad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad b = \begin{bmatrix} 10 \\ 5 \\ -1 \end{bmatrix}$$

Then, the system can be described as:

$$Ax = b$$

## Solutions to Systems of Linear Equations (con't...)

### ■ Solution by Matrix Inverse:

$$\begin{aligned} Ax &= b \\ A^{-1}Ax &= A^{-1}b \\ x &= A^{-1}b \end{aligned}$$

### ■ MATLAB:

```
>> A = [ 3 2 -1; -1 3 2; 1 -1 -1];
>> b = [ 10; 5; -1];
>> x = inv(A)*b
x =
-2.0000
 5.0000
-6.0000
```

Answer:

$x_1 = -2, x_2 = 5, x_3 = -6$

**NOTE:**

left division:  $A \backslash b \rightarrow b \div A$

### ■ Solution by Matrix Division:

The solution to the equation  $Ax = b$  can be computed using **left division**.

### ■ MATLAB:

```
>> A = [ 3 2 -1; -1 3 2; 1 -1 -1];
>> b = [ 10; 5; -1];
>> x = A\b
x =
-2.0000
 5.0000
-6.0000
```

Answer:

$x_1 = -2, x_2 = 5, x_3 = -6$

right division:  $x/y \rightarrow x \div y$

### ■ **plot(x):**

This command is used to plot graphs of different functions and systems. For example, the graph of sin of linspace is plotted below with help of plot command.

## Plotting

- For more information on 2-D plotting, type **help graph2d**

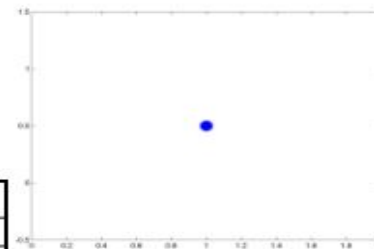
- Plotting a point:

```
>> plot ( variablename, 'symbol')
```

the function **plot ()** creates a graphics window, called a **Figure window**, and named by default **"Figure No. 1"**

- Example : Complex number

```
>> z = 1 + 0.5j;
>> plot (z, '.')
```



commands for axes:

command	description
axis ([xmin xmax ymin ymax])	Define minimum and maximum values of the axes
axis square	Produce a square plot
axis equal	equal scaling factors for both axes
axis normal	turn off axis square, equal
axis (auto)	return the axis to defaults





## Plotting (con't...)

- Plotting Curves:
  - **plot (x,y)** – generates a linear plot of the values of x (horizontal axis) and y (vertical axis).
  - **semilogx (x,y)** – generate a plot of the values of x and y using a logarithmic scale for x and a linear scale for y.
  - **semilogy (x,y)** – generate a plot of the values of x and y using a linear scale for x and a logarithmic scale for y.
  - **loglog(x,y)** – generate a plot of the values of x and y using logarithmic scales for both x and y



## Plotting (con't...)

- Multiple Curves:
  - **plot (x, y, w, z)** – multiple curves can be plotted on the same graph by using multiple arguments in a plot command. The variables x, y, w, and z are vectors. Two curves will be plotted: y vs. x, and z vs. w.
  - **legend ('string1', 'string2',...)** – used to distinguish between plots on the same graph
- Multiple Figures:
  - **figure (n)** – used in creation of multiple plot windows. place this command before the plot() command, and the corresponding figure will be labeled as "Figure n"
  - **close** – closes the figure n window.
  - **close all** – closes all the figure windows.
- Subplots:
  - **subplot (m, n, p)** – m by n grid of windows, with p specifying the current plot as the p<sup>th</sup> window



## Plotting (con't...)

### Example: (polynomial function)

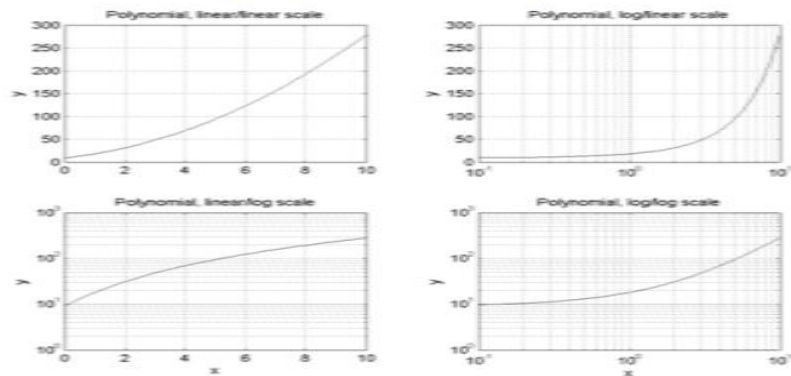
plot the polynomial using linear/linear scale, log/linear scale, linear/log scale, & log/log scale:

$$y = 2x^2 + 7x + 9$$

```
% Generate the polynomial:
x = linspace (0, 10, 100);
y = 2*x.^2 + 7*x + 9;

% plotting the polynomial:
figure (1);
subplot (2,2,1), plot (x,y);
title ('Polynomial, linear/linear scale');
ylabel ('y'), grid;
subplot (2,2,2), semilogx (x,y);
title ('Polynomial, log/linear scale');
ylabel ('y'), grid;
subplot (2,2,3), semilogy (x,y);
title ('Polynomial, linear/log scale');
xlabel ('x'), ylabel ('y'), grid;
subplot (2,2,4), loglog (x,y);
title ('Polynomial, log/log scale');
xlabel ('x'), ylabel ('y'), grid;
```

## Plotting (con't...)



## Plotting (con't...)

- Adding new curves to the existing graph:
- Use the **hold** command to add lines/points to an existing plot.
  - hold on – retain existing axes, add new curves to current axes. Axes are rescaled when necessary.
  - hold off – release the current figure window for new plots
- Grids and Labels:

Command	Description
grid on	Adds dashed grids lines at the tick marks
grid off	removes grid lines (default)
grid	toggles grid status (off to on, or on to off)
title ('text')	labels top of plot with text in quotes
xlabel ('text')	labels horizontal (x) axis with text in quotes
ylabel ('text')	labels vertical (y) axis with text in quotes
text (x,y,'text')	Adds text in quotes to location (x,y) on the current axes, where (x,y) is in units from the current plot.

**Exercise 1:**

Use Matlab command to obtain the following

- Extract the fourth row of the matrix generated by magic (6).
- Show the results of 'x' multiply by 'y' and 'y' divides by 'x'.  
Given  $x = [0:0.1:1.1]$  and  $y = [10:21]$
- Generate random matrix 'r' of size 4 by 5 with number varying between -8 and 9

**Output (a):**

```
Command Window
>> a=magic(6)

a =

    35     1     6    26    19    24
     3    32     7    21    23    25
    31     9     2    22    27    20
     8    28    33    17    10    15
    30     5    34    12    14    16
     4    36    29    13    18    11

>> a(4,:)

ans =

     8    28    33    17    10    15

fx >> |
```

**Output (b):**

```

Command Window

>> x = [0:0.1:1.1];
>> y = [10:21];
>> m=x.*y

m =

Columns 1 through 10

    0    1.1000    2.4000    3.9000    5.6000    7.5000    9.6000   11.9000   14.4000   17.1000

Columns 11 through 12

   20.0000   23.1000

>> d=y./x

d =

Columns 1 through 10

   Inf  110.0000   60.0000   43.3333   35.0000   30.0000   26.6667   24.2857   22.5000   21.1111

Columns 11 through 12

   20.0000   19.0909

fx >> |

```

**Output (c):**

```

Command Window

>> r=randi([-8,9],4,5)

r =

    -1    -6    -7    -8     5
    -4     8    -4    -8     3
    -1     9    -2    -5     0
    -7     2     6     3     1

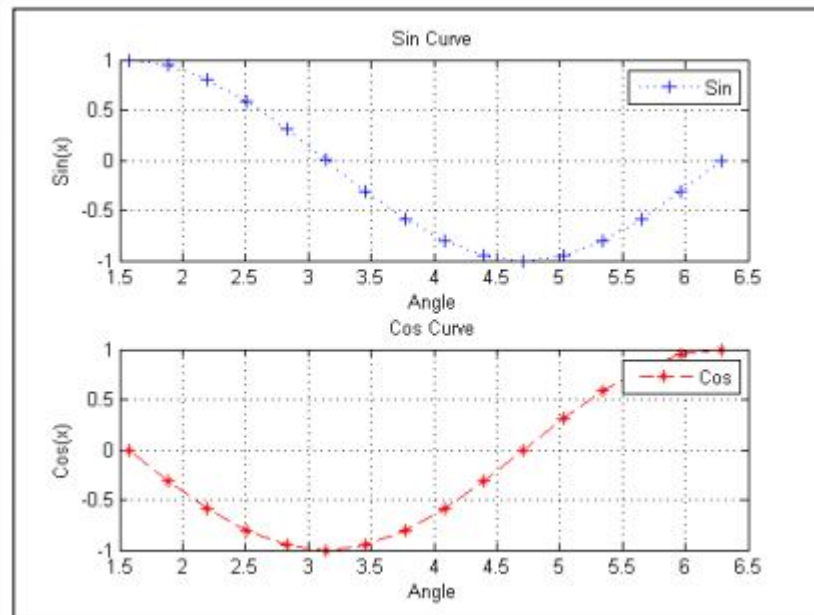
fx >> |

```

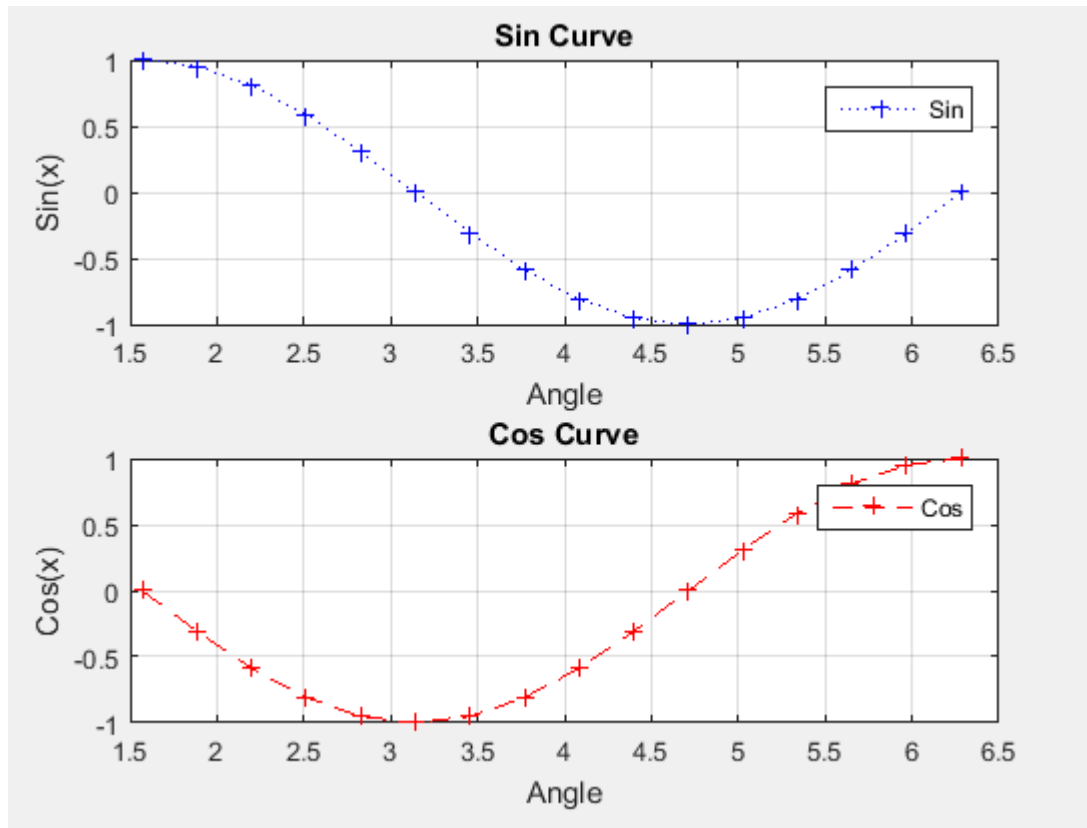
**Exercise 2:**

Use MATLAB commands to get exactly as the figure shown below

```
x=pi/2:pi/10:2*pi;
y=sin(x);
z=cos(x);
```

**Output:****Command Window**

```
>> x = pi/2 : pi/10 : 2*pi;
>> y = sin(x);
>> z = cos(x);
>> subplot(2,1,1),plot(x,y,':b+');
>> grid on;
>> title('Sin Curve');
>> xlabel('Angle');
>> ylabel('Sin(x)');
>> legend('Sin');
>> subplot(2,1,2),plot(x,z,'--r+');
>> title('Cos Curve');
>> xlabel('Angle');
>> ylabel('Cos(x)');
>> legend('Cos');
>> grid on;
>> |
~~
```



### **Conclusion:**

In this lab, I learned how to use command window of MATLAB. I also learned how to create matrix and perform different operation on it i.e extract any row or column. I also learned how to use `rand(n)` and `magic(n)` commands. `rand(n)` command gives  $n \times n$  matrix having elements varying from 0 to 1. I learned how to plot graph in MATLAB and change the color and style of curve.