Rawal Institute of Engineering & Technology

IT WORKSHOP

MATLAB FILE

Computer Science &Engineering

Submitted To: **Submitted By:**

INDEX

S. No.	Topic	Date	Page No.	Signature
1.	**Module 1:** **- MATLAB Introduction** **-Features** **-Applications**			
2.	**Module 2 :Introduction of** **-Commands** **-Data types** **-Data conversion**			
3.	**Module 3 :** **-Operators** **-Mathematical functions** **-Functions**			
4.	**Module 4 :** **Matrix manipulations using MATLAB**			
5.	**Module 5 :Programming in MATLAB** **-If** **-Switch** **-For** **-While** **-Break**			
6.	**Module 6 :Plot Different Types of Graphs**			

Module - 1

Aim: MATLAB Introduction

MATLAB is a programming platform designed specifically for engineers and scientists to analyze and design systems and productsthat transform our world. The heart of MATLAB is the MATLAB language, a matrix-based language allowing the most natural expression of computational mathematics.

Features of MATLAB

- **MATLAB is a high-level language:** MATLAB Supports Object oriented programming. It also supports different types of programming constructs like Control flow statements (IFELSE, FOR, WHILE). MATLAB also supports structures like in C programming, Functional programming (writing functions to contain commonly used code and later callingthem). It also contains Input / Output statements like disp()and input().

- **Interactive graphics:** MATLAB has inbuilt graphics to enhance user experience. We can actually visualize whatever data is there in forms of plots and figures. It also supports processing of image and displaying them in 2D or 3D formats. We can visualize and manipulate our data across any of the three dimensions (1D, 2D, and 3D). We can plot the functions and customize them also according to our needs like changing bullet points, line color and displaying/not displaying grid.

- **A large library of Mathematical functions:** MATLAB has a huge inbuilt library of functions required for mathematical analysis of any data. It has common math functions like sqrt. factorial etc. It has functions required for statistical analysis like median, mode and std (to find standard deviation), and much more. MATLAB also has functions for signal processing like filter, butter(Butterworth filter design) audio read, Conv, xcorr, fft, fftshift etc. It also supports image processing and some common functions required for image processing in MATLAB are rgb2gray,rgb2hsv, adaptthresh etc.

- **Interactive environment:** MATLAB offers interactive environment by providing a GUI (Graphical user interface) and different types of tools like signal analyses and tuners. MATLAB also has tools for debugging and the development of any software. Importing and exporting files becomes easy in MATLAB through the GUI. We can view the workspace data as we progress in the development of our software and modify it according to our needs.

- **MATLAB can interface with different languages:** We can write a set of codes (libraries) in languages like PERLand JAVA, and we can call those libraries from within the MATLAB itself. MATLAB also supports ActiveX and .NETlibraries.

- **Machine Learning, Deep Learning, and Computer vision:** The most demanding technologies like Machine learning, Deep learning, and Computer vision can be done in MATLAB. We can create and interconnect layers of a deep neural network; we can build custom training loops and training layers with automatic differentiation. For machine learning, we can use the DBSCAN algorithm to discover clusters and noise in DATA. For computer vision, we can do object tracking; object recognition, gesture recognition, and processing 3D point clouds.

Applications of MATLAB

- **Statistics and machine learning (ML):** This toolbox in MATLAB can be very handy for the programmers. Statistical methods such as descriptive or inferential can be easily implemented. So is the case with machine learning. Various models can be employed to solve modern-day problems. The algorithms used can also be used for big data applications.
- **Curve fitting:** The curve fitting toolbox helps to analyze the pattern of occurrence of data. After a particular trend which can be a curve or surface is obtained, its future trends can be predicted. Further plotting, calculating integrals, derivatives, interpolation, etc can be done.
- **Control systems:** Systems nature can be obtained. Factors such as closedloop, open-loop, its controllability and observability, Bode plot, Nyquist plot, etc can be obtained.

Various controlling techniques such as PD, PI and PID can be visualized. Analysis can be done in the time domain or frequency domain.

- **Signal Processing:** Signals and systems and digital signal processing are taught in various engineering streams. But MATLAB provides the opportunity for proper visualization of this. Various transformssuch as Laplace, Z, etc can be done on any given signal. Theorems can be validated. Analysis can be done in the timedomain or frequency domain. There are multiple built-in functions that can be used.
- **Mapping** Mapping has multiple applications in various domains. For example, in Big data, the MapReduce tool is quite important which has multiple applications in the real world. Theft analysis or financial fraud detection, regression models, contingency analysis, predicting techniques in social media, data monitoring, etc can be done by data mapping

Module- 2

Aim: Introduction of commands, Data types & Data conversion

Commands for managing the session

Commands	Purpose
Clc	Clears command window
Clear	Removes variables from memory
Exit	Checks for existence of file or variable
Global	Declares variable to be global
Help	Searches for a help topic
Lookfor	Searches help entries for a keyword
Quit	Stops Matlab
Who lists current variable	Lists current variable
Whos	Lists current variable (long display)
Which	Check about files directory

Commands for working with system

Commands	Purpose
Cd	Changes current directory
Date	Displays current date
Delete	Deletes a file
Diary	Switches on/off diary file recording
Dir	Lists all files in current directory
Load	Loads workspace variables from a file
Path	Displays search path
Pwd	Displays current directory
Save	Saves workspace variables in a file
Type	Displays contents a file
What	Lists all Matlab files in the current Directory
Wklread	Reads wk1 spreadsheet file

Data Types

Data types	Description
Int8	8 bit signed integer
Uint8	8 bit unsigned integer
Int16	16 bit signed integer
Uint16	16 bit signed integer
Int32	32 bit signed integer
Uint32	32 bit unsigned integer
Int64	64 bit signed integer
Uint64	64 bit unsigned integer
Single	Single precision numerical difference
Double	Double precision numerical difference
Char	Strings are stored in the form of vectors
Cell array	Array of index cell, each of capable of storing an arrayof a different direction and data type
Structure	C like structure each structure having named fieldscapable of storing
Function handle	An array of different dimension of a data type pointerto a function
User classer	Objects constructed from user defined class
Java classer	Objects constructed from java class

Data Conversion

Function	Description
uintN (eg. Uint8)	Convert a character toan integer code that represents that character
Str2num	Convert a character type to a numeric type
Str2double	Similar to str2num, butoffers better performance and works with string arrays and cell arrays of character vectors
hex2num	Convert a numeric type to a character type of specified precision, returning acharacter array that MATLAB can evaluate
Hex2dec	Convert a charactertype of hexadecimalbase to positive integer
Bin2dec	Convert a character type of binary numberto decimal number
Base2dec	Convert a charactertype of any base number from 2 through 36 to a decimal number

String to number

```
A = str2num('2 4 6 8')

%%
```

```
Command Window

A =

     2     4     6     8

fx >>
```

String to Double

```
x=str2double('123.45e7')

%%
```

```
Command Window

x =

  1.2345e+009

fx >>
```

Hex to number

```
x=hex2num('A')
x=hex2num('B')
%%
```

```
Command Window

x =
 -1.4917e-154

x =
 -1.7272e-077

fx >>
```

Hex to Decimal

```
x=hex2dec('A')
x=hex2dec('B')
%%
```

```
Command Window
x =
    10
x =
    11
fx >>
```

Binary to decimal

```
x=bin2dec('1010')
x=bin2dec('010 111')
%%
```

```
Command Window
x =
    10
x =
    23
fx >>
```

Base to Decimal

```
x=base2dec('12',8)
x=base2dec('212',3)
%%
```

```
Command Window
x =
    10
x =
    23
fx >>
```

Program no. 3
Aim: Operators, Mathematical functions, Functions, pre-defined functions and Operator Precedence

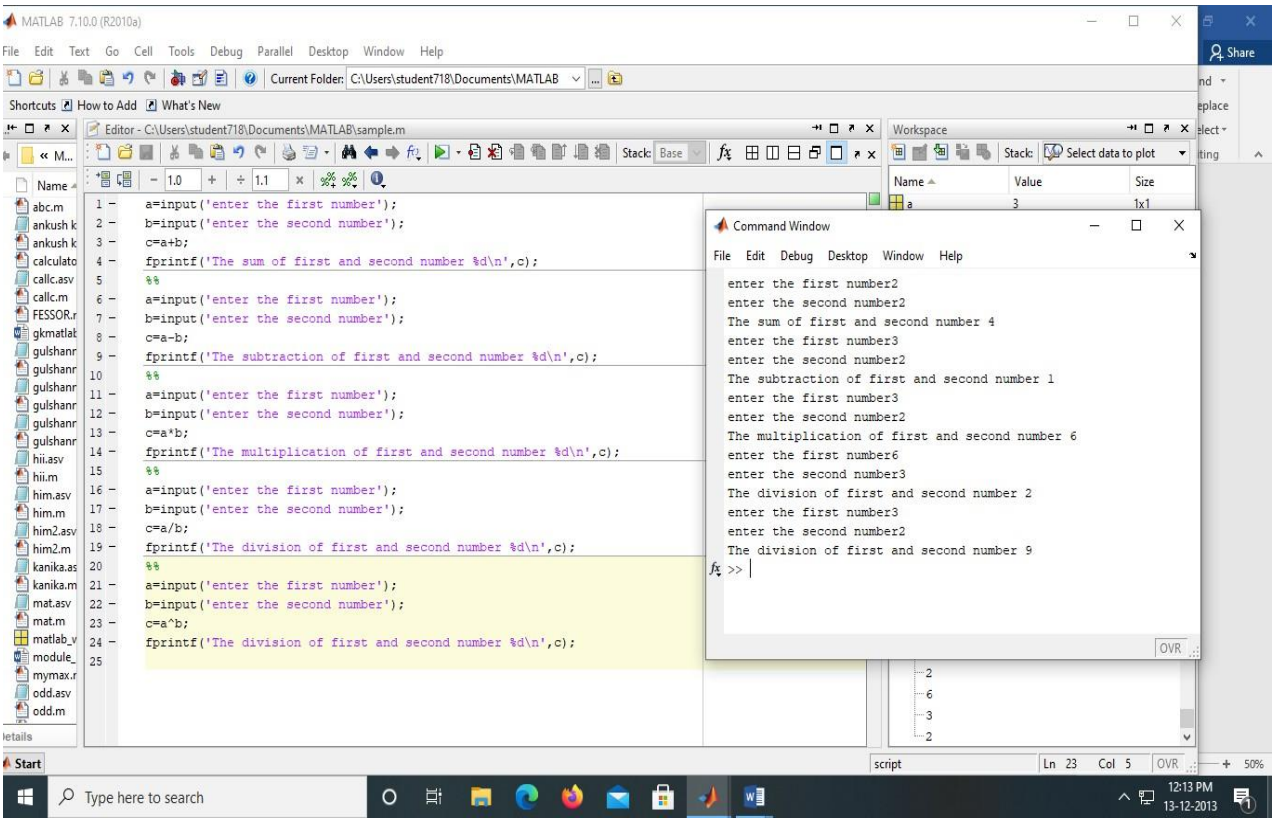
Operators: - It is a symbol that tells the compiler to perform variousnumerical or logical manipulations in MATLAB.

Types of operators:-

- **Arithmetic Operator**

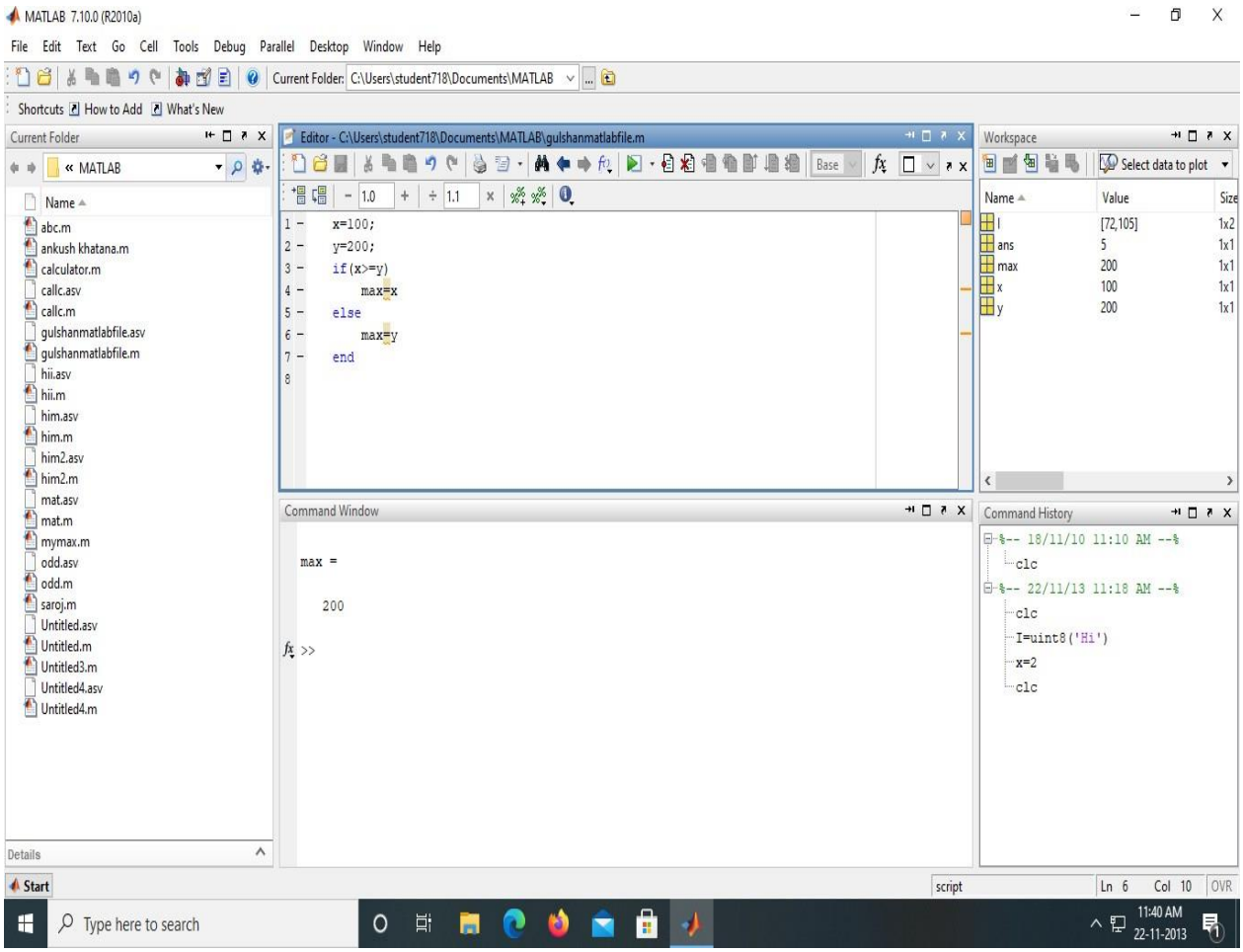
Sr. No.	Operatrors	Operations	examples
1	Addition	+	2+3
2	Subtraction	-	3-2
3	Multiplication	*	3*2
4	Division	/	6/2
5	Exponential	^	6^2

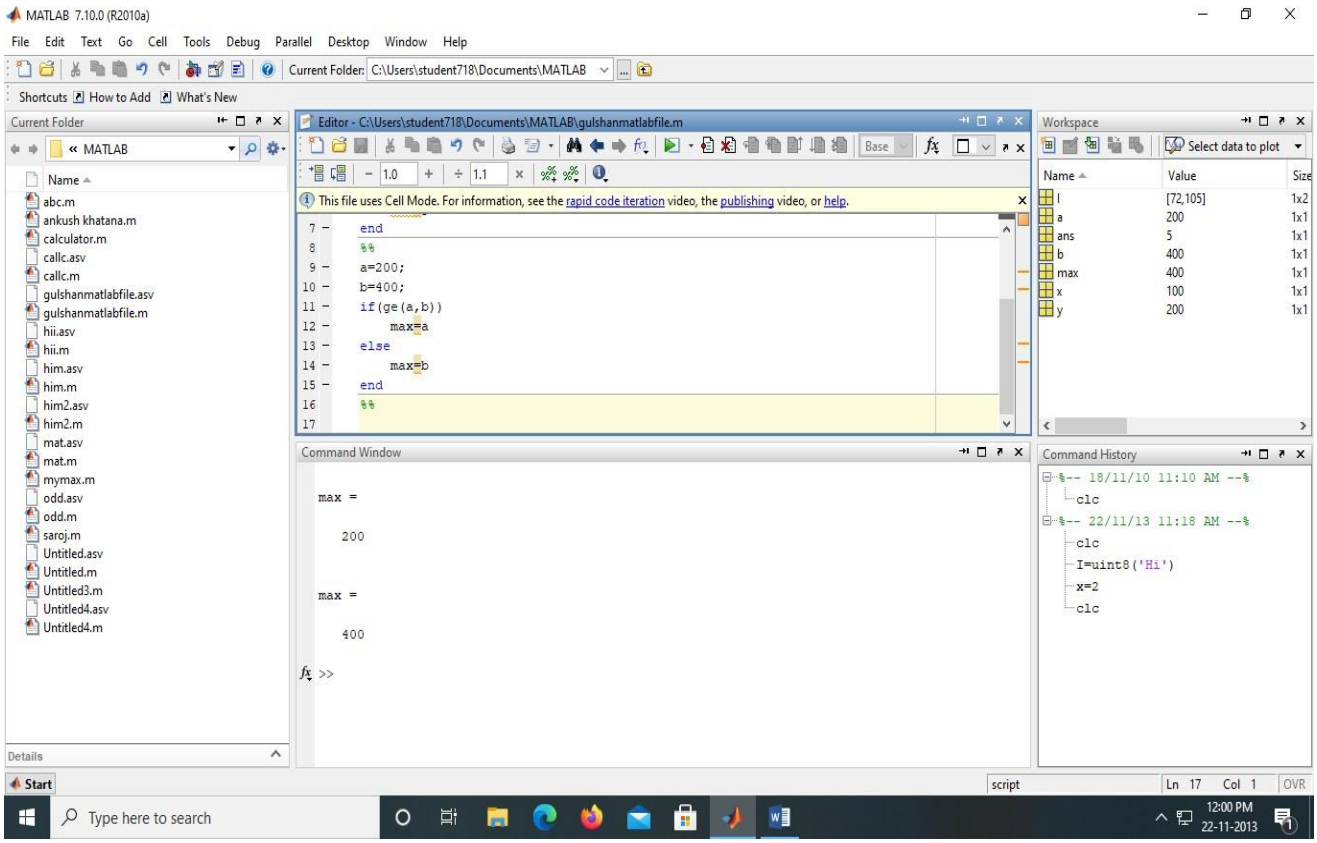
OUTPUT:



Relational and logical Operators

S.No.	Operators	Description
1	>	Greater than to
2	<	Less than to
3	>=	Greater than and equal to
4	<=	Less than and equal to
5	%	Percentage
6	==	Equal to
7	~=	Not Equal to
8	~	Logical “NOT”
9	&	Logical “AND”
10	|	Logical “OR”





Bitwise operators

S.No.	Operators	Description
1	Bitand(a,b)	Bit-wise AND
2	Bitor(a,b)	Bit-wise OR
3	Bitxor(a,b)	Bit-wise XOR
4	Bitcmp(a)	Bit-wise complement
5	Bitget(a,pos)	Get bit at specified position
6	Bitset(a,pos)	Set bit at specified location
7	Bitshift(a,k)	Shift bits specified number of places
8	Swapbytes	Swap byte ordering

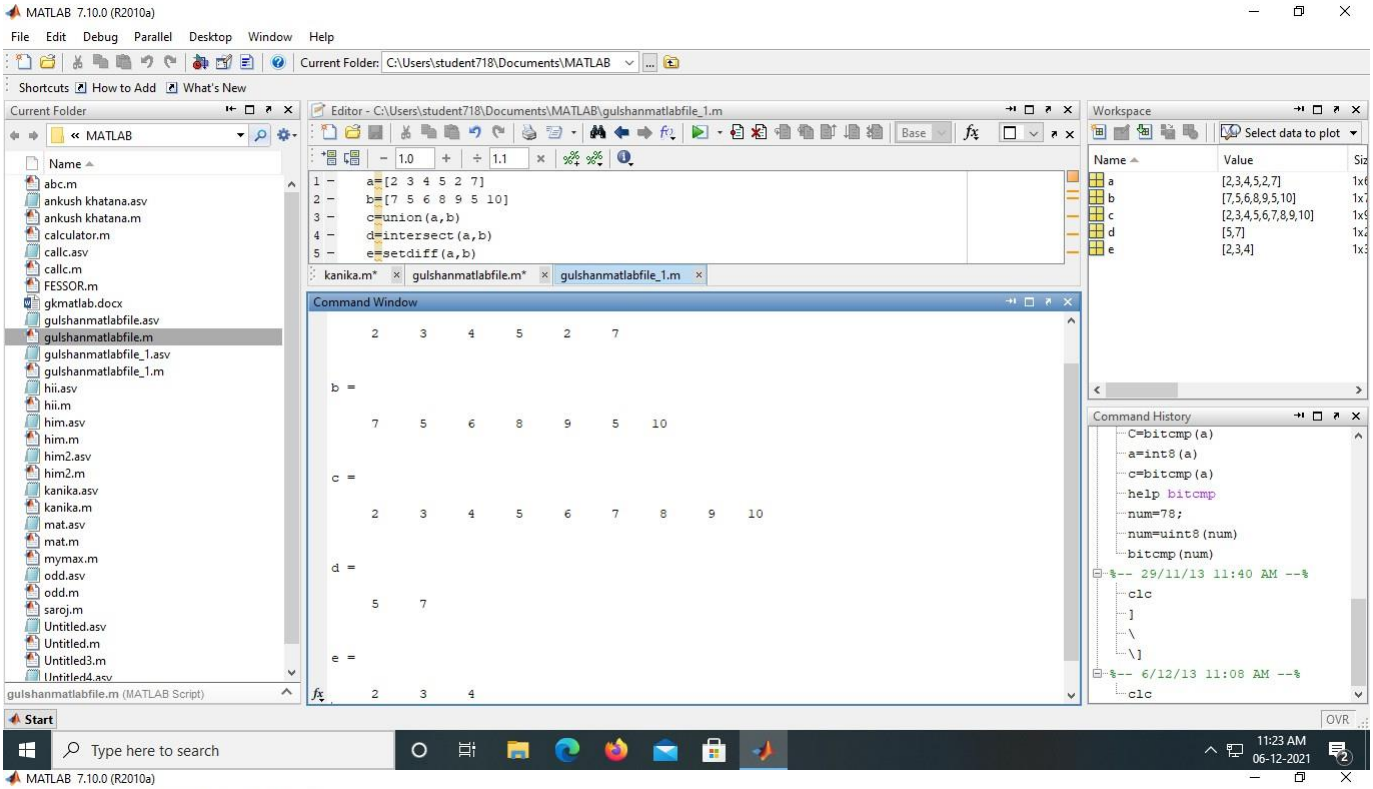
Set Operations

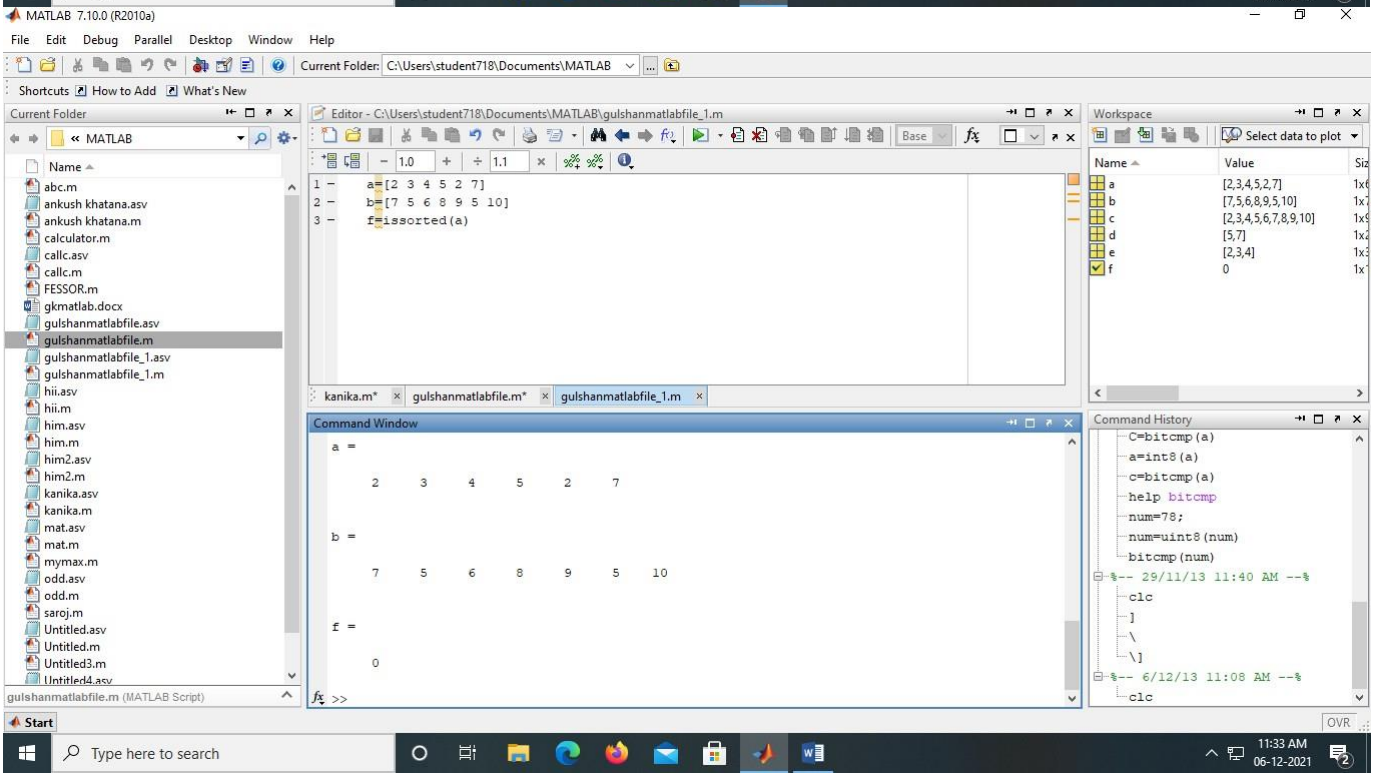
- Union (Sets of union of two arrays)
- Intersection (Unique values in array)
- Testing for set membership

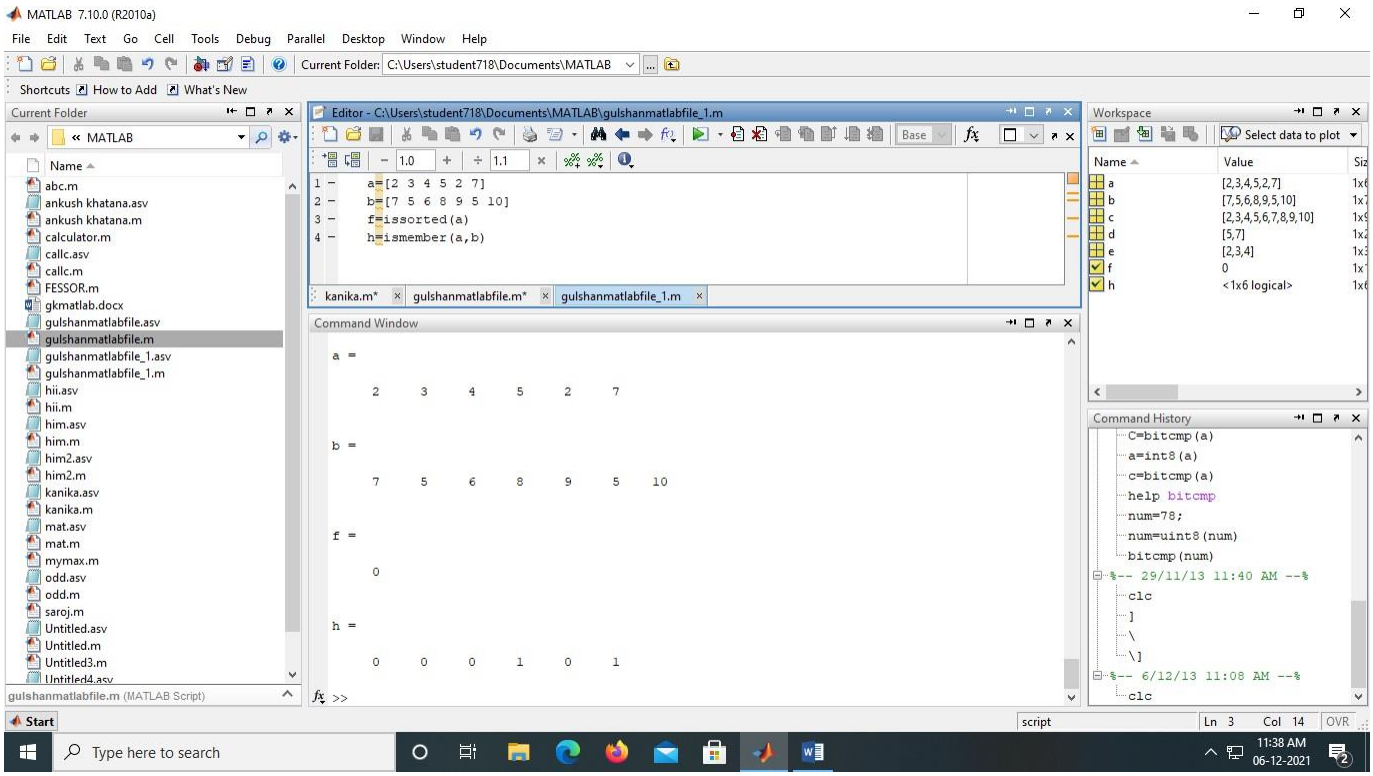
Sr.no.	Functions	Desciption
1	**Intersect(a,b)**	Set intersection of two arrays, Returns the common values to both Aand B in sorted order
2	**Intersect(a,b,'rows')**	Treats each row of A ,each row of B assingle entities, Returns the rows common to both Aand B in sorted order
3	**Ismember(a,b)**	Returns an array of same size as a, containing 1(true) where elements of aare found in b. elsewhere, it returns 0 (false)
4	**Ismember(a,b,'rows')**	Treats each row of a and each row of bas single entities and returns a vector containing 1(true) where the rows of matrix A are also rows of B. elsewhere,it returns 0 (false)
5	**Issorted(a)**	Returns logical 1(true) if element of a are in sorted order and logical 0(false)otherwise. Input a can be a vector or an N by 1 or 1 by N cell array of strings. A is considered to be sorted ifA and the output of sort(a) are equal.
6	**Issorted(a,'rows')**	Returns logical 1(true) if the rows of two-dimensional matrix a are in sorted order and logical 0(false) otherwise, matrix A is considered to be sorted if Aand output of sortrows(a) are equal.
7	**Setdiff(a,b)**	Sets difference of two arrays, Returns the values in A that are not in B. The Values in the returned array arein sorted order.
8	**Setdiff(a,b,'rows')**	Treats each row of A and each row of B as single entities, Returns the row from A that are not in B. The rows of the returned matrix arein sorted order.
The 'rows' option does not support cell arrays		

Functions

Sr. No.	Function	Desciption
1	Intersect(A,B)	Set intersection of twoarrays
2	Intersect(A,B,'rows')	Treat each row of A andeach row of B
3	Ismember(A,B)	Returns an array the same size as A, containing 1(true)where elements of A are found in B elsewhere, it returns 0(false)
4	Ismember(A,B,'rows')	Treats each rows of A andeach row of B as single entities
5	Issorted(A)	Returns logical 1(true) if theelements of A are in sorted order and logical 0(flase) otherwise input A can be a vector or an N-by-1 or 1-by-N cell array of strings. **A is considered to be sorted ifA** and output of sort(A) are equal
6	Issorted(A,'rows')	Returns logical 1(true) if therows of two dimensional matrix A is in sorted order, and logical 0(flase) otherwise . **Matrix A is considered to be sorted if**A and the output of sortrows(A) are equal.
7	Setdiff(A,B)	Sets difference of two arrays
8	Setdiff(A,B,'rows')	Treats each row of A and each row of B as single entities and returns the rowsfrom A that are not in B. the rows of the returned matrix are in sorted order **The 'rows' option does notsupport cell arrays**
9	Setxor	Sets exclusive OR of twoarrays
10	Union	Set union of two arrays
11	Unique	Unique values in array


MATLAB 7.10.0 (R2010a)
File Edit Debug Parallel Desktop Window Help
Current Folder: C:\Users\student718\Documents\MATLAB
Shortcuts How to Add What's New
Editor - C:\Users\student718\Documents\MATLAB\gulshanmatlabfile_1.m
1 - a=[2 3 4 5 2 7]
2 - b=[7 5 6 8 9 5 10]
3 - c=union(a,b)
4 - d=intersect(a,b)
5 - e=setdiff(a,b)
kanika.m* gulshanmatlabfile.m* gulshanmatlabfile_1.m
Command Window
a = 2 3 4 5 2 7
b = 7 5 6 8 9 5 10
c = 2 3 4 5 6 7 8 9 10
d = 5 7
e = 2 3 4
Workspace
Select data to plot
Name Value
a [2,3,4,5,2,7]
b [7,5,6,8,9,5,10]
c [2,3,4,5,6,7,8,9,10]
d [5,7]
e [2,3,4]
Command History
C=bitcmp(a)
a=int8(a)
c=bitcmp(a)
help bitcmp
num=78;
num=uint8(num)
bitcmp(num)
%-- 29/11/13 11:40 AM --%
clc
]
\
\]
%-- 6/12/13 11:08 AM --%
clc
gulshanmatlabfile.m (MATLAB Script)
Start
OVR
Type here to search
11:23 AM
06-12-2021


MATLAB 7.10.0 (R2010a)
File Edit Debug Parallel Desktop Window Help
Current Folder: C:\Users\student718\Documents\MATLAB
Shortcuts How to Add What's New
Editor - C:\Users\student718\Documents\MATLAB\gulshanmatlabfile_1.m
1 - a=[2 3 4 5 2 7]
2 - b=[7 5 6 8 9 5 10]
3 - f=issorted(a)
kanika.m* gulshanmatlabfile.m* gulshanmatlabfile_1.m
Command Window
a = 2 3 4 5 2 7
b = 7 5 6 8 9 5 10
f = 0
>>
Workspace
Select data to plot
Name Value
a [2,3,4,5,2,7]
b [7,5,6,8,9,5,10]
c [2,3,4,5,6,7,8,9,10]
d [5,7]
e [2,3,4]
f 0
Command History
C=bitcmp(a)
a=int8(a)
c=bitcmp(a)
help bitcmp
num=78;
num=uint8(num)
bitcmp(num)
%-- 29/11/13 11:40 AM --%
clc
]
\
\]
%-- 6/12/13 11:08 AM --%
clc
gulshanmatlabfile.m (MATLAB Script)
Start
OVR
Type here to search
11:33 AM
06-12-2021


MATLAB 7.10.0 (R2010a)
File Edit Text Go Cell Tools Debug Parallel Desktop Window Help
Current Folder: C:\Users\student718\Documents\MATLAB
Shortcuts How to Add What's New
Current Folder
« MATLAB
Name
abc.m
ankush khatana.asv
ankush khatana.m
calculator.m
callc.asv
callc.m
FESSOR.m
gkmatlab.docx
gulshanmatlabfile.asv
gulshanmatlabfile.m
gulshanmatlabfile_1.asv
gulshanmatlabfile_1.m
hii.asv
hii.m
him.asv
him.m
him2.asv
him2.m
kanika.asv
kanika.m
mat.asv
mat.m
mymax.m
odd.asv
odd.m
saroj.m
Untitled.asv
Untitled.m
Untitled3.m
Untitled4.asv
gulshanmatlabfile.m (MATLAB Script)
Editor - C:\Users\student718\Documents\MATLAB\gulshanmatlabfile_1.m
1 - a=[2 3 4 5 2 7]
2 - b=[7 5 6 8 9 5 10]
3 - f=issorted(a)
4 - h=ismember(a,b)
kanika.m*
gulshanmatlabfile.m*
gulshanmatlabfile_1.m
Command Window
a =
2 3 4 5 2 7
b =
7 5 6 8 9 5 10
f =
0
h =
0 0 0 1 0 1
>>
Workspace
Select data to plot
Name
Value
a [2,3,4,5,2,7]
b [7,5,6,8,9,5,10]
c [2,3,4,5,6,7,8,9,10]
d [5,7]
e [2,3,4]
f 0
h <1x6 logical>
Command History
C=bitcmp(a)
a=int8(a)
c=bitcmp(a)
help bitcmp
num=78;
num=uint8(num)
bitcmp(num)
%-- 29/11/13 11:40 AM --%
clc
]
\
\]
%-- 6/12/13 11:08 AM --%
clc
Start
script
Ln 3 Col 14 OVR
Type here to search
11:38 AM
06-12-2021

MODULE-4

Aim: Matrix manipulations using MATLAB

1. Write a Program to create and print the following matrices:

a. 4X4 Matrix Identity Matrix

Sol:

A = eye(4,4);

Output:

```
Diagonal Matrix

   1   0   0   0
   0   1   0   0
   0   0   1   0
   0   0   0   1
```

b. 3X2 Matrix with all elements as zeros

Sol:

B= zeros(3,2);

```
B =

   0   0
   0   0
   0   0
```

c. 3X2 Matrix with all elements as ones

Sol:

C = Ones(3,2);

```
Command Window
>> b

b =

     0     0
     0     0
     0     0

>> c

c =

     1     1
     1     1
     1     1

fx >>
```

2. Write a Program to create and print the following matrices:

a. 5X4 matrix with random element upto 100 (say matrix X)

sol:

```
X=randi([0 100],5,4);
```

Output

```
X =

   70   89   60   73
   31   90   14    7
    9   71   60   86
   57   16   18   28
   81   75    8    1
```

b. 5X4 matrix with random element upto 100 (say matrix Y)

sol:

```
Y=randi([100 200],5,4);
```

Output:

```
Y =

  175   177   118   137
  169   131   171   186
  111   146   185   162
  120   166   163   117
  109   158   135   133
```

c. Compute matrix Z=X+Y

sol:

```
Z=X+Y;
```

Output

```
Z =

  165   158   187   140
  145   177   182   264
  136   268   220   128
  148   247   268   201
  177   231   174   110
```

d. Transpose matrix A into Q

sol:

Q=X';

Output

```
Q =

    22    83     7   100    13
    73    47    25    81    65
    39    48    80     2    20
    51    46    92    37    91
```

e. compute D=Q*Y;

OUTPUT:

```
D =

   30221   33056   35402   31450
   16940   18244   19034   15236
   29108   31076   33089   32153
   45237   49328   53007   47030
```

3. Generate a random matrix L of dimension 5X6:

a. Write code to display all the rows one by one.

Sol:

L=randi([0 50],5,6);

row1=L(1,:);

row2=L(2,:);

row3=L(3,:);

row4=L(4,:);

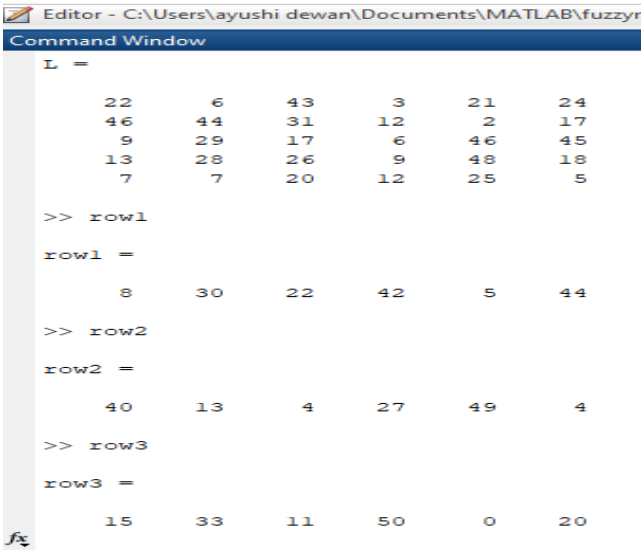
row5=L(5,:);

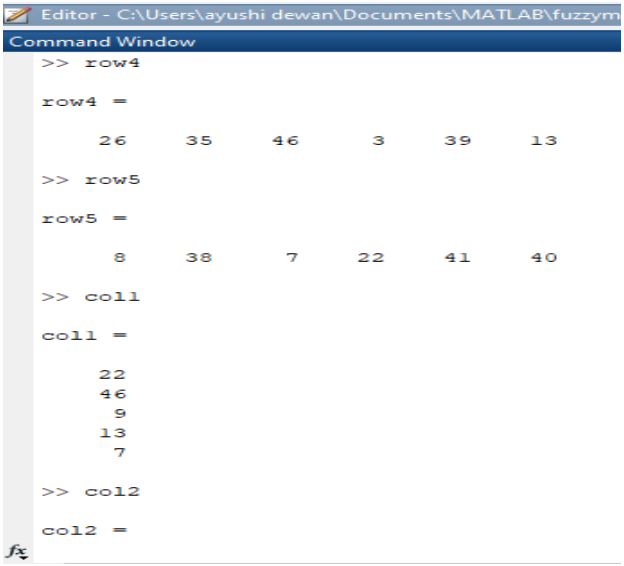
b. Write code to display all the columns one by one.

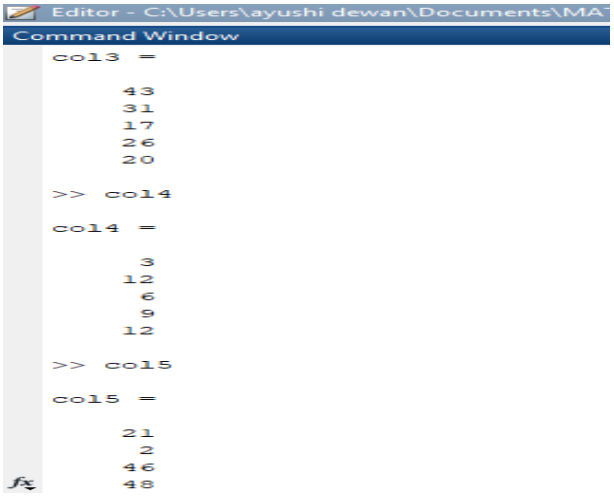
Sol:

```
L=randi([0 50],5,6);
col1=L(:,1);
col2=L(:,2);
col3=L(:,3);
col4=L(:,4);
col5=L(:,5);
```

Output:







c. Extract sub matrices from row 2-4 and column 3-6 from random matrix A of 5X6.

Sol:

A=randi([0 50],5,6);

Output:

```
A =

   29   36   10   49   50    5
    5   46   21    7   39   46
   45    5   40   11    9   13
    6   16   46   19   22   21
   46    8   36    4   29   38
```

B=A([2:4],[3:6])

Output

```
B =

   21    7   39   46
   40   11    9   13
```

4. Generate a 7X5 matrix with all its elements as the sum of row and column. For example, a_{53}=8 and a_{41}=5. Check if this matrix is symmetric.

Sol:

disp Q4;

n = 7 ;

x = 1:n ;

row = repmat(x',1,n) ;

col = repmat(x,n,1) ;

sumij = row+col;

disp (sumij);

OUTPUT

```
     2     3     4     5     6     7     8
     3     4     5     6     7     8     9
     4     5     6     7     8     9    10
     5     6     7     8     9    10    11
     6     7     8     9    10    11    12
     7     8     9    10    11    12    13
     8     9    10    11    12    13    14
```

5. Write a program to read a year and to check if it is leap Year or not.

Sol:

function year

year = input (' Enter the year : ')

if mod(year,4) == 0 disp ('true')

elseif mod(year,100) == 0 disp ('false')

elseif mod(year,400) == 0 disp ('true')

else disp ('false')

end

Output:

```
>> leapyear
 Enter the year :
2022

year =

        2022

false
>> leapyear
 Enter the year :
2024

year =

        2024

true
>>
```

MODULE-5

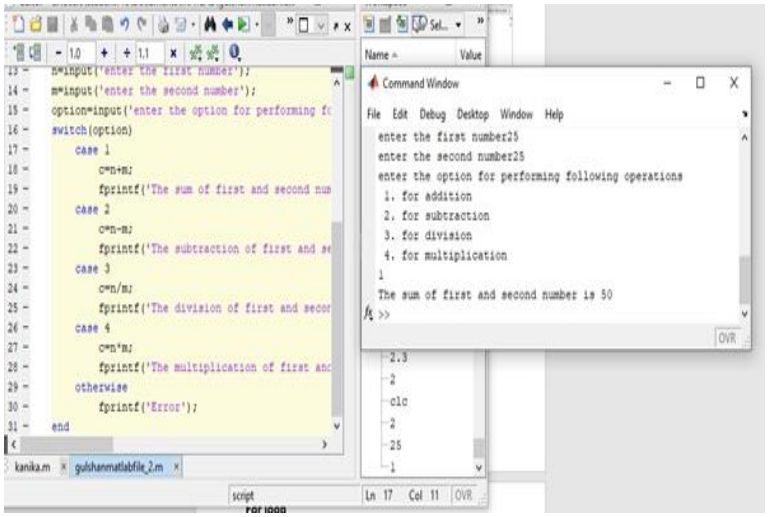
Aim: Programming in MATLAB

Flow control

- IF
- SWITCH
- FOR
- WHILE
- BREAK

IF:- It is a type of statement which is used in conditional programs in all the MATLAB as well as cprogramming language

SWITCH



FOR LOOP

```
1 -   n=input('enter the number');
2 -  for i=1:10;
3 -      fprintf('%d*%d=%d\n',n,i,n*i);
4 -  end
5     %%
6 -   n=input('enter the number');
7 -   i=1;
8 -  while (i<=10)
9 -      fprintf('%d*%d=%d\n',n,i,n*i);
10 -     i=i+1;
11 -  end
```

kanika.m | gulshanmatlabfile_2.m

Command Window

```
2*8=16
2*9=18
2*10=20
enter the number2
2*1=2
2*2=4
2*3=6
2*4=8
2*5=10
2*6=12
2*7=14
2*8=16
2*9=18
2*10=20
fx >>
```

- ## WHILE LOOP


```
1 -   n=input('enter the number');
2 -  for i=1:10;
3 -      fprintf('%d*%d=%d\n',n,i,n*i);
4 -  end
5     %%
6 -   n=input('enter the number');
7 -   i=1;
8 -  while (i<=10)
9 -      fprintf('%d*%d=%d\n',n,i,n*i);
10 -     i=i+1;
11 -  end
```

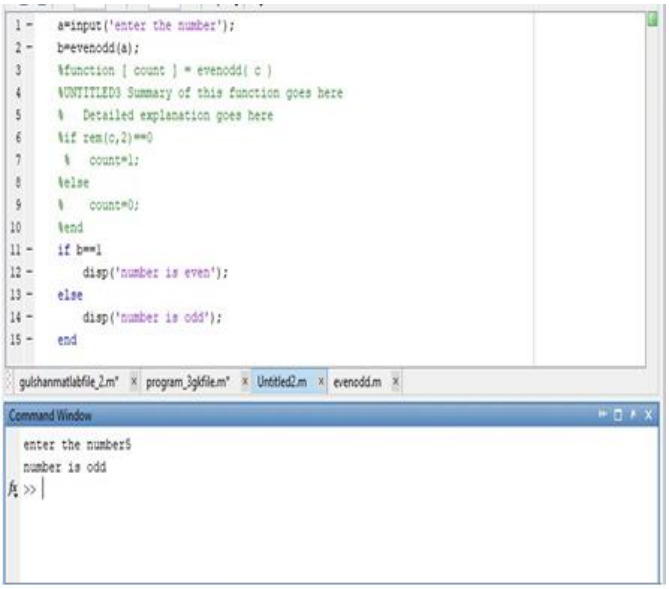
kanika.m | gulshanmatlabfile_2.m

Command Window

```
2*8=16
2*9=18
2*10=20
enter the number2
2*1=2
2*2=4
2*3=6
2*4=8
2*5=10
2*6=12
2*7=14
2*8=16
2*9=18
2*10=20
fx >>
```

- **BREAK:-** It terminates the loop statement and transfers execution to the statement immediately following the loop

- **CONTINUE:-** It causes the loop to skip the reminder of itsbody & immediately retest its condition prior to reinitiating.



MODULE-6

Aim: To plot Different Types of Graphs

Plot:- plot(x , y) creates a 2-D line plot of the data in y versusthe corresponding values in x.

- To plot a set of coordinates connected by line segments,specify x and y as vectors of the same length.
- To plot multiple sets of coordinates on the same set of axes,specify at least one of x or y as a matrix.

Methods for creating a graph:-

a) **Plot(x , y)-> plot(x , y)->** creates a 2-D line plot of the datain y versus the corresponding values in x

b) **Plot(x , y , LineSpec)->** creates the plot using the specifiedline style, marker and color.

c) **Plot(x1 , y1 , . . . , x_n , y_n)->** plots multiple pairs of x and ycoordinates on the same set of axes.

d) **Plot(x1 , y1 , lineSpec1 , . . . , x_n , y_n , LineSpec)->** assignsspecific line styles, markers and colors to each x-y pair. You can specify LineSpec for some x-y pairs and omit it for others.

e) **Plot(y)->** plots y against an implicit set of x-coordinates

- If y is a vector, x-coordinates range from 1 to length(y).
- If y is a matrix, plot contains one line for each column in y.The x-coordinates range from 1 to the number of rows in y.

f) **Plot(y , LineSpec)->** specific line style, marker and color.

g) **Plot(_ , Name , Value)->** specific line properties using one ormore name-value arguments. The properties apply to all the plotted lines. Specify the name-value arguments after all the arguments in any of the previous syntaxes.

h) **Plot(ax , _)->** displays the plot in the target axes. Specify theaxes as the first argument in any of the previous syntaxes.

i) **P = plot(_)->** returns a line object or an array of Line objects.Use p to modify properties of the plot after creating it.

Specific line, color and markers:-

Lines

S. no.	Symbol	Style of line
1	-	Solid
2	- -	Dashed
3	:	Dotted
4	-.	Dash-Dotted

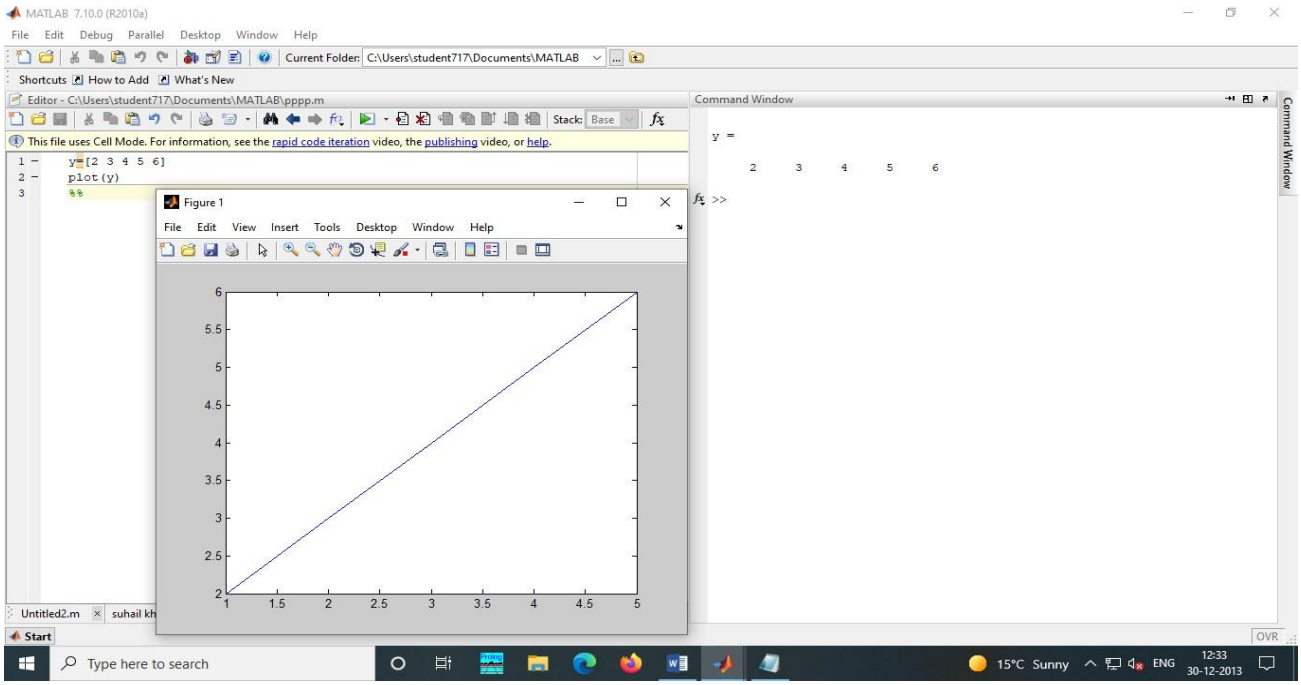
Colors

S. no.	Symbol	Colour
1	K	Black
2	R	Red
3	B	Blue
4	G	Green
5	C	Cyan
6	M	Magenta
7	Y	Yellow

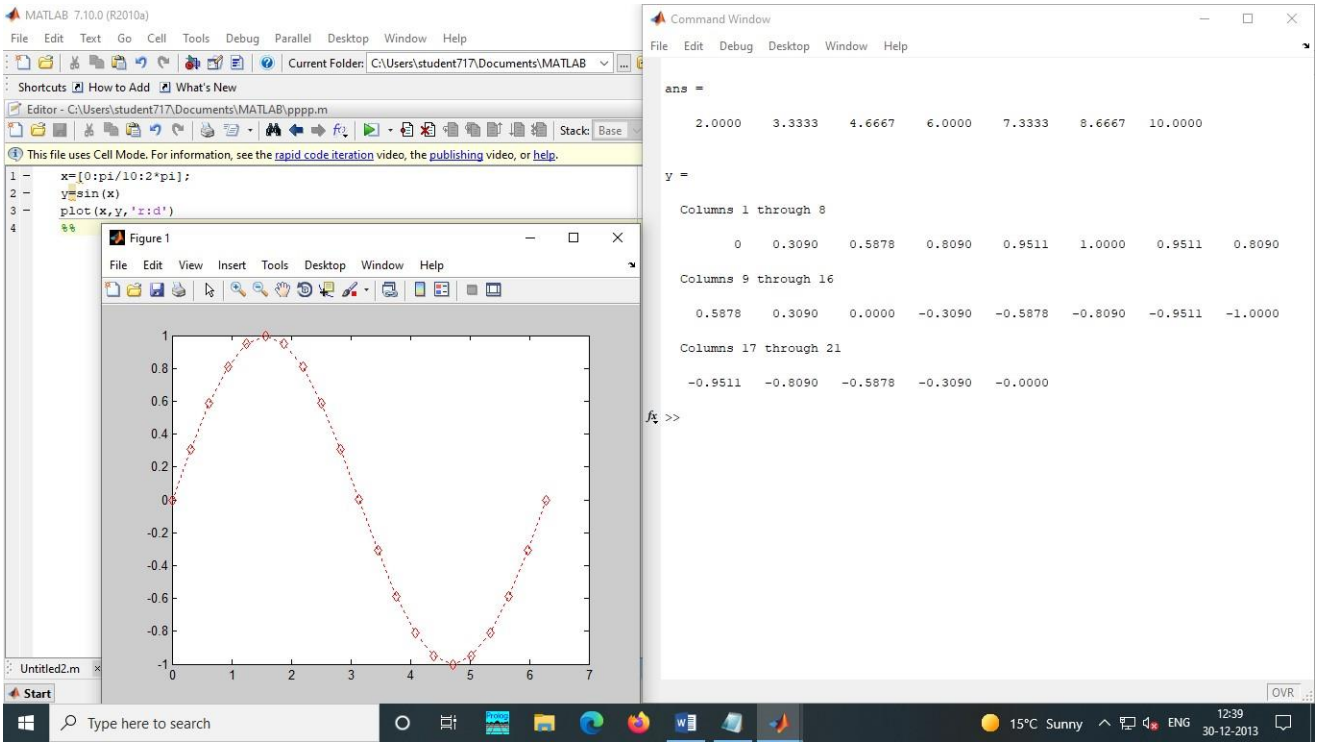
Markers

Sr. no.	Symbol	Marker
1	+	Plus sign
2	O	Circle
3	*	Astrick
4	.	Point
5	X	Cross
6	S	Square
7	D	Diamond
8	^	Upward/Pointing triangle
9	v	Downward
10	>	Right
11	<	Left
12	P	Five
13	H	Six

1. Plot(y)



2. Plot(x, y, LineSpec)

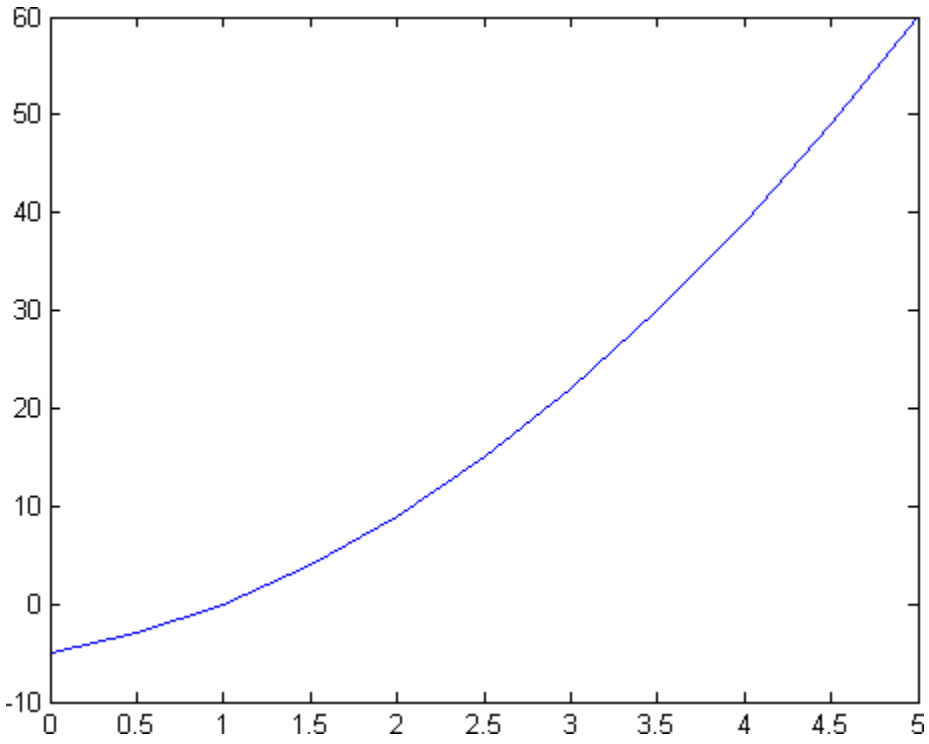


3. A quick example of plot command: Draw a curve

Sol:

a = [0:0.5:5];

b = 2*a.^2 + 3*a -5;plot(a,b)



Remarks:

(1) In "plot(a,b)", the array "a" should contain the data of the coordinate or "grid point)on the x-axis and "b" should be the corresponding values on the y-axis.

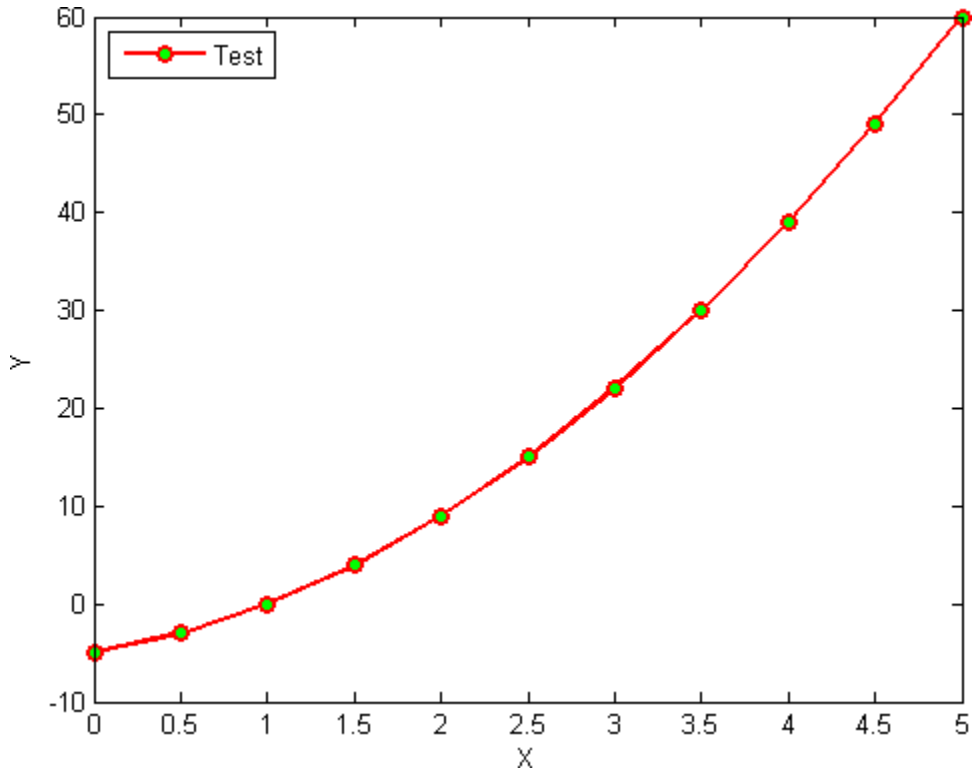
(2) After a plot is made, it can be further modified by using the interactive tool for graphics. For example, the labels of the x and y axes can be manually added to the plot.

(3) The plot can be saved in various formats (jpg, tif, eps, etc.).

4. Refine the plot: Line pattern, color, and thickness

Sol:

```
a = [0:0.5:5];
b = 2*a.^2 + 3*a -5;
plot(a,b,'-or','MarkerFaceColor','g','LineWidth',2)
xlabel('X'); ylabel('Y'); legend('Test','Location','NorthWest')
```

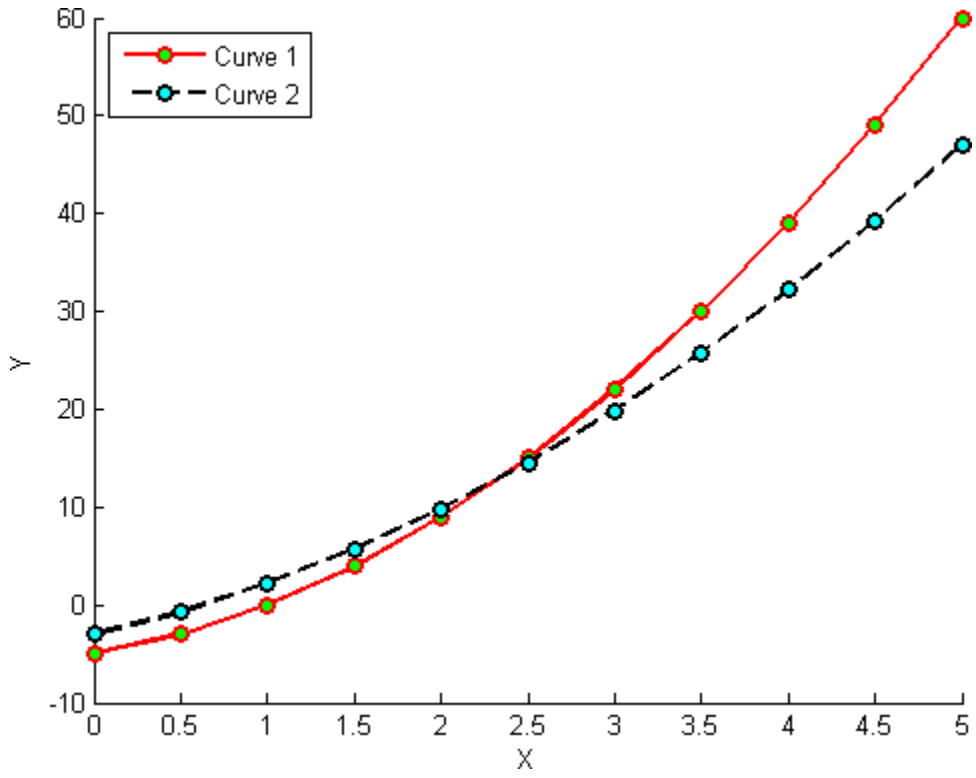


Remarks: The '-or' in the *plot* command set the line pattern. In this case, it's solid linewith circular symbol. The circular symbol is filled with green color ('g' for 'MarkerFaceColor'). The legend of the plot is set to locate at teh upper-left corner ('Location'' set to 'NorthWest') inside the frame.

5. Draw multiple curves

Sol:

```
a = [0:0.5:5];
b = 2*a.^2 + 3*a -5;
c= 1.2*a.^2+4*a-3;hold on
plot(a,b,'-or','MarkerFaceColor','g','LineWidth',2)
plot(a,c,'--ok','MarkerFaceColor','c','LineWidth',2)
xlabel('X'); ylabel('Y'); legend('Curve 1','Curve 2','Location','NorthWest')
```

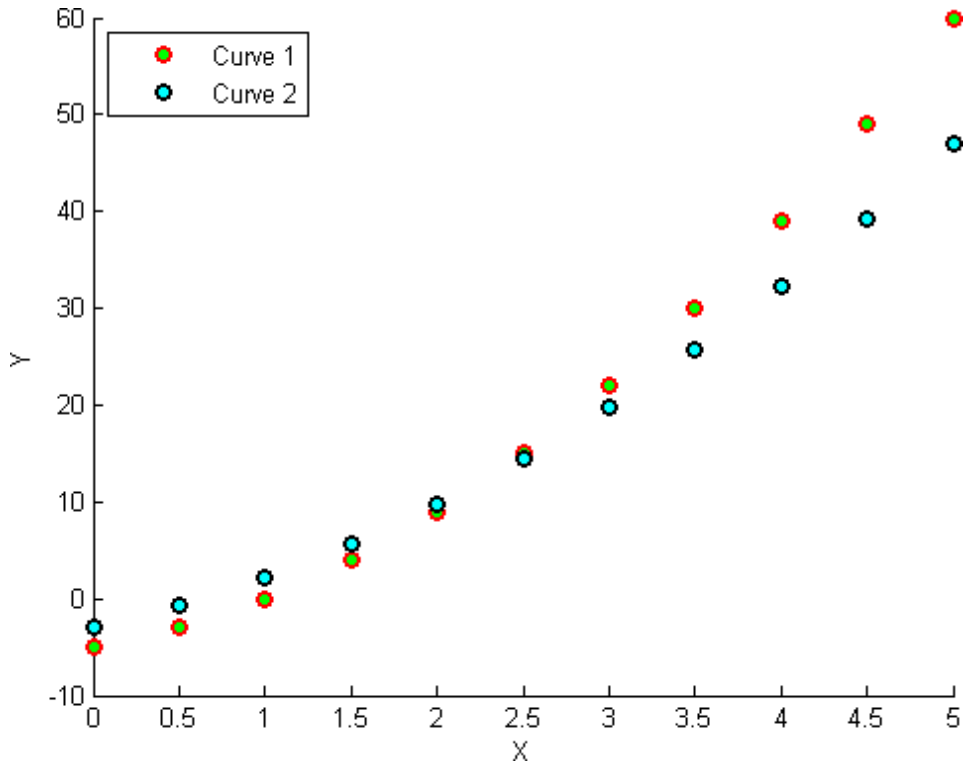


Remark: Without the "hold on" command, the second *plot* will override the first one andacts to erase the curve produced by the latter.

6. Draw symbols

Sol:

```
a = [0:0.5:5];
b = 2*a.^2 +3*a-5;
c=1.2*a.^2+4*a-3;
hold on
plot(a,b,'or','MarkerFaceColor','g','LineWidth',2)
plot(a,c,'ok','MarkerFaceColor','c','LineWidth',2)
xlabel('X'); ylabel('Y'); legend('Curve 1','Curve 2','Location','NorthWest')
```



7. Plot with multiple panels

Sol:

```
a = [0:0.5:5];
b = 2*a.^2 + 3*a -5;
c=1.2*a.^2+4*a-3;subplot(1,2,1)
plot(a,b,'-or','MarkerFaceColor','g','LineWidth',2)
xlabel('X'); ylabel('Y'); legend('Curve ','Location','NorthWest')subplot(1,2,2)
plot(a,c,'--ok','MarkerFaceColor','c','LineWidth',2)
xlabel('X'); ylabel('Y'); legend('Curve 2','Location','NorthWest')
```