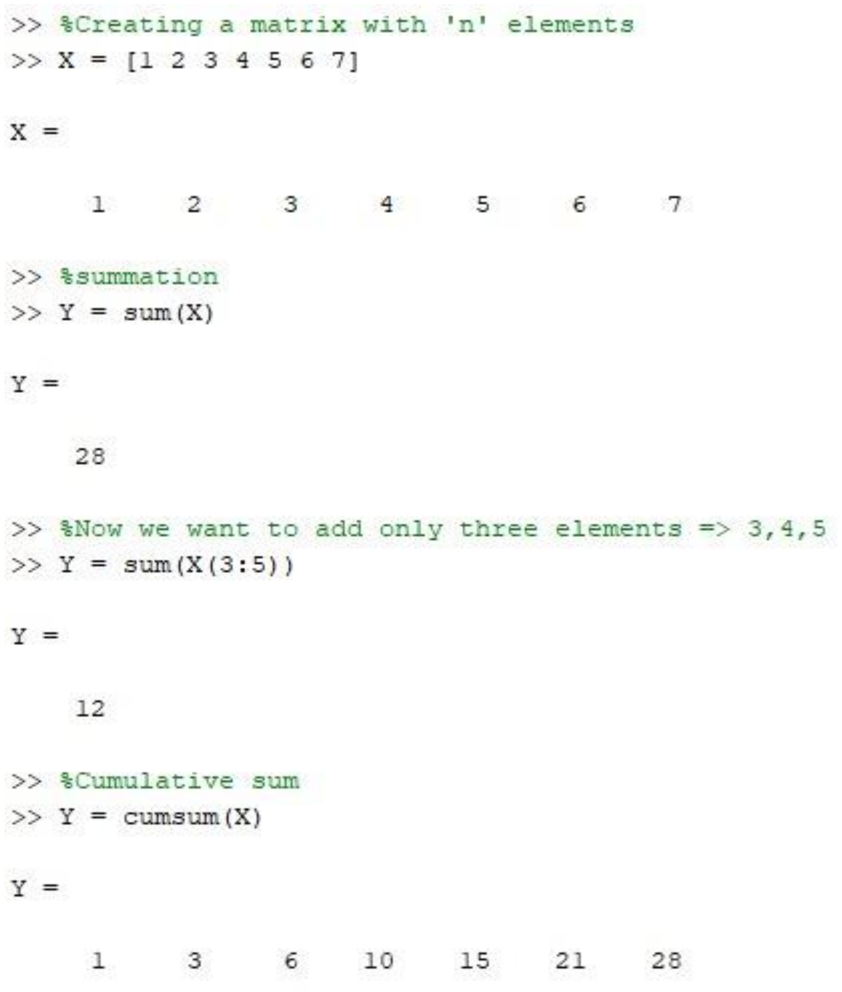
PRACTICAL – 3(A)

AIM: Generating a set of commands on a given vector to add up the values of the given elements. The size of the vector is 1xN (N = number of columns)

PROCEDURE:



```
>> %Creating a matrix with 'n' elements
>> X = [1 2 3 4 5 6 7]

X =

     1     2     3     4     5     6     7

>> %summation
>> Y = sum(X)

Y =

    28

>> %Now we want to add only three elements => 3,4,5
>> Y = sum(X(3:5))

Y =

    12

>> %Cumulative sum
>> Y = cumsum(X)

Y =

     1     3     6    10    15    21    28
```

DISCUSSION:

(i) **Summation -** For the summation of all matrix elements, we use the command Y = sum(X).
For e.g.: X = [1 2 3 4 5 6 7]
1+2+3+4+5+6+7 = 28

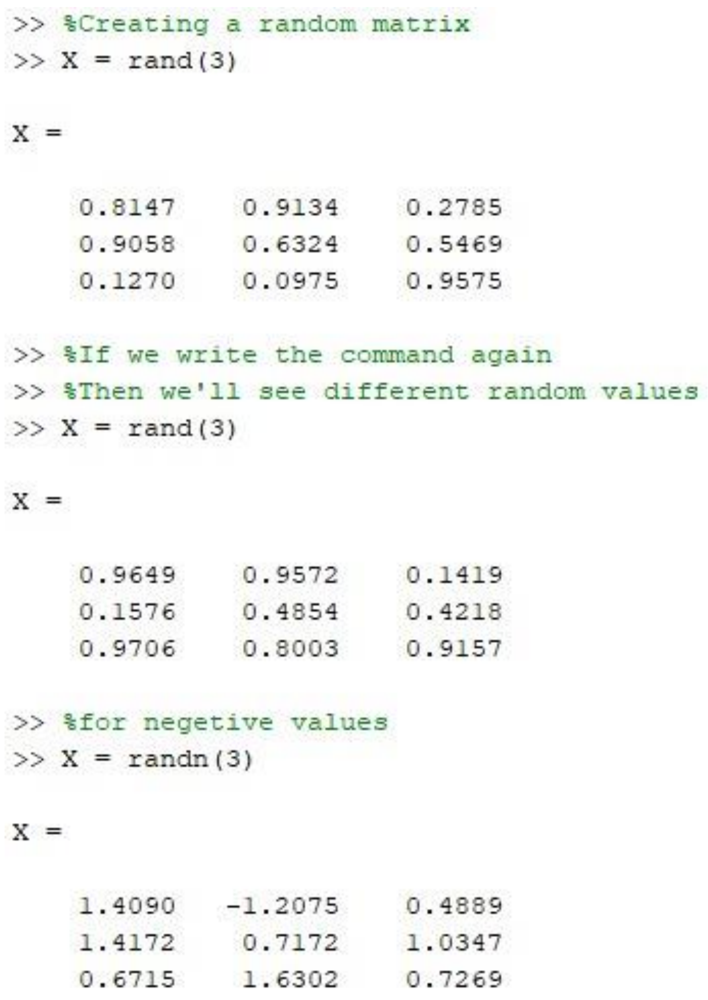
(ii) **Summation of specific elements of a matrix –** Here we use the command Y = sum(X(3:5)) => This command will add the elements from 3 to 5.

(iii) **Cumulative sum = >** A cumulative sum is a sequence of partial sums of a given sequence. For example, the cumulative sums of the sequence {1,2,3,4,…}are {1, 1+2, 1+2+3, 1+2+3+4,..}

PRACTICAL – 3(B)

AIM: Generating a random sequence using rand() function /randn() function and plotting them.

PROCEDURE:



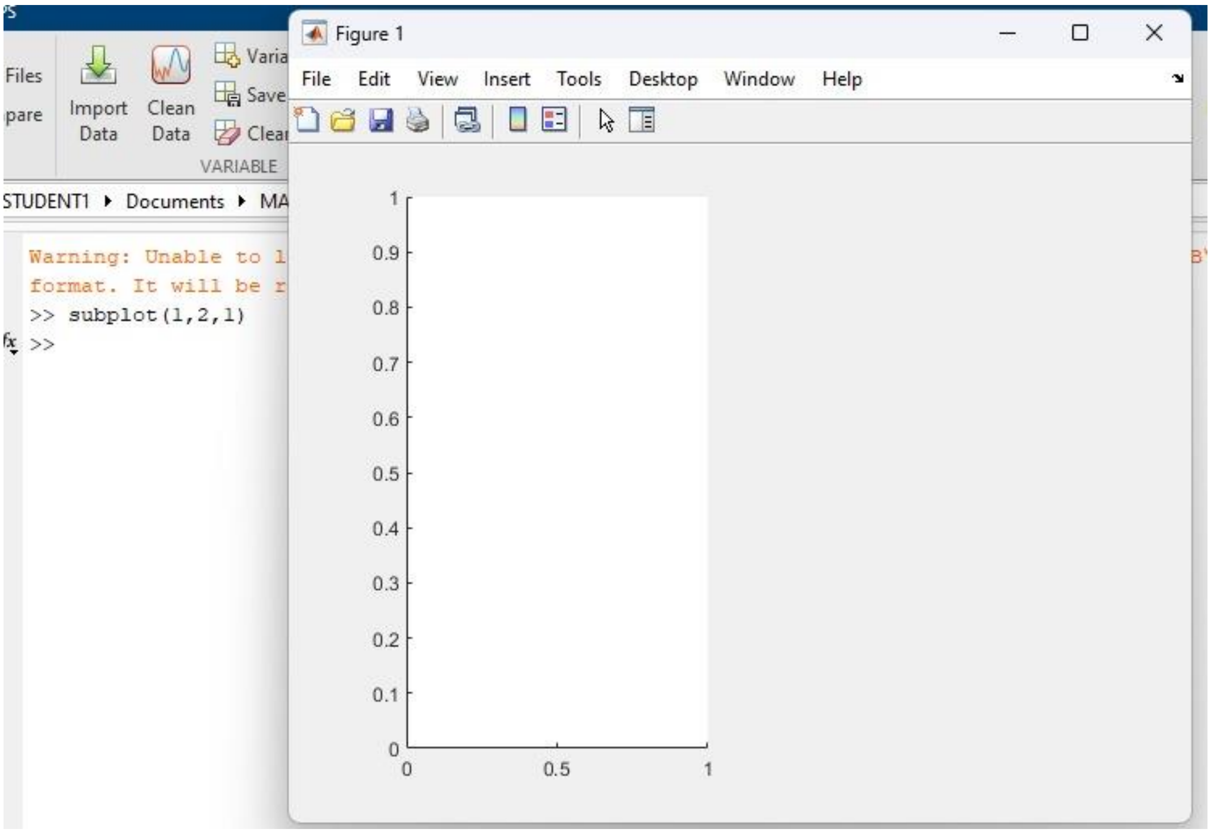
```
>> %Creating a random matrix
>> X = rand(3)

X =

    0.8147    0.9134    0.2785
    0.9058    0.6324    0.5469
    0.1270    0.0975    0.9575

>> %If we write the command again
>> %Then we'll see different random values
>> X = rand(3)

X =

    0.9649    0.9572    0.1419
    0.1576    0.4854    0.4218
    0.9706    0.8003    0.9157

>> %for negetive values
>> X = randn(3)

X =

    1.4090   -1.2075    0.4889
    1.4172    0.7172    1.0347
    0.6715    1.6302    0.7269
```

DISCUSSION:

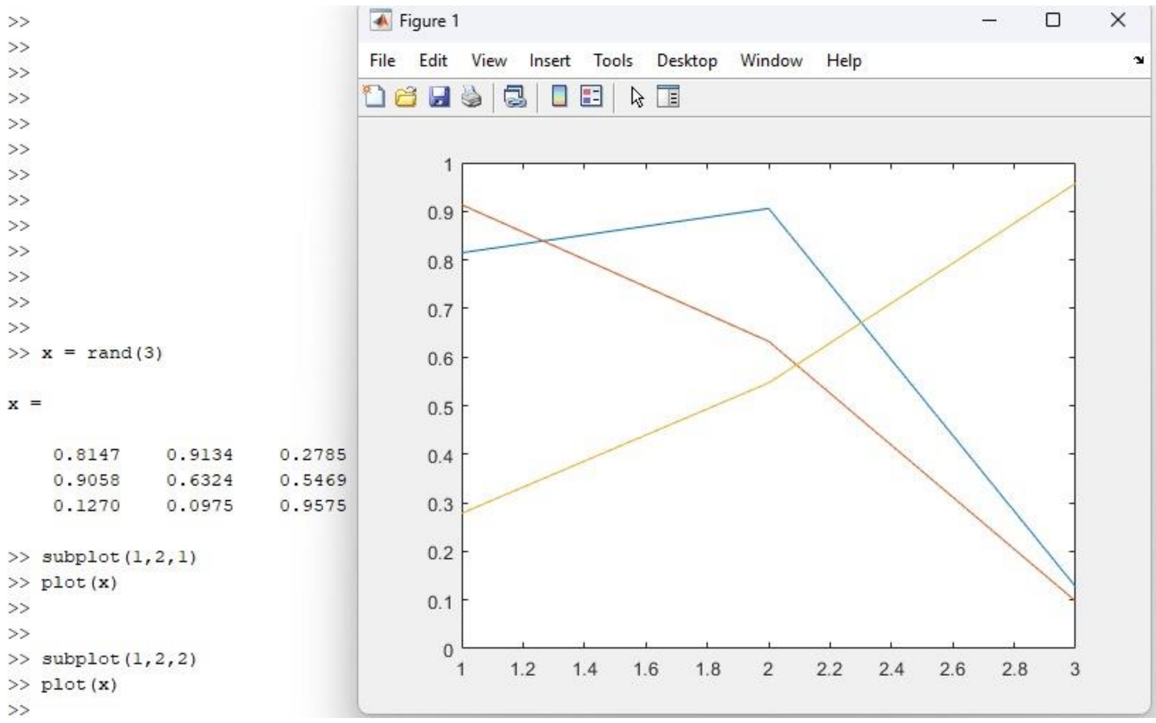
for 3x3 random matrix, we use the command X = randn(3) (their values will be between -3 to 3)

PLOTTING THESE VALUES:

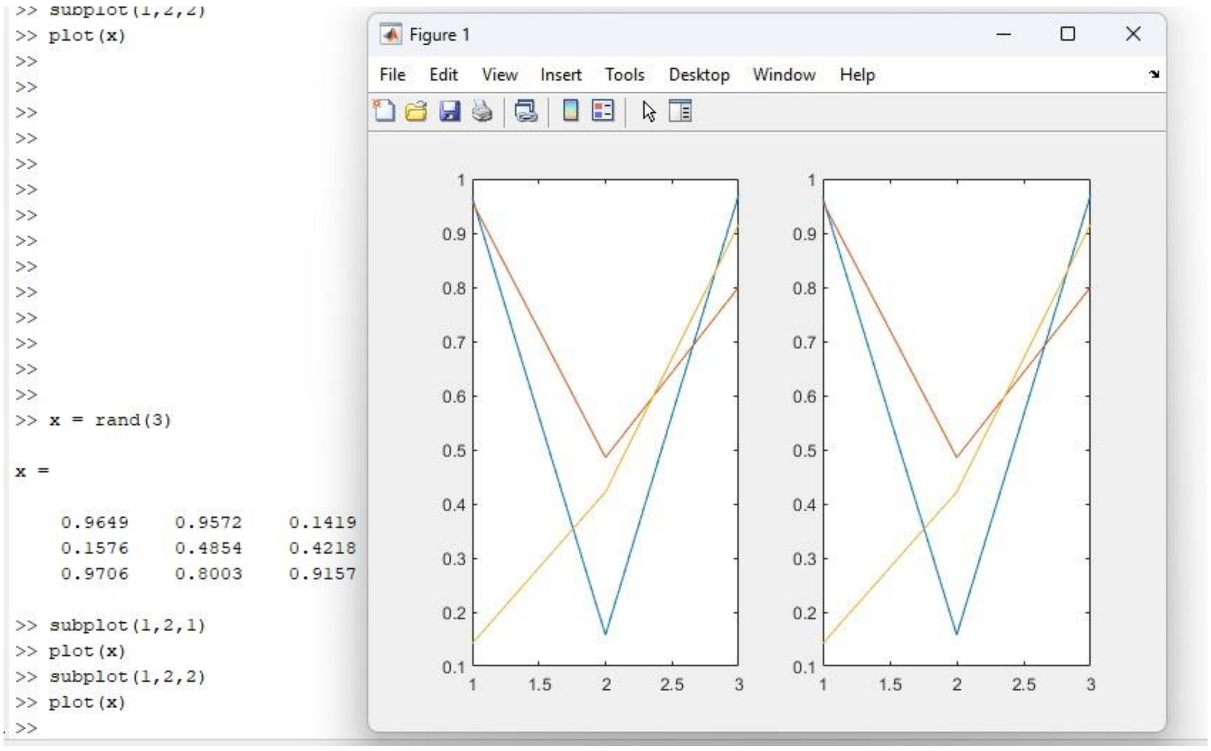
For plotting the above random matrix we use the command Subplot(1,2,1) and plot(x)



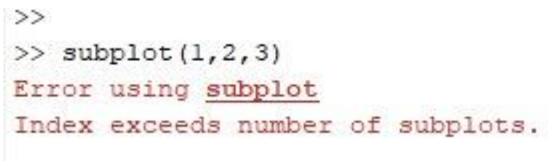
```
Warning: Unable to l
format. It will be r
>> subplot(1,2,1)
>>
```



```
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>> x = rand(3)

x =

    0.8147    0.9134    0.2785
    0.9058    0.6324    0.5469
    0.1270    0.0975    0.9575

>> subplot(1,2,1)
>> plot(x)
>>
>>
>> subplot(1,2,2)
>> plot(x)
>>
```



```
>> subplot(1,2,2)
>> plot(x)
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>> x = rand(3)

x =

    0.9649    0.9572    0.1419
    0.1576    0.4854    0.4218
    0.9706    0.8003    0.9157

>> subplot(1,2,1)
>> plot(x)
>> subplot(1,2,2)
>> plot(x)
>>
```



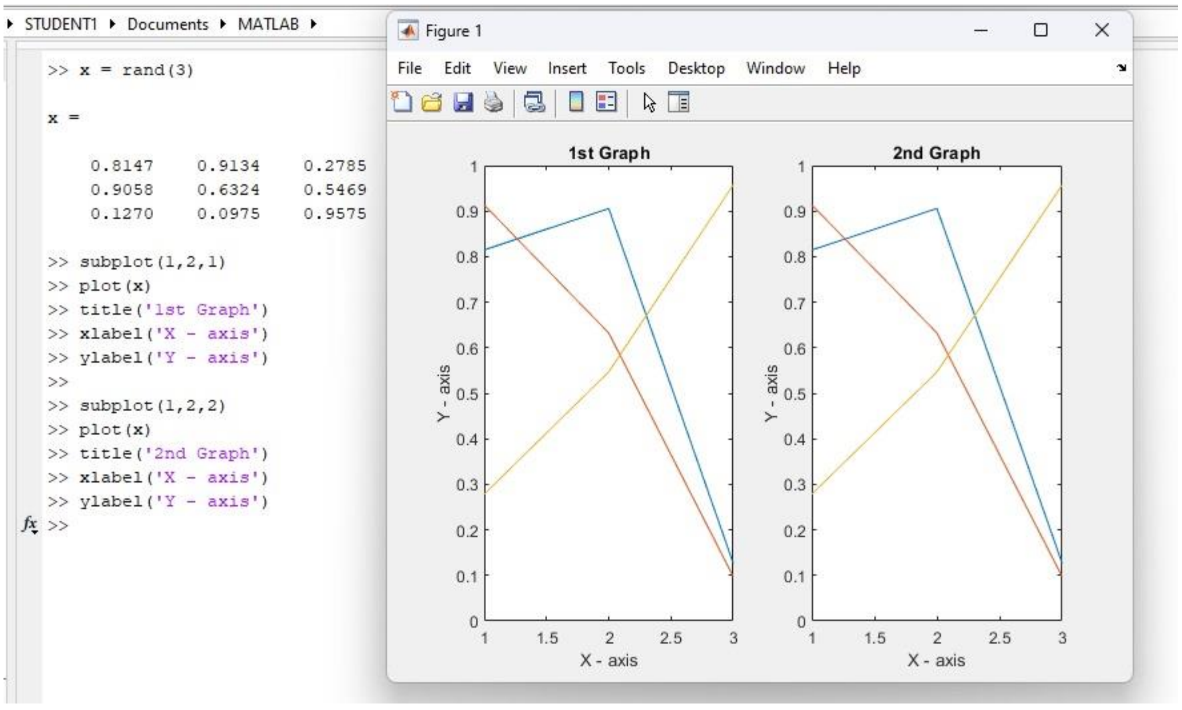
```
>>
>> subplot(1,2,3)
Error using subplot
Index exceeds number of subplots.
```

LABELING THE GRAPH

For labeling, the above-given graph, we use the command title(‘ 1st graph’)

X label (‘x-axis’)

Y label (‘y



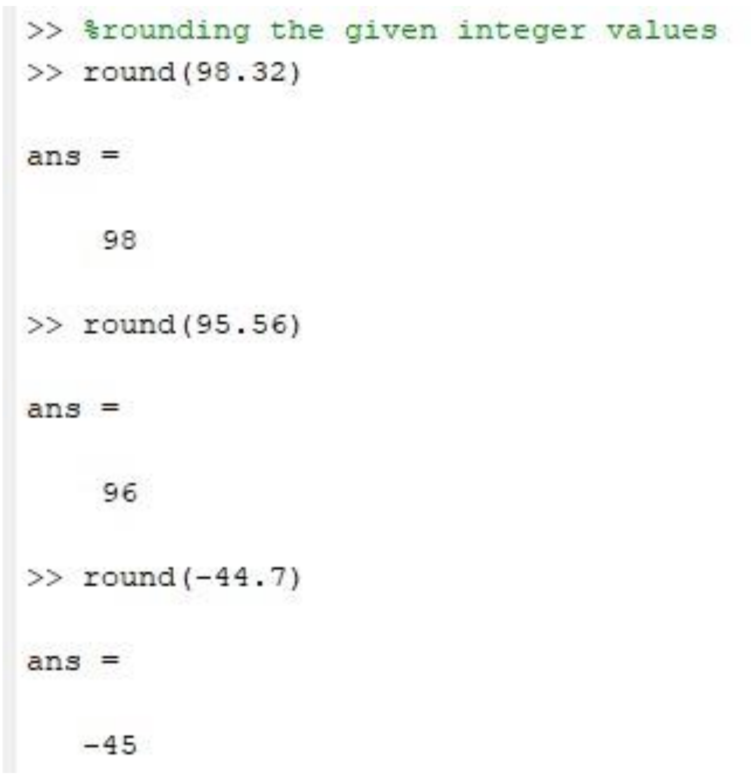
STUDENT1 ▸ Documents ▸ MATLAB ▸
Figure 1
File Edit View Insert Tools Desktop Window Help
1st Graph
2nd Graph
Y - axis
X - axis

```
>> x = rand(3)

x =

    0.8147    0.9134    0.2785
    0.9058    0.6324    0.5469
    0.1270    0.0975    0.9575

>> subplot(1,2,1)
>> plot(x)
>> title('1st Graph')
>> xlabel('X - axis')
>> ylabel('Y - axis')
>>
>> subplot(1,2,2)
>> plot(x)
>> title('2nd Graph')
>> xlabel('X - axis')
>> ylabel('Y - axis')
fx >>
```

PRACTICAL 4(A)

AIM: Evaluating a given expression and rounding it to the nearest integer value using round, floor, ceil, and fix commands.

PROCEDURE:



```
>> %rounding the given integer values
>> round(98.32)

ans =

    98

>> round(95.56)

ans =

    96

>> round(-44.7)

ans =

   -45
```

DISCUSSION:

We use the command round() for rounding the given value. **E.g.** if we have to round the value 95.44, then we will use the command round(95.44) and the result will be 95. If we have to round the value 95.54, then we will use the command round(95.44) and the result will be 96.

Floor: lower value

Ceil: upper value

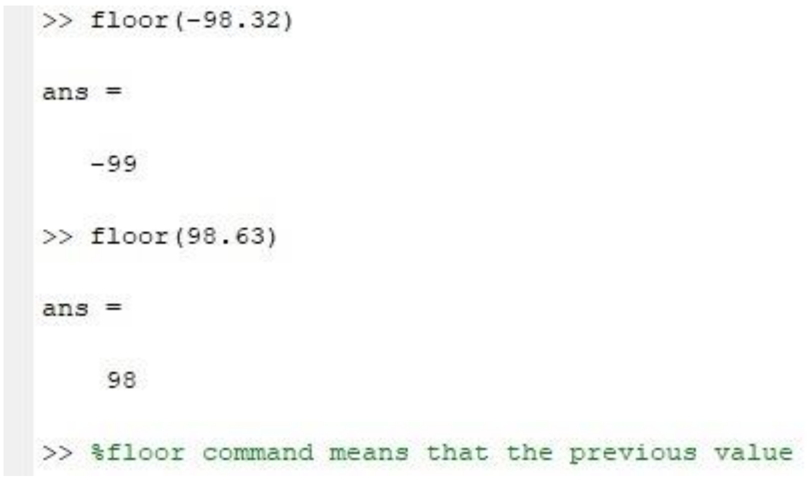
Floor Command: floor command will round the value to its lower value.

DISCUSSION:

B= floor(A) rounds the element of A to the nearest integer less than or equal to A. **E.g.** floor(98.63). the result will be 98.

floor(-98.63) will be -99

PROCEDURE:

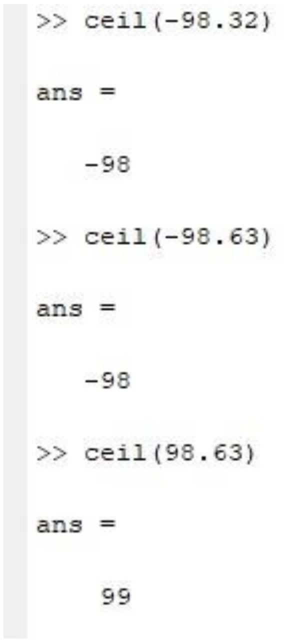


```
>> floor(-98.32)

ans =

   -99

>> floor(98.63)

ans =

    98

>> %floor command means that the previous value
```

Ceil command:

B= ceil(A) rounds the element of A to the nearest greater than or equal to A.

PROCEDURE:



```
>> ceil(-98.32)

ans =

   -98

>> ceil(-98.63)

ans =

   -98

>> ceil(98.63)

ans =

    99
```

Fix command:

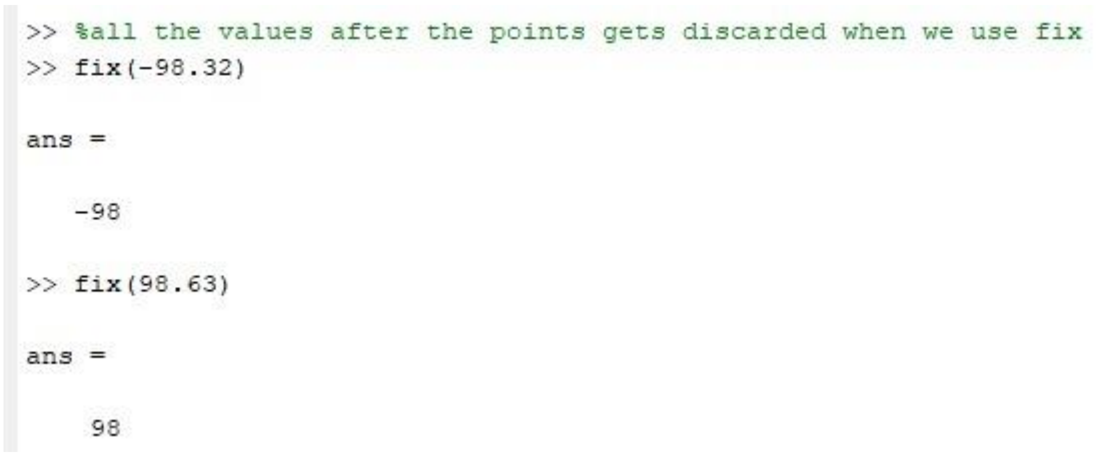
The fix command will discard all the values after the decimal.

DISCUSSION:

all the values after the decimal will be discarded if we use the command-

fix(). **E.g.** fix(98.44) will be 98.

PROCEDURE:



```
>> %all the values after the points gets discarded when we use fix
>> fix(-98.32)

ans =

   -98

>> fix(98.63)

ans =

    98
```

RESULT:

Matrices were created and different rounding functions (round, floor, fix, ceil) were applied to them and results were obtained.

CONCLUSION:

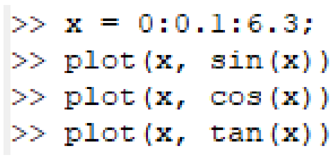
Various rounding functions were successfully implemented.

Experiment4(B)

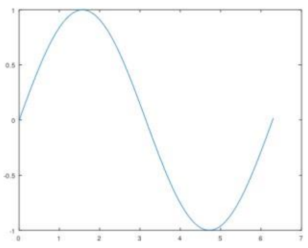
Aim: Togenerateaplotoftrigonometricfunctionssin(x),cos(x)andtan(x).

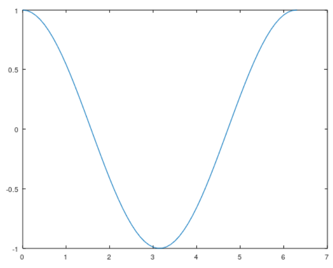
SoftwareUsed:MATLAB9.12.0(R2022a)

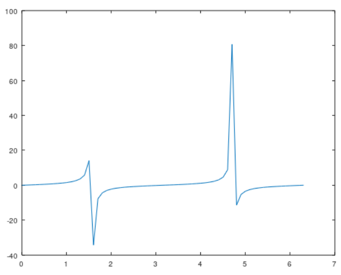
Procedure:



```
>> x = 0:0.1:6.3;
>> plot(x, sin(x))
>> plot(x, cos(x))
>> plot(x, tan(x))
```







Discussion:

To plot trigonometric functions in MATLAB, you can use the built-intrigonometric functions such as sine, cosine, and tangent. Here are the basicsteps:

1) Definetherangeofvaluesfortheindependentvariable,usuallyx.
2) Compute the values of the trigonometric function using the built-infunctionssuchassin,cos,ortan.Theinputtothesefunctionsisusuallytheve ctor ofxvalues.
3) Use the plot function to create a line plot of the function. The first inputto the plot function is the vector of x values, and the second input is thevectorofyvalues.Forexample,plot(x,cos(x)).

EXPERIMENT-5

AIM: a) Creating a vector Xn = ((-1)^n+1)/(2n-1) and adding up to 100 elements of the vector.

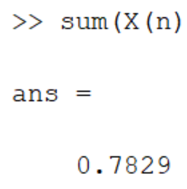
b) Plotting X, X^3, e^X, e^(x^2)

SOFTWARE USED: MATLAB(R2022b)

PROCEDURE:

```
>> n=1:100

n =

  Columns 1 through 14

     1     2     3     4     5     6     7     8     9    10    11    12    13    14

  Columns 15 through 28

    15    16    17    18    19    20    21    22    23    24    25    26    27    28

  Columns 29 through 42

    29    30    31    32    33    34    35    36    37    38    39    40    41    42

  Columns 43 through 56

    43    44    45    46    47    48    49    50    51    52    53    54    55    56

  Columns 57 through 70

    57    58    59    60    61    62    63    64    65    66    67    68    69    70

  Columns 71 through 84

    71    72    73    74    75    76    77    78    79    80    81    82    83    84

  Columns 85 through 98

    85    86    87    88    89    90    91    92    93    94    95    96    97    98

  Columns 99 through 100

    99   100

>> X(n)=((-1).^(n+1))./(2*n-1)

X =

  Columns 1 through 8

    1.0000   -0.3333    0.2000   -0.1429    0.1111   -0.0909    0.0769   -0.0667

  Columns 9 through 16

    0.0588   -0.0526    0.0476   -0.0435    0.0400   -0.0370    0.0345   -0.0323

  Columns 17 through 24

    0.0303   -0.0286    0.0270   -0.0256    0.0244   -0.0233    0.0222   -0.0213
  Columns 25 through 32

    0.0204   -0.0196    0.0189   -0.0182    0.0175   -0.0169    0.0164   -0.0159

  Columns 33 through 40

    0.0154   -0.0149    0.0145   -0.0141    0.0137   -0.0133    0.0130   -0.0127

  Columns 41 through 48

    0.0123   -0.0120    0.0118   -0.0115    0.0112   -0.0110    0.0108   -0.0105

  Columns 49 through 56

    0.0103   -0.0101    0.0099   -0.0097    0.0095   -0.0093    0.0092   -0.0090

  Columns 57 through 64

    0.0088   -0.0087    0.0085   -0.0084    0.0083   -0.0081    0.0080   -0.0079

  Columns 65 through 72

    0.0078   -0.0076    0.0075   -0.0074    0.0073   -0.0072    0.0071   -0.0070

  Columns 73 through 80

    0.0069   -0.0068    0.0067   -0.0066    0.0065   -0.0065    0.0064   -0.0063

  Columns 81 through 88

    0.0062   -0.0061    0.0061   -0.0060    0.0059   -0.0058    0.0058   -0.0057

  Columns 89 through 96

    0.0056   -0.0056    0.0055   -0.0055    0.0054   -0.0053    0.0053   -0.0052

  Columns 97 through 100

    0.0052   -0.0051    0.0051   -0.0050
```

SUM OF VECTOR: -



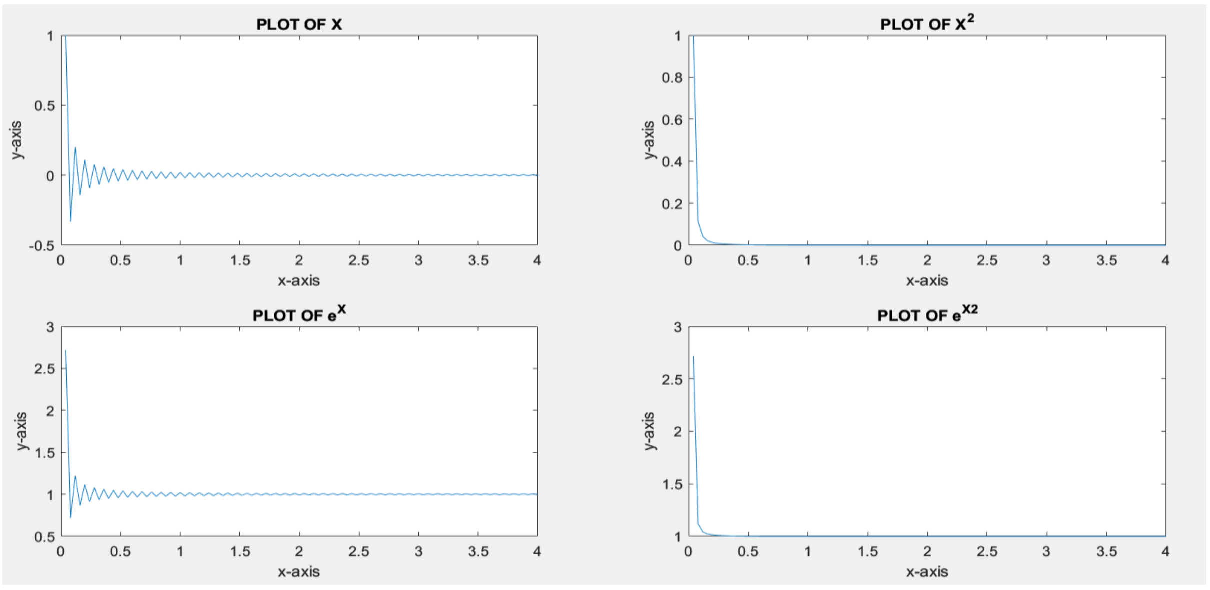
```
>> sum(X(n))

ans =

    0.7829
```

-

PLOT OF X, X^2, e^X, e^(X^2): -

```
>> z=0.04:0.04:4

z =

  Columns 1 through 8

    0.0400    0.0800    0.1200    0.1600    0.2000    0.2400    0.2800    0.3200

  Columns 9 through 16

    0.3600    0.4000    0.4400    0.4800    0.5200    0.5600    0.6000    0.6400
```

```
 Columns 17 through 24

   0.6800    0.7200    0.7600    0.8000    0.8400    0.8800    0.9200    0.9600

  Columns 25 through 32

   1.0000    1.0400    1.0800    1.1200    1.1600    1.2000    1.2400    1.2800

  Columns 33 through 40

   1.3200    1.3600    1.4000    1.4400    1.4800    1.5200    1.5600    1.6000

  Columns 41 through 48

   1.6400    1.6800    1.7200    1.7600    1.8000    1.8400    1.8800    1.9200

  Columns 49 through 56

   1.9600    2.0000    2.0400    2.0800    2.1200    2.1600    2.2000    2.2400

  Columns 57 through 64

   2.2800    2.3200    2.3600    2.4000    2.4400    2.4800    2.5200    2.5600

  Columns 65 through 72

   2.6000    2.6400    2.6800    2.7200    2.7600    2.8000    2.8400    2.8800

  Columns 73 through 80

   2.9200    2.9600    3.0000    3.0400    3.0800    3.1200    3.1600    3.2000

  Columns 81 through 88

   3.2400    3.2800    3.3200    3.3600    3.4000    3.4400    3.4800    3.5200

  Columns 89 through 96

   3.5600    3.6000    3.6400    3.6800    3.7200    3.7600    3.8000    3.8400

  Columns 97 through 100

   3.8800    3.9200    3.9600    4.0000

>> subplot(2,2,1)
>> plot(z,X)
>> title("PLOT OF X")
>> xlabel("x-axis")
>> ylabel("y-axis")
>> subplot(2,2,2)
>> plot(z,X.^2)
>> title("PLOT OF X^2")
>> xlabel("x-axis")
>> ylabel("y-axis")
>> subplot(2,2,3)

>> plot(z,exp(X))

>> title("PLOT OF e^X")

>> xlabel("x-axis")

>> ylabel("y-axis")

>> subplot(2,2,4)

>> plot(z,exp(X.^2))

>> title("PLOT OF e^(X^2)")

>> xlabel("x-axis")

>> ylabel("y-axis")
```



DISCUSSION:

The vector was constructed using for loop and the given formula and sum of all elements was calculated using sum(). Then functions X, X^2, e^X, e^(X^2) were plotted over the interval 0<X<4 using plot().

RESULT:

Vector X was created and subsequently X, X^2, e^X, e^(X^2) were plotted over the interval 0<X<4.

CONCLUSION:

Several functions X, X^2, e^X, e^(X^2) were successfully plotted for a vector X.

Criteria	Total Marks	Marks Obtained	Comments
Concept (A)	2		
Implementation (B)	2		
Performance (C)	2		
Total	6		

EXPERIMENT - 6

AIM:

Generating a Sinusoidal Signal of a given frequency (say, 100Hz) and Plotting with Graphical Enhancements: Titling, Labeling, Adding Text, Adding Legends, Adding New Plots to Existing Plot, Printing Text in Greek Letters, Plotting as Multiple and Subplot.

SOFTWARE USED: MATLAB 7.12.0(R2011a)

PROCEDURE:

>> % Generating parameter t

```
>> t=0:0.05*pi:2*pi

t =
Columns 1 through 9
         0    0.1571    0.3142    0.4712    0.6283    0.7854    0.9425    1.0996    1.2566
  Columns 10 through 18
    1.4137    1.5708    1.7279    1.8850    2.0420    2.1991    2.3562    2.5133    2.6704
  Columns 19 through 27
    2.8274    2.9845    3.1416    3.2987    3.4558    3.6128    3.7699    3.9270    4.0841
  Columns 28 through 36
    4.2412    4.3982    4.5553    4.7124    4.8695    5.0265    5.1836    5.3407    5.4978
  Columns 37 through 41
    5.6549    5.8119    5.9690    6.1261    6.2832
```

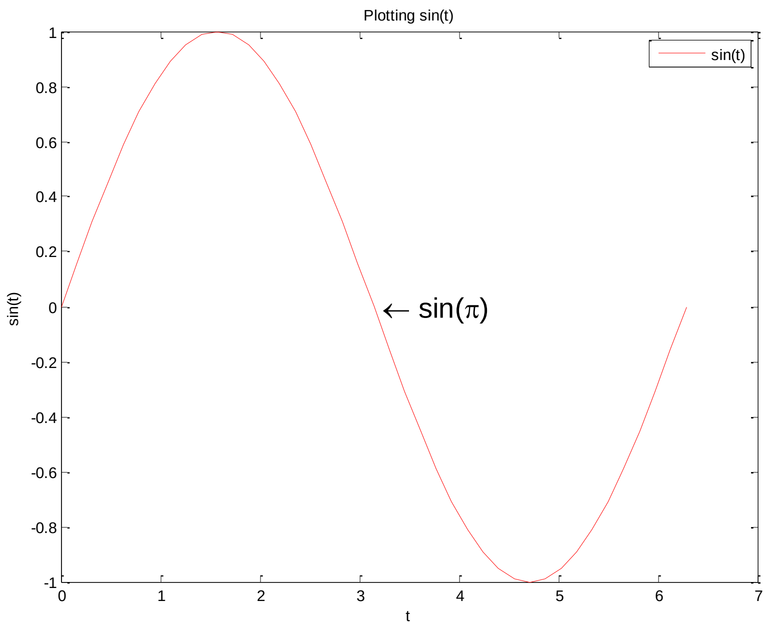
>> % Generating sin(t)

```
>> X=sin(t)

X =
Columns 1 through 9
         0    0.1564    0.3090    0.4540    0.5878    0.7071    0.8090    0.8910    0.9511
  Columns 10 through 18
    0.9877    1.0000    0.9877    0.9511    0.8910    0.8090    0.7071    0.5878    0.4540
  Columns 19 through 27
    0.3090    0.1564    0.0000   -0.1564   -0.3090   -0.4540   -0.5878   -0.7071   -0.8090
  Columns 28 through 36
   -0.8910   -0.9511   -0.9877   -1.0000   -0.9877   -0.9511   -0.8910   -0.8090   -0.7071
  Columns 37 through 41
   -0.5878   -0.4540   -0.3090   -0.1564   -0.0000
```

>> % Plotting with graphical enhancements

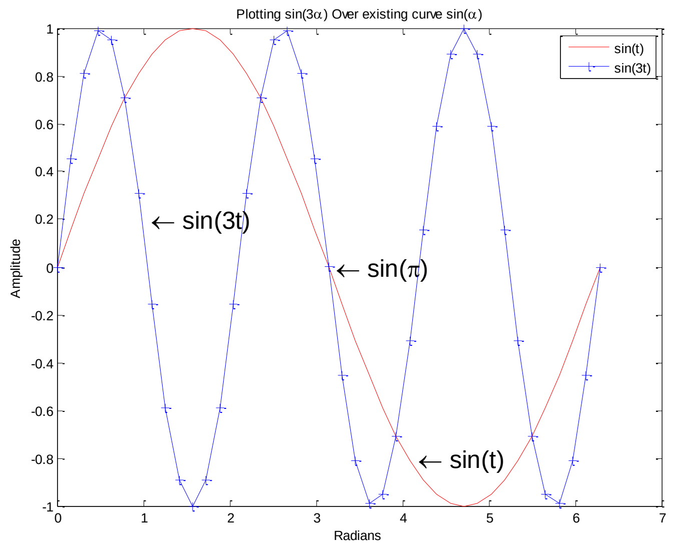
>> % Plotting sin(t) (With Titling, Labeling, Adding Text and Legends)

```
>> plot(t,X,'R-')
>> xlabel('t')
>> ylabel('sin(t)')
>> title('Plotting sin(t)')
>> legend('sin(t)')
>> text(pi,0,' \leftarrow sin(\pi)','FontSize',18)
```



>> % Adding new plots to existing plots,Printing Text in Greek Letters
>> % Adding sin(3*t) to sin(t)

```
>> hold on
>> Y=sin(3*t)
Y =
  Columns 1 through 9
         0    0.4540    0.8090    0.9877    0.9511    0.7071    0.3090   -0.1564   -0.5878
  Columns 10 through 18
   -0.8910   -1.0000   -0.8910   -0.5878   -0.1564    0.3090    0.7071    0.9511    0.9877
  Columns 19 through 27
    0.8090    0.4540    0.0000   -0.4540   -0.8090   -0.9877   -0.9511   -0.7071   -0.3090
  Columns 28 through 36
    0.1564    0.5878    0.8910    1.0000    0.8910    0.5878    0.1564   -0.3090   -0.7071
  Columns 37 through 41
   -0.9511   -0.9877   -0.8090   -0.4540   -0.0000

>> plot(t,Y,'B +-')
>> xlabel('Radians')
>> ylabel('Amplitude')
>> legend('sin(t)','sin(3t)')
>> title('Plotting sin(3{\alpha}) Over existing curve sin({\alpha})')
>> text(4.1,-0.8,' \leftarrow sin(t)','FontSize',18)
>> text(1,0.2,' \leftarrow sin(3t)','FontSize',18)
```



DISCUSSION:

Generating and plotting a Sinusoidal Signal: Input parameter t was generated using colon operator (start:step:stop).Sinusoidal Signal was generated using sin() and plotted using plot().
Plotting with Graphical Enhancements:
Titling: title('string') outputs the string at the top and in the center of the current axes.

Labeling: xlabel('string') labels the *x*-axis of the current axes. ylabel('string') labels the *y*-axis of the current axes.

Adding Text: text(x,y,z,'string','PropertyName',PropertyValue..): adds the string in quotes to the location defined by the coordinates and uses the values for the specified text properties.
Adding Legends: legend('string1','string2',..): displays a legend in the current axes using the specified strings to label each set of data.
Adding New Plots to Existing Plot: hold on: It retains the current plot and certain axes properties so that subsequent graphing commands add to the existing graph.
Printing Text in Greek Letters: You can define text that includes symbols and Greek letters using the text function, assigning the character sequence to the String property of text objects. You can also include these character sequences in the string arguments of the title, xlabel, ylabel, and zlabel functions. Example: xlabel('Time \musec.') displays Time µsec on graph.

RESULT:
Sinusoidal Signals were successfully generated and Plotted and several Graphical Enhancements like Titling, Labeling, Adding Text, Adding Legends, Adding New Plots to Existing Plot, Printing Text in Greek Letters were successfully performed.

CONCLUSION:

Sinusoidal Signal were successfully generated and plotted with several graphical enhancements.

Internal Assessment (Mandatory Experiment) Sheet for Lab Experiment
Department of Computer Science & Engineering
Amity University, Noida (UP)

Marking Criteria

Criteria	Total Marks	Marks Obtained	Comments
Concept (A)	2		
Implementation (B)	2		
Performance (C)	2		
Total	6		

EXPERIMENT – 8

AIM: Writing brief Scripts starting each Script with a request for input (using input) to Evaluate the function h(T) using if-else statement, where

$$h(T) = (T - 10)\ ,\ \text{for } 0 < T < 100$$

$$= (0.45\,T + 900),\ \text{for } T > 100.$$

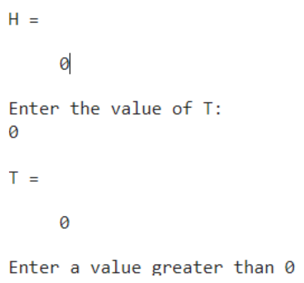
Exercise: Testing the Scripts written using A). T = 5, h = -5 and B). T = 110, h =949.5

SOFTWARE USED: MATLAB 7.12.0 (R2011a)

PROCEDURE:

```
>> % The Script

H=0

T=input('Enter the value of T: ')

if(T==0)

 disp('Enter a value greater than 0')

else if(0<T && T<100)

 fprintf('For T = %d',T)

 H=(T-10)

else if(T>100)

 fprintf('For T = %d',T)

 H=((0.45*T)+900)

 end

 end

end

fprintf('H = %d',H)
```

Output:



```
H =

     0

Enter the value of T:
0

T =

     0

Enter a value greater than 0
```

DISCUSSION:

If-else statements: If expression1 evaluates as false and expression2 as true, MATLAB executes the one or more commands denoted here as statements2. A true expression has either a logical true or nonzero value.

CONCLUSION:

The experiment calculated value of T and H according to the given if-else condition.

EXPERIMENT – 10

AIM: Basic 2D and 3D plots:

o parametric space curve.

o polygons with vertices.

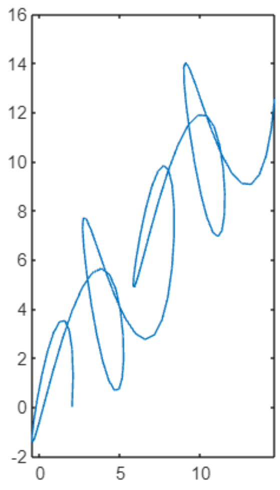
o 3D contour lines, pie and bar charts.

SOFTWARE USED: MATLAB R2019b

PROCEDURE:

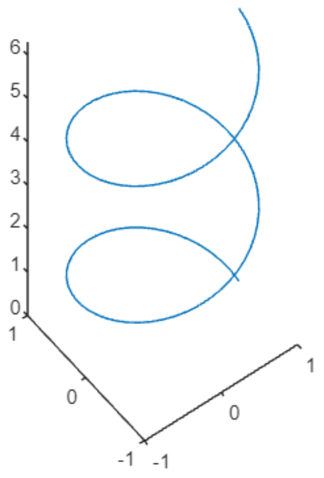
%2D PLOT

```
clc;
clear all;
t=linspace(0,4*pi);
x=t+2*cos(2*t);
y=t+3*sin(3*t);
subplot(1,2,1)
plot(x,y);
```



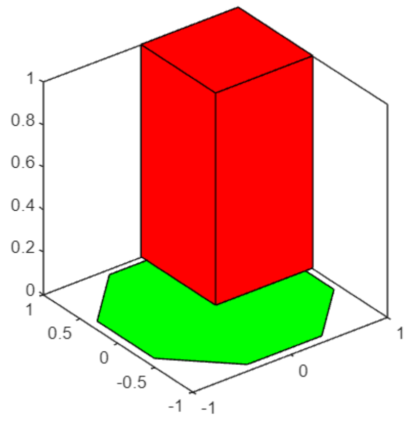
%3D PLOT

```
t=linspace(0,4*pi);
x=cos(t);
y=sin(t);
z=t/2;
subplot(1,2,2)
plot3(x,y,z);
```



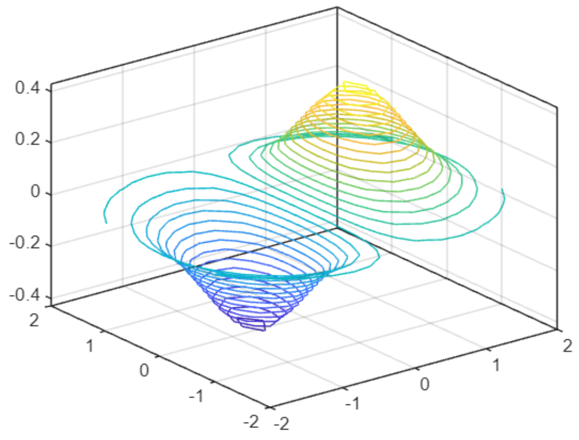
% polygon with vertices

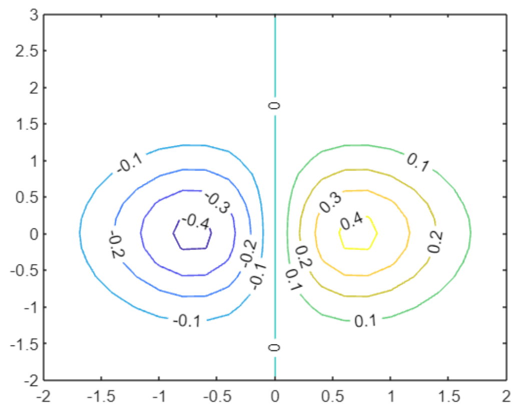
```
t = (1/16:1/8:1)'*2*pi;

x = cos(t);

y = sin(t);

fill(x,y,'g')

axis square

%3d

patch([0 0 1 1],[0 1 1 0],[1 1 1 1],'r')

patch([0 1 1 0],[0 0 0 0],[0 0 1 1],'r')

patch([0 0 0 0],[0 1 1 0],[0 0 1 1],'r')

view(-37.5, 30)

axis square
```



%3-D Contour Lines

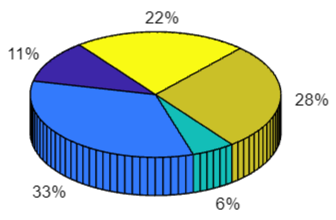
```
x = -2:0.25:2;

[X,Y] = meshgrid(x);

Z = X.*exp(-X.^2-Y.^2);

contour3(X,Y,Z,30)

% 2d

x = -2:0.2:2;

y = -2:0.2:3;

[X,Y] = meshgrid(x,y);

Z = X.*exp(-X.^2-Y.^2);

figure

contour(X,Y,Z,'ShowText','on')
```





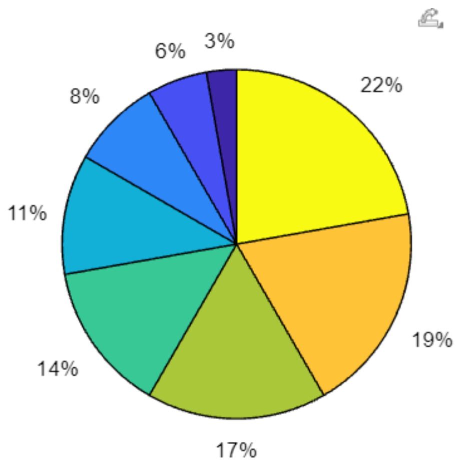
%3D Pie Chart

```
n=[1,3,0.5,2.5,2];
figure;
pie3(n);
```



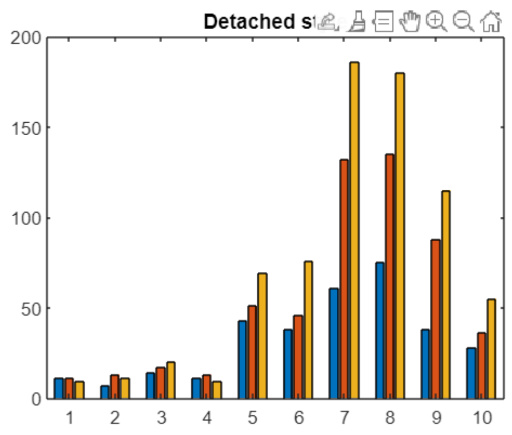
%2d pie

```
x= [1,2,3,4,5,6,7,8];
pie(x);
```



%2d bar

```
load count.dat
z=count(1:10,:);
figure;
bar(z);
title('Detached style');
```



% 3d bar

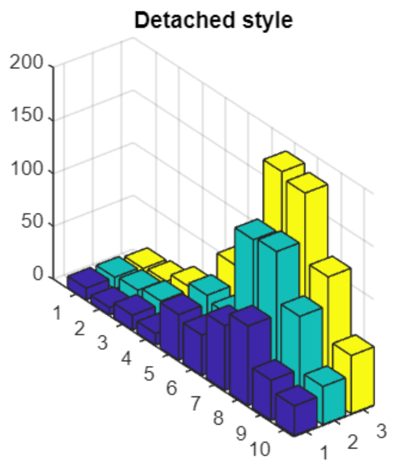
load count.dat

z=count(1:10,:);

figure;

bar3(z);

title('Detached style');



DISCUSION:

Contour () and pie () can be used as functions to design various charts.

RESULT:

The result is displayed above.

CONCLUSION:

Basic 2D and 3D plots have been plotted, bar and pie charts have also been plotted.

EXPERIMENT – 7

AIM:

Solving First Order Ordinary Differential Equation using Built-in Functions.

Consider the following Ordinary Differential Equation:

x(dy/dx) + 2y = x3

where, dy/dx = (x3

-2y) / x 1<x<3 and y=4.2

SOFTWARE USED: MATLAB 7.12.0 (R2011a)

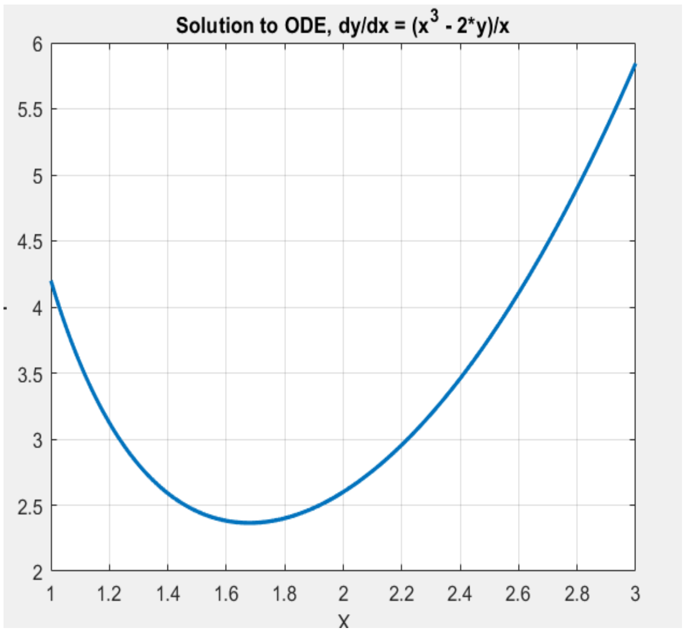
Commands:

```
>>% Clearing Everything
>> clc;clear all;clf
>> % Creating an Ordinary Differential Equation
>> ode1 = @ (x,y) (x^3 - 2*y) / x
ode1 =
 @(x,y)(x^3-2*y)/x
>> x =
 1.0000 1.0100 1.0200 1.0300 1.0400 1.0500
 1.0600 1.0700 1.0800 1.0900 1.1000 1.1100
 1.1200 1.1300 1.1400 1.1500 1.1600 1.1700
 1.1800 1.1900 1.2000 1.2100 1.2200 1.2300
 1.2400 1.2500 1.2600 1.2700 1.2800 1.2900
 1.3000 1.3100 1.3200 1.3300 1.3400 1.3500
 1.3600 1.3700 1.3800 1.3900 1.4000 1.4100
 1.4200 1.4300 1.4400 1.4500 1.4600 1.4700
 1.4800 1.4900 1.5000 1.5100 1.5200 1.5300
 1.5400 1.5500 1.5600 1.5700 1.5800 1.590
y =
 4.2000 4.1272 4.0569 3.9889 3.9232 3.8596
 3.7982 3.7388 3.6813 3.6257 3.5720 3.5200
 3.4697 3.4211 3.3741 3.3287 3.2848 3.2424
 3.2013 3.1617 3.1234 3.0864 3.0506 3.0161
 2.9828 2.9506 2.9196 2.8897 2.8608 2.8330
 2.8063 2.7805 2.7557 2.7318 2.7089 2.6869
 2.6657 2.6454 2.6260 2.6074 2.5896 2.5726
 2.5564 2.5409 2.5262 2.5122 2.4990 2.4864
 3.9658 4.0005 4.0356 4.0711 4.1069 4.1431
 4.1797 4.2166 4.2539 4.2915 4.3295 4.3679
 4.4067 4.4458 4.4853 4.5252 4.5654 4.6060
 4.6470 4.6883 4.7300 4.7721 4.8146 4.8574
 4.9006 4.9442 4.9881 5.0325 5.0772 5.1223
 5.1678 5.2136 5.2598 5.3064 5.3534 5.4008
 5.4485 5.4967 5.5452 5.5941 5.6434 5.6931
>>% Plotting the ordinary differential equation
>> plot(x,y,'linewidth',2)
>> xlabel('X'),ylabel('Y'),grid on
>> title('Solution to ODE, dy/dx = (x^3 - 2*y)/x')
```

DISCUSSION:

Ode45: ode45 is based on an explicit Runge-Kutta (4,5) formula, the Dormand-Prince pair. It is a one-step solver - in computing y(tn), it needs only the solution at the immediately preceding time point, y(tn-1)

RESULT:



CONCLUSION:

The solution to the ordinary differential equation using ode45 is calculated and plot is made on graph.

Internal Assessment (Mandatory Experiment) Sheet for Lab Experiment Department of Computer Science & Engineering Amity University, Noida (UP)			
Marking Criteria			
Criteria	**Total Marks**	**Marks Obtained**	**Comments**
Concept (A)	2		
Implementation (B)	2		
Performance (C)	2		
Total	6		

EXPERIMENT - 9

AIM:

Generating a Square Wave from sum of Sine Waves of certain Amplitude and Frequencies.

SOFTWARE USED: MATLAB 7.12.0(R2011a)

PROCEDURE:

```
>> % Defining Constant A
>> A=3
A =
 3
```

>> % Generating parameter t

```
>> t=0:0.05*pi:2*pi
t =
 Columns 1 through 9
 0 0.1571 0.3142 0.4712 0.6283 0.7854 0.9425 1.0996 1.2566
 Columns 10 through 18
 1.4137 1.5708 1.7279 1.8850 2.0420 2.1991 2.3562 2.5133 2.6704
 Columns 19 through 27
 2.8274 2.9845 3.1416 3.2987 3.4558 3.6128 3.7699 3.9270 4.0841
 Columns 28 through 36
 4.2412 4.3982 4.5553 4.7124 4.8695 5.0265 5.1836 5.3407 5.4978
 Columns 37 through 41
 5.6549 5.8119 5.9690 6.1261 6.2832
```

>> % Generating & Plotting Square wave using Fourier Series Expansion

```
>> sum=0
sum =
 0
>> for n=1:2:100
Y=(sin(n*t))/n
sum=sum+Y
```

```
end
Y =
 Columns 1 through 9
 0 0.0016 -0.0031 0.0046 -0.0059 0.0071 -0.0082 0.0090 -0.0096
 Columns 10 through 18
 0.0100 -0.0101 0.0100 -0.0096 0.0090 -0.0082 0.0071 -0.0059 0.0046
 Columns 19 through 27
 -0.0031 0.0016 -0.0000 -0.0016 0.0031 -0.0046 0.0059 -0.0071 0.0082
 Columns 28 through 36
 -0.0090 0.0096 -0.0100 0.0101 -0.0100 0.0096 -0.0090 0.0082 -0.0071
sum =
 Columns 1 through 9
 0 0.8171 0.7692 0.7964 0.7769 0.7925 0.7792 0.7910 0.7801
 Columns 10 through 18
 0.7905 0.7804 0.7905 0.7801 0.7910 0.7792 0.7925 0.7769 0.7964
 Columns 19 through 27
 0.7692 0.8171 0.0000 -0.8171 -0.7692 -0.7964 -0.7769 -0.7925 -0.7792

>> C=(4*A*sum)/pi
C =
 Columns 1 through 9
 0 3.1211 2.9383 3.0420 2.9675 3.0270 2.9764 3.0214 2.9799
 Columns 10 through 18
 3.0193 2.9809 3.0193 2.9799 3.0214 2.9764 3.0270 2.9675 3.0420
 Columns 19 through 27
 2.9383 3.1211 0.0000 -3.1211 -2.9383 -3.0420 -2.9675 -3.0270 -2.9764
 >> plot(t,C)
```

>> % Generating and Plotting sine wave over the existing curve

```
>> hold on
>> D=sin(t)
D =
 Columns 1 through 9
 0 0.1564 0.3090 0.4540 0.5878 0.7071 0.8090 0.8910 0.9511
 Columns 10 through 18
 0.9877 1.0000 0.9877 0.9511 0.8910 0.8090 0.7071 0.5878 0.4540
 Columns 19 through 27
 0.3090 0.1564 0.0000 -0.1564 -0.3090 -0.4540 -0.5878 -0.7071 -0.8090
 Columns 28 through 36
 -0.8910 -0.9511 -0.9877 -1.0000 -0.9877 -0.9511 -0.8910 -0.8090 -0.7071
 Columns 37 through 41
 -0.5878 -0.4540 -0.3090 -0.1564 -0.0000
>> plot(t,D,'R+-')
>> xlabel('Radians')
>> ylabel('Amplitude')
>> title('Generating Square Wave')
>> text(pi/2,1.125,' \downarrow sin(t)','FontSize',18)
>> text(3,3,' \leftarrow Square Wave','FontSize',18)
```

DISCUSSION:

Input parameter t was generated using colon operator (start:step:stop).

Generating Square Wave Using Fourier Series Expansion:

The Fourier series expansion for a square-wave is made up of a sum of odd harmonics. The more waves you add more smooth the square wave will become.

X(t)= 4*A/ π [sin(w*t) + (1/3)*sin(3*w*t) + (1/5)*sin(5*w*t) + ..]

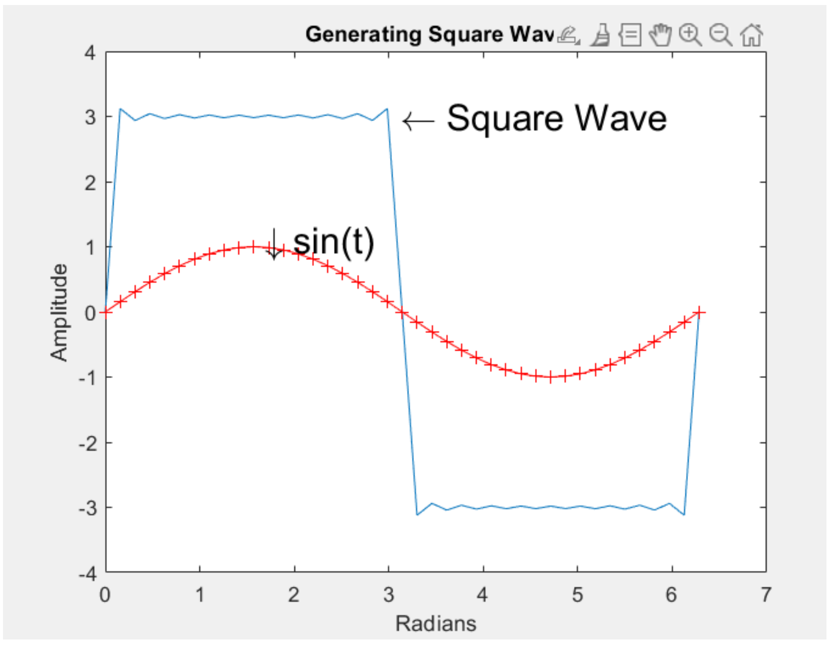
Adding New Plots to Existing Plot:

hold on: It retains the current plot and certain axes properties so that subsequent graphing commands add to the existing graph.

RESULT:

Square Wave was generated from sum of Sine Waves of certain Amplitude and Frequencies using Fourier Series Expansion.

\



CONCLUSION:

Square Wave was successfully generated and plotted.

Internal Assessment (Mandatory Experiment) Sheet for Lab Experiment Department of Computer Science & Engineering Amity University, Noida (UP)			
Marking Criteria			
Criteria	**Total Marks**	**Marks Obtained**	**Comments**
Concept (A)	2		
Implementation (B)	2		
Performance (C)	2		
Total	6		