FATIMA JINNAH WOMEN UNIVERSITY
Department Of Software Engineering

Numerical Analysis

LAB MANUAL

Submitted By :

Laiba Sohail(2021-BSE-016)
Neha Amjad(2021-BSE-024)
Tanzeela Asghar(2021-BSE-032)

Submitted To :

Sir Muhammad Shahzad

Section :

BSE (4A)

Date:

14TH JULY 2023

Tasks:

>>v=[1 4 7 10 13]

```
v =

   1    4    7   10   13
```

>>w=[1;4;7;10;13]

```
w =

   1
   4
   7
  10
  13
```

>>v(1:3)

```
ans =

   1    4    7
```

>>v(3:end)

```
ans =

   7   10   13
```

Task (1): what does the following command do?

>> v(:)

It is the most commonly used operator in MATLAB. It is used to specify iterations, generate vectors, and subscript arrays. If you want to generate a row vector with numbers in the range of 1 to 10.
1:10
MATLAB performs the instruction and provides a row vector containing the integers from 1 to 10 –
ans =
1 2 3 4 5 6 7 8 9 10

To create a matrix that has multiple rows, separate the rows with semicolons. a = [1 2 3; 4 5 6; 7 8 10]

a =

1 2 3

4 5 6

7 8 10

The element of row I and column j of the matrix A is denoted by A(I,j). >>A(3,3)

```
ans =

  10
```

suppose we want to enter a vector x consisting of points (0, 0.1, 0.2, 0.3, ... 5). We can use the command

>>x=0:0.1:5;

```
x =

 Columns 1 through 14

       0   0.1000   0.2000    0.3000    0.4000    0.5000    0.6000    0.7000    0.8000    0.9000    1.0000
1.1000   1.2000   1.3000

 Columns 15 through 28

    1.4000   1.5000   1.6000    1.7000    1.8000    1.9000    2.0000    2.1000    2.2000    2.3000    2.4000
2.5000   2.6000   2.7000
```

```
Columns 29 through 42

  2.8000   2.9000   3.0000   3.1000   3.2000   3.3000   3.4000   3.5000   3.6000   3.7000   3.8000
3.9000   4.0000   4.1000

Columns 43 through 51

  4.2000   4.3000   4.4000   4.5000   4.6000   4.7000   4.8000   4.9000   5.0000
```

Task (2-a): Generate a row vector x of 10 points linearly spaced between and including 0 and 10. Also perform the same task with the help of ":" operator and by specifying the increments.

```
x=0:1:10

x =

   0    1    2    3    4    5    6    7    8    9   10
```

Task (2-b): Generate a row vector y of 101 points linearly spaced between and including 0 and 2pi. Also perform the same task with the help of ":" operator and by specifying the increments.

```
y=0:3.6:360

y =

 Columns 1 through 14

       0   3.6000   7.2000   10.8000   14.4000   18.0000   21.6000   25.2000   28.8000   32.4000   36.0000
39.6000   43.2000   46.8000

 Columns 15 through 28

  50.4000   54.0000   57.6000   61.2000   64.8000   68.4000   72.0000   75.6000   79.2000   82.8000
86.4000   90.0000   93.6000   97.2000

 Columns 29 through 42
```

```
 100.8000  104.4000  108.0000  111.6000  115.2000  118.8000  122.4000  126.0000  129.6000
133.2000  136.8000  140.4000  144.0000  147.6000

 Columns 43 through 56

 151.2000  154.8000  158.4000  162.0000  165.6000  169.2000  172.8000  176.4000  180.0000
183.6000  187.2000  190.8000  194.4000  198.0000

 Columns 57 through 70

 201.6000  205.2000  208.8000  212.4000  216.0000  219.6000  223.2000  226.8000  230.4000
234.0000  237.6000  241.2000  244.8000  248.4000

 Columns 71 through 84

 252.0000  255.6000  259.2000  262.8000  266.4000  270.0000  273.6000  277.2000  280.8000
284.4000  288.0000  291.6000  295.2000  298.8000

 Columns 85 through 98

 302.4000  306.0000  309.6000  313.2000  316.8000  320.4000  324.0000  327.6000  331.2000
334.8000  338.4000  342.0000  345.6000  349.2000

 Columns 99 through 101

 352.8000  356.4000  360.0000
```

The colon operator can be used to pick up the certain row or column. For example, the statement A (m:n, k:l) specifies m to n columns k to l.

>>A (2,:)

```
ans =

   4     5     6
```

>>A(:2:3)

```
a(:,3)

ans =
```

```
3
6
10
```

<u>Task (3): Explain output of the following command?</u>

<u>>>A(:,2)=[]</u>

The output of the following command is that it prints the second column of the matrix A.

<u>Task (3-a): Extract a submatrix B consisting of rows 1 and 3 and columns 1 and 2 of matrix A</u>.

```
b=a([1 3],[1 2])

b =

   1    2
   7    8
```

<u>>>A=[1 2 3;4 5 6;7 8 9]</u>

<u>>>B=A([2 3],[1 2])</u>

```
A =

   1    2    3
   4    5    6
   7    8    9

b =

   4    5
   7    8
```

<u>>>size(A)</u>

<u>[m,n]=size(A)</u>

```
size(A)

ans =

   3    3
```

TASK 1:

Write a code which takes two matrices as an input from user in variable A and B respectively and give back addition, multiplication, inverse, determinant, transpose, of those matrices

Solution:

```
A=input('Enter a first matrix : ')
B=input('Enter a second matrix : ')
display('The addition of your matrices is')
C = A+B
display('The multiplication of your matrices is');
D = A*B
display('The inverse of your first matrix is');
E = inv(A)
display('The inverse of your second matrix is');
F = inv(B)
display('The determinant of your first matrix is');
G = det(A)
display('The determinant of your second matrix is');
H = det(B)
display('The transpose of your first matrix is');
I = [A]'
display('The transpose of your second matrix is');
J = [B]'
```

```
Command Window

>> Na_lab1_TAsk1
Enter a first matrix : [1 2;3 4]

A =

     1     2
     3     4

Enter a second matrix : [6 7;5 6]

B =

     6     7
     5     6

The addition of your matrices is

C =

     7     9
     8    10
```

```
The multiplication of your matrices is

D =

    16    19
    38    45

The inverse of your first matrix is

E =

   -2.0000    1.0000
    1.5000   -0.5000

The inverse of your second matrix is

F =

    6.0000   -7.0000
   -5.0000    6.0000
```

```
The determinant of your first matrix is

G =

    -2

The determinant of your second matrix is

H =

    1.0000
```

```
The transpose of your first matrix is

I =

     1     3
     2     4

The transpose of your second matrix is

J =

     6     5
     7     6

>>
```

TASK 2:

Write a MATLAB code which takes a number as input from user.
If the given number is less than 0 and integer, then it should display

- Square of that number

If the given number is greater than 0 and integer, then it should display

- Square root of that number

If the given number is less than 0 and not integer, then it should display

- Cube of that number

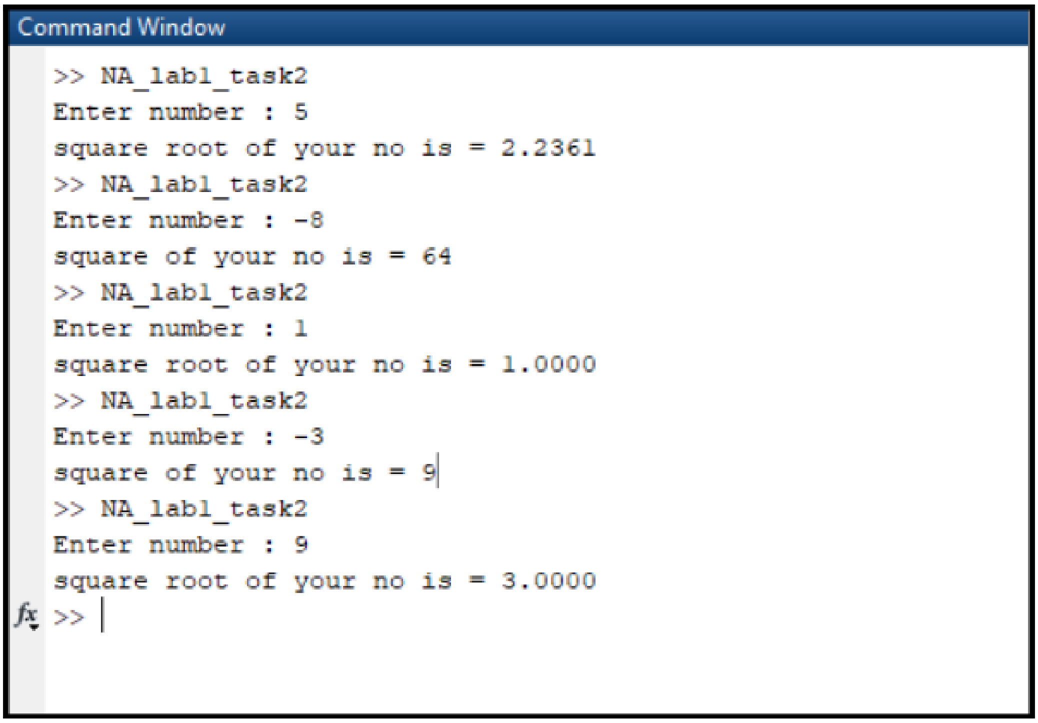
If the given number is greater than 0 and not integer, then it should display

- Cube root of that number

Else it should display
"Check your inputs"

Solution:

```
x =input('Enter number : ');
r=rem(x,1);
if(x<0 && rem(x,1)==0)
fprintf('square of your no is = %d\n',x^2);
elseif(x>0 && rem(x,1)==0)
fprintf('square root of your no is = %0.4f\n', x^(1/2));
elseif(x<0 && rem(x,1)~=0)
fprintf('cube of your no is = %d\n',x^3);
elseif(x>0 && rem(x,1)~=0)
fprintf('cube root of your no is = %0.4f\n', x^(1/3));
else
display('Check your inputs');
end
```



```
Command Window
>> NA_lab1_task2
Enter number : 5
square root of your no is = 2.2361
>> NA_lab1_task2
Enter number : -8
square of your no is = 64
>> NA_lab1_task2
Enter number : 1
square root of your no is = 1.0000
>> NA_lab1_task2
Enter number : -3
square of your no is = 9
>> NA_lab1_task2
Enter number : 9
square root of your no is = 3.0000
>>
```

<u>Task 3</u>

Roots(a) will find the roots of the desired equation.

Task (3-c): find the roots of the following equations, first with the help of quadratic formula and then with the help of roots commands so that may be compared and verified. 882 + 488 + 4

3882 + 488 + 10

<u>Solution:</u>

```
a=input('Enter a =');
b=input('Enter b =');
c=input('Enter c =');
d=b*b-4*a*c;
if (d<0)
    fprintf ('Equation has no real root');
else
    r1=(-b+sqrt(d))/(2*a);
    r2=(-b-sqrt(d))/(2*a);
    if (d==0)
        fprintf('Equation has one root');
        fprintf('\n%f',r1);
    else
        fprintf('The equation has two roots')
        fprintf('\n%f %f',r1,r2);
    end
end
```

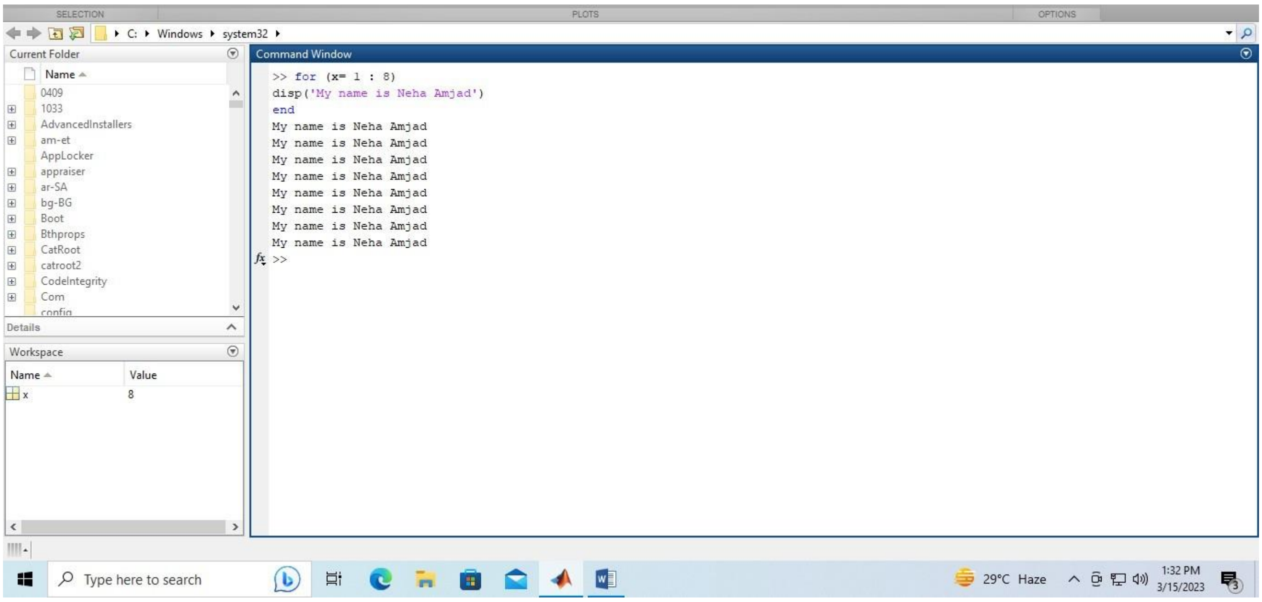
```
Enter a = 2
Enter b = 6
Enter c = 7
Equation has no real root>>
```

```
Enter a = 1
Enter b = 6
Enter c = 4
The equation has two roots
-0.763932 -5.236068>>
```

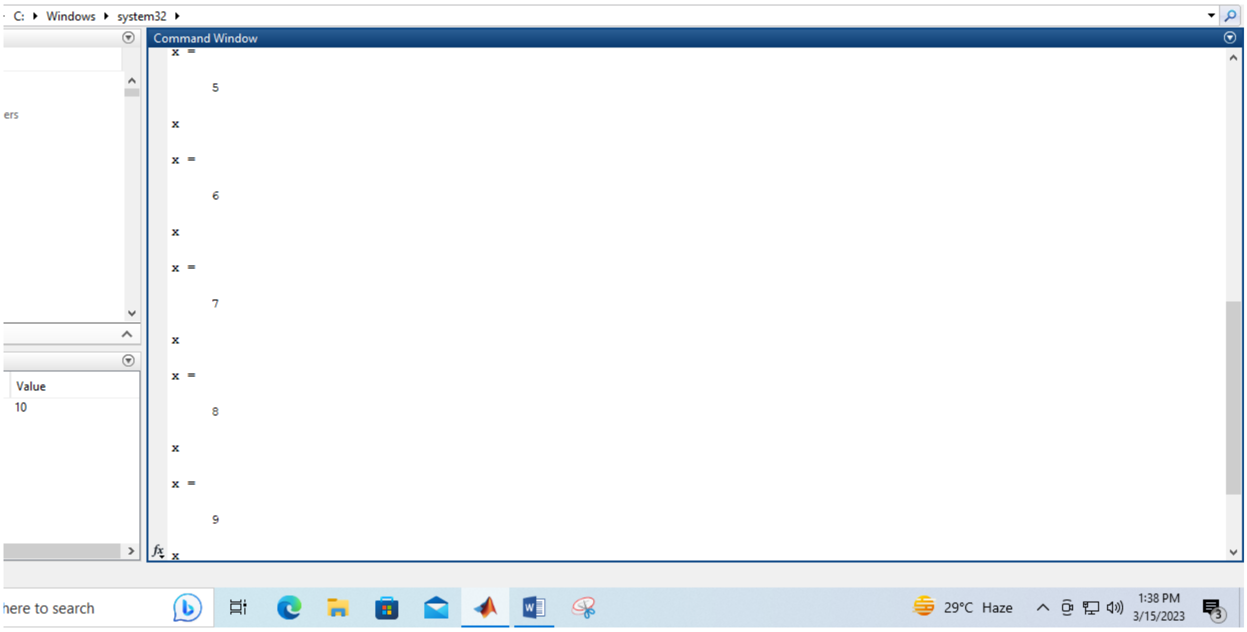
```
Enter a = 1
Enter b = 2
Enter c = 1
Equation has one root
-1.000000>>
```

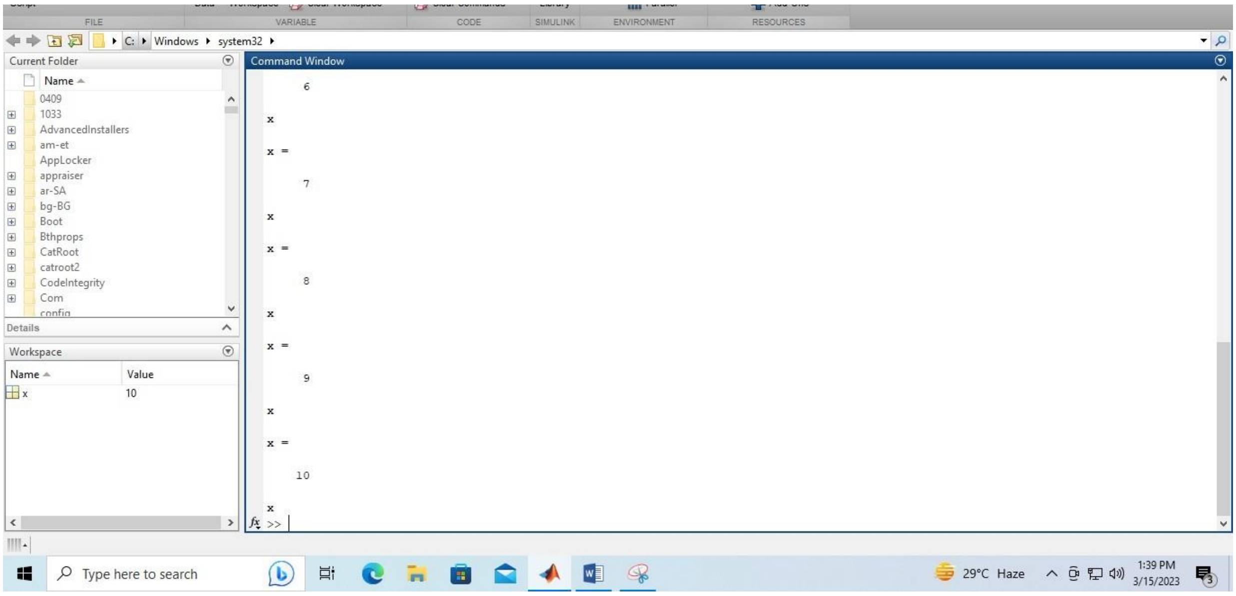
"LAB#2"

TASK 1:

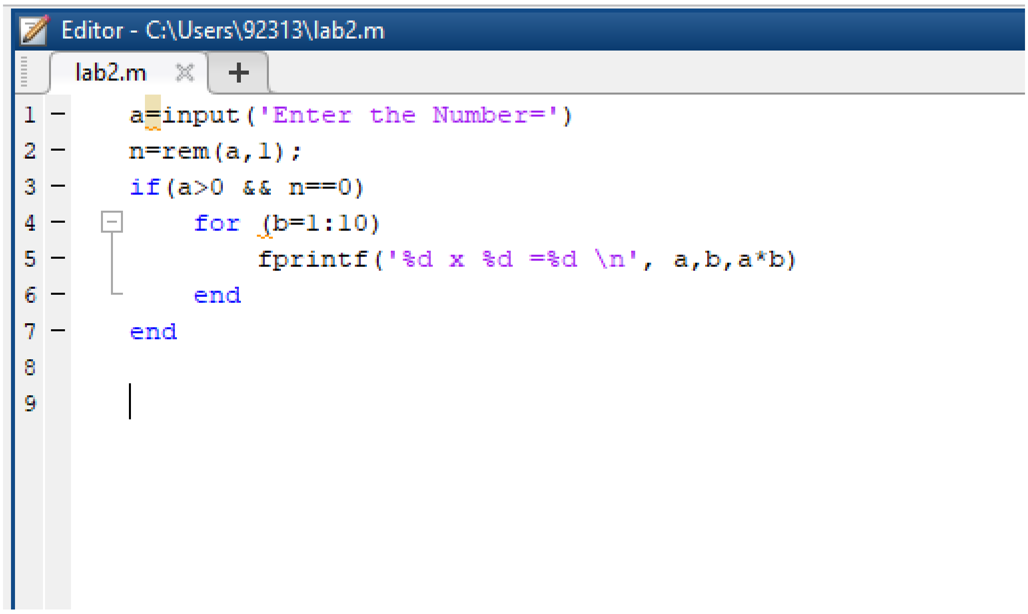

Command Window
>> for (x= 1 : 8)
disp('My name is Neha Amjad')
end
My name is Neha Amjad
My name is Neha Amjad
My name is Neha Amjad
My name is Neha Amjad
My name is Neha Amjad
My name is Neha Amjad
My name is Neha Amjad
My name is Neha Amjad
Workspace
Name Value
x 8

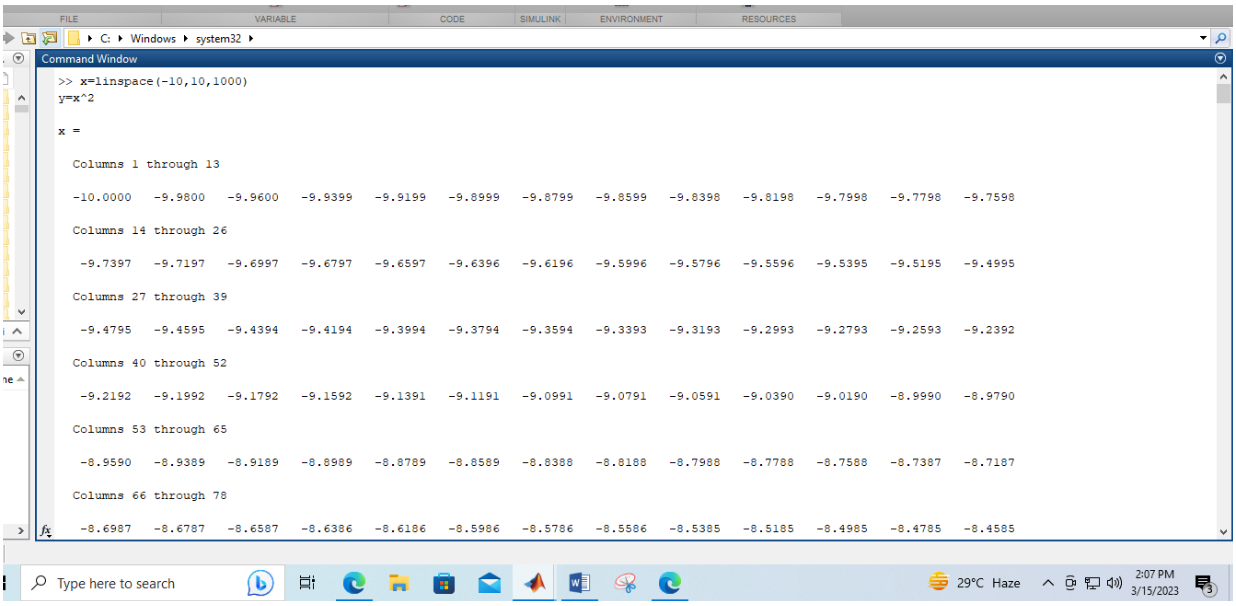

Command Window
>> x=1
x =
1
>> while(x<10)
x=x+1
disp('x')
end
x =
2
x
x =
3
x
x =
4
x
Workspace
Name Value
x 10


C: ▸ Windows ▸ system32 ▸
Command Window
x =
5
x
x =
6
x
x =
7
x
x =
8
x
x =
9
x
Value
10
here to search
29°C Haze
1:38 PM
3/15/2023


FILE
VARIABLE
CODE
SIMULINK
ENVIRONMENT
RESOURCES
C: ▸ Windows ▸ system32 ▸
Current Folder
Name
0409
1033
AdvancedInstallers
am-et
AppLocker
appraiser
ar-SA
bg-BG
Boot
Bthprops
CatRoot
catroot2
CodeIntegrity
Com
config
Details
Workspace
Name
Value
x
10
Command Window
6
x
x =
7
x
x =
8
x
x =
9
x
x =
10
x
>>
Type here to search
29°C Haze
1:39 PM
3/15/2023

TASK 2:\


Editor - C:\Users\92313\lab2.m
lab2.m
a=input('Enter the Number=')
n=rem(a,1);
if(a>0 && n==0)
for (b=1:10)
fprintf('%d x %d =%d \n', a,b,a*b)
end
end


FILE
VARIABLE
CODE
SIMULINK
ENVIRONMENT
RESOURCES
C: Windows system32
Command Window
>> x=linspace(-10,10,1000)
y=x^2
x =
Columns 1 through 13
-10.0000 -9.9800 -9.9600 -9.9399 -9.9199 -9.8999 -9.8799 -9.8599 -9.8398 -9.8198 -9.7998 -9.7798 -9.7598
Columns 14 through 26
-9.7397 -9.7197 -9.6997 -9.6797 -9.6597 -9.6396 -9.6196 -9.5996 -9.5796 -9.5596 -9.5395 -9.5195 -9.4995
Columns 27 through 39
-9.4795 -9.4595 -9.4394 -9.4194 -9.3994 -9.3794 -9.3594 -9.3393 -9.3193 -9.2993 -9.2793 -9.2593 -9.2392
Columns 40 through 52
-9.2192 -9.1992 -9.1792 -9.1592 -9.1391 -9.1191 -9.0991 -9.0791 -9.0591 -9.0390 -9.0190 -8.9990 -8.9790
Columns 53 through 65
-8.9590 -8.9389 -8.9189 -8.8989 -8.8789 -8.8589 -8.8388 -8.8188 -8.7988 -8.7788 -8.7588 -8.7387 -8.7187
Columns 66 through 78
-8.6987 -8.6787 -8.6587 -8.6386 -8.6186 -8.5986 -8.5786 -8.5586 -8.5385 -8.5185 -8.4985 -8.4785 -8.4585
Type here to search
29°C Haze
2:07 PM
3/15/2023

```
Command Window
>> lab2
Enter the Number=12

a =

    12

12 x 1 =12
12 x 2 =24
12 x 3 =36
12 x 4 =48
12 x 5 =60
12 x 6 =72
12 x 7 =84
12 x 8 =96
12 x 9 =108
12 x 10 =120
>>
```

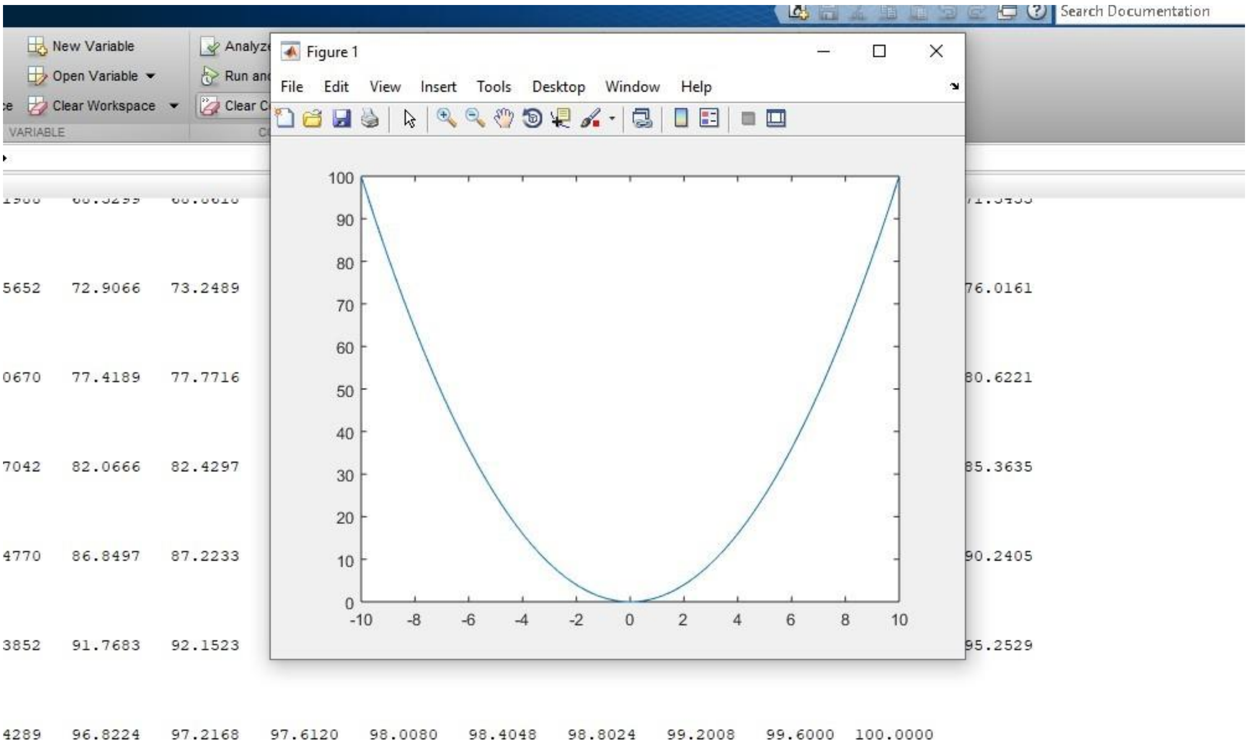
TASK 3:

```
Command Window
>> y=x.^2

y =

  Columns 1 through 13

  100.0000   99.6000   99.2008   98.8024   98.4048   98.0080   97.6120   97.2168   96.8224   96.4289   96.0361   95.6441   95.2529

  Columns 14 through 26

   94.8625   94.4730   94.0842   93.6962   93.3090   92.9227   92.5371   92.1523   91.7683   91.3852   91.0028   90.6213   90.2405

  Columns 27 through 39

   89.8605   89.4814   89.1030   88.7255   88.3487   87.9728   87.5976   87.2233   86.8497   86.4770   86.1050   85.7339   85.3635

  Columns 40 through 52

   84.9940   84.6253   84.2573   83.8902   83.5239   83.1583   82.7936   82.4297   82.0666   81.7042   81.3427   80.9820   80.6221

  Columns 53 through 65

   80.2629   79.9046   79.5471   79.1904   78.8345   78.4794   78.1251   77.7716   77.4189   77.0670   76.7159   76.3656   76.0161

  Columns 66 through 78

   75.6674   75.3195   74.9724   74.6261   74.2806   73.9359   73.5920   73.2489   72.9066   72.5652   72.2245   71.8846   71.5455
```

Type here to search   29°C Haze   2:08 PM 3/15/2023

```
   71.8846   72.2245   72.5652   72.9066   73.2489   73.5920   73.9359   74.2806   74.6261   74.9724   75.3195   7!

  Columns 937 through 949

   76.3656   76.7159   77.0670   77.4189   77.7716   78.1251   78.4794   78.8345   79.1904   79.5471   79.9046   8(

  Columns 950 through 962

   80.9820   81.3427   81.7042   82.0666   82.4297   82.7936   83.1583   83.5239   83.8902   84.2573   84.6253   8

  Columns 963 through 975

   85.7339   86.1050   86.4770   86.8497   87.2233   87.5976   87.9728   88.3487   88.7255   89.1030   89.4814   8!

  Columns 976 through 988

   90.6213   91.0028   91.3852   91.7683   92.1523   92.5371   92.9227   93.3090   93.6962   94.0842   94.4730   9

  Columns 989 through 1000

   95.6441   96.0361   96.4289   96.8224   97.2168   97.6120   98.0080   98.4048   98.8024   99.2008   99.6000   10(

>> plot(x,y)
>>
fx >>
```

Type here to search



```
>> f=@(x) x^2

f =

    @(x)x^2

>> fplot(f)
Error using fplot (line 51)
Not enough input arguments.

>> x=5

x =

     5

>> f(5)

ans =

    25

>> fplot(f)
Error using fplot (line 51)
Not enough input arguments.

>>
```

TASK 4

EDITOR PUBLISH VIEW

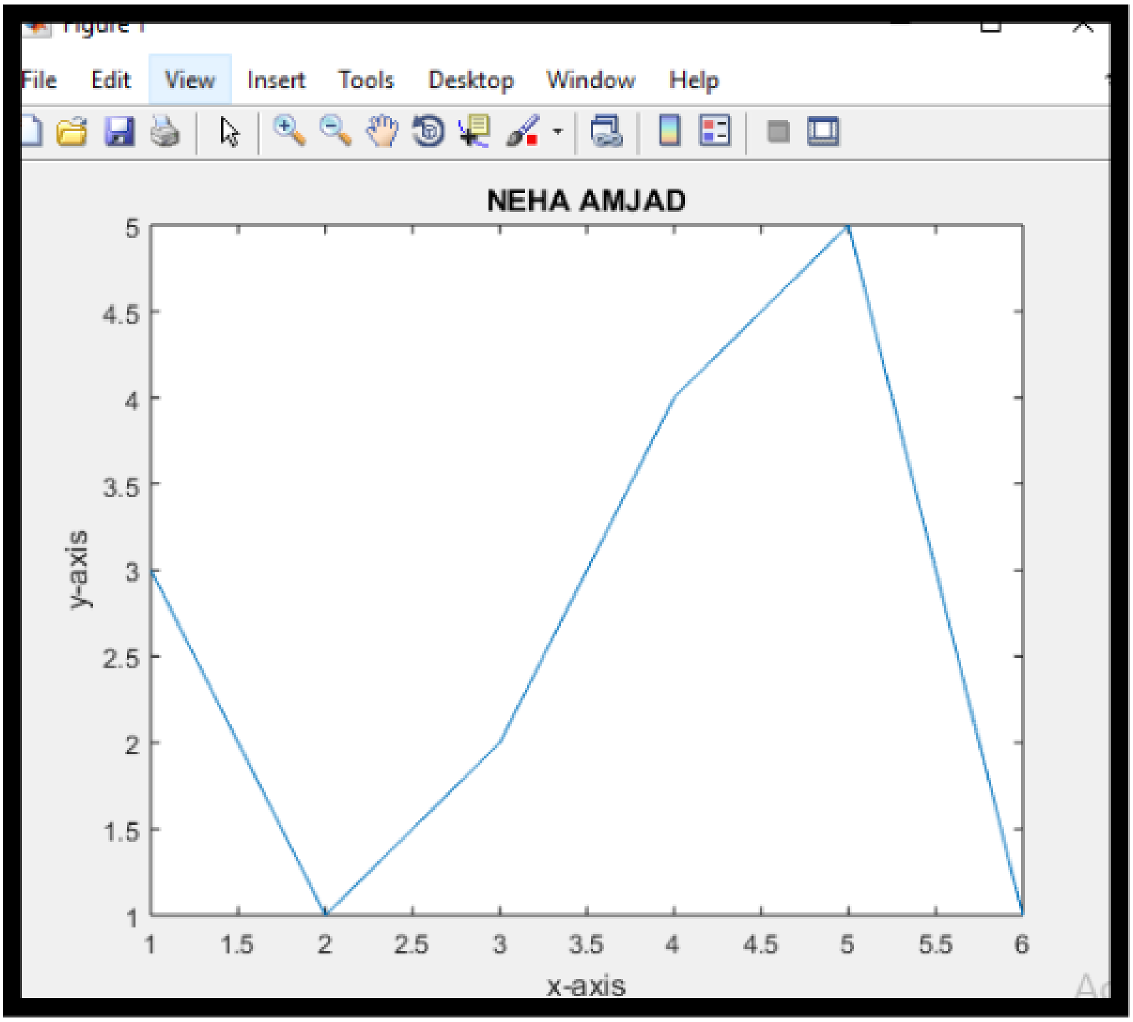
lab2.m Untitled2 +

```
A=[2 3 4;4 4 6;9 8 10]
B=[12;7;6]
disp('value of X= ')
X=inv(A)*B
f=@(x,y,z) (2*x+3*y+4*z)
f(-7,8,0.5)
```

	Value
A	[2,3,4;4,4,6;9,8,10]
ans	12
B	[12;7;6]
f	@(x,y,z)(2*x+3*y+4*z)
X	[-7.0000;8.0000;0.5000]

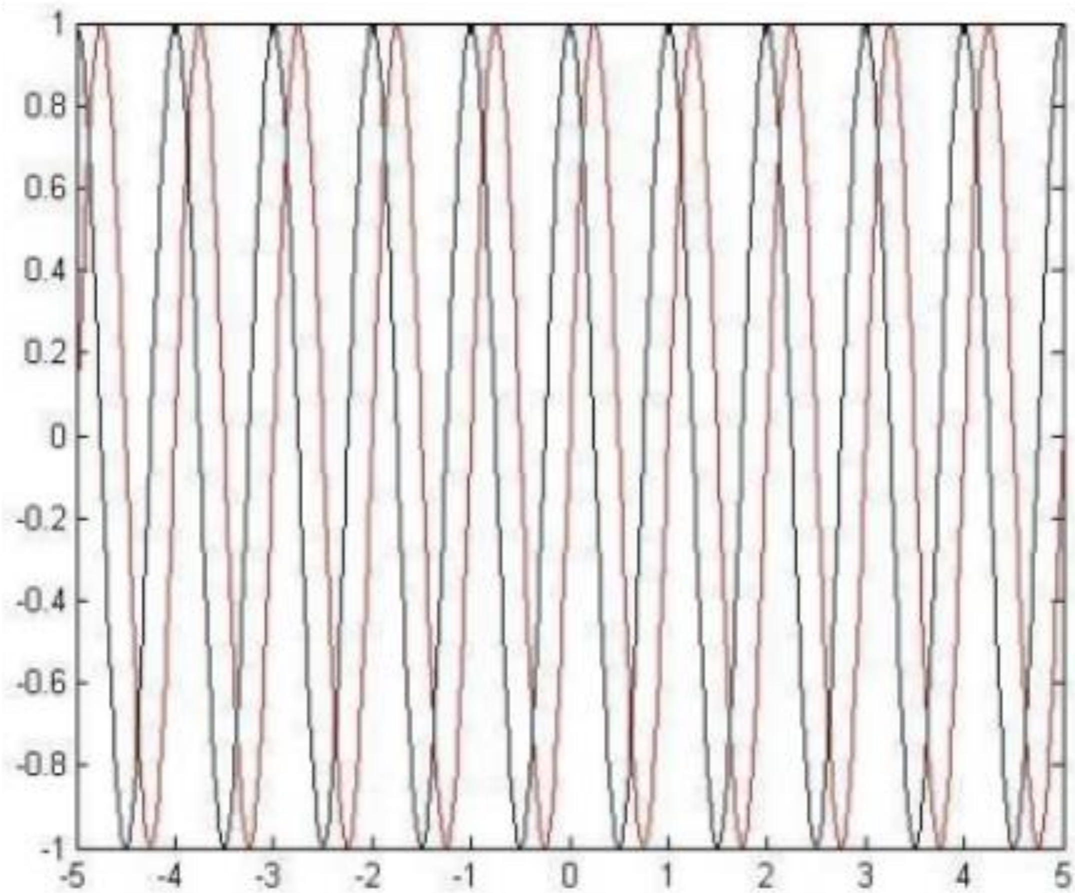
LAB #3

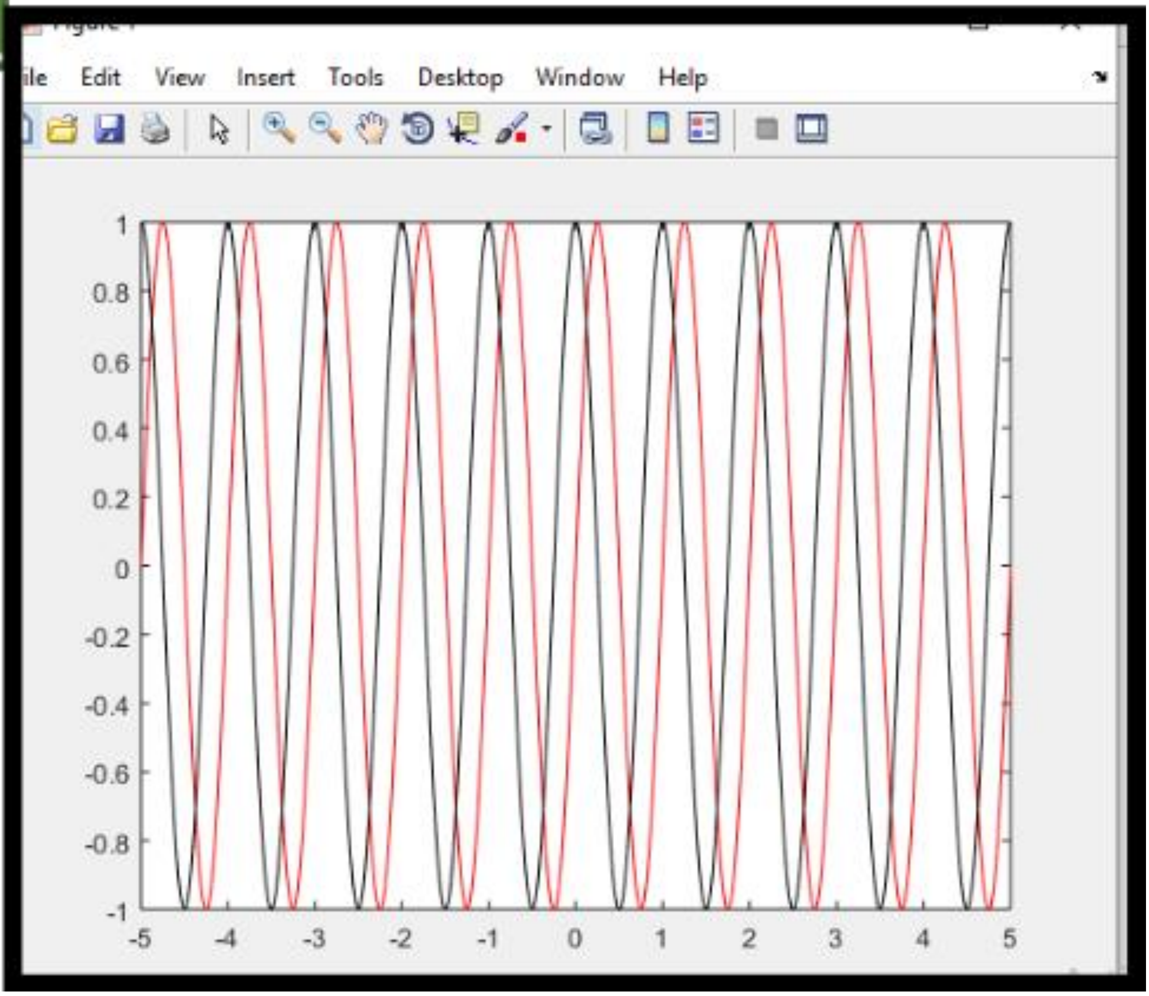
```
>>  x=[1;2;3;4;5;6]

x =

     1
     2
     3
     4
     5
     6

>> y=[3;1;2;4;5;1]

y =

     3
     1
     2
     4
     5
     1

>> plot(x,y)
```



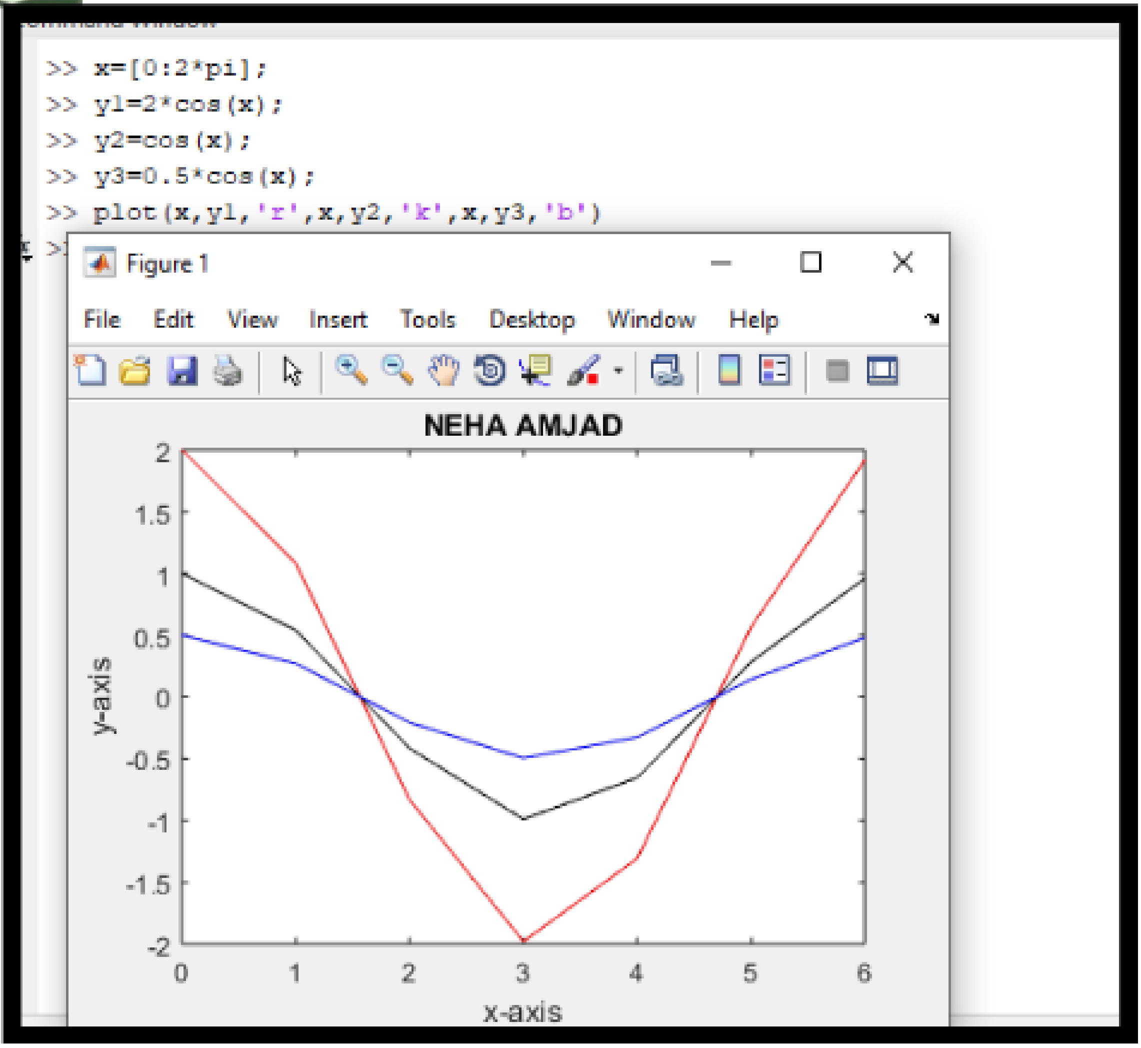
Type the following commands and observe the plot shown in figure 1.1

```
» t = -5:.01:5;
» y = sin (2*pi*t);
» plot (t,y);

» z = cos (2*pi*t);

» plot(t,y,'r',t,z,'k')
```

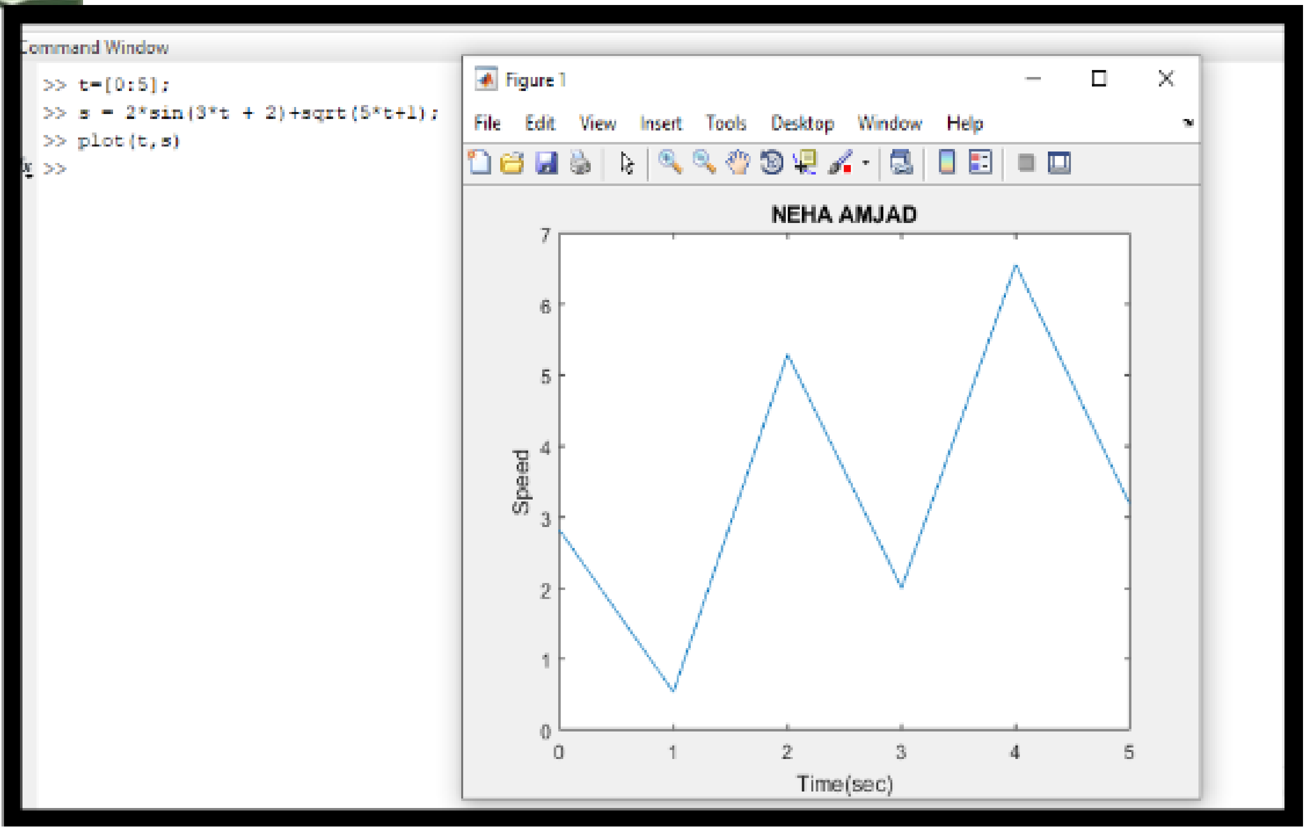

1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-5
-4
-3
-2
-1
0
1
2
3
4
5



Task (1): Plot the following cosine functions 88 = 2 cos(x), = cos(x), and = 0.5cos(x), in the
$_{1}88_{2}88_{3}$
interval $0 \leq x \leq 2\pi$. The title of the graph should show your name whereas x and y axis should be labeled appropriately.



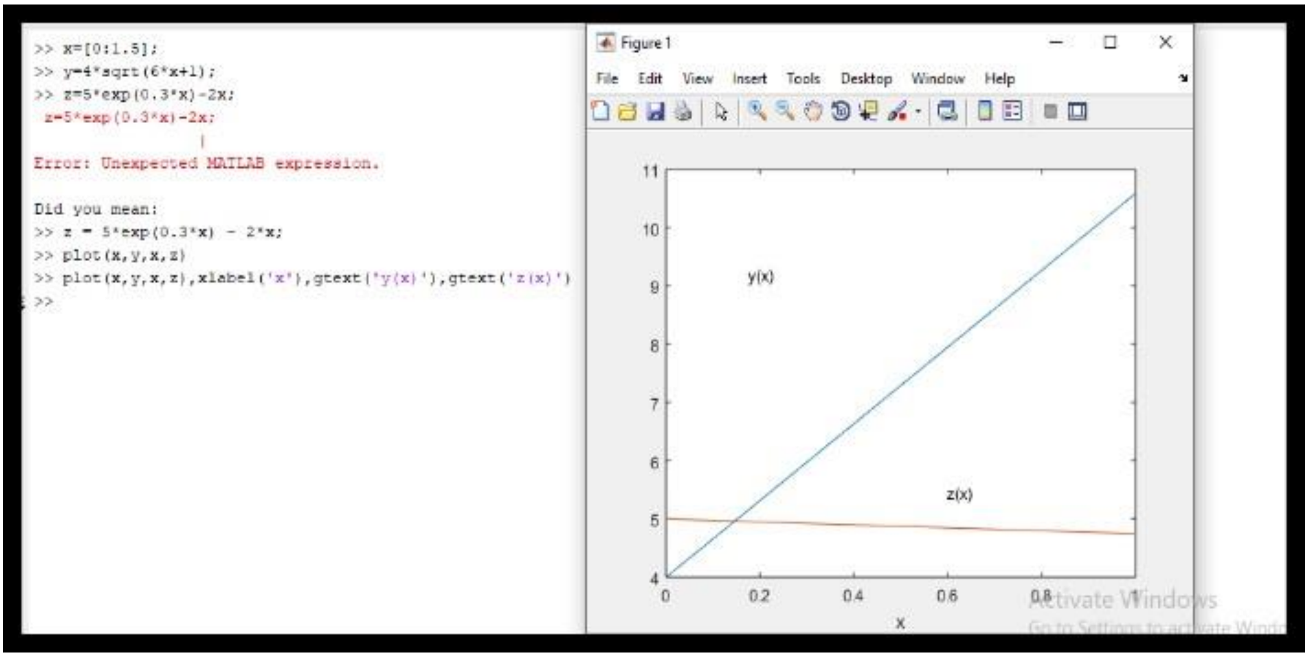
Task (2): Use Matalb to plot the function s = 2 sin (3t + 2) + [redacted] over the interval $0 \le t \le$ 5. Put a title on the plot, and properly label the axes. The variable s represents speed in feet per second; the variable t represents time in second.



0.388 — 288

Task (3): Use Matalb to plot the function y = 4 [redacted] and z = 5 88 over the interval 0 ≤ t

≤ 1. 5. Properly label the plot and each curve. The variable x represents distance in meters.



SUBPLOT:

Study the following code and observe the output figure 1.2.

x= (0:0.5:10);
y=sin (x)
z=cos (x)subplot
(1,2,1) plot (x,y)
subplot
(1,2,2) plot
(x,z)

The subplot command [subplot(m,n,p)] divides the output window into the desired number of panes e.g. subplot (3,2,5) creates an array of six panes, three panes deep and two panes across, and directs the next plot to appear in the fifth pane (in the bottom-left corner).

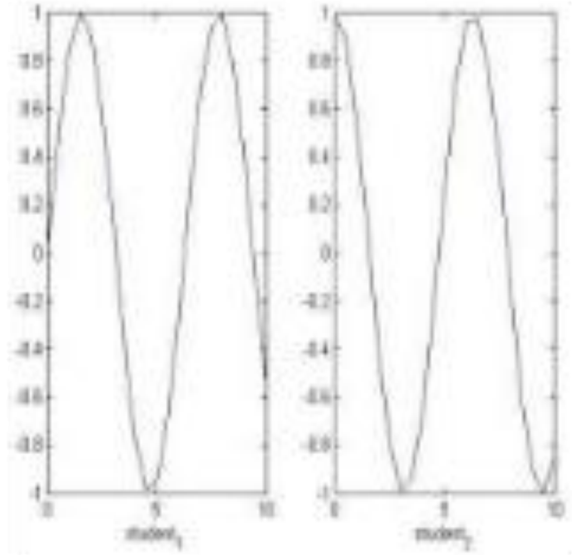
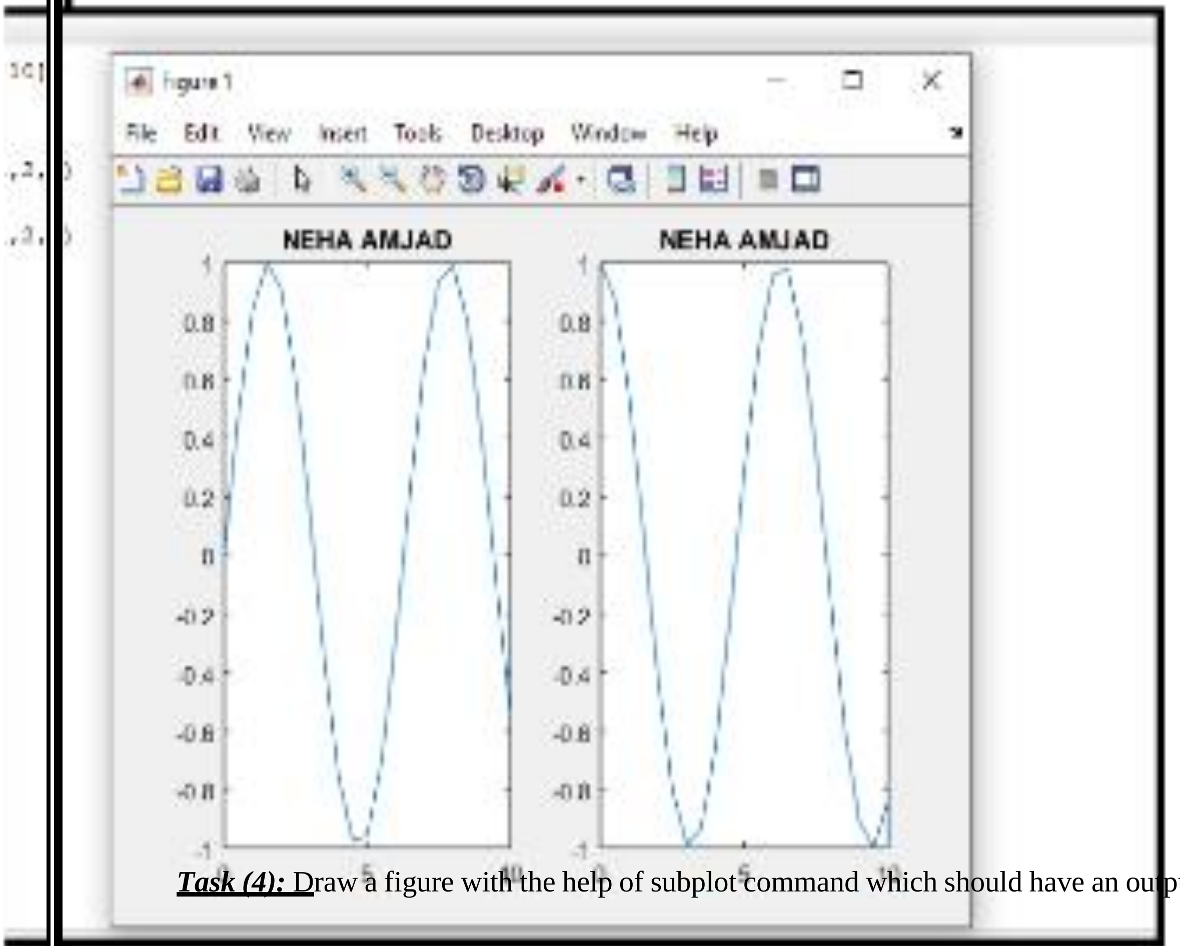


Figure 1.2



Task (4): Draw a figure with the help of subplot command which should have an output as shown in figure 1.3

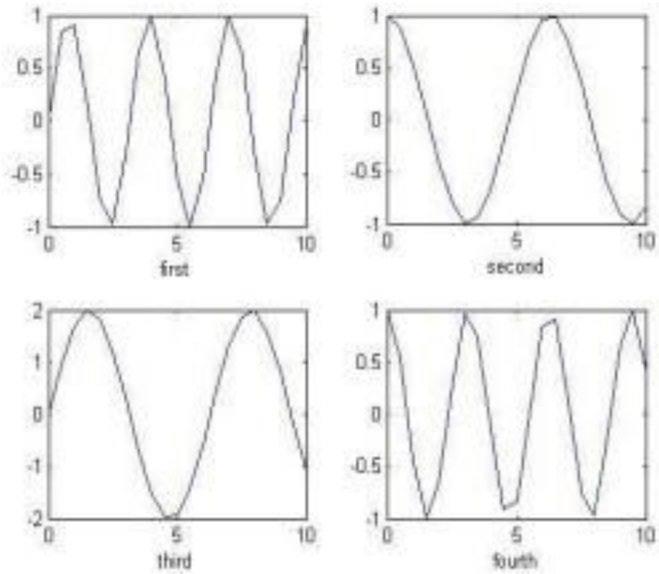
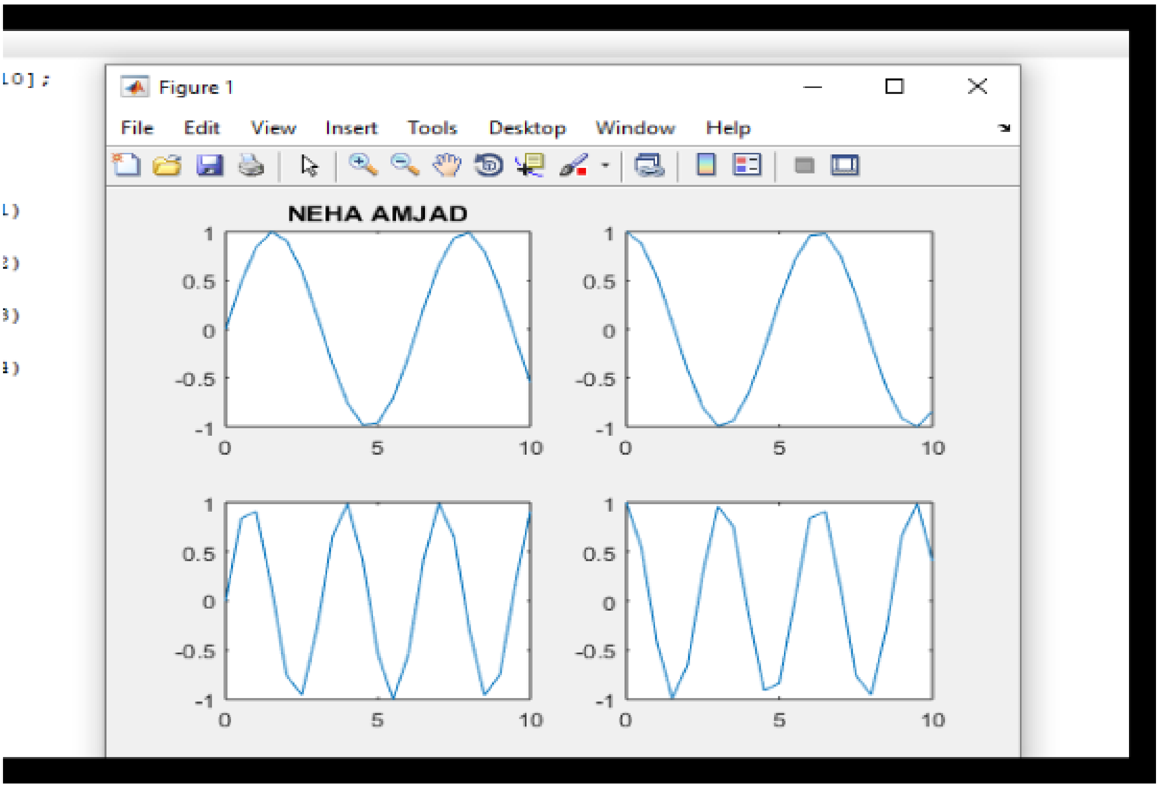


Figure 1.3



Task (5): Draw a figure with the help of subplot command which should have an output figure as follows

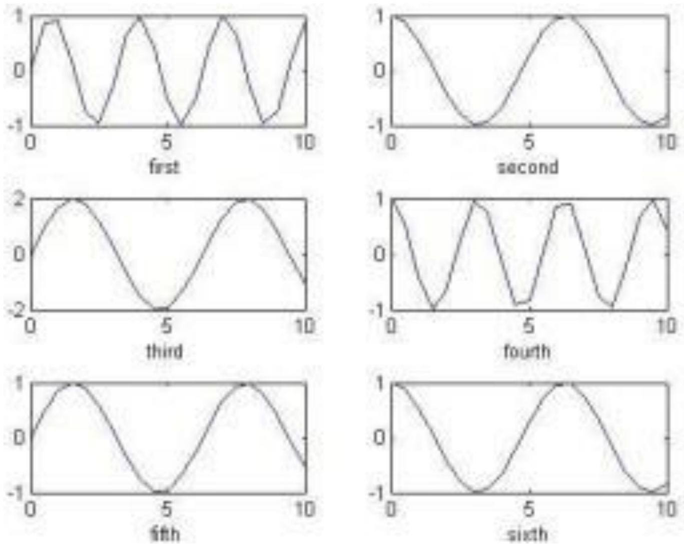


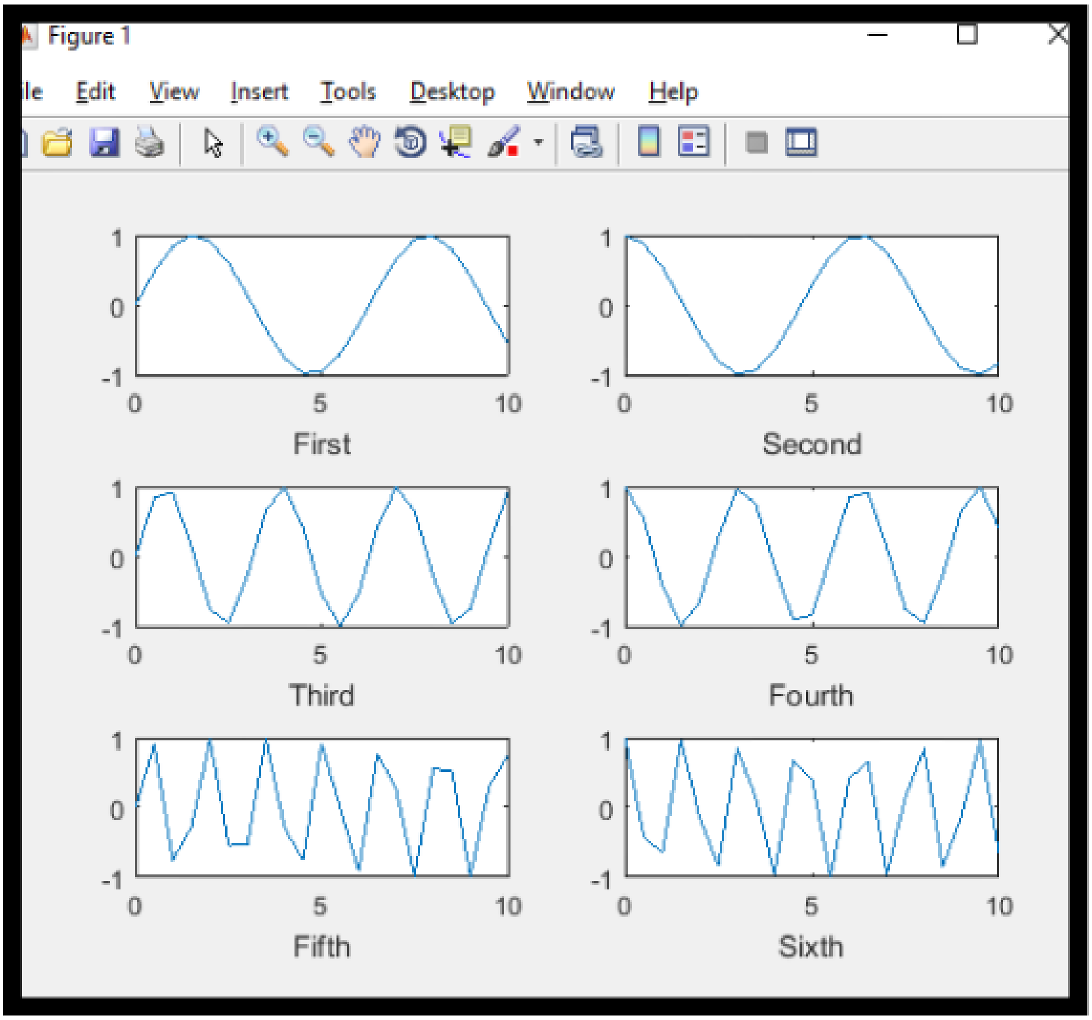
Figure 1.4

Editor - D:\university\numerical Analysis Lab\lab3_t4.m

lab3_e1.m | lab3_t1.m | lab3_t2.m | lab3_t3.m | lab3_e3.m | lab3_t4.m

```
clc
clear all
x=0:0.5:10;
y1=sin(x);
y2=cos(x);
y3=sin(2*x);
y4=cos(2*x);
y5=sin(4*x);
y6=cos(4*x);
subplot(3,2,1)
plot(x,y1)
xlabel('First')
```

```
subplot(3,2,2)
plot(x,y2)
xlabel('Second')
subplot(3,2,3)
plot(x,y3)
xlabel('Third')
subplot(3,2,4)
plot(x,y4)
xlabel('Fourth')
subplot(3,2,5)
plot(x,y5)
xlabel('Fifth')
subplot(3,2,6)
plot(x,y6)
xlabel('Sixth')
```



Stem

Stem() method in MATLAB is a type of plotting method to represent any type of data in a discrete form. This method generates a plot in the form of vertical lines being extended from thebases line, having little circles at tips which represents the exact value of the given data.

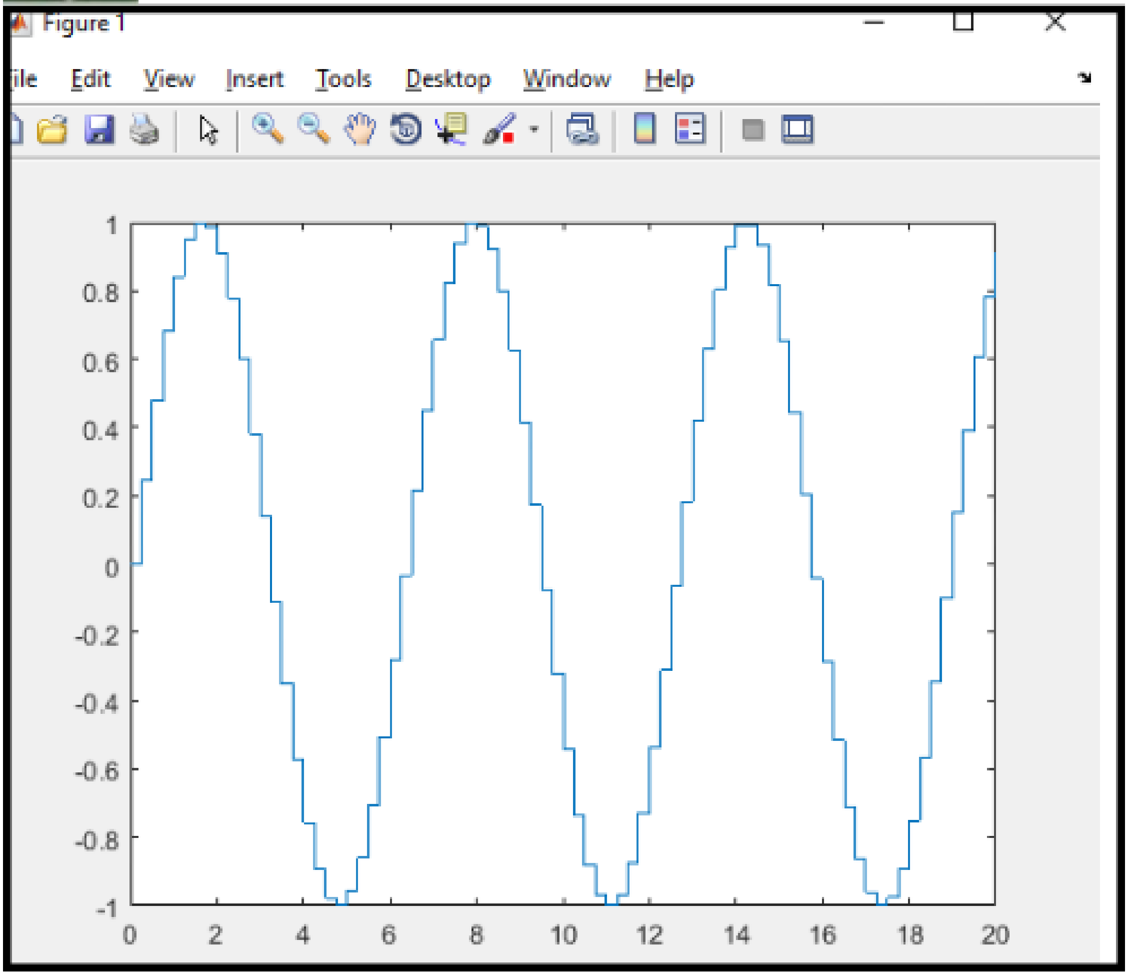
Stairs

Stairstep plots are useful for drawing time-history plots of digitally sampled data systems. stairs(Y) draws a stairstep graph of the elements of Y, drawing one line per column formatrices. When Y is a vector, the *x*-axis scale ranges from 1 to length(Y).

Editor - D:\university\numerical Analysis Lab\gr1.m

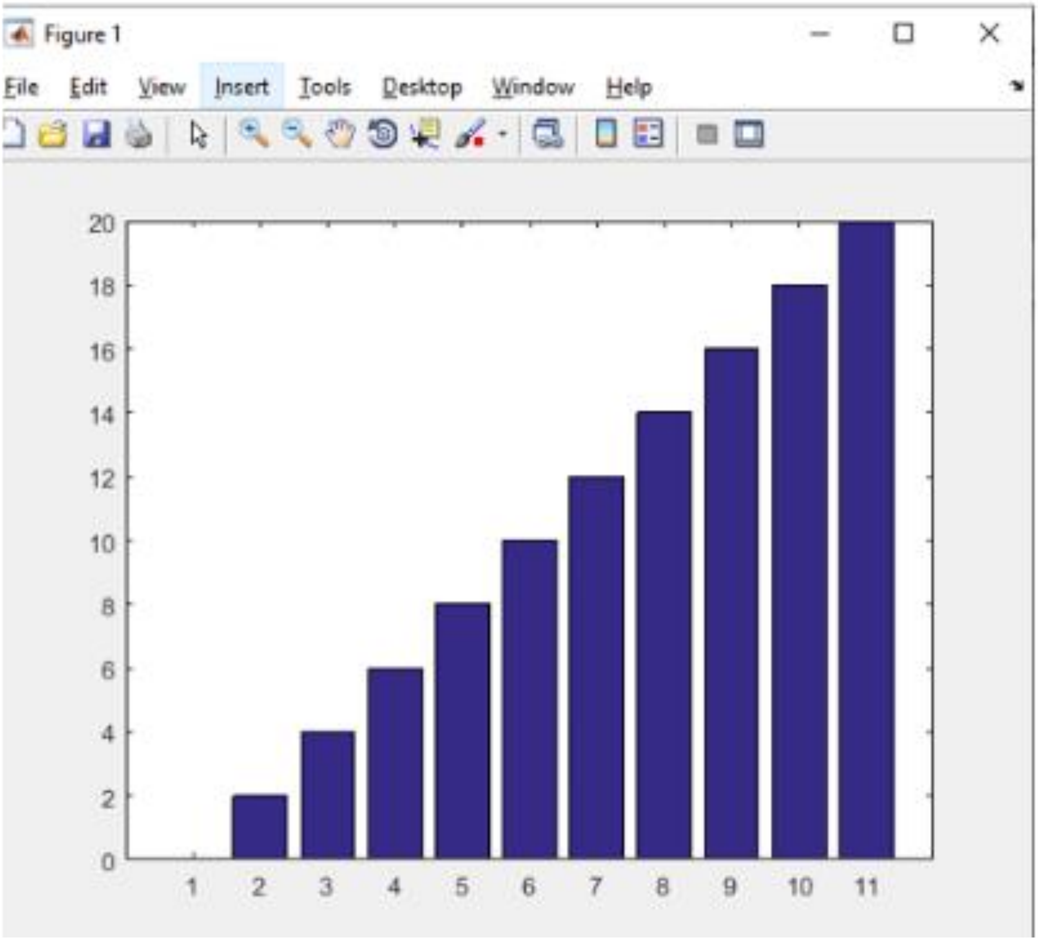
graph1.m gr1.m

```
clc
clear all
x=0:0.25:20;
stairs(x,sin(x));
```



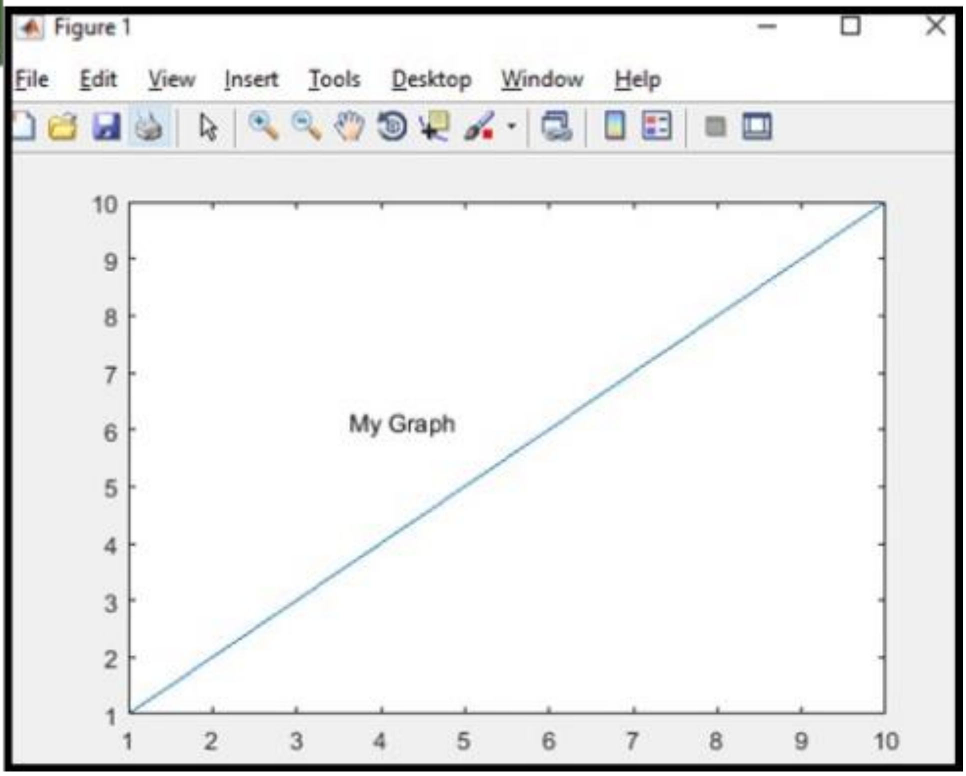
Bar

Bar plot is a simple visual representation of data in the form of multiple bars. Higher the value, higher is the length of the bar. These bars can take both positive and negative values as per our data.



Getext

Using gtext add text to figure using mouse



Figure

For creating the figure object, MATLAB creates a separate window. The characteristics of thisnew window can be controlled using figure properties specified as arguments

Example of stem and figure:

```
figure(1)
y=[0 1 2 4 2 1 0];
subplot(2,2,1);
stem(y)
subplot(2,2,2);
plot(y)

figure(2)
y=[0 1 2 4 2 1 0];
stem(y)
plot(y)
```

Graph

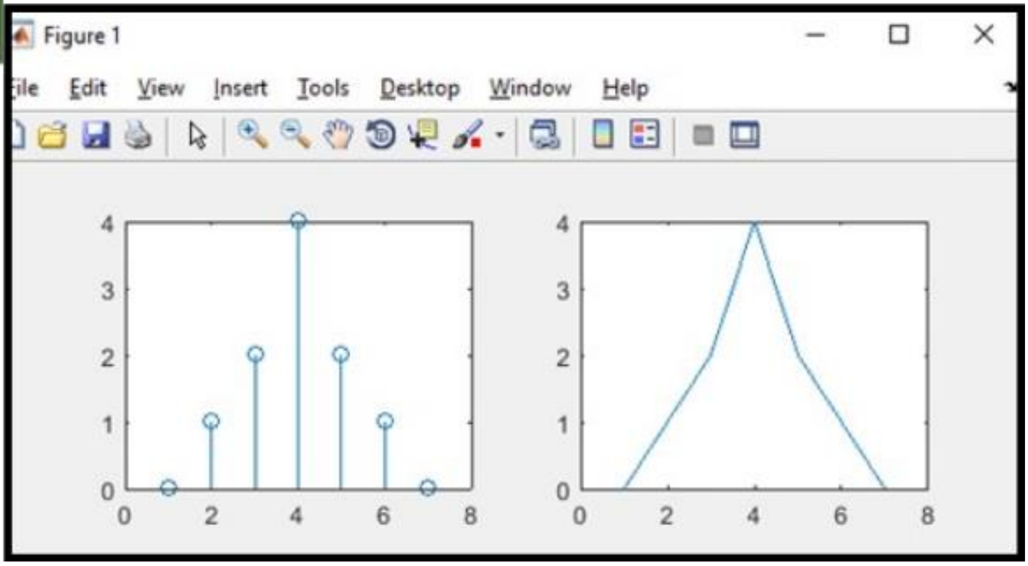


Figure 1
File Edit View Insert Tools Desktop Window Help
0 1 2 3 4
0 2 4 6 8
0 1 2 3 4
0 2 4 6 8

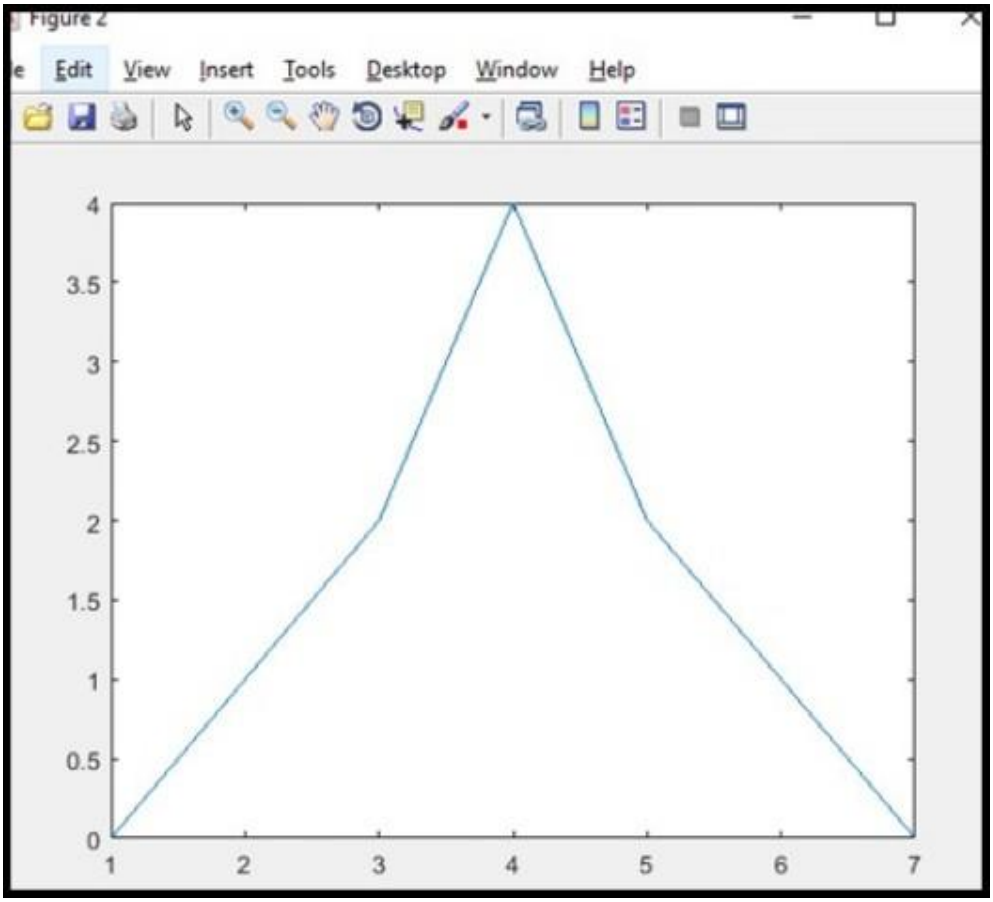


Figure 2
Edit View Insert Tools Desktop Window Help
0 0.5 1 1.5 2 2.5 3 3.5 4
1 2 3 4 5 6 7

M-File functions

M-file functions are programs (or *routines*) that accept *input* arguments and return output arguments.

The first line of M-File function starts with the keyword **function**. It gives the function name and order of arguments.

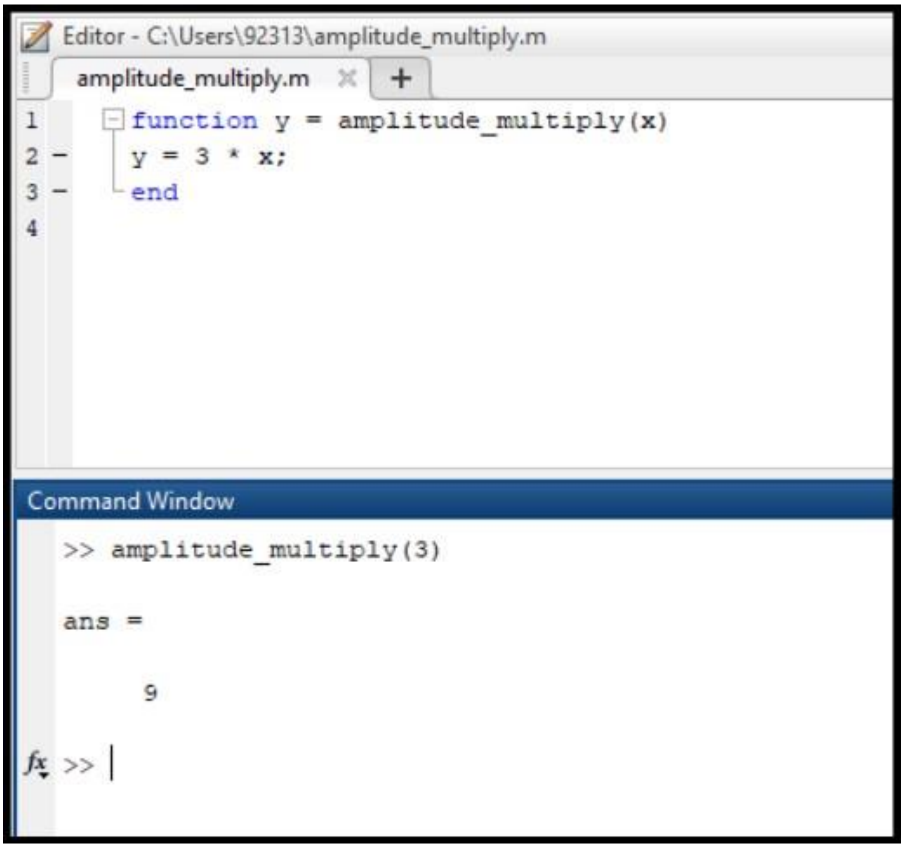
function [outputs] = function_name(inputs)

Note:** Name of the text file that you save will consist of the function name with the extension .m* ***Example 1:

function y =

amplitude_multiply(x) y = 3 * x;

end

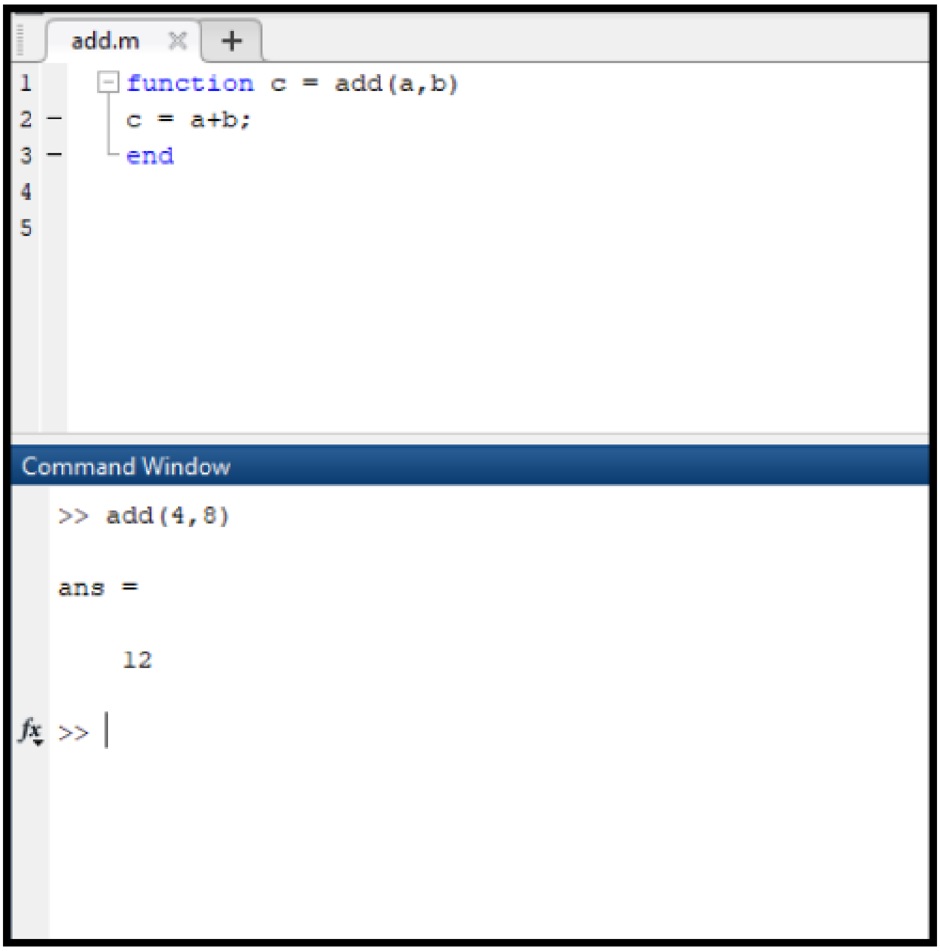


Example 2:

function c =

add(a,b) c = a+b;

end



Anything written after the function line and before the first instruction of function definition, becomes function's help. With this you can use a function which is defined by you in the same way as you use a built in function.

As a function may have more than one output. These are enclosed in square brackets. For instance in example 3 , the function ***circle*** computes the area A and the circumference C of a circle, given its radius as an input argument.

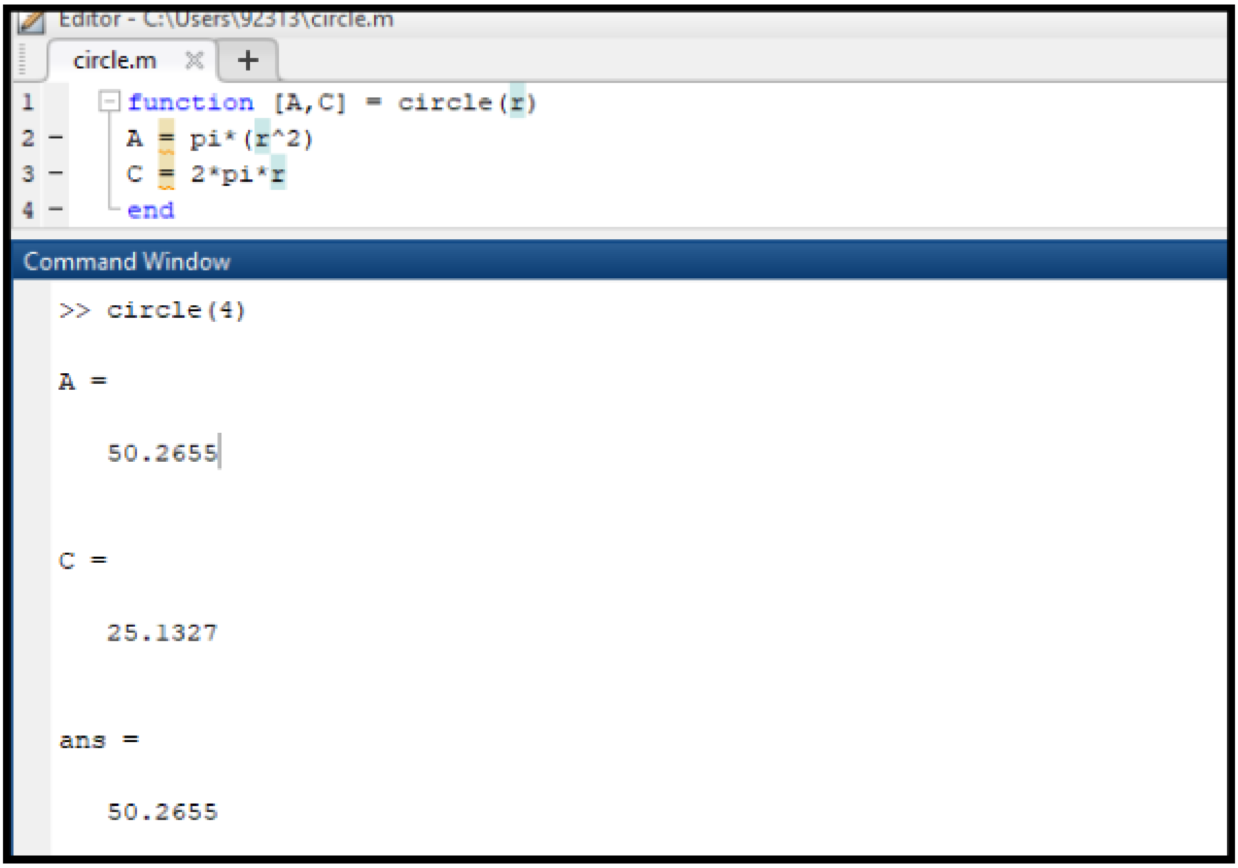
Example:

function [A,C] = circle(r)

% The function (circle) is used to calculate the area and circumference

of a circle A = pi*(r^2);

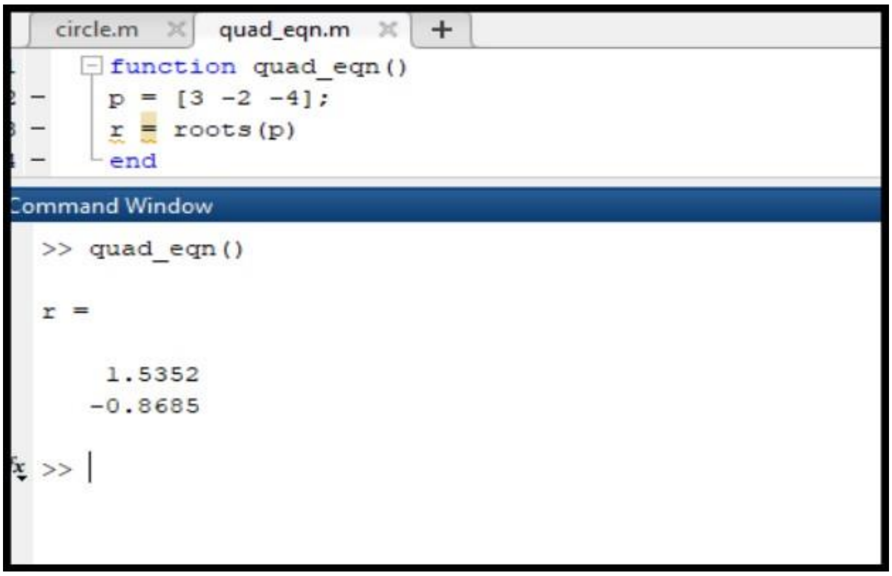
C = 2*pi*r; end



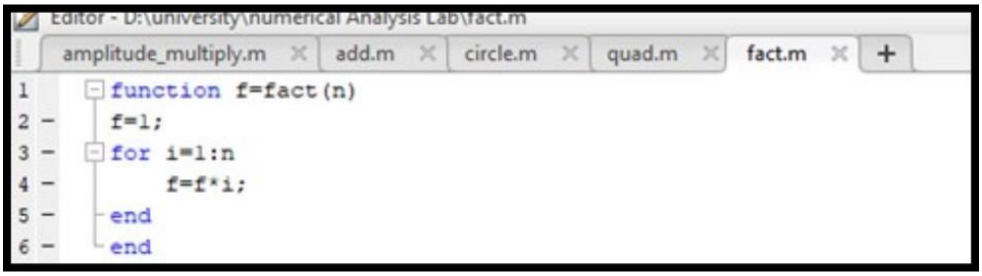
Now move on to the command window and call the circle function as you call other built in functions. Find the Area and Circumference of a circle of radius r=4. Make sure you understand what happens when you omit an output argument.

You can prompt your own error messages for a specific user defined function with the help of nargin and nargout commands. Nargin is a command which takes care of the number of input arguments of a user defined function whereas nargout deals with the output arguments.

Task (1):Create a function that finds the roots of a quadratic equation. Also write a help document of few lines.



Task (2): Create a function that can take three numbers as input and give you one output. This function can perform addition, subtraction and multiplication of these three numbers.



```
Editor - D:\university\numerical Analysis Lab\fluids.m
amplitude_multiply.m | add.m | circle.m | quad.m | fact.m | fluids.m

function f=fluids(x,n)
c=1;
f=c;
for i=1:n
    c=1/i;
    f=f+c*(x^i);
end
end
```

```
Command Window
>> fluids(2,2)

ans =

     5

>>
```

<u>Task (3)</u>: Create a function that can calculate CGPA of your previous semester. You need to take the GPA of your five courses.

```
Editor - D:\university\numerical Analysis Lab\cgpa.m
amplitude_multiply.m | add.m | circle.m | quad.m | fact.m | fluid

function g=cgpa(g1,g2,g3,g4,g5)
g=(g1+g2+g3+g4+g5)/5;
end
```

```
>> cgpa(3.7,3.8,3.3,3.8,3.9)

ans =

    3.7000

>>
```

Tasks

Lab#4

Task#1 (Without Function)

Apply Newton Raphson method in Matlab and show iterations and roots in output.

Code:

Editor - D:\Untitled3.m

Untitled3.m

```
clc
clear all
syms x
eq=input('Enter quation:','s');
f =str2func(['@(x)' eq]);
d=diff(f,x);
x(1)=input('Enter x0:');
dp=input('Enter upto which decimal point you want to find root:');
e=10^(-dp);

for i=1:200
    x(i+1)=x(i)-f(x(i))/subs(d,x(i));
    disp(double(x(i+1)));
    if(abs(f(x(i+1)))<e)
        root=x(i+1);
       break;
    end

end
disp('roots:')
disp(double(x(i+1)));
```

Command Prompt:

Command Window

```
Enter quation:x.^2-2*x-2
Enter x0:2
Enter upto which decimal point you want to find root:4
     3

    2.7500

    2.7321

    2.7321

roots:
    2.7321
```

Task#2 (With Function)

Apply Newton Raphson method in Matlab and show iterations and roots in output.

Code:

Editor - D:\university\numerical Analysis Lab\Newton_Raphson.m

Untitled3.m | bisection.m | Newton_Raphson.m

```
function root=Newton_Raphson(eq)
syms x
%eq=input('Enter quation:','s');
f =str2func(['@(x)' eq]);
d=diff(f,x);
x(1)=input('Enter x0:');
dp=input('Enter upto which decimal point you want to find root:');
e=10^(-dp);

for i=1:200
    x(i+1)=x(i)-f(x(i))/subs(d,x(i));
    disp(double(x(i+1)));
    if(abs(f(x(i+1)))<e)
        root=x(i+1);
       break;
    end

end
disp('roots:')
disp(double(x(i+1)));
```

Command Prompt:

```
>> Newton_Raphson('x.^2-2*x-2')
Enter x0:2
Enter upto which decimal point you want to find root:4
     3

    2.7500

    2.7321

    2.7321

roots:
    2.7321
```

Lab#5

Question: Given a non-linear equation $2^x - 5x + 2 = 0$. Use Newton-Raphson Method to find the root of this equation correct upto 4 decimal places $(\varepsilon = 10^{-4})$

Solution: First we must bracket the root. For this choose random alternative integers and compute function values at those integers.

x	-2	-1	0	1	2
$f(x)$	12.5	7.5	3	-1	-4
Signs of $f(x)$	+	+	+	-	-

Also Compute derivative of given function

$$f'(x) = 2^x\left(\ln\left(2\right)\right) - 5$$

Now start newton's method by taking initial guess either **0 or 1**

Task 1

```
f = @(x) 2.^x - 5*x + 2;
df = @(x)log(2)*(2^x)-5;
e=10^-4;
x0 = 0;
n = 10;
if df(x0)~=0
for i=1:10
    x1=x0-(f(x0)/df(x0))
fprintf('p%d =%0.4f\n',i, x0);
x0=x1;
end
else
    fprintf('root not found')
end
```

```
x1 =

    0.6966

p1 =0.0000

x1 =

    0.7321

p2 =0.6966

x1 =

    0.7322

p3 =0.7321

x1 =

    0.7322

p4 =0.7322

x1 =

    0.7322
```

Command Window

```
p5 =0.7322

x1 =

    0.7322

p6 =0.7322

x1 =

    0.7322

p7 =0.7322

x1 =

    0.7322

p8 =0.7322

x1 =

    0.7322

p9 =0.7322

x1 =
```

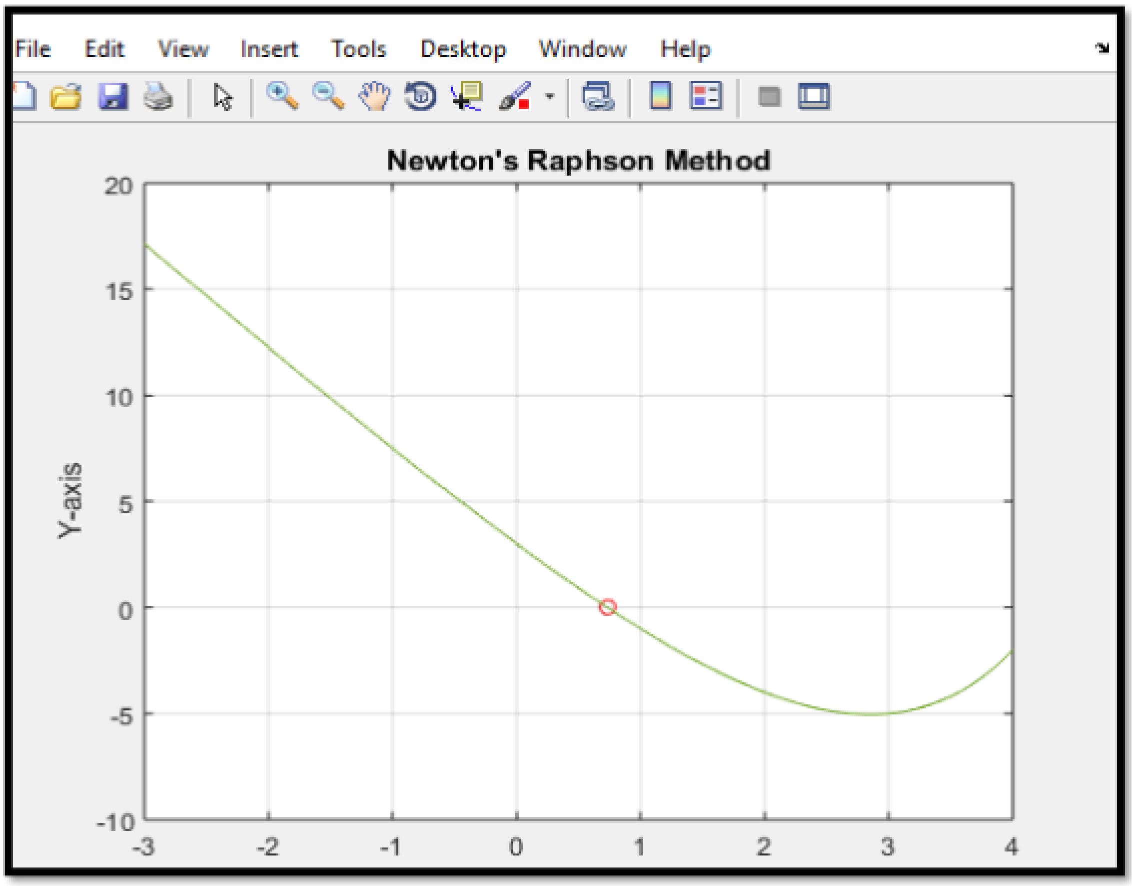
Task 2

Untitled.m Untitled2.m

```
f = @(x) 2.^x - 5*x + 2;
df = @(x)log(2)*(2^x)-5;
e=10^-4;
x0 = 0;
n = 10;
if df(x0)~=0
for i=1:10
    x1=x0-(f(x0)/df(x0))
fprintf('p%d =%0.4f\n',i, x0);
x0=x1;
if abs (f(x0))<e
    break;
end
end
fprintf('Root found at x = %0.4f\n', x0);
    hold on
    plot(x0, f(x0), 'ro')
else
    fprintf('root not found')
end
fplot(@(x)2.^x - 5*x + 2 ,[-3,4])
grid on
```

```
>> Untitled2

x1 =

    0.6966

p1 =0.0000

x1 =

    0.7321

p2 =0.6966

x1 =

    0.7322

p3 =0.7321
Root found at x = 0.7322
```

Task 3



“LAB #6”

TASK #1:

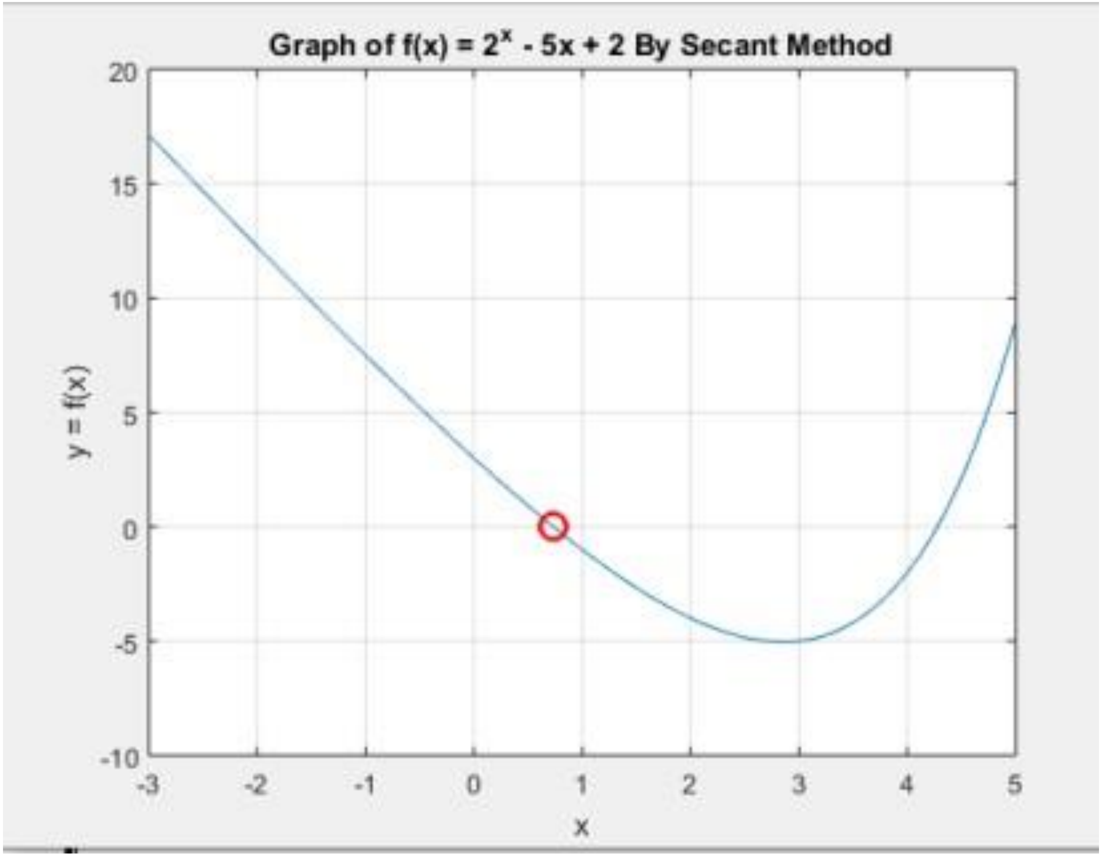
Given a non-linear equation 2^x-5x+2=0. Use secant method to find root of this equation correct upto 4 decimal places (e=10^-4).

```
clc;
clear all;
f = @(x) 2.^x - 5*x + 2;
e = 10^-4;
x0 = 0;
x1 = 1;
n = 10;
if f(x0)*f(x1)<0
    for i=1:n
        df = (f(x1) - f(x0)) / (x1 - x0);
        x2 = x1 - f(x1) / df;
        fprintf('x%d = %0.4f\n', i, x2)
        if abs(f(x2)) < e
            break
        end
        x0 = x1;
        x1 = x2;
    end
    fprintf('Root found at x = %0.4f\n', x2)
else
    fprintf('No sign change detected, cannot use Secant method\n')
end
```

```
        x1 = x2;
    end
    fprintf('Root found at x = %0.4f\n', x2)
else
    fprintf('No sign change detected, cannot use Secant method\n')
end
f = @(x) 2.^x - 5*x + 2;
fplot(f,[-3,5])
grid on
title('Graph of f(x) = 2^x - 5x + 2 By Secant Method')
xlabel('x')
ylabel('y = f(x)')
grid on
hold on
plot(x2,f(x2),'ro', 'MarkerSize', 10, 'LineWidth', 2)
```

Command Window

```
x1 = 0.7500
x2 = 0.7317
x3 = 0.7322
Root found at x = 0.7322
>>
```



"LAB #7,8"

REGULA-FALSI METHOD:

regulaFalsi.m

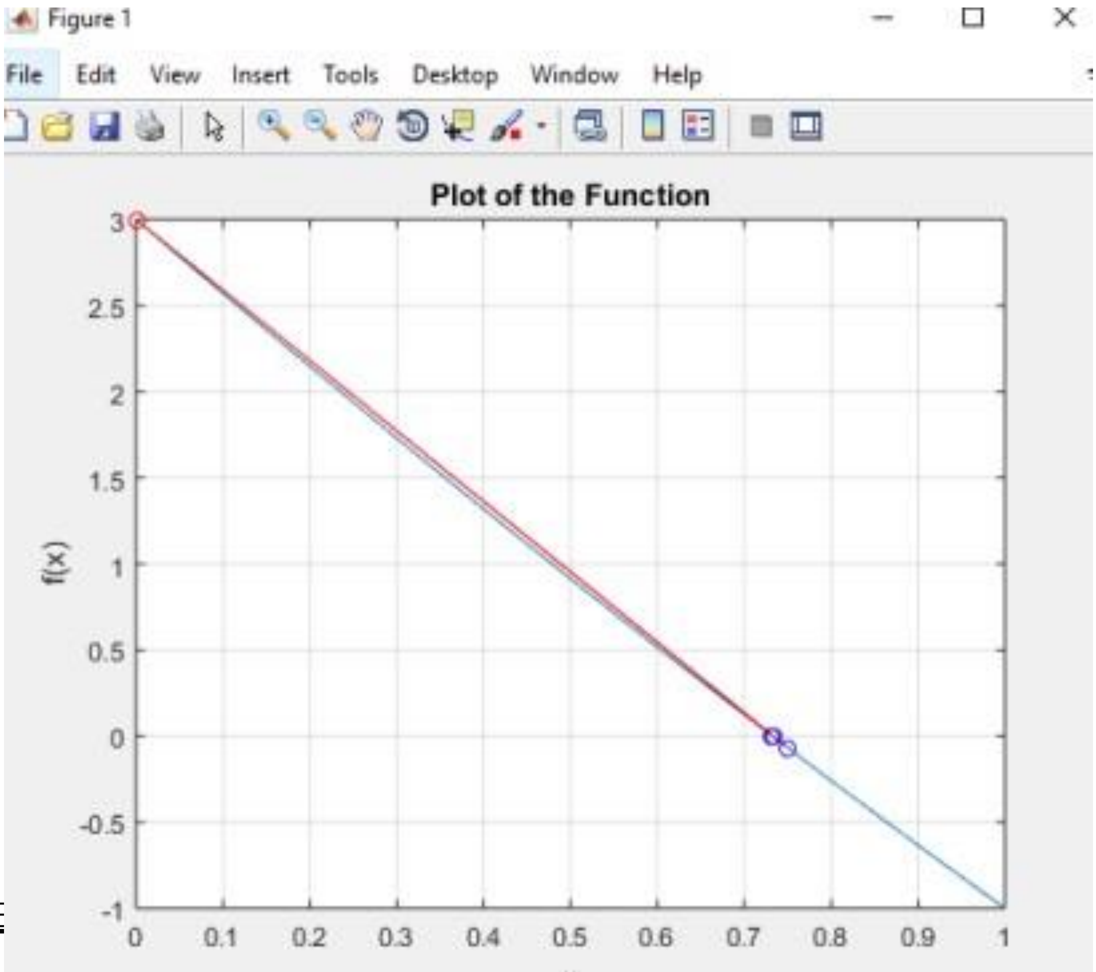
```
f = input('Enter your Function:');
a = input('Enter Left side of your root guess: ');
b = input('Enter right side of your root guess: ');
n = input('Enter the number of iterations you want: ');
e = input('Enter your desired tolerance: ');
x = linspace(a, b, 1000);
y = f(x);

figure;
plot(x, y);
hold on;
grid on;
xlabel('x');
ylabel('f(x)');
title('Plot of the Function');

if f(a)*f(b) < 0 && a < b
    for i = 1:n
        c = (a*f(b) - b*f(a)) / (f(b) - f(a));

        if abs(f(c)) < e
            break;
        elseif f(a)*f(c) < 0
            b = c;
        else
            a = c;
```

Editor - C:\Users\92313\regulaFalsi.m

regulaFalsi.m

```
    for i = 1:n
        c = (a*f(b) - b*f(a)) / (f(b) - f(a));

        if abs(f(c)) < e
            break;
        elseif f(a)*f(c) < 0
            b = c;
        else
            a = c;
        end
        plot([a, b], [f(a), f(b)], 'ro-');
        plot(c, f(c), 'bo');
        drawnow;

        fprintf('Iteration %d: Left = %.4f, Right = %.4f\n', i, a, b);
    end

    fprintf('Root of the equation: %.4f\n', c);
else
    disp('No root between the given brackets');
end
```

```
Command Window
>> regulaFalsi
Enter function:@(x) 2.^x-5*x+2
Enter Left side of your root guess: 0
Enter right side of your root guess: 1
Enter the number of iterations you want: 10
Enter your desired tolerance: 10^-4
Iteration 1: Left = 0.0000, Right = 0.7500
Iteration 2: Left = 0.0000, Right = 0.7333
Iteration 3: Left = 0.0000, Right = 0.7323
Root of the equation: 0.7322
```



FIXE

Editor - D:\university\numerical Analysis Lab\fixedpoint.m

fixedpoint.m

```
f=input('Enter the function:');
x0=input('Enter the initial value:');
n=input('Enter the number of interations');
e=10^-4;
for i=1:n;
    x1=f(x0);
    fprintf('p%d=%0.4f\n',i,x1);
    if abs(x1-x0)<e
        break;
    end;
    x0=x1;

end;
i=-4:0.1:4;
grid on;
hold on;
plot(i,f(i));
plot(x1,f(x1),'r*');
```

```
p12=0.5676
p13=0.5669
p14=0.5673
p15=0.5671
p16=0.5672
p17=0.5671
fx >>
```

<u>“LAB 9&10”</u>

TASK 1:

Question: Solve the following system of linear equations using Cramer's rule

$$10x_1 + 3x_2 + 1x_3 = 19$$
$$3x_1 + 10x_2 + 2x_3 = 29$$
$$1x_1 + 2x_2 + 10x_3 = 35$$

▸ C: ▸ Windows ▸ system32 ▸

Editor - C:\Users\92313\run12.m

```
A = input('Enter coefficient matrix: ');
B = input('Enter source matrix: ');

N = length(B);
X = zeros(N, 1);

det_A = det(A);

for i = 1:N
    temp_A = A;
    temp_A(:, i) = B;
    det_temp_A = det(temp_A);
    X(i) = det_temp_A / det_A;
end

disp('Solution:');
disp(X);

```

Command Window

```
>> run12
Enter coefficient matrix: [10 3 1;3 10 2;1 2 10]
Enter source matrix: [19;29;35]
Solution:
    1.0000
    2.0000
    3.0000

fx >>
```

2. Gauss-Elimination Method:

To understand gauss-elimination method let's take an example of 3x3 system.

$$10x_1 + 3x_2 + 1x_3 = 19$$
$$3x_1 + 10x_2 + 2x_3 = 29$$
$$1x_1 + 2x_2 + 10x_3 = 35$$

I'm

lab10.m

```matlab
A = input('Enter coefficient matrix: ');
B = input('Enter source matrix: ');

% Check if dimensions are consistent
if size(A, 1) ~= size(B, 1)
    error('Dimensions of matrices A and B are not consistent.');
end

% Create augmented matrix
augmented_matrix = [A, B];

N = size(A, 1); % Number of rows

% Forward elimination
for k = 1:N-1
    for i = k+1:N
        factor = augmented_matrix(i,k) / augmented_matrix(k,k);
        augmented_matrix(i,:) = augmented_matrix(i,:) - factor * augmented_matrix(k,:);
    end
end

% Back substitution
X = zeros(N, 1);
X(N) = augmented_matrix(N, N+1) / augmented_matrix(N, N);
for k = N-1:-1:1
```

Command Window

```
>> lab10
Enter coefficient matrix: [10 3 1; 3 10 2; 1 2 10]
Enter source matrix: [19; 29; 35]
Augmented Matrix:
   10.0000    3.0000    1.0000   19.0000
         0    9.1000    1.7000   23.3000
         0         0    9.5824   28.7473

Solution:
    1.0000
    2.0000
    3.0000

fx >>
```

"Lab 11"
(Gauss Jordan)

Code:

Editor - C:\Users\92313\lab13.m

lab13.m

```
A=input('Enter coefficient MAtrix');
B=input('Source Matrix');
N=length(B);
x= zeros(N,1);
Aug=[A B]
for j=1:N
    Aug(j,:)=Aug(j,:)/Aug(j,j)
    for i=1:N
if i~=j
    m=Aug(i,j)
    Aug(i,:)=Aug(i,:)- m*Aug(j,:)
end
    end
end
```

```
Aug =

     1     2     2
     0    -1     0


Aug =

     1     2     2
     0     1     0


m =

     2


Aug =

     1     0     2
     0     1     0
```

OUTPUT:(BY 3 MATRIX)

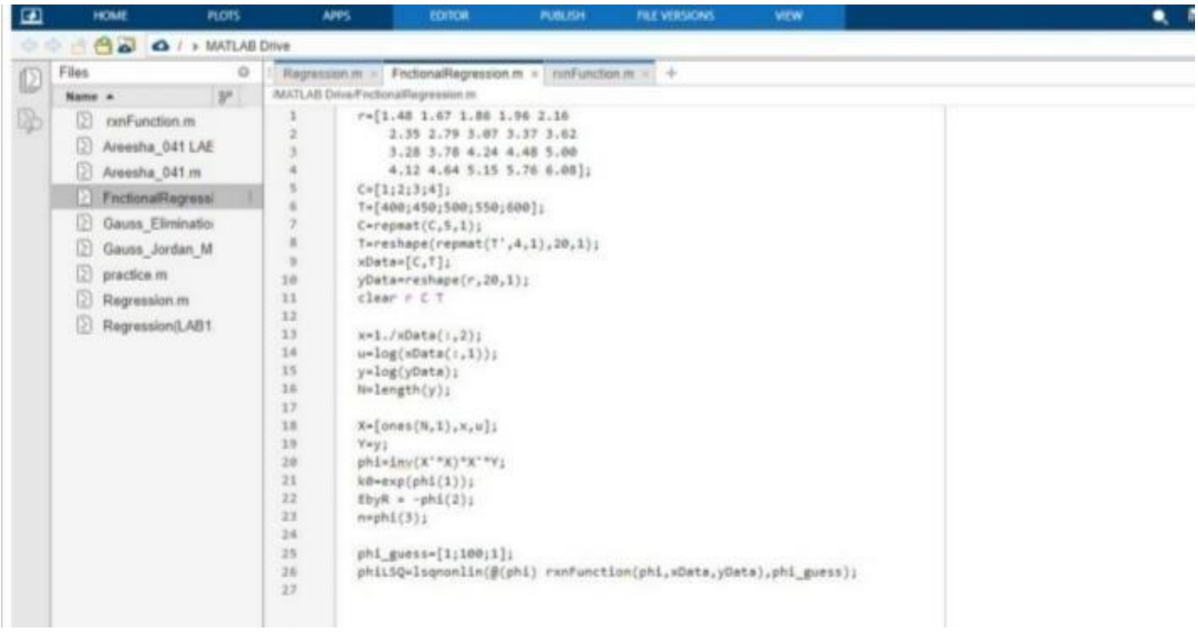
```
>> lab13
Enter coefficient MAtrix[10 3 1;3 10 2;1 2 10]
Source Matrix[19;29;35]

Aug =

    10     3     1    19
     3    10     2    29
     1     2    10    35


Aug =

    1.0000    0.3000    0.1000    1.9000
    3.0000   10.0000    2.0000   29.0000
    1.0000    2.0000   10.0000   35.0000


m =

     3
```

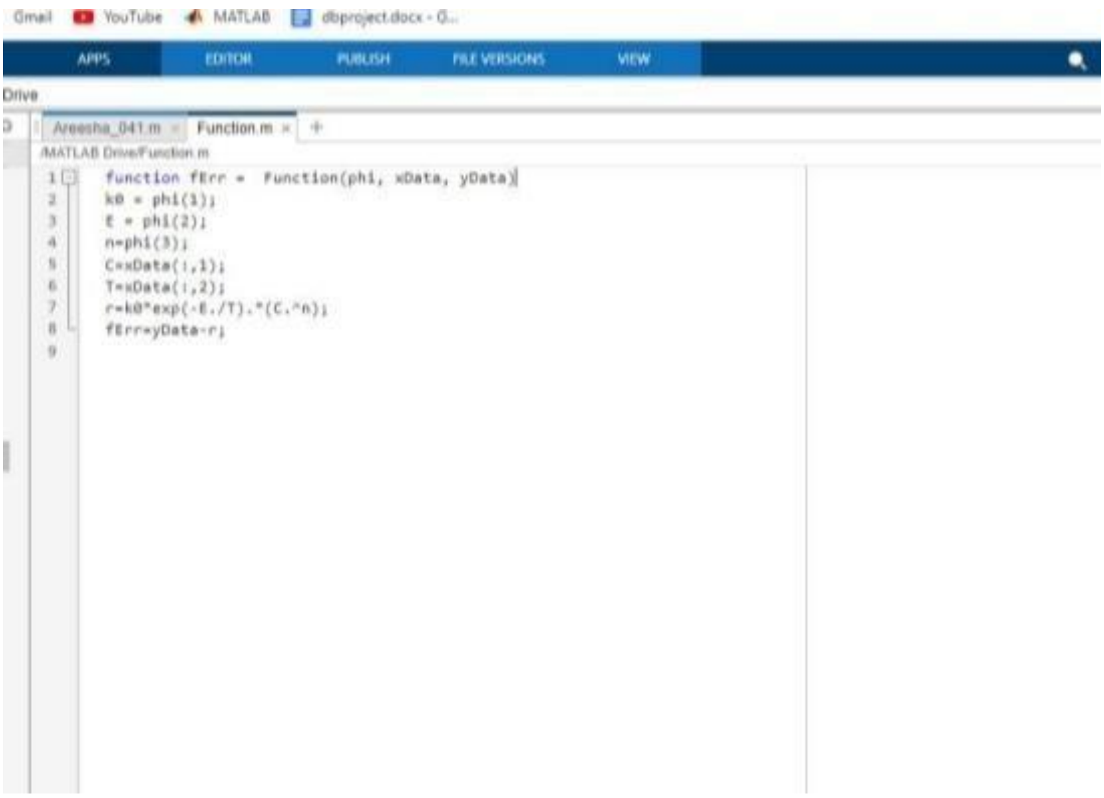
```
Aug =

    1.0000    0.3000    0.1000    1.9000
         0    9.1000    1.7000   23.3000
    1.0000    2.0000   10.0000   35.0000


m =

     1


Aug =

    1.0000    0.3000    0.1000    1.9000
         0    9.1000    1.7000   23.3000
         0    1.7000    9.9000   33.1000


Aug =

    1.0000    0.3000    0.1000    1.9000
         0    1.0000    0.1868    2.5604
```

```
m =

    0.3000


Aug =

    1.0000         0    0.0440    1.1319
         0    1.0000    0.1868    2.5604
         0    1.7000    9.9000   33.1000


m =

    1.7000


Aug =

    1.0000         0    0.0440    1.1319
         0    1.0000    0.1868    2.5604
         0         0    9.5824   28.7473


Aug =
```

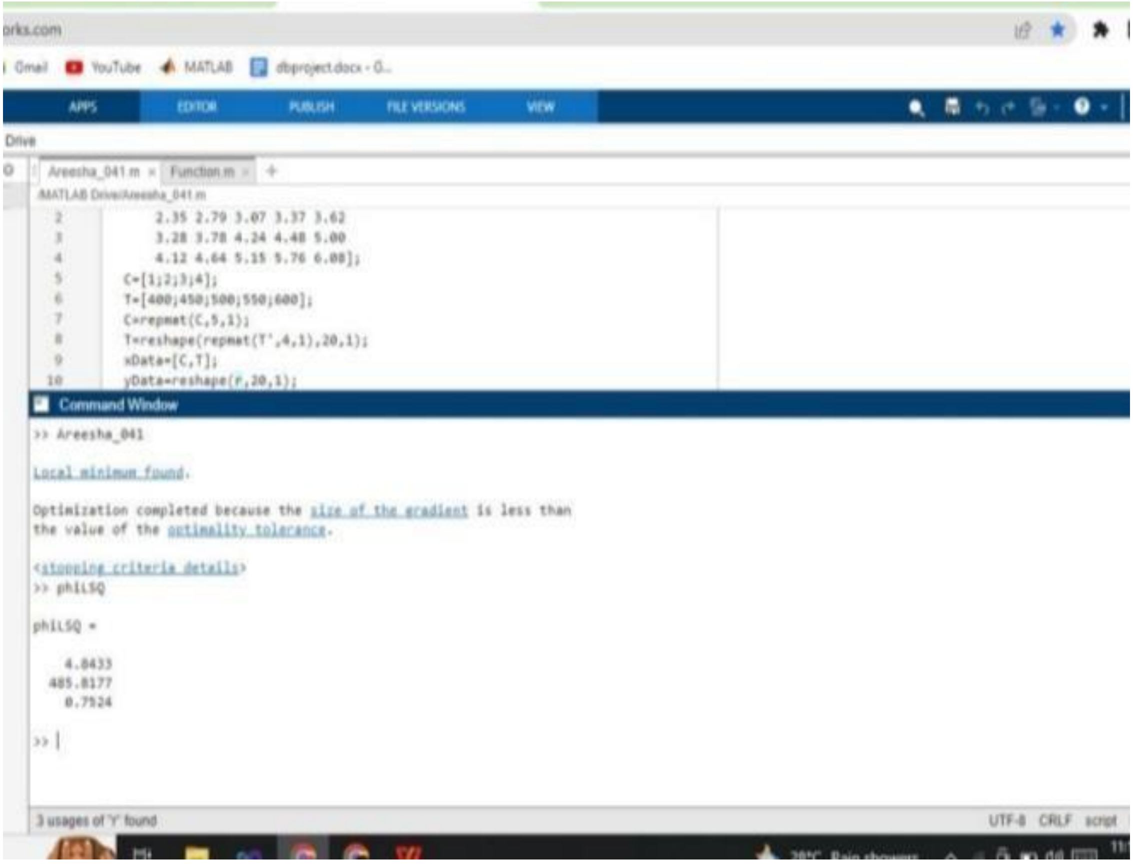
```
m =

    0.0440


Aug =

    1.0000         0         0    1.0000
         0    1.0000    0.1868    2.5604
         0         0    1.0000    3.0000


m =

    0.1868


Aug =

     1     0     0     1
     0     1     0     2
     0     0     1     3

>> 
```

"LAB 13"
Non-Linear & Functional Regression





Output :



orks.com
Gmail
YouTube
MATLAB
dbproject.docx - G...
APPS
EDITOR
PUBLISH
FILE VERSIONS
VIEW
Drive
Areesha_041.m
Function.m
/MATLAB Drive/Areesha_041.m
2.35 2.79 3.07 3.37 3.62
3.28 3.78 4.24 4.48 5.00
4.12 4.64 5.15 5.76 6.08];
C=[1;2;3;4];
T=[400;450;500;550;600];
C=repmat(C,5,1);
T=reshape(repmat(T',4,1),20,1);
xData=[C,T];
yData=reshape(Y,20,1);
Command Window
>> Areesha_041
Local minimum found.
Optimization completed because the size of the gradient is less than
the value of the optimality tolerance.
<stopping criteria details>
>> phiLSQ
phiLSQ =
4.8433
485.8177
0.7524
>>
3 usages of 'Y' found
UTF-8
CRLF
script