MATLAB PROGRAMMING LAB FILE
CSEU4103

Submitted By:
Kanishk Yadav
Course: M. Tech (AIML)
Session: 2024 – 26
Enrollment No: A501144824006

Submitted To:
Dr Abhinaba Sinha
ASET Dept.

INDEX

Practical 1

Aim: a) Introduction to MATLAB

MATLAB, short for “MATrix LABoratory”, is a high-level programming and numerical computing environment used for a wide range of applications in science, engineering, and mathematics. Developed by MathWorks, MATLAB provides a comprehensive set of tools for data analysis, algorithm development, visualization, and the creation of applications.
Some important key features of MATLAB:

1. Interactive Environment: MATLAB offers an interactive environment where you can perform computations, run scripts, and execute commands in a command-line interface. This makes it a popular choice for quick prototyping and experimentation.
2. High-Level Language: MATLAB is built around a high-level scripting language, which is easy to learn and use. It supports various data types, including matrices, arrays, and structures, and provides extensive mathematical and linear algebra functions.
3. Built-in Functions: MATLAB comes with a vast collection of built-in functions and toolboxes for specialized tasks like signal processing, image processing, machine learning, and more. These functions enable you to perform complex tasks without writing extensive code.
4. Data Visualization: MATLAB is known for its powerful data visualization capabilities. You can create 2D and 3D plots, customize graphs, and generate publication-quality figures to represent your data effectively.
5. Numerical Computation: MATLAB excels at numerical computing and linear algebra. It's commonly used for solving complex mathematical equations, performing simulations, and implementing numerical algorithms.
6. Interoperability: MATLAB offers interoperability with other programming languages like C, C++, Java, and Python. You can call functions and exchange data between MATLAB and these languages.
7. Parallel and GPU Computing: MATLAB supports parallel computing and GPU acceleration, allowing you to speed up computationally intensive tasks.
8. Community and Resources: MATLAB has a strong user community, and there are numerous online resources, forums, and documentation to help users get started and solve problems.
9. Education and Research: MATLAB is extensively used in academia for teaching and research in fields like engineering, physics, and data analysis.

MATLAB Home window:-
This tab mainly comprises of the tools that allow us to have a new start such as new script, new live script, import data and many more.

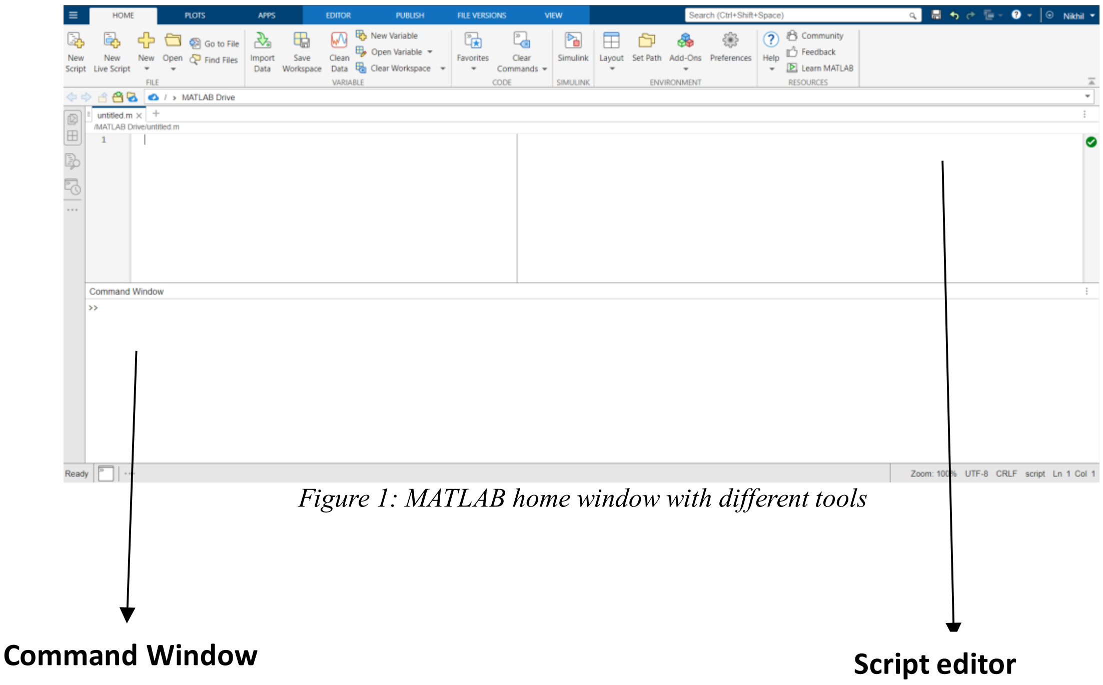


Figure 1: MATLAB home window with different tools

- **Command Window:** Command Window is a fundamental component of the user interface and plays a crucial role in interacting with the software. It serves as an interactive environment where you can enter commands, run scripts, and get immediate feedback.

- **Script editor:-** it is a dedicated environment for creating and editing MATLAB scripts and functions. The Script Editor is where you can write, edit, and save your MATLAB code

MATLAB Plot window:
This tab consists of tools that help us to plot various graphs and with more functions.

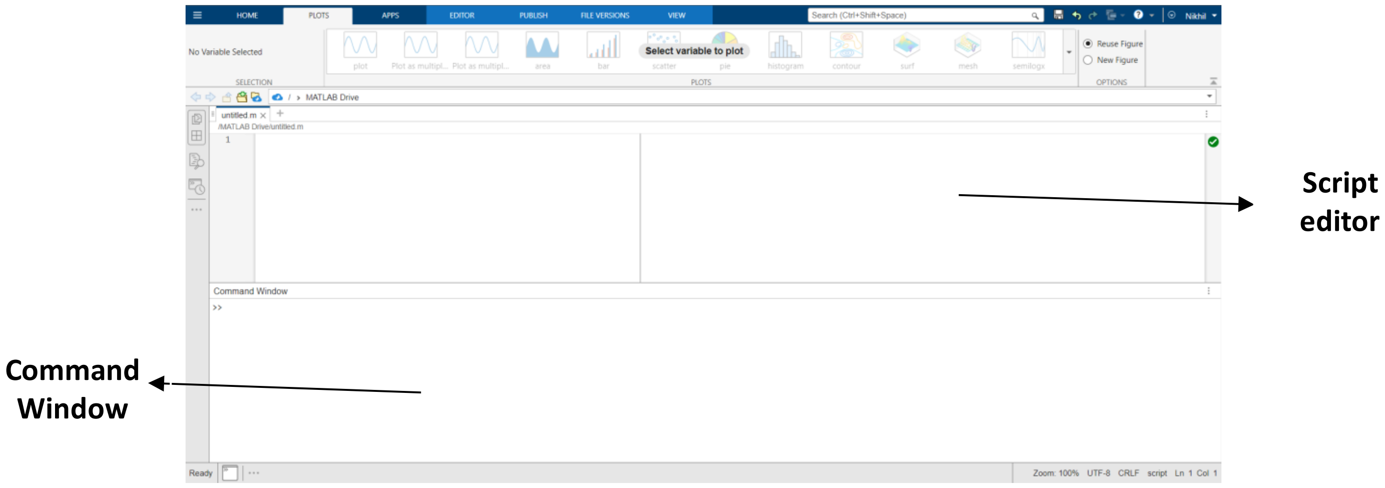


Figure 2: MATLAB plot window with different graph and other tools for data visualization

MATLAB editor window:-

This tab consists of workspace which display variables, commands window which gives the output and the processing and the editor where we declare the variable and perform the task.

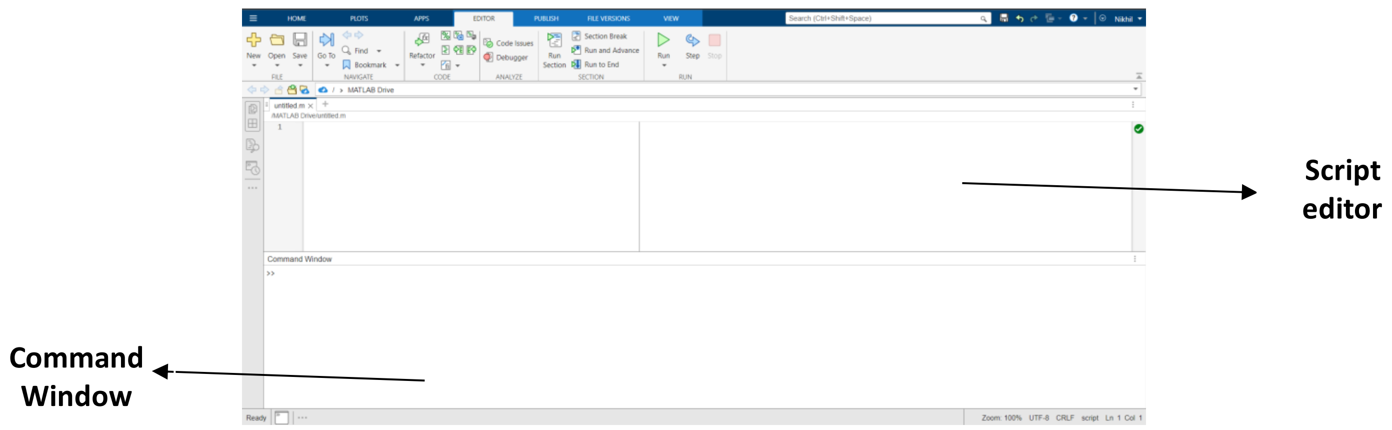


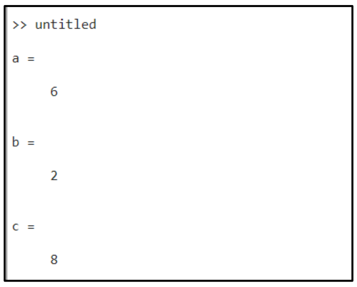
Figure 3: MATLAB Editor window

Aim: b) Basic Operations in MATLAB

1) Define two variables and find their sum, difference, multiplication, division

Addition:

a = 6
b = 2
c = a + b

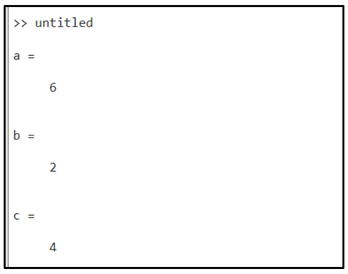


```
>> untitled

a =

     6


b =

     2


c =

     8
```

Figure 4: Addition of 2 numbers

Subtraction:

a = 6
b = 2
c = a - b

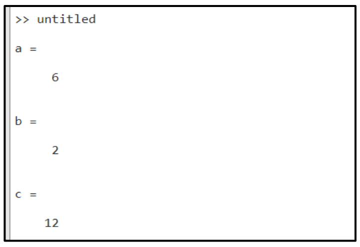


```
>> untitled

a =

     6


b =

     2


c =

     4
```

Figure 5: Subtraction of 2 numbers

Multiplication:

a = 6
b = 2
c = a * b

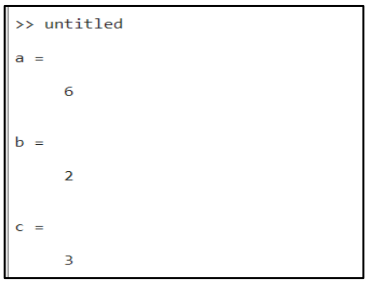


```
>> untitled

a =

     6


b =

     2


c =

    12
```

Figure 6: Multiplication of 2 numbers

Division:

a = 6
b = 2
c = a / b



```
>> untitled

a =

     6


b =

     2


c =

     3
```

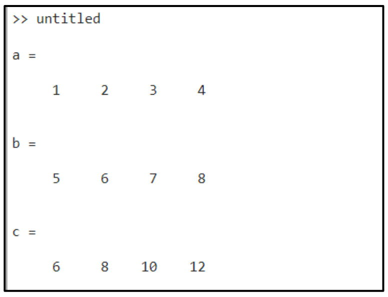
Figure 7: Division of 2 numbers

2) Matrix definition, addition, subtraction, product, division, element wise operations, determinant, inverse and transpose.

A matrix is defined by its dimensions, which are typically expressed as "m x n," where "m" represents the number of rows, and "n" represents the number of columns. Each element in a matrix is denoted by its position, using two subscripts (i, j), where "i" refers to the row number and "j" refers to the column number. These elements can be real numbers, complex numbers, or variables. The collection of these elements is enclosed within square brackets or parentheses.

Addition:

a = [1,2,3,4]
b = [5,6,7,8]
c = a + b

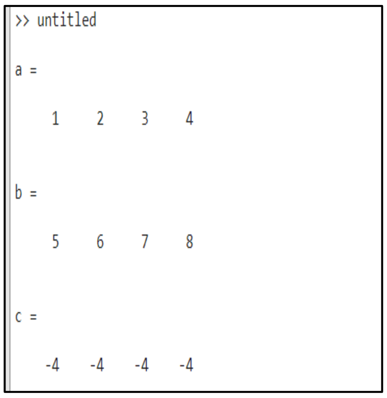


```
>> untitled

a =

     1     2     3     4

b =

     5     6     7     8

c =

     6     8    10    12
```

Figure 8: Addition of 2 Matric

Subtraction:

a = [1,2,3,4]
b = [5,6,7,8]
c = a – b

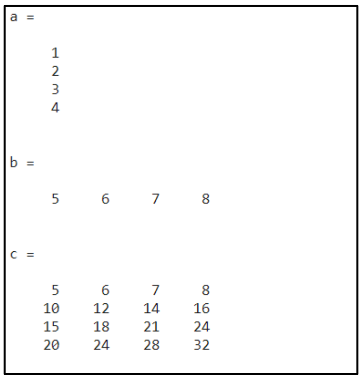


```
>> untitled

a =

     1     2     3     4

b =

     5     6     7     8

c =

    -4    -4    -4    -4
```

Figure 9: Subtraction of 2 numbers

Multiplication:

a = [1
  2
  3
  4]
b = [5,6,7,8]
c = a * b

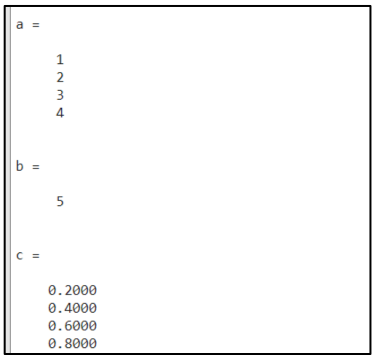


```
a =

     1
     2
     3
     4

b =

     5     6     7     8

c =

     5     6     7     8
    10    12    14    16
    15    18    21    24
    20    24    28    32
```

Figure 10: Mulitplication of 2 matrics

Division of a matric by a number:

a = [1
  2
  3
  4]
b = 5
c = a / b

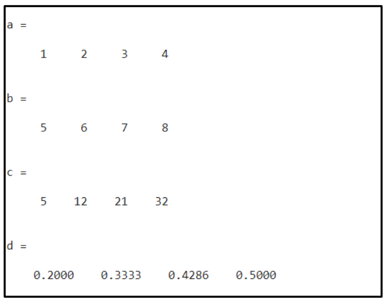


```
a =

     1
     2
     3
     4

b =

     5

c =

    0.2000
    0.4000
    0.6000
    0.8000
```

Figure 11: Division of a matric by a number

Element wise operations:

```
a = [1,2,3,4]
b = [5,6,7,8]
c = a .* b
d = a ./ b
```

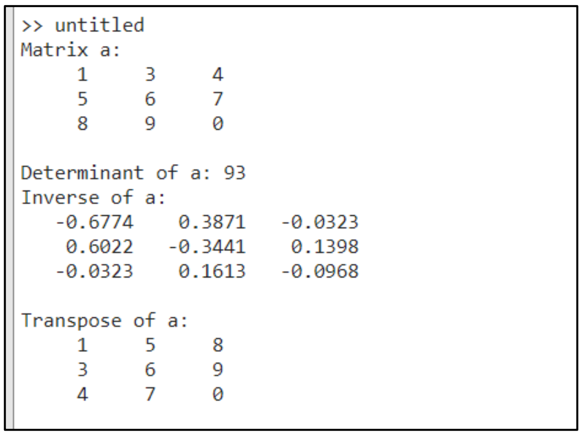


```
a =

    1     2     3     4

b =

    5     6     7     8

c =

    5    12    21    32

d =

   0.2000    0.3333    0.4286    0.5000
```

Figure 12: Element wise operations

Determinant, inverse and transpose:

```
a = [1, 3, 4;
     5, 6, 7;
     8, 9, 0];
b = det(a); %det = determinant
c = inv(a); %inv = inverse
d = transpose(a);%transpose = transpose
disp('Matrix a:');%disp = display
disp(a);
disp(['Determinant of a: ' num2str(b)]);
disp('Inverse of a:');
disp(c);
disp('Transpose of a:');
disp(d);
```



```
>> untitled
Matrix a:
     1     3     4
     5     6     7
     8     9     0

Determinant of a: 93
Inverse of a:
   -0.6774    0.3871   -0.0323
    0.6022   -0.3441    0.1398
   -0.0323    0.1613   -0.0968

Transpose of a:
     1     5     8
     3     6     9
     4     7     0
```

Figure 13: determinant, inverse and transpose

Practical No. 2

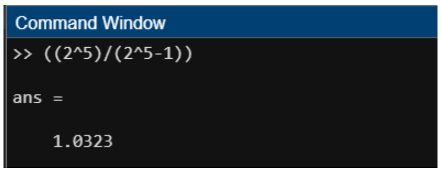
Aim: Implementation of MATLAB program for various defining different mathematical functions.

Define the following mathematical expressions using command window:

1. $2^5/2^5\text{-}1$ = 1.0333
2. $3\{\frac{5^{\frac{1}{2}}-1}{(5^{\frac{1}{2}}-1)2}\}$ = 4.7447
3. e^3 = 20.0855
4. $\sin\frac{\pi}{6}$ = 0.5000
5. $\cos\frac{\pi}{6}$ = 0.8600
6. $\log10_{e^3}$ = 1.3029
7. $\log10_{10^5}$ = 5
8. $\tan\frac{\pi}{2}$ = 1.6331e + 16
9. $\sin(\frac{\pi}{2})^2+\cos(\frac{\pi}{2})^2$ = -0.2152
10. $\pi^{\frac{1}{3}}-1$ = 0.0472
11. $\pi(\pi^{\frac{1}{3}}-1)^2$ = 2.1413
12. $(\cosh 32\pi)^2+(\sinh 32\pi)^2$ 1.0000 + 0.0000i
13. $\frac{1+3i}{1-3i}$ = 0.4000
14. $\frac{\pi i}{e^4}$ = - 0.02888
15. $\frac{\pi}{e^{4i}}$ = 23.2134
16. lne^3 = 3

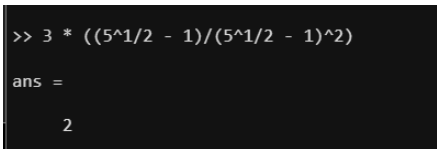
Solution:

((2^5) / (2^5 – 1))



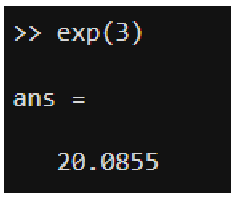
```
Command Window
>> ((2^5)/(2^5-1))

ans =

    1.0323
```

3*((5^1/2 – 1) / (5^1/2 – 1) ^2)



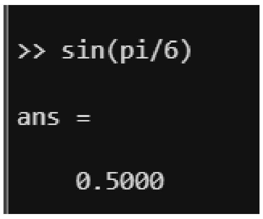
```
>> 3 * ((5^1/2 - 1)/(5^1/2 - 1)^2)

ans =

     2
```

exp(3)



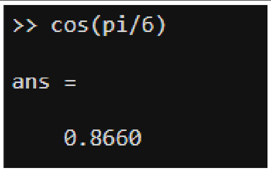
```
>> exp(3)

ans =

   20.0855
```

sin(pi/6)



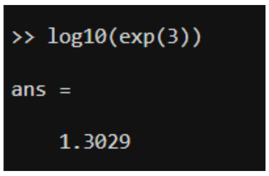
```
>> sin(pi/6)

ans =

    0.5000
```

cos(pi/6)



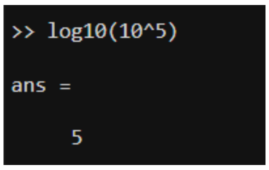
```
>> cos(pi/6)

ans =

    0.8660
```

log10(exp(3))



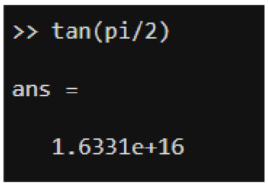
```
>> log10(exp(3))

ans =

    1.3029
```

log10(exp(3))



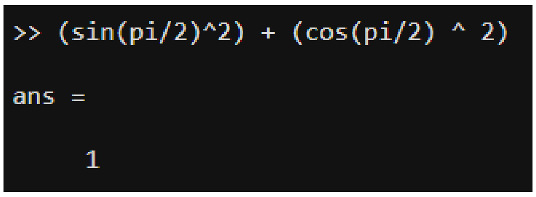
```
>> log10(10^5)

ans =

     5
```

tan(pi/2)



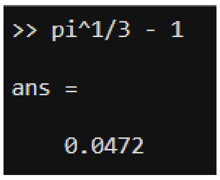
```
>> tan(pi/2)

ans =

   1.6331e+16
```

(sin(pi/2) ^ 2) + (cos(pi/2) ^ 2)



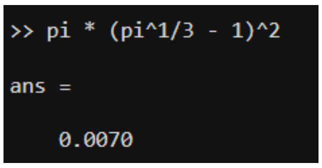
```
>> (sin(pi/2)^2) + (cos(pi/2) ^ 2)

ans =

     1
```

pi^1/3 – 1



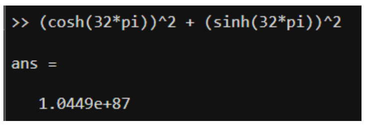
```
>> pi^1/3 - 1

ans =

    0.0472
```

pi * (pi^1/3 - 1) ^ 2



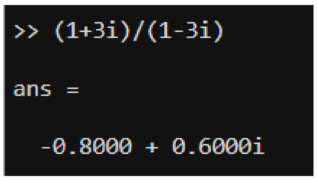
```
>> pi * (pi^1/3 - 1)^2

ans =

    0.0070
```

(cosh(32*pi))^2 + (sinh(32*pi))^2



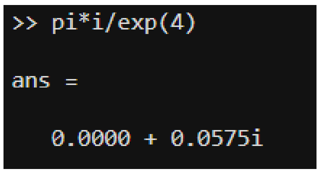
```
>> (cosh(32*pi))^2 + (sinh(32*pi))^2

ans =

   1.0449e+87
```

(1+3i)/(1-3i)



```
>> (1+3i)/(1-3i)

ans =

  -0.8000 + 0.6000i
```

pi*i/exp(4)



```
>> pi*i/exp(4)

ans =

   0.0000 + 0.0575i
```

Practical No. 3

Aim: Introduction to plots using MATLAB programming

1. **Plot and label a circle of unit radius.**

```
r  = 4;
theta = linspace(0,2*pi,100);
x = r*cos(theta);
y = r*sin(theta);
plot(x,y);
axis('equal');
title('circle of radius r = ',num2str(r))
```

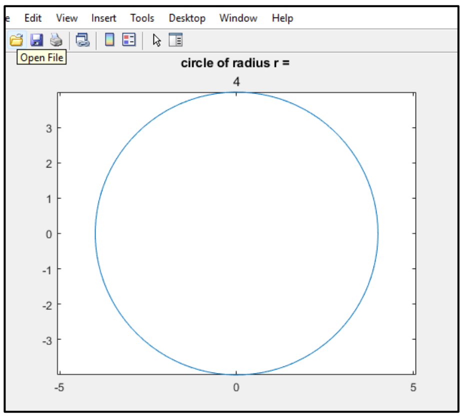


Figure 14: A circle of unit radius

2. **Plot y=sin (x), graph for 0<=x<=2pi having interval of 100 linearly spaced points in the given interval. Label the axes and give suitable tile.**

```
theta = linspace(0,2*pi,100)
x = sin(theta);
plot(x);
xlabel("X_Axis");
ylabel("Y_Axis");
title("Sin Graph")
```

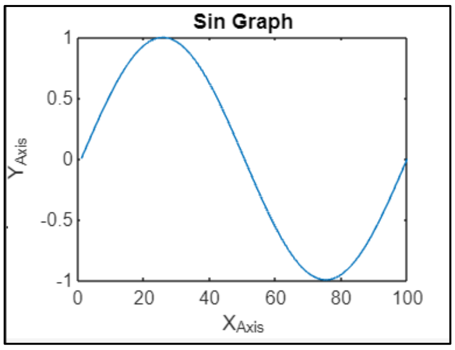


Figure 15: Sine graph

3. **On the same plot of ques 2, plot and label sin (x^2) graph for 0<=x<=2pi having interval of 100 linearly spaced points in the given interval. Label the axes and give suitable tile. Use + marker and red color to plot this line.**

```
x = linspace(0, 2*pi, 100);
y1 = sin(x);
y2 = sin(x.^2);
figure;
plot(x, y1, 'b-', 'LineWidth', 2);
hold on;
plot(x, y2, 'r+', 'MarkerSize', 6);
xlabel('x');
ylabel('y');
title('Plot of y = sin(x) and y = sin(x^2)');
legend('y = sin(x)', 'y = sin(x^2)');
grid on;
xlim([0, 2*pi]);
ylim([-1.2, 1.2]);
hold off;
```

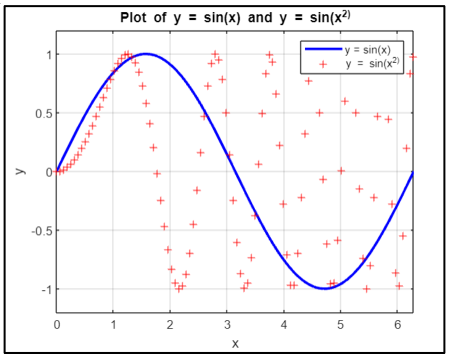


Figure 16: Plot of sinx and sinx^2

4. **Plot and label e $^{-0.4x}$ sin(x) graph for 0<=x<=4pi having interval of 10 and 100 points.**

```
theta = linspace(0,4*pi,10)
x = exp(-0.4*theta).*sin(theta);
plot(x,'g');
hold on;
theta = linspace(0,4*pi,100)
x = exp(-0.4*theta).*sin(theta);
plot(x,'r');
xlabel("X_Axis");
ylabel("Y_Axis");
title("Graph")
```

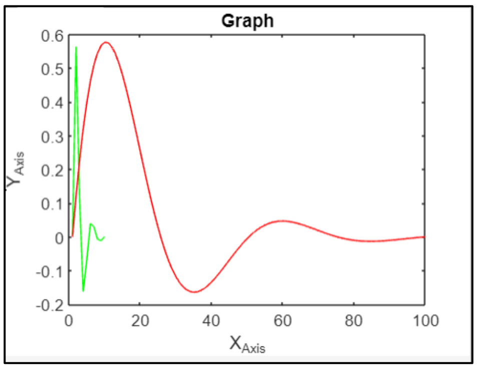


*Figure 17: Graph of (e^-0.4) * (sinx)*

Note:
plot() = plot a graph
xlabel() = X-axis label
ylabel() = y-axis label
title() = title of graph
linspace() = gives direct control over the numbers of points and always includes the endpoints
axis() = manipulates axis properties
hold on = retains plots in current axis so that new plots added to the axis do not delete existing plot
hold off = sets the hold state to off so that new plots added to the axes clear existing plots and reset all axes properties
legend = creates a legend with descriptive labels for each plotted data series.
xlim(limits), ylim(limits) = sets the x-axis and y-axis limits for the current axes or chart.

Practical No. 4

Aim: Implement different matrix transformations using MATLAB

1. **Define a 3x3 matrix A of random values using rand function.**
2. **Print the element in 2nd row and 1st column of matrix A.**
3. **Replace the element in the 3rd row and 1st column of matrix A by 0.**
4. **Obtain a submatrix B from matrix A which contains elements from 2nd row to 3rd row and 1st column to 3rd column of matrix A.**
5. **Obtain a submatrix C from matrix A which contains elements from 2nd row to 3rd row and 2nd column to 3rd column of matrix A.**

Solution:

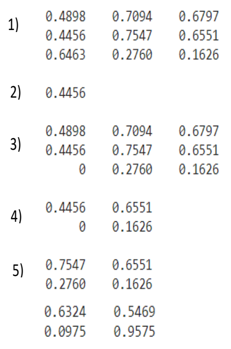
1)
```
a = rand(3, 3);%rand() = random
values in the matric
disp(a);
element = a(2, 1);
disp(element);
```

2)
```
a(3, 1) = 0;
disp(a);
```

3)
```
submatrix1 = a(2:3, [1, 3]);% slicing
of element
disp(submatrix1);
```

4)
```
submatrix2 = a(2:3, 2:3);
disp(submatrix2);
```

5)
```
submatrix3 = a(2:3, 2:3);
disp(submatrix3);
```

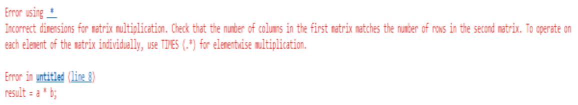


```
1)  0.4898    0.7094    0.6797
    0.4456    0.7547    0.6551
    0.6463    0.2760    0.1626

2)  0.4456

    0.4898    0.7094    0.6797
3)  0.4456    0.7547    0.6551
         0    0.2760    0.1626

    0.4456    0.6551
4)       0    0.1626

5)  0.7547    0.6551
    0.2760    0.1626

    0.6324    0.5469
    0.0975    0.9575
```

Figure 1: performing all the operation mentioned above

1) **Define a new 3x3 matrix P using rand function. Perform matrix multiplication between matrices A and P.**

```
P = rand(3, 3);
A = [];
result = A * P;
```



```
Error using *
Incorrect dimensions for matrix multiplication. Check that the number of columns in the first matrix matches the number of rows in the second matrix. To operate on
each element of the matrix individually, use TIMES (.*) for elementwise multiplication.

Error in untitled (line 8)
result = a * b;
```

Figure 18: error displayed

2) **Define a 3x3 matrix whose all elements are zero using zeros function**

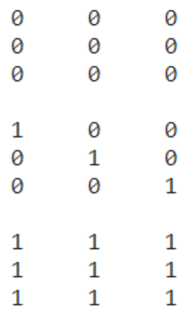
```
x = zeros(3,3);%all elements are zeros
disp(x)
```

3) **Define a 3x3 matrix whose all elements are one using ones function**

```
y = eye(3);%diagonal elements are 1 and
zeros elsewhere
disp(y)
```

4) **Define a 3x3 Identity matrix using eye function.**

```
z = ones(3, 3);%all elements are one's
disp(z)
```

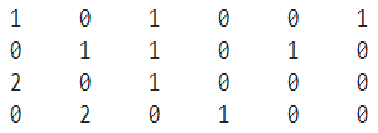


```
     0     0     0
     0     0     0
     0     0     0

     1     0     0
     0     1     0
     0     0     1

     1     1     1
     1     1     1
     1     1     1
```

Figure 19: Zeros, eyes and ones matrices

5) **Define the matrix [1,1,0,0,1,0; 1,1,0,0,0,1; 2,0,1,0,0,0; 0,2,0,1,0,0]**

```
W = zeros(4, 6);
W = W + eye(4, 6);
W(1, 3) = 1;
W(1, 6) = 1;
W(2, 3) = 1;
W(2, 5) = 1;
W(3, 1) = 2;
W(4, 2) = 2;
disp(W)
```



```
     1     0     1     0     0     1
     0     1     1     0     1     0
     2     0     1     0     0     0
     0     2     0     1     0     0
```

Figure 20: Changing elements in a ZEROS matric

6) **Define a 5x7 matrix with random elements.**

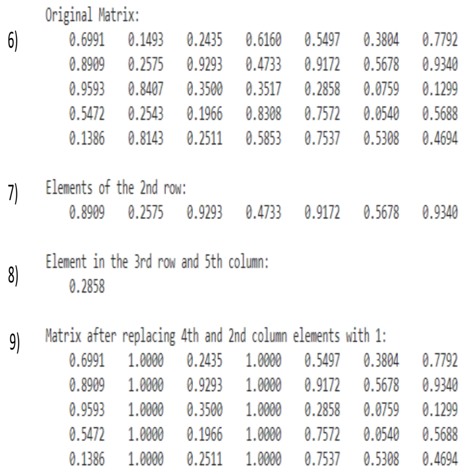
```
matrix = rand(5, 7);
disp('Original Matrix:');
disp(matrix);
```

7) **Extract all the elements of the second row.**

```
row_2_elements = matrix(2, :);
disp('Elements of the 2nd row:');
disp(row_2_elements);
```

8) **Extract the element in the 3rd row and 5th column.**

```
element_3_5 = matrix(3, 5);
disp('Element in the 3rd row and 5th
column:');
disp(element_3_5);
matrix(:, [4, 2]) = 1;
```



```
    Original Matrix:
6)      0.6991    0.1493    0.2435    0.6160    0.5497    0.3804    0.7792
        0.8909    0.2575    0.9293    0.4733    0.9172    0.5678    0.9340
        0.9593    0.8407    0.3500    0.3517    0.2858    0.0759    0.1299
        0.5472    0.2543    0.1966    0.8308    0.7572    0.0540    0.5688
        0.1386    0.8143    0.2511    0.5853    0.7537    0.5308    0.4694

7)  Elements of the 2nd row:
        0.8909    0.2575    0.9293    0.4733    0.9172    0.5678    0.9340

8)  Element in the 3rd row and 5th column:
        0.2858

9)  Matrix after replacing 4th and 2nd column elements with 1:
        0.6991    1.0000    0.2435    1.0000    0.5497    0.3804    0.7792
        0.8909    1.0000    0.9293    1.0000    0.9172    0.5678    0.9340
        0.9593    1.0000    0.3500    1.0000    0.2858    0.0759    0.1299
        0.5472    1.0000    0.1966    1.0000    0.7572    0.0540    0.5688
        0.1386    1.0000    0.2511    1.0000    0.7537    0.5308    0.4694
```

Figure 1: Extracting elements

9) Replace the element in the 4th row and 2nd column by 1.

```
disp('Matrix after replacing 4th and 2nd
column elements with 1:');
disp(matrix);
```

10) Create a matrix G using A = [2,6 ; 3,9], B = [1,2 ; 3,4], C = [-5,5 ; 5,3]. Every element should be on its diagonal.

```
A = [2, 6; 3, 9];
B = [1, 2; 3, 4];
C = [-5, 5; 5, 3];
G = diag([A(1, 1), B(1, 2), C(2, 2)]);
```

11) Find the size of matrix G.

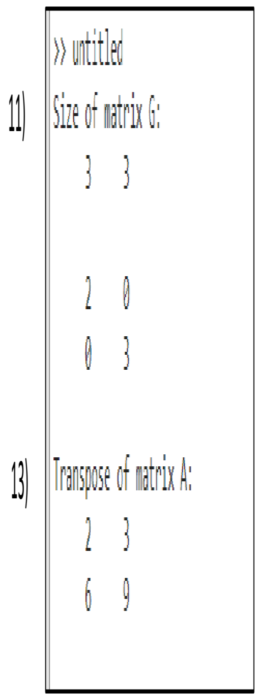
```
size_G = size(G); % size of matrix G
disp('Size of matrix G:');
disp(size_G);
```

12) Delete the last row and last column of matrix G.

```
G(2, :) = [];  % Delete the last row
G(:, 2) = [];  % Delete the last  column
disp(G);
```

13) Obtain the transpose of matrix A.

```
A transpose = A.';
disp('Transpose of matrix A:');
disp(A_transpose);
```



```
>> untitled
11) Size of matrix G:
     3     3

     2     0
     0     3

13) Transpose of matrix A:
     2     3
     6     9
```

Figure 1: performing deletion and finding the size

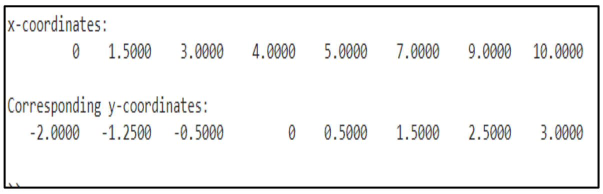
Practical No. 6

Aim: Implementation of mathematical equations using MATLAB program.

1) **The equation of a straight line is y=mx+c, where m and c are constants. Compute the y-coordinates of a line with slope m=0.5 and the intercept c=-2 at the following x-coordinates: x=0, 1.5, 3, 4, 5, 7, 9, 10.**

   Solution:

```
m = 0.5;
c = -2;
x_coordinates = [0, 1.5, 3, 4, 5, 7, 9, 10];
y_coordinates = m * x_coordinates + c;
disp('x-coordinates:');
disp(x_coordinates);
disp('Corresponding y-coordinates:');
disp(y_coordinates);
```



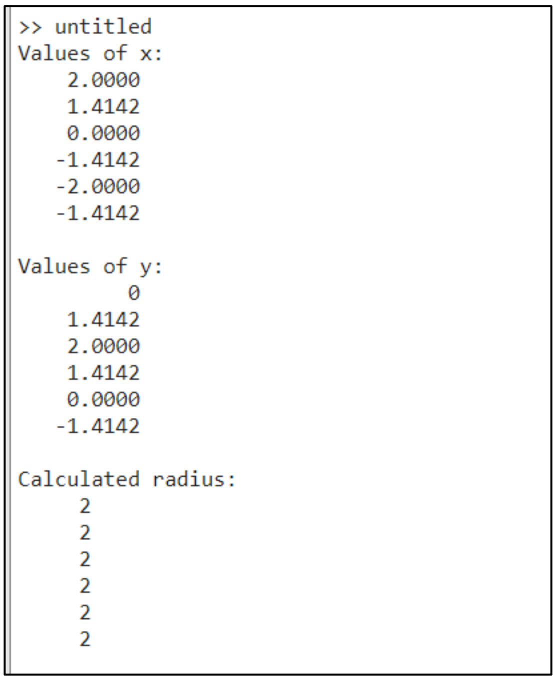
```
x-coordinates:
         0    1.5000    3.0000    4.0000    5.0000    7.0000    9.0000   10.0000

Corresponding y-coordinates:
   -2.0000   -1.2500   -0.5000         0    0.5000    1.5000    2.5000    3.0000
```

Figure 21: Finding the y-coordinates at different x-coordinates

2) **All points with coordinates x = rcos(theta) and y = rsin(theta), where r is a constant, lie on a circle with radius r, i.e., they satisfy the equation $x^2 + y^2 = r^2$. Create a column vector for theta with the values 0, pi/4, pi/2, 3pi/4, pi, 5pi/4. Take r=2 and compute column vectors x and y. Now check that x and y indeed satisfy the equation of a circle, by computing the radius $r = (x^2 + y^2)^{1/2}$.**

   Solution:

```
theta = [0; pi/4; pi/2; 3*pi/4; pi; 5*pi/4];% column vector with the given values.
r = 2;
% Calculate x and y using the given values of theta and r
x = r * cos(theta);
y = r * sin(theta);
% Calculate the radius using the equation r = sqrt(x^2 + y^2)
radius = sqrt(x.^2 + y.^2);
% Display the values of x, y, and the calculated radius
disp('Values of x:');
disp(x);
disp('Values of y:');
disp(y);
disp('Calculated radius:');
disp(radius);
```



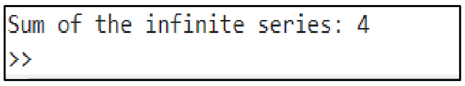
```
>> untitled
Values of x:
    2.0000
    1.4142
    0.0000
   -1.4142
   -2.0000
   -1.4142

Values of y:
         0
    1.4142
    2.0000
    1.4142
    0.0000
   -1.4142

Calculated radius:
     2
     2
     2
     2
     2
     2
```

Figure 22: calculating radius with the help of X and Y values

Write a MATLAB program to print the sum of infinite geometric series.

Solution:

```
a = 2;  % First term
r = 0.5;  % Common ratio
% Calculate the sum of the infinite geometric series
if abs(r) < 1
    S = a / (1 - r);
    disp(['Sum of the infinite series:num2str(S)]);
else
     disp('The series does not converge (|r| >= 1).');
end
```



```
Sum of the infinite series: 4
>>
```

Figure 23: Sum of infinite geometric series

Draw a circle of unit radius. Mark the centre of the circle by '+'.
Solution:

```
figure;% Create a figure
radius = 1;
center_x = 0;
center_y = 0;
theta = linspace(0, 2 * pi, 100);  % 100 points around
the circle

% Calculate the coordinates of the points on the circle
x = radius * cos(theta) + center_x;
y = radius * sin(theta) + center_y;

% Plot the circle
plot(x, y, 'b');  % 'b' specifies a blue line
hold on;

% Mark the center with a '+'
plot(center_x, center_y, 'r+', 'MarkerSize', 10);

% Set axis limits to ensure the circle is fully visible
axis equal;  % Equal aspect ratio
axis([-1.5  1.5  -1.5  1.5]);   % Adjust these limits as
needed

% Add labels and title
xlabel('x-axis');
ylabel('y-axis');
title('Circle with Center Marked');
```

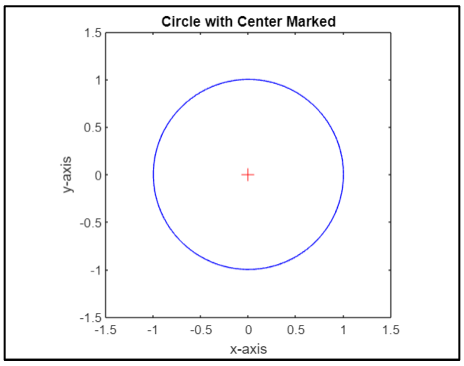


Figure 24: Circle with center marked

Modify the above code such that the user is prompted to give an arbitrary radius.
Solution:

```
radius = input('Enter the radius of the circle: ');
theta = linspace(0, 2 * pi, 100);
x = radius * cos(theta);
y = radius * sin(theta);
figure;
plot(x, y, 'b');
hold on;
plot(0, 0, 'r+', 'MarkerSize', 10);
axis equal;
axis([-radius - 1, radius + 1, -radius - 1, radius + 1]);
xlabel('x-axis');
ylabel('y-axis');
title('Circle with Center Marked');
hold off;
```

```
>> untitled
Enter the radius of the circle:
4
```

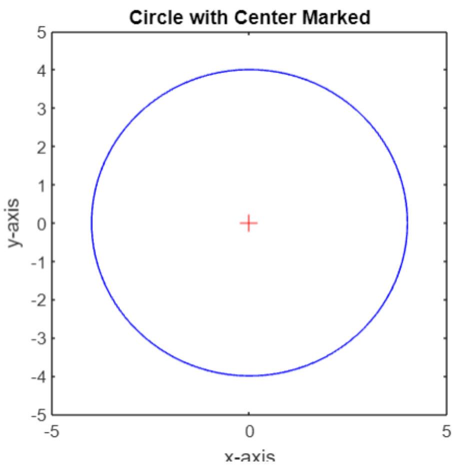


Figure 25: Entering the arbitrary radius and plot

Write a function which accepts the radius and the centre of a circle and returns the value of x and y. Also plot the circle with it's centre marked by '+'.

Solution:

```
function [x, y] = untitled(radius, centerX, centerY)
theta = linspace(0, 2 * pi, 100);
x = radius * cos(theta) + centerX;
y = radius * sin(theta) + centerY;
figure;
plot(x, y, 'b');
hold on;
plot(centerX, centerY, 'r+', 'MarkerSize', 10);
axis equal;
axis([centerX - radius - 1, centerX + radius + 1, centerY - radius - 1, centerY + radius + 1]);
xlabel('x-axis');
ylabel('y-axis');
title('Circle with Center Marked');
grid on;
hold off;
end
radius = 3;
centerX = 2;
centerY = -1;
[x, y] = untitled(radius, centerX, centerY);
```

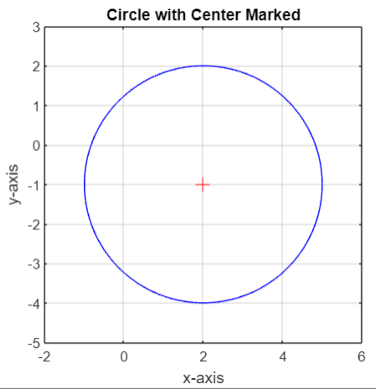
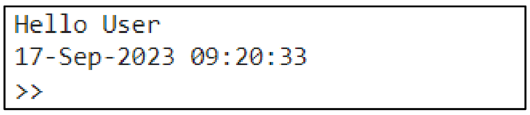


Figure 26: circle plot

Write a script file which when executed, greets you, displays the date and time.
Solutions:

```
disp("Hello User");
disp(string(datetime))
```



```
Hello User
17-Sep-2023 09:20:33
>>
```

Figure 27: user name and date/time

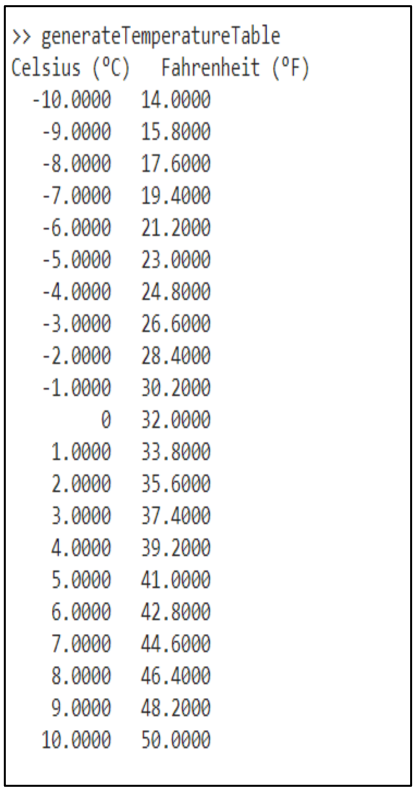
Practical 7

Aim: Write a MATLAB program to implement different functions using script files.

1) **Write a function that outputs a conversion table for Celsius and Fahrenheit temperatures. The input of the function should be two numbers Ti and Tf, specifying the lower and upper range of the table in Celsius. The output should be a two-column matrix. The first column showing the temperature in Celsius from Ti to Tf in the increment of 1 degree Celsius and the second column showing the corresponding temperature in Fahrenheit.**

Solution:

```
Function conversionTable =
generateTemperatureTable(Ti, Tf)
    % Check if Ti is greater than Tf, and swap values if
    necessary
    if Ti > Tf
        [Ti, Tf] = deal(Tf, Ti);
    end
    % Initialize empty arrays to store Celsius and
    Fahrenheit values
    celsiusValues = Ti:Tf;
    fahrenheitValues = (celsiusValues * 9/5) + 32;
    % Create the conversion table as a two-column
    matrix conversionTable = [celsiusValues',
    fahrenheitValues'];
    % Display the conversion table
    disp('Celsius   Fahrenheit');
    disp(conversionTable);
end
Ti = -10;  % Lower limit in Celsius
Tf = 10;   % Upper limit in Celsius
generateTemperatureTable(Ti, Tf);
```



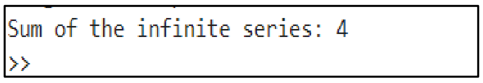
```
>> generateTemperatureTable
Celsius (°C)   Fahrenheit (°F)
  -10.0000   14.0000
   -9.0000   15.8000
   -8.0000   17.6000
   -7.0000   19.4000
   -6.0000   21.2000
   -5.0000   23.0000
   -4.0000   24.8000
   -3.0000   26.6000
   -2.0000   28.4000
   -1.0000   30.2000
         0   32.0000
    1.0000   33.8000
    2.0000   35.6000
    3.0000   37.4000
    4.0000   39.2000
    5.0000   41.0000
    6.0000   42.8000
    7.0000   44.6000
    8.0000   46.4000
    9.0000   48.2000
   10.0000   50.0000
```

Figure 28: conversion from Celsius to Fahrenheit

2) **Write a function to print the sum of infinite geometric series**

Solution:

```
function sum = infiniteGeometricSeries(a, r)
% Check if the series converges (|r| < 1)
  if abs(r) >= 1
        error('The series does not converge
(|r| >= 1).');
  end
% Calculate the sum of the infinite geometric
series
    Sum = a / (1 - r);
end
a = 2;  % First term
r = 0.5;  % Common ratio
S = infiniteGeometricSeries(a, r);
```



```
Sum of the infinite series: 4
>>
```

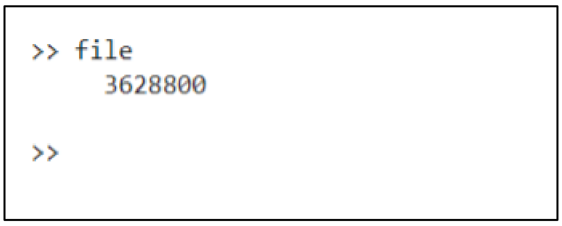
Figure 29: Sum of infinite geometric series

```
disp(['Sum of the infinite series: ' num2str(S)]);
```

3) Write a function factorial to compute the factorial of any integer.

Solution:

```
function file(x)
                fact=1;
                for i = 1:x
                        fact = fact * i;
                end
                disp(fact)
        end
        x = 10;
        file(x);
```



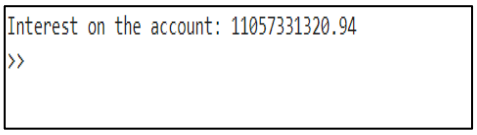
```
>> file
   3628800

>>
```

Figure 30: factorial of 10

4) The interest you get at the end of n years, at a flat annual rate of r% depends on how the interest is compounded. If the interest is added to your account k times a year and the principal amount you invested is X_o, then at the end of n years you would have $X = X_o (1 + r/k)^{kn}$ amount of money in your account. Write a function to compute the interest (X - Xo) on your account for a given X, n, r and k.

Solution:

```
function interest = computeInterest(X, Xo, n, r, k)
  % Check if the provided values are valid
  if Xo <= 0 || X <= 0 || n <= 0 || r < 0 || k <= 0
        error('Invalid input values. Ensure that Xo, X,
n, r, and k are positive and n > 0.');
 end

 % Calculate the interest using the formula
    interest = X - Xo * (1 + r / k)^(k * n);
  end
Xo = 1000;  % principal amount
n = 5;      % Number of years
r = 5;      % Annual interest rate in percentage
k = 4;      % Number of times interest is compounded
per year
X = 1000 * (1 + r / k)^(k * n);  % Calculate the final
amount based on the formula
interest = computeInterest(X, Xo, n, r, k);
disp(['Interest on the account: ' num2str(interest)]);
```



```
Interest on the account: 11057331320.94
>>
```

Figure 31: Interest

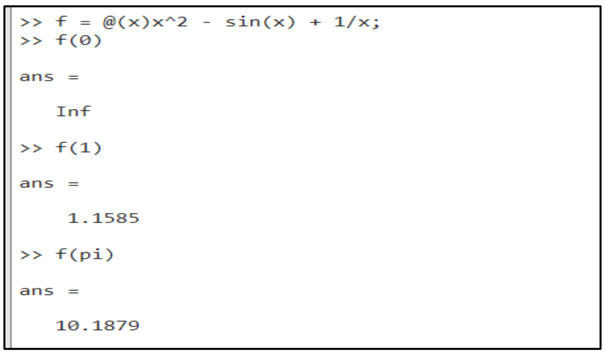
Practical 8

Aim: Create and evaluate anonymous functions using MATLAB Command window.

1. **Create the function f(x) = x^2 – sin(x) +(1/x).**

   f = @(x)x^2 - sin(x) + 1/x;

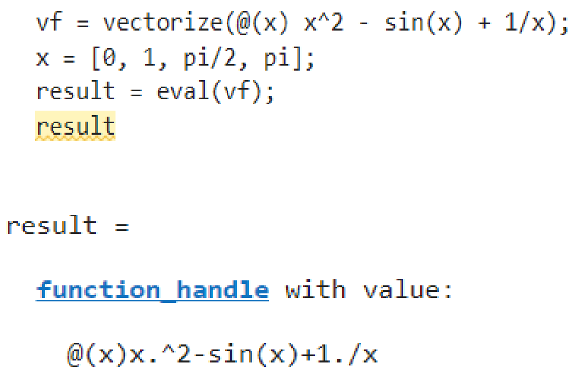
a) Find f (0), f (1) and f (pi/2).



```
>> f = @(x)x^2 - sin(x) + 1/x;
>> f(0)

ans =

   Inf

>> f(1)

ans =

    1.1585

>> f(pi)

ans =

   10.1879
```

Figure 32: values at f(0),f(1) and f(pi/2)

b) Vectorize f and evaluate f (x) where x = [0 1 pi/2 pi]



```
vf = vectorize(@(x) x^2 - sin(x) + 1/x);
x = [0, 1, pi/2, pi];
result = eval(vf);
result


result =

  function_handle with value:

    @(x)x.^2-sin(x)+1./x
```

Figure 33: vectorize f

c) Create x = linspace(-1,1), evaluate f (x), and plot x vs f (x)

```
x = linspace(-1, 1, 100);% 100 points between -1 and 1
f_x = x.^2 - sin(x) + 1./x;
figure;
plot(x, f_x, 'b', 'LineWidth', 2);  % Plot x vs f(x) in blue with a line width  of 2
xlabel('x');
ylabel('f(x)');
title('Plot of f(x) = x^2 - sin(x) + 1/x');
grid on;
```

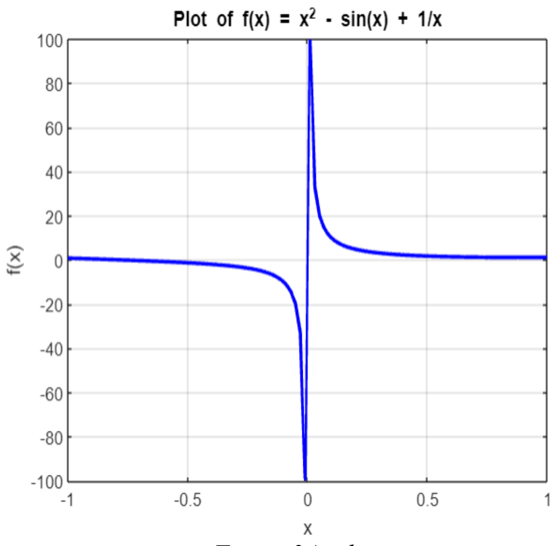


Figure 34: plot

d) Combine the following three commands into a single command to produce the plot that you will get at the end of the third command.

x = linspace(-1, 1);
f_x = x.^2 - sin(x) + 1./x;
plot(x, f_x);
plot(linspace(-1, 1), linspace(-1, 1).^2 - sin(linspace(-1, 1))

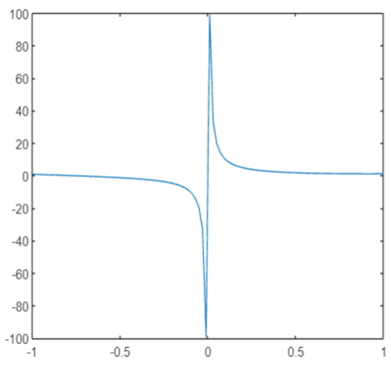


Figure 35: plot

e) Use fplot to plot f (x) over x from -pi to pi

f = @(x) x.^2 - sin(x) + 1./x;
fplot(f, [-pi, pi]);
xlabel('x');
ylabel('f(x)');
title('Plot of f(x) = x^2 - sin(x) + 1/x');
grid on;

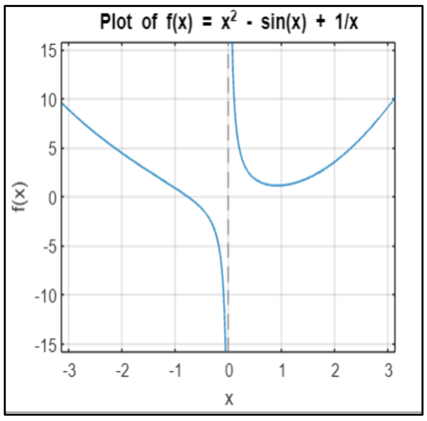
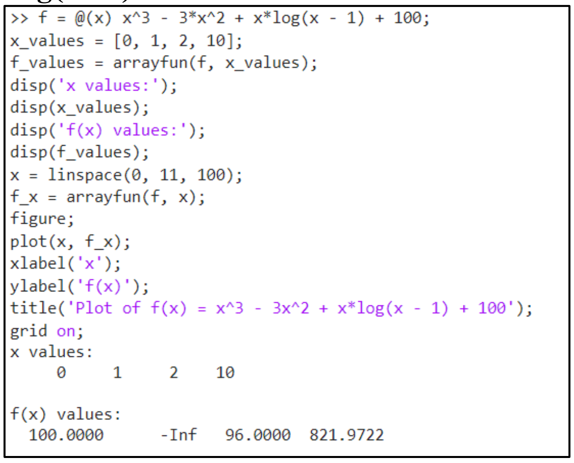


Figure 36: plot

2. Create a function f(x) = x^3 - $3x^2$ + x log(x - 1) + 100

a) Evaluate the function at x =0, 1, 2,10 in an array form



```
>> f = @(x) x^3 - 3*x^2 + x*log(x - 1) + 100;
x_values = [0, 1, 2, 10];
f_values = arrayfun(f, x_values);
disp('x values:');
disp(x_values);
disp('f(x) values:');
disp(f_values);
x = linspace(0, 11, 100);
f_x = arrayfun(f, x);
figure;
plot(x, f_x);
xlabel('x');
ylabel('f(x)');
title('Plot of f(x) = x^3 - 3x^2 + x*log(x - 1) + 100');
grid on;
x values:
     0     1     2    10

f(x) values:
  100.0000      -Inf   96.0000  821.9722
```

Figure 37: value of f(x) at different x values

b) Plot x,f(x)

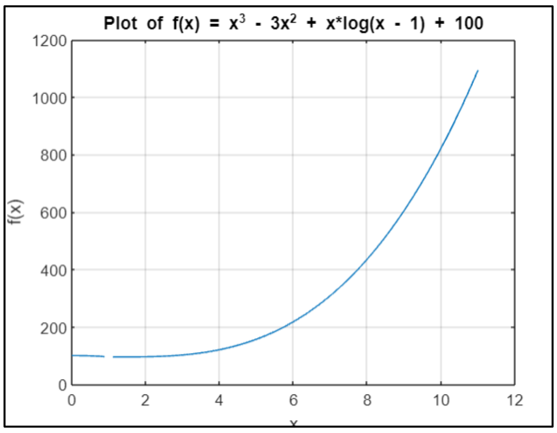


Figure 38: plot

3. Define x and y to be symbolic variables. Define a function f as $f=(x+y)^3$.

a) expand algebraic expressions for f(x)

```
% a. Expand algebraic expression for f(x)
expanded_f = expand(f);
disp('Expanded expression for f(x):');
disp(expanded_f);
```

b) find factors of algebraic expression f(x)

```
% b. Find factors of the algebraic expression f(x)
factors_f = factor(f);
disp('Factors of f(x):');
disp(factors_f);
```

c) Substitute y = pi-x in expression f.

```
% c. Substitute y = pi - x in expression f
f_substituted = subs(f, y, pi - x);
disp('Substituting y = pi - x in f:');
disp(f_substituted);
```

d) Differentiate f with respect to x.

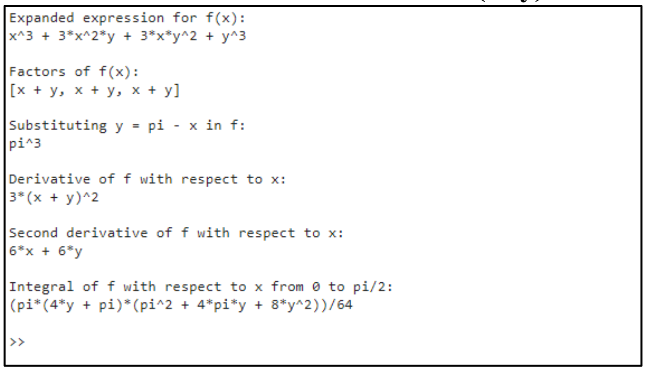
```
% d. Differentiate f with respect to x
df_dx = diff(f, x);
disp('Derivative of f with respect to x:');
disp(df_dx);
```

e) Find the second derivative of f with respect to x

```
% e. Find the second derivative of f with respect to x
d2f_dx2 = diff(df_dx, x);
disp('Second derivative of f with respect to x:');
disp(d2f_dx2);
```

f) Integrate z with respect to x from 0 to pi/2

```
% f. Integrate z with respect to x from 0 to pi/2
integrated_f = int(f, x, 0, pi/2);
disp('Integral of f with respect to x from 0 to pi/2:');
disp(integrated_f);
```



```
Expanded expression for f(x):
x^3 + 3*x^2*y + 3*x*y^2 + y^3

Factors of f(x):
[x + y, x + y, x + y]

Substituting y = pi - x in f:
pi^3

Derivative of f with respect to x:
3*(x + y)^2

Second derivative of f with respect to x:
6*x + 6*y

Integral of f with respect to x from 0 to pi/2:
(pi*(4*y + pi)*(pi^2 + 4*pi*y + 8*y^2))/64

>>
```

Figure 39: Output of the above question

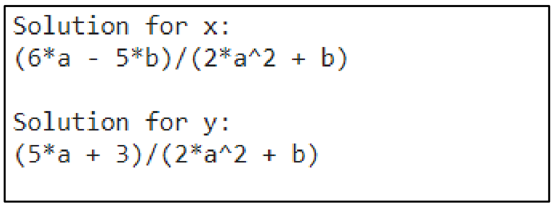
4. Solve two simultaneous algebraic equations for x and y:

ax + by - 3 = 0

-x + 2ay - 5 = 0

Solution:

```
% Define the symbolic variables and coefficients
syms x y a b

% Define the equations
eq1 = a*x + b*y - 3 == 0;
eq2 = -x + 2*a*y - 5 == 0;

% Solve the equations for x and y
solution = solve([eq1, eq2], [x, y]);

% Display the solutions
x_solution = solution.x;
y_solution = solution.y;

disp('Solution for x:');
disp(x_solution);
disp('Solution for y:');
disp(y_solution);
```



```
Solution for x:
(6*a - 5*b)/(2*a^2 + b)

Solution for y:
(5*a + 3)/(2*a^2 + b)
```

Figure 40: Solution for the equations

Practical 9

Aim: To write a MATLAB program to generate various signals and sequences such as unit impulse, unit step, unit ramp, exponential, sinusoidal, saw tooth, triangular, sinc signal, square functions.

Unit impulse

```
N = 10;
x = zeros(1, N);
x(5) = 1;
stem(x);%stem( Y ) plots the data sequence,
Y , as stems that extend from a baseline
along the x-axis.
xlabel('n');
ylabel('Amplitude');
title('Unit Impulse signal');
grid on;% grid on displays the major grid
lines for the current axes returned by the gca
command.
```

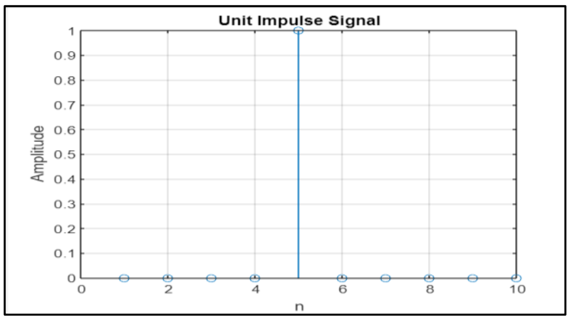


Figure 44: Output of a unit impulse

Unit Step:

```
t = -5:0.01:5;
u = zeros(size(t));
u(t >= 0) = 1;
plot(t, u, 'b', 'LineWidth', 2);
title('Unit Step Signal');
xlabel('Time (t)');
ylabel('Amplitude');
grid on;
axis([-5 5 -0.2 1.2]);
```

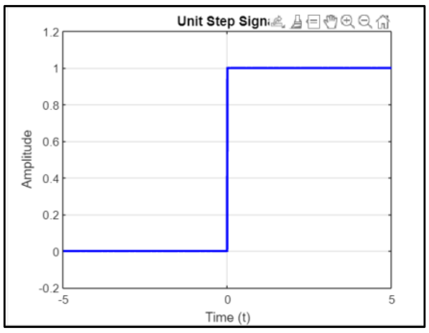


Figure 45: Output of unit step

Unit ramp:-

```
n = 0:10;
x = n;
stem(n, x, 'r', 'filled');
xlabel('Time (n)');
ylabel('Amplitude');
title('Unit Ramp Signal');
grid on;
```

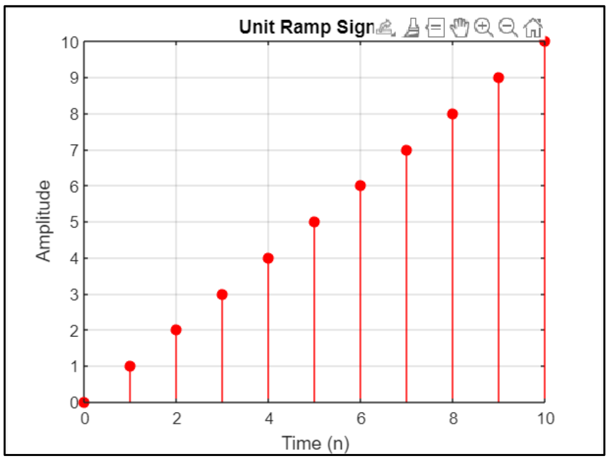


Figure 46: Output of unit ramp

Exponential

```
A = 1;
alpha = 0.5;
n = 0:10;
x = A * exp(alpha * n);
stem(n, x);
xlabel('Time Index (n)');
ylabel('Amplitude');
title('Exponential Signal');
grid on;
```

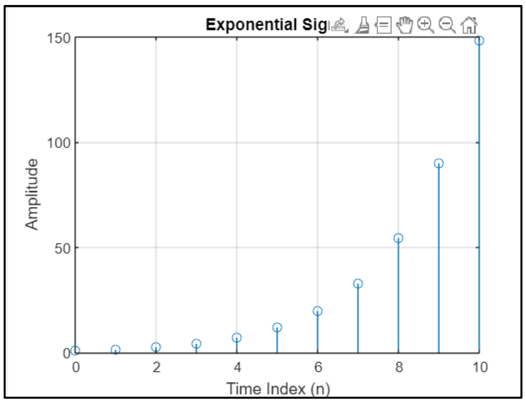


Figure 47: output of the exponential

Sinusoidal:

```
frequency = 2;
amplitude = 1;
duration = 2;
sampling_rate = 1000;
t = linspace(0, duration, duration *
sampling_rate);
signal = amplitude * sin(2 * pi * frequency
* t);
figure;
plot(t, signal);
xlabel('Time (s)');
ylabel('Amplitude');
title('Sinusoidal Signal');
grid on;
```

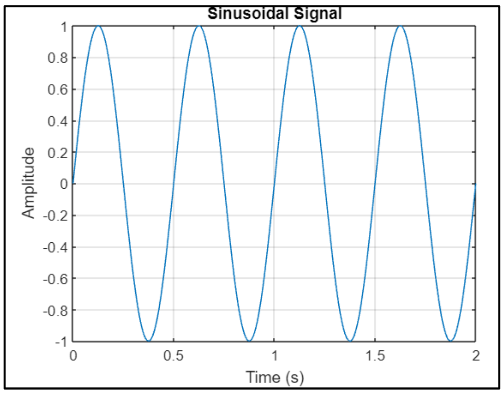


Figure 48: Sinusoidal graph

Saw tooth (2*pi*50*t), freq 50 Hz, sample rate is 1 khz, generate 10 periods

```
frequency = 50;
sample_rate = 1000;
duration = 10 / frequency;
t = 0:1/sample_rate:duration;
sawtooth_signal =
sawtooth(2*pi*frequency*t);
plot(t, sawtooth_signal);
title('Sawtooth Signal');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;
xlim([0, 1/frequency]);
xticks(0:1/frequency:duration);
xticklabels(0:10);
```

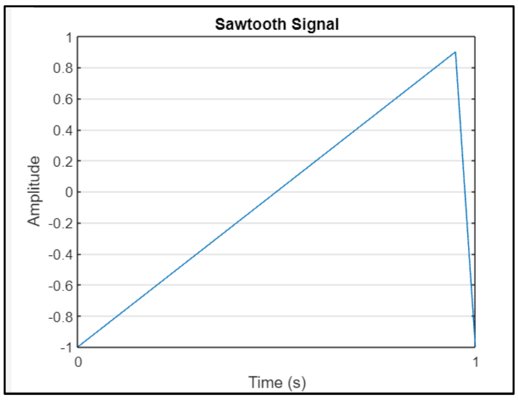


Figure 49: saw tooth graph

Triangular (2*pi*50*t,1/2), freq 50 Hz, sample rate is 1 khz, generate 10 periods

```
frequency = 50;
sample_rate = 1000;
duration = 10 * (1 / frequency);
t = 0:1/sample_rate:duration;
triangular_signal = sawtooth(2 * pi *
frequency * t, 0.5);
plot(t, triangular_signal);
title('Triangular Signal');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;
```

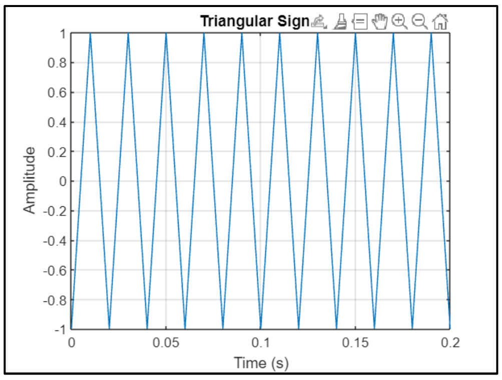


Figure 50: triangular signal graph

Sinc signal (sinc(x))

```
t = -10:0.01:10;
sinc_signal = sinc(t);
figure;
plot(t, sinc_signal, 'b', 'LineWidth', 2);
title('Sinc Signal');
xlabel('Time (t)');
ylabel('Amplitude');
grid on;
```

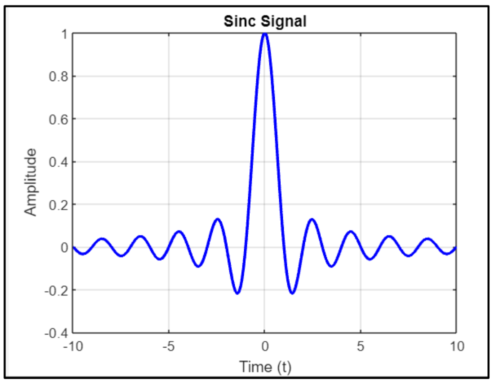


Figure 51: Sinc signal

Square function (t/T*2*pi) genrate & plot square wave with periods 0.5 & amplitude

```
0.81
T = 0.5;
amplitude = 0.81;
t = 0:0.001:2*T;
square_wave = amplitude * square(2 * pi * t
/ T);
plot(t, square_wave, 'b', 'LineWidth', 2);
xlabel('Time (s)');
ylabel('Amplitude');
title('Square Wave');
grid on;
axis([0 2*T -amplitude-0.1 amplitude+0.1]);
grid on;
```

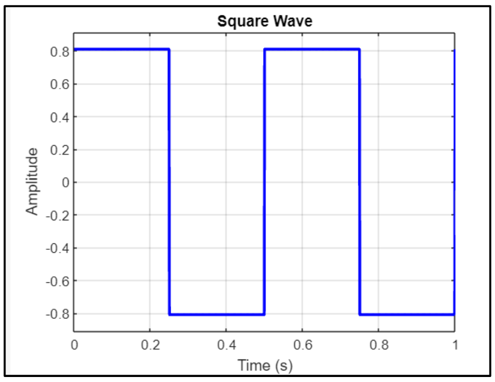


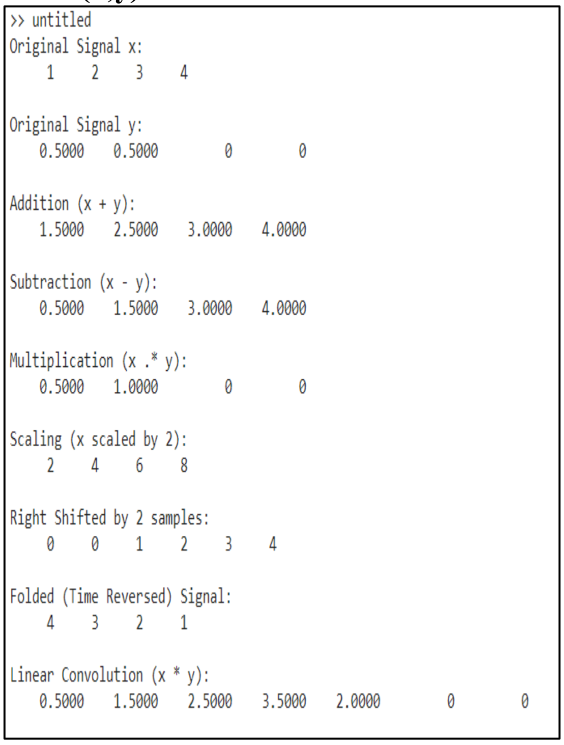
Figure 52: Square wave

Practical No.:-10

Aim: To perform operations on signals and sequences such as addition, multiplication, scaling, shifting, folding.

1. **Linear convolution x=[----]; y=[----]; z=conv(x,y)**

```
% Define two signals x and y
x = [1 2 3 4];
y = [0.5 0.5 0 0];
% Addition
z_addition = x + y;
% Subtraction
z_subtraction = x - y;
% Multiplication (Modulation)
z_multiplication = x .* y;
% Scaling
scale_factor = 2.0;
z_scaling = x * scale_factor;
% Shifting (Right Shift by 2)
k = 2;
z_shifted = [zeros(1, k) x];
% Folding (Time Reversal)
z_folded = fliplr(x);
% Linear Convolution
z_convolution = conv(x, y, 'full');
% Display results
disp('Original Signal x:');
disp(x);
disp('Original Signal y:');
disp(y);
disp('Addition (x + y):');
disp(z_addition);
disp('Subtraction (x - y):');
disp(z_subtraction);
disp('Multiplication (x .* y):');
disp(z_multiplication);
disp(['Scaling (x scaled by '
num2str(scale_factor) '):']);
disp(z_scaling);
disp(['Right Shifted by ' num2str(k) '
samples:']);
disp(z_shifted);
disp('Folded (Time Reversed) Signal:');
disp(z_folded);
disp('Linear Convolution (x * y):');
disp(z_convolution);
```



```
>> untitled
Original Signal x:
     1     2     3     4

Original Signal y:
    0.5000    0.5000         0         0

Addition (x + y):
    1.5000    2.5000    3.0000    4.0000

Subtraction (x - y):
    0.5000    1.5000    3.0000    4.0000

Multiplication (x .* y):
    0.5000    1.0000         0         0

Scaling (x scaled by 2):
     2     4     6     8

Right Shifted by 2 samples:
     0     0     1     2     3     4

Folded (Time Reversed) Signal:
     4     3     2     1

Linear Convolution (x * y):
    0.5000    1.5000    2.5000    3.5000    2.0000         0         0
```

Figure 41: Output of the above code

2. **Auto correlation n=0:15; x=0.84.^n; xcorr(x); cross correlation n=0:15; x=0.84.^n; y=circshift(x,5); xcorr(x,y);**

Auto-Correlation:

```
n = 0:15;
x = 0.84.^n;
auto_corr = xcorr(x);
% Plot the auto-correlation
stem(-15:15, auto_corr);
title('Auto-correlation of x');
xlabel('Lag');
ylabel('Auto-correlation');
```

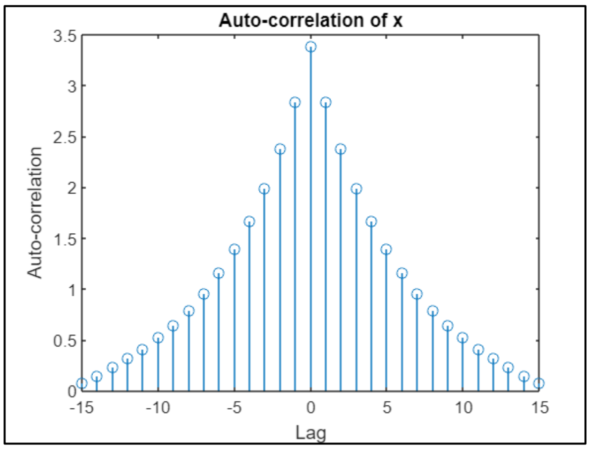


Figure 42: AutoCorrelation output

Cross Correlation:

```
n = 0:15;
x = 0.84.^n;
y = circshift(x, 5); % Shifting x by 5
positions
cross_corr = xcorr(x, y);
% Plot the cross-correlation
stem(-15:15, cross_corr);
title('Cross-correlation between x and y');
xlabel('Lag');
ylabel('Cross-correlation');
```

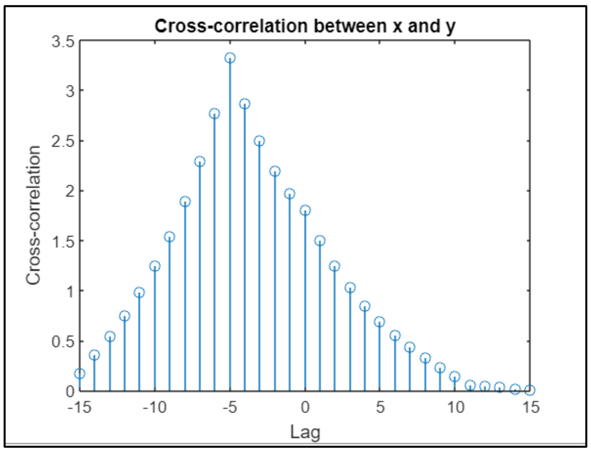
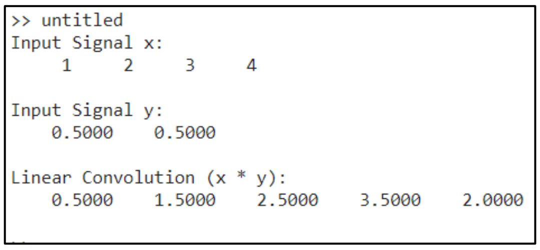


Figure 43: Cross Correlation output

Practical No.:11

Aim: (a) Write a program to find the output with linear convolution operation using matlab software.

```
 % Define two input signals
 x = [1 2 3 4];
 y = [0.5 0.5];

 % Perform linear convolution
 z = conv(x, y, 'full');

 % Display the input signals and the
convolution result
 disp('Input Signal x:');
 disp(x);

 disp('Input Signal y:');
 disp(y);

 disp('Linear Convolution (x * y):');
 disp(z);

 % Plot the convolution result
 figure;
 subplot(3, 1, 1);
 stem(x, 'b', 'LineWidth', 1.5);
 title('Input Signal x');
 xlabel('Sample Index');
 ylabel('Amplitude');

 subplot(3, 1, 2);
 stem(y, 'r', 'LineWidth', 1.5);
 title('Input Signal y');
 xlabel('Sample Index');
 ylabel('Amplitude');

 subplot(3, 1, 3);
 stem(z, 'g', 'LineWidth', 1.5);
 title('Convolution Result (x * y)');
 xlabel('Sample Index');
 ylabel('Amplitude');

 % Adjust plot spacing
 sgtitle('Linear Convolution');
```



```
>> untitled
Input Signal x:
     1     2     3     4

Input Signal y:
    0.5000    0.5000

Linear Convolution (x * y):
    0.5000    1.5000    2.5000    3.5000    2.0000
```

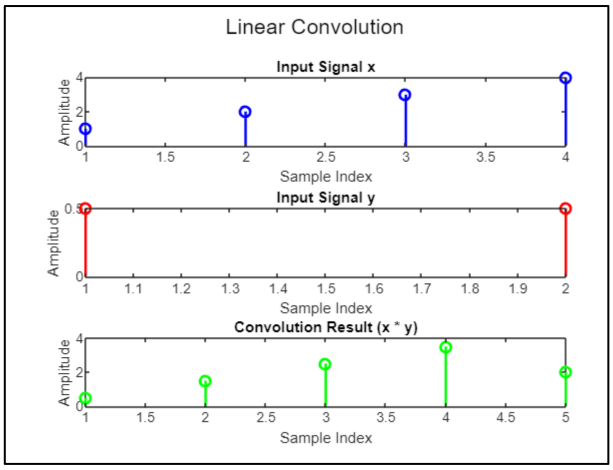
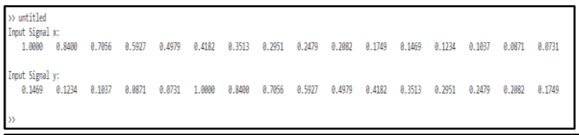


Figure 44: Linear Correlation

(b) Write a program to compute auto correlation and cross correlation between signals and sequences.

```
 % Define two input signals
 n = 0:15;
 x = 0.84.^n;
 y = circshift(x, 5); % Shifting x by 5
positions to create y

 % Compute auto-correlation of x
 auto_corr_x = xcorr(x);

 % Compute cross-correlation between x
and y
 cross_corr_xy = xcorr(x, y);

 % Display the input signals and the
correlation results
 disp('Input Signal x:');
 disp(x);

 disp('Input Signal y:');
 disp(y);

 % Plot auto-correlation of x
 figure;
 subplot(2, 1, 1);
 stem(-15:15, auto_corr_x, 'b', 'LineWidth',
1.5);
 title('Auto-correlation of x');
 xlabel('Lag');
 ylabel('Auto-correlation');

 % Plot cross-correlation between x and y
 subplot(2, 1, 2);
 stem(-15:15, cross_corr_xy, 'r',
'LineWidth', 1.5);
 title('Cross-correlation between x and y');
 xlabel('Lag');
 ylabel('Cross-correlation');

 % Adjust plot spacing
 sgtitle('Auto-correlation and Cross-
correlation');
```



```
>> untitled
Input Signal x:
    1.0000    0.8400    0.7056    0.5927    0.4979    0.4182    0.3513    0.2951    0.2479    0.2082    0.1749    0.1469    0.1234    0.1037    0.0871    0.0731

Input Signal y:
    0.1469    0.1234    0.1037    0.0871    0.0731    1.0000    0.8400    0.7056    0.5927    0.4979    0.4182    0.3513    0.2951    0.2479    0.2082    0.1749

>>
```

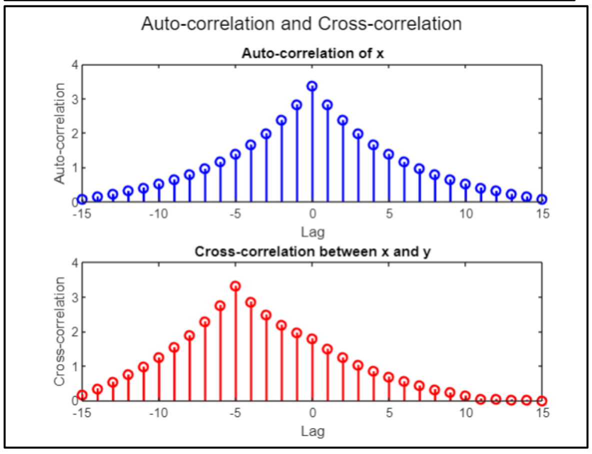


Figure 45: Auto Correlation and Cross Correlation