**MATLAB PROGRAMMING LAB FILE**

**CSEU4103**

****

**Submitted By: Submitted To:**

Kanishk Yadav Dr Abhinaba Sinha

Course: M. Tech (AIML) ASET Dept.

Session: 2024 – 26

Enrollment No: A501144824006

**INDEX**

Z

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S. No.** | **Program** | **Page No.** | **Date** | **Teacher’s Sign** |
| 1 | **a) Introduction to MATLAB** | 3-7 | **20-08-2024** |  |
|  | **b) Basic Operations in MATLAB** | **20-08-2024** |  |
| 2 | **Implementation of MATLAB program for various defining different mathematical functions** | 8-9 | **10-09-2024** |  |
| 3 | **Introduction to plots using MATLAB programming** | 10-11 | **17-09-2024** |  |
| 4 | **Implement different matrix transformations using MATLAB** | 12-14 | **24-09-2024** |  |
| 5 | **Implementation of mathematical equations using MATLAB program** | 15-17 | **01-10-2024** |  |
| 6 | **Write a MATLAB program to implement different functions using script files.** | 18-19 | **08-10-2024** |  |
| 7 | **Create and evaluate anonymous functions using MATLAB Command window.** | 20-23 | **15-10-2024** |  |
| 8 | **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.** | 24-26 | **22-10-2024** |  |
| 9 | **To perform operations on signals and sequences such as addition, multiplication, scaling, shifting, folding.** | 27-28 | **05-11-2024** |  |
| 10 | a) **Write a program to find the output with linear convolution operation using matlab software** | 29-30 | **12-11-2024** |  |
|  | b) **Write a program to compute auto correlation and cross correlation between signals and sequences.** | **12-11-2024** |  |

**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.

A screenshot of a computer

Description automatically generated

Figure 1: MATLAB home window with different tools

**Command Window**

**Script editor**

* **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.

A screenshot of a computer

Description automatically generated

**Command**

**Window**

**Script editor**

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.

A screenshot of a computer

Description automatically generated

**Command**

**Window**

**Script editor**

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 | A white background with black text  Description automatically generated  Figure 4: Addition of 2 numbers |

**Subtraction:**

|  |  |
| --- | --- |
| a = 6  b = 2  c = a - b | A white background with black text  Description automatically generated  Figure 5: Subtraction of 2 numbers |

**Multiplication:**

|  |  |
| --- | --- |
| a = 6  b = 2  c = a \* b | A white background with black text  Description automatically generated  Figure 6: Multiplication of 2 numbers |

**Division:**

|  |  |
| --- | --- |
| a = 6  b = 2  c = a / b | A white background with black and white clouds  Description automatically generated  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 | A white background with black numbers  Description automatically generated  Figure 8: Addition of 2 Matric |

**Subtraction:**

|  |  |
| --- | --- |
| a = [1,2,3,4]  b = [5,6,7,8]  c = a – b | A white background with black numbers  Description automatically generated  Figure 9: Subtraction of 2 numbers |

**Multiplication:**

|  |  |
| --- | --- |
| a = [1  2  3  4]  b = [5,6,7,8]  c = a \* b | A white background with numbers and lines  Description automatically generated  Figure 10: Mulitplication of 2 matrics |

**Division of a matric by a number:**

|  |  |
| --- | --- |
| a = [1  2  3  4]  b = 5  c = a / b | A white paper with black numbers and a few black numbers  Description automatically generated with medium confidence  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 number on a white background  Description automatically generated  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); | A white background with numbers and a few black text  Description automatically generated  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. 25/25-1 = 1.0333
2. = 4.7447
3. = 20.0855
4. sin = 0.5000
5. cos = 0.8600
6. log = 1.3029
7. log = 5
8. tan = 1.6331e + 16
9. sin( = -0.2152
10. = 0.0472
11. π)2 = 2.1413
12. 1.0000 + 0.0000i
13. = 0.4000
14. = - 0.02888
15. = 23.2134
16. = 3

**Solution:**

|  |  |
| --- | --- |
| ((2^5) / (2^5 – 1)) | **A screenshot of a black screen  Description automatically generated** |

|  |  |
| --- | --- |
| 3\*((5^1/2 – 1) / (5^1/2 – 1) ^2) | **A screenshot of a black screen  Description automatically generated** |

|  |  |
| --- | --- |
| exp(3) | A black background with white text  Description automatically generated |

|  |  |
| --- | --- |
| sin(pi/6) | **A screenshot of a black screen  Description automatically generated** |

|  |  |
| --- | --- |
| cos(pi/6) | **A screenshot of a computer program  Description automatically generated** |

|  |  |
| --- | --- |
| log10(exp(3)) | **A screenshot of a computer program  Description automatically generated** |

|  |  |
| --- | --- |
| log10(exp(3)) | **A screenshot of a computer program  Description automatically generated** |

|  |  |
| --- | --- |
| tan(pi/2) | **A screenshot of a computer program  Description automatically generated** |

|  |  |
| --- | --- |
| (sin(pi/2) ^ 2) + (cos(pi/2) ^ 2) | A black background with white text  Description automatically generated |

|  |  |
| --- | --- |
| pi^1/3 – 1 | A black screen with white text  Description automatically generated |

|  |  |
| --- | --- |
| pi \* (pi^1/3 - 1) ^ 2 | A black screen with white text  Description automatically generated |

|  |  |
| --- | --- |
| (cosh(32\*pi))^2 + (sinh(32\*pi))^2 | A black screen with white text  Description automatically generated |

|  |  |
| --- | --- |
| (1+3i)/(1-3i) | A black background with white text  Description automatically generated |

|  |  |
| --- | --- |
| pi\*i/exp(4) | A black screen with white text  Description automatically generated |

**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 |

1. **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") | A graph with a blue line  Description automatically generated  Figure 15: Sine graph |

1. **On the same plot of ques 2, plot and label sin (x2) 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 |

1. **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);   1. a(3, 1) = 0;   disp(a);   1. submatrix1 = a(2:3, [1, 3]);% slicing of element   disp(submatrix1);   1. submatrix2 = a(2:3, 2:3);   disp(submatrix2);   1. submatrix3 = a(2:3, 2:3);   disp(submatrix3); |  |

|  |  |
| --- | --- |
| 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; | Figure 18: error displayed |
| 1. **Define a 3x3 matrix whose all elements are zero using zeros function**   x = zeros(3,3);%all elements are zeros  disp(x) | Figure 19: Zeros, eyes and ones matrices |
| 1. **Define a 3x3 matrix whose all elements are one using ones function**   y = eye(3);%diagonal elements are 1 and zeros elsewhere  disp(y) |
| 1. **Define a 3x3 Identity matrix using eye function.**   z = ones(3, 3);%all elements are one’s  disp(z) |
| 1. **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) | Figure 20: Changing elements in a ZEROS matric |
| 1. **Define a 5x7 matrix with random elements.**   matrix = rand(5, 7);  disp('Original Matrix:');  disp(matrix); |  |
| 1. **Extract all the elements of the second row.**   row\_2\_elements = matrix(2, :);  disp('Elements of the 2nd row:');  disp(row\_2\_elements); |
| 1. **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; |
| 1. **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); |
| 1. **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)]); |  |
| 1. **Find the size of matrix G.**   size\_G = size(G); % size of matrix G  disp('Size of matrix G:');  disp(size\_G); |
| 1. **Delete the last row and last column of matrix G.**   G(2, :) = []; % Delete the last row  G(:, 2) = []; % Delete the last column  disp(G); |
| 1. **Obtain the transpose of matrix A.**   A transpose = A.';  disp('Transpose of matrix A:');  disp(A\_transpose); |

**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); | Figure 21: Finding the y-coordinates at different x-coordinates |

1. **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 x2 + y2 = r2. 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 = (x2 + y2) 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); | 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 | 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; | 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)) | 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); | Figure 28: conversion from Celsius to Fahrenheit |

1. **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);  disp(['Sum of the infinite series: num2str(S)]); | Figure 29: Sum of infinite geometric series |

1. **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); | Figure 30: factorial of 10 |

1. **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 Xo , then at the end of n years you would have X = Xo (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)]); | Figure 31: Interest |

**Practical 8**

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

1. **Create the function f(x) = x2 – sin(x) +(1/x).**

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

|  |  |
| --- | --- |
| 1. **Find f (0), f (1) and f (pi/2).** | Figure 32: values at f(0),f(1) and f(pi/2) |
| 1. **Vectorize f and evaluate f (x) where x = [0 1 pi/2 pi]** | Figure 33: vectorize f |
| 1. **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 |
| 1. **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 |
| 1. **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 |

1. **Create a function f(x) = x3 - 3x2 + x log(x - 1) + 100**

|  |  |
| --- | --- |
| 1. **Evaluate the function at x =0, 1, 2,10 in an array form** | Figure 37: value of f(x) at different x values |
| 1. **Plot x,f(x)** | Figure 38: plot |

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

|  |  |
| --- | --- |
| 1. **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); | Figure 39: Output of the above question |
| 1. **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); |
| 1. **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); |
| 1. **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); |
| 1. **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); |
| 1. **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); |

1. **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); | 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); | Figure 41: Output of the above code |

1. **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'); | 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'); | Figure 45: Auto Correlation and Cross Correlation |