# LAB FILE

# BASIC SIMULATION LAB (ES 204)

# Image result for amity university logo

**SUBMITTED BY:**

# AASTHA GAUTAM (A2305220479)

# B.Tech. (CSE) 3 CSE7 Y

**SUBMITTED TO:**

**Ms. Shikha Bathla**

**AMITY SCHOOL OF ENGINEERING AND TECHNOLOGY**

**AMITY UNIVERSITY, UTTAR PRADESH**

**EXPERIMENT 1**

**AIM :**

Creating a One-Dimensional Array (Row / Column Vector)

Creating a Two-Dimensional

Array (Matrix of given size) and

(A). Performing Arithmetic Operations - Addition, Subtraction, Multiplication and Exponentiation.

(B). Performing Matrix operations - Inverse, Transpose, Rank.

**TOOLS USED :**

MATLAB/Octave.

**THEORY :**

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include Math and computation, Algorithm development, Data acquisition, Modeling, simulation, and rototyping, Data analysis, exploration, and visualization, Scientific and engineering graphics, Application development, including graphical user interface building. MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This allows you to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a SUGGESTED PROGRAM: in a scalar non interactive language such as C or Fortran. The name MATLAB stands for matrix laboratory.

Starting MATLAB on Windows platforms: start MATLAB by double-clicking the MATLAB shortcut icon on your Windows desktop.

Quitting MATLAB: To end your MATLAB session, select File > Exit MATLAB in the desktop, or type quit in the Command Window.

You can run a script file named my\_script.m each time MATLAB quits.

Alternative, octave-online.net can be used instead of MATLAB.

Consider two matrices A and B. If A is an m x n matrix and B is a n x p matrix, they could be multiplied together to produce an m x n matrix C. Matrix multiplication is possible only if the number of columns n in A is equal to the number of rows n in B. In matrix multiplication, the elements of the rows in the first matrix are multiplied with corresponding columns in the second matrix. Each element in the (i, j)th position, in the resulting matrix C, is the summation of the products of elements in ith row of first matrix with the corresponding element in the jth column of the second matrix. In MATLAB, matrix multiplication is performed by using the \* operator. The inverse of a matrix does not always exist. If the determinant of the matrix is zero, then the inverse does not exist, and the matrix is singular. In MATLAB, inverse of a matrix is calculated using the inv function. Inverse of a matrix A is given by inv(A).

The transpose operation switches the rows and columns in a matrix. It is represented by a single quote('). The rank function provides an estimate of the number of linearly independent rows or columns of a full matrix. k = rank(A) returns the number of singular values of A that are larger than the default tolerance.

Text

Description automatically generatedText

Description automatically generated

Graphical user interface, text, application

Description automatically generatedA screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated**A picture containing text, electronics

Description automatically generated**

**RESULT :**

One dimensional matrix and two dimensional matrix have been formed. All the arithmetic operation (addition, subtraction, multiplication, division and exponentiation) and matrix operations (inverse, rank, transpose) have been successfully performed

**PRECAUTIONS :**

1. Connection should be strong.

2. Put the instructions carefully to make the Maurice.

3. Perform the operations carefully.

4. Order of Matrices should be correct for various operations.

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

**EXPERIMENT 2**

**AIM**

**Performing Matrix Manipulations:**

Concatenating, Indexing, sorting, Shifting, Reshaping, Resizing and Flipping about Vertical axis / Horizontal axis.

**Performing Relational Operations -** >, <, ==, <=, >=, ~=

**Performing Logical Operations-** ~, &, | , XOR

**DESCRIPTION**

* **Concatenation:** It refers to Combining two vectors or matrices. There are two types of Concatenation:

**1. Horizontal Concatenation –** Column wise Concatenation. In this, two vectors or matrices are combined through columns. Rows of the Matrices / Vectors should be same. C=[A, B] In this syntax “,” Represents Horizontal Concatenation.

C=[dim, A, B] in this syntax, “dim” Represents Concatenation. Dim=2 refers to horizontal concatenation.

Built in Function can be used, which is horzcat(A, B).

**2. Vertical Concatenation -** Row wise Concatenation. In this, two vectors or matrices are combined through rows. Columns of the Matrices / Vectors should be same.

C=[A;B] In this syntax “ ; “ Represents Vertical Concatenation.

C=[dim, A, B] in this syntax, “dim” Represents Concatenation. Dim=1 refers to Vertical concatenation.

Built in function can be used, which is vertcat(A, B)

* **Indexing:** Accessing elements of Matrices with the help of their position (index) is known as Indexing. C=A(dim1, dim2) In this syntax, dim1 and dim2 are the indices of the matrix. In order to access specify block of elements we can use “ : “ operator- For example

C=A(dim1: dim2, dim3: dim4) or C=A(:) or C=A(: , :)

* **Sorting:** It refers to Sorting elements of the matrices in either ascending or descending order. By Default, the Sorting technique follows ascending order. sort(A) sorts the elements in ascending order.

sort(A, ‘descend’) sorts the elements in descending order.

* **Reshaping:** Reshape / Resize a matrix with order MxN into NxM (in an order where numbers of elements will be same) by retaining all the elements. It is different from transpose as it reshapes the elements from column.

C=reshape(A, [dim1, dim2]). Suppose matrix A is of dim [3x4] then possible reshaping can be C=reshape(A, [4,3]) or C=reshape(A, [2,6]) or C=reshape(A, [1,12]) or C=reshape(A, [12,1]).

* **Shifting:** **Circular Shift-** Shifts the elements of matrix or arrays by ‘N’ numbers of time. C=circshift(A,N) circularly shifts the elements in array A by N positions. If N is an integer, then circshift shifts along the first dimension of A whose size does not equal 1.

If N is a vector of integers, then each element of N indicates the shift amount in corresponding dimension of A. the

* **Fliping:** Fliping Elements along rows or columns. Fliping is of 2 types: fliplr and flipud. In flipud- FLIP Up and Down is flipping elements along rows i.e., C= flipud(A) returns A with its rows flipped up-down (that is, about a horizontal axis). In fliplr- FLIP Left and Right is Fliping Elements along columns i.e., C=fliplr(A) return A with its Column flipped left-right ( that is, about Vertical axis).
* **Relational Operators:** Relational operators Compare the elements in two arrays. Result is equal to 1 (if the comparison is TRUE) and 0 (if the comparison is FALSE) i.e., Bool Values. It compares element of matrix A with element of matrix B wrt to Index. Element by element comparison. Syntax are as follows –

Table

Description automatically generated

* **Logical Operators:** Logical Operators checks for the elements in an array or matrix. Result is 1 (if the operation is true) and 0 (if the operation is false)

& ,and - checks for non-zero element in both matrices. Both elements at particular index should be non-zero (A&B)

| ,or - checks for non-zero element in both matrices. One of the elements at particular index should be non-zero (A|B)

xor - checks for non-zero element in both matrices. Both elements at particular index should be 1 (xor(A,B))

~,not – This returns a logical array of the same size as A. The arrays Contains 1 for every Zero value and Contains 0 for every non-zero values (~A)

**Programs and Outputs**

* **SOURCE CODE**

a = [1 2 3 4];

b = [5 6 7 8];

c = [a b];

d = [a; b];

e = [b a];

A = [1 1 2 2; 3 3 4 4; 5 5 6 6; 7 7 8 8];

disp("");

disp("Arrays");

disp("");

disp(a);

disp("");

disp(b);

disp("");

disp(A);

disp("");

disp("Concatenating");

disp("");

disp(c);

disp("");

disp(d);

disp("");

disp("Array Indexing");

disp("");

fprintf('d(1,2) = %d',d(1,2));

disp("");

disp("");

disp("Array Sorting");

disp("");

disp(" Original");

disp(e);

disp("");

disp(" Ascending");

disp(sort(e,'ascend'));

disp("");

disp(" Descending");

disp(sort(e,'descend'));

disp("");

disp("Shifting");

disp("");

disp(" Original");

disp(A);

disp("");

disp(" circshift(A,2)");

disp(circshift(A,2));

disp("");

disp(" circshift(A,1,1)");

disp(circshift(A,1,1));

disp("");

disp("Array Reshaping");

disp("");

disp(" Original");

disp(A);

disp("");

disp(" reshape(A,[8 2])");

disp(reshape(A,[8 2]));

disp("");

disp(" reshape(A,[8 2])");

disp(reshape(A,[2 8]));

disp("");

disp("Flipping");

disp("");

disp(" Original");

disp(A);

disp("");

disp(" flipud(A)");

disp(flipud(A));

disp("");

disp(" fliplr(A)");

disp(fliplr(A));

disp("");

disp("Rotating");

disp("");

disp(" Original");

disp(A);

disp("");

disp(" rot90(A)");

disp(rot90(A));

disp("");

disp(" rot90(rot90(A))");

disp(rot90(rot90(A)));

a = [1 0 1 1 0];

b = [0 0 0 1 1];

disp("-----------");

disp("a =");

* **OUTPUT**

**A picture containing calendar

Description automatically generated**

**Calendar

Description automatically generated**

**CONCLUSION :**

One dimensional matrix and two dimensional matrix have been formed. All the arithmetic operation (addition, subtraction, multiplication, division and exponentiation) and matrix operations (inverse, rank, transpose) have been successfully performed.

**PRECAUTIONS:**

1)Connection should be uninterrupted.

2)Put the instructions carefully to make the Matrices.

3)Perform the operations carefully.

4)Order of Matrices should be correct for various operation

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

**EXPERIMENT 3**

**AIM :**

a) To generate Random Sequence and plot them

b) To calculate sum matrix, cumulative sum matrix and plot the sum matrix.

**TOOLS USED :**

MATLAB/Octave.

**THEORY :**

When you use software to generate random numbers, the outcomes aren't truly random. Software solutions like MATLAB®, on the other hand, use methods to make your results appear random and independent. The results also pass different randomization and independence statistical tests. Pseudorandom and pseudo-independence are terms used to describe these values that appear to be random and independent. These numbers can be used as if they were truly random and independent. The ability to repeat a random number computation at any moment is one of the advantages of employing pseudorandom, pseudo independent numbers. In testing or diagnostic scenarios, this can be beneficial.

Although repeatability is beneficial, it is feasible to repeat your results when you actually want something different. There are numerous options for avoiding this issue. Several examples are included in the documentation to demonstrate how to ensure that your results are different when that is your goal. Rand, randn, randi, and randperm are all random number functions that use the same random number generator. The generator resets itself to the same state every time you start MATLAB®. As a result, if you run a command like rand(2,2) right after startup, you'll get the same result every time. Furthermore, any script or function that calls the random number functions produces the same result every time you restart.

**PROCEDURE :**

Rand, randi, randn, and randperm are the four basic random number functions.

1). The rand function returns a set of real values ranging from 0 to 1 selected from a uniform distribution.

For example, r1 = rand(1000,1) contains real floating-point integers selected from a uniform distribution and is a 1000-by-1 column vector. The open interval encompasses all of the values in r1 (0, 1). These data have a histogram that is essentially flat, indicating a very uniform sampling of numbers.

2). The randi function returns a discrete uniform distribution of double integer values.

For example, r2 = randi(10,1000,1) has integer values drawn from a discrete uniform distribution with a range of 1,2,...,10; r2 is a 1000-by-1 column vector containing integer values taken from a discrete uniform distribution with a range of 1,2,...,10. These numbers have a histogram that is essentially flat, indicating a very uniform sampling of integers between 1 and 10. These numbers have a histogram that is essentially flat, indicating a very uniform sampling of integers between 1 and 10.

3). The randn function returns an array of real floating-point values drawn from a normal distribution. For example, r3 = randn(1000,1); r3 is a 1000-by-1 column vector containing numbers drawn from a standard normal distribution. A r3 histogram resembles a normal distribution with a mean of 0 and a standard deviation of 1.

4). The randperm function can be used to generate arrays of random integer values with no repeats. For example, r4 = randperm(15,5); r4 is a 1-by-5 array containing randomly selected integer values on the closed interval, [1, 15]. Unlike randi, which can yield an array with repeated values, randperm can’t return an array with repeated values. If you call any of these functions again, you'll get different results. This characteristic is useful for constructing a variety of random value arrays.

**PROGRAMS:**

**1).**

x=rand(5,5);

display(x);

plot(x);

title("Random Sequence");

xlabel("Random variable");

ylabel('f(x)');

**2).**

z=randn(3,3);

display(z);

plot(z);

title("random function");

xlabel("random variable");

ylabel('f(x)');

**3).**

A=magic(4,4);

X=cumsum(A,1);

Y=cumsum(A,2);

Z=sum(A);

V=sum(A,1);

U=sum(A,2);

**4.)**

z=randi(3,3);

display(z);

plot(z);

title("random function");

xlabel("random variable");

ylabel('f(x)');

**OUTPUTS:**

**Diagram

Description automatically generatedA picture containing graphical user interface

Description automatically generated**

A picture containing engineering drawing

Description automatically generatedA picture containing calendar

Description automatically generated

**RESULT :**

A) Random Sequence is generated and plotted.

B) Sum matrix, cumulative sum matrix are calculated and the sum matrix is plotted.

**PRECAUTION :**

1. Connection should be Strong.

2. Put the instructions carefully to make the matrix.

3. Perform the operations carefully.

4. Order of matrix should be correct for various operations.

|  |  |  |  |
| --- | --- | --- | --- |
| CRITERIA | TOTAL MARKS | MARKS OBTAINED | COMMENTS |
| CONCEPT(A) | 2 |  |  |
| IMPLEMENTATION(B) | 2 |  |  |
| PERFORMANCE(C) | 2 |  |  |
| TOTAL | 6 |  |  |

**EXPERIMENT– 4**

**AIM:**

Evaluating a given expression and rounding it to the nearest integer value using Round, Floor, Ceil and Fix functions; Also, generating and Plots of Trigonometric Functions - sin(t), cos(t), tan(t), sec(t), cosec(t) and cot(t) for a given duration, ‘t’.

**TOOLS USED:**

OCTAVE

**THEORY:**

* **Round-**

Round to nearest integer

Syntax: Y = round(X)

Description: Y = round(X) rounds the elements of X to the nearest integers. For complex X, the imaginary and real parts are rounded independently.

* **Floor-**

Round towards minus infinity

Syntax B = floor(A)

Description B = floor(A) rounds the elements of A to the nearest integers less than or equal to A. For complex A, the imaginary and real parts are rounded independently.

* **Ceil-**

Round toward infinity

Syntax: B = ceil(A)

Description: B = ceil(A) rounds the elements of A to the nearest integers greater than or equal to A. For complex A, the imaginary and real parts are rounded independently.

* **Fix-**

Round towards zero

Syntax: B = fix(A)

Description: B = fix(A) rounds the elements of A toward zero, resulting in an array of integers. For complex A, the imaginary and real parts are rounded independently.

* **Trigonometric Functions-**

The trigonometric functions are used along with their names and have the angle value as the parameters in them. A range for the value of the parameter is defined to attain their graphical representations.

Syntax: A = ‘trigonometric function name’(Value)

Ex. sin(30), cos(115), tan(-45)

These functions include sin, cos, tan, cot, cosec and sec function.

**PROCEDURE:**

To plot the graph of a function, you need to take the following steps:

1. Define x, by specifying the range of values for the variable x, for which the function is to be plotted.

2. Define the function, y = f(x).

3. Call the plot command, as plot(x, y).

**OBSERVATIONS:**

**Code:**

disp("Original number:")

X = randn (5)

disp("round(X):")

Y = round(X)

disp("floor(X):")

Y = floor(X)

disp("fix(X):")

Y = fix(X)

disp("Sin(x):")

t = [0:0.1:2\*pi];

a = sin(t);

plot(t,a)

disp("Cos(x):")

t = [0:0.1:2\*pi];

a = cos(t);

plot(t,a)

disp("tan(x):")

x1 = -pi/2+0.01:0.01:pi/2-0.01;

x2 = pi/2+0.01:0.01:(3\*pi/2)-0.01;

plot(x1,tan(x1),x2,tan(x2))

disp("cot(x):")

x1 = -pi+0.01:0.01:-0.01;

x2 = 0.01:0.01:pi-0.01;

plot(x1,cot(x1),x2,cot(x2))

disp("Sec(x):")

x1 = -pi/2+0.01:0.01:pi/2-0.01;

x2 = pi/2+0.01:0.01:(3\*pi/2)-0.01;

plot(x1,sec(x1),x2,sec(x2))

disp("cot(x):")

x1 = -pi+0.01:0.01:-0.01;

x2 = 0.01:0.01:pi-0.01;

plot(x1,cot(x1),x2,cot(x2))

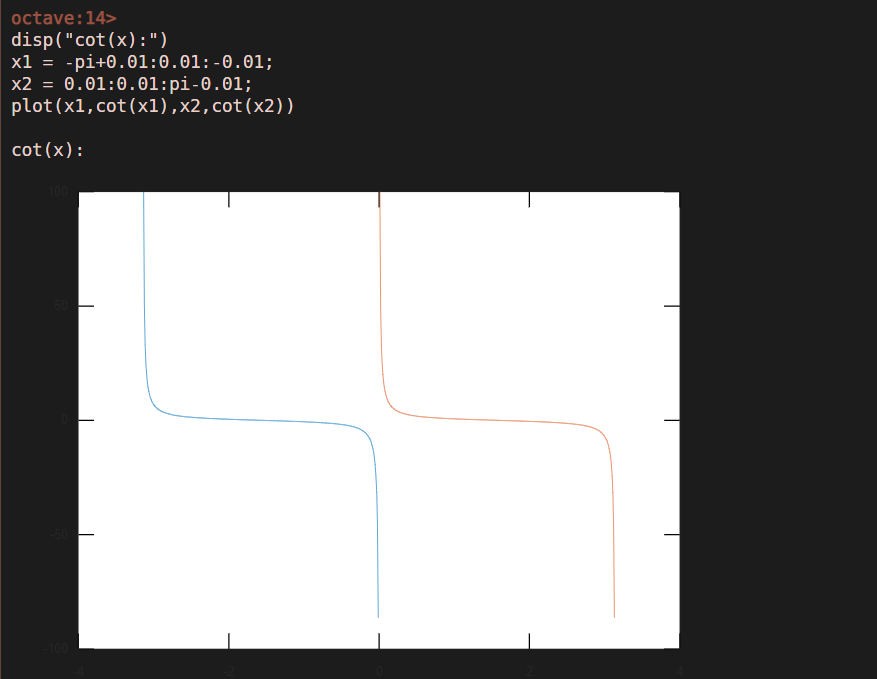
Calendar

Description automatically generated with low confidence**OUTPUT:**

Chart, histogram

Description automatically generatedChart, histogram

Description automatically generated

A picture containing shape

Description automatically generated

A picture containing diagram

Description automatically generatedShape

Description automatically generated with low confidence

A picture containing diagram

Description automatically generatedChart

Description automatically generated

A picture containing graphical user interface

Description automatically generatedChart

Description automatically generated with medium confidence

Diagram

Description automatically generated with medium confidence**INTERPRETATION OF GRAPHS:**

when we plot the graphs of trigonometric functions, we observe that with change in amplitude and frequency, the graphs of various trigonometric functions also vary. While plotting the graph for fix function, we see that graph is broken. We also observe that we are getting a straight line starting from 0 till the last number we get after plotting the graph of the round function.

**RESULTS AND DISCUSSIONS:**

rounding off the numbers to the nearest integers using various functions was done successfully. Moreover, graphs of various functions like trigonometric functions, logarithmic functions, rounding functions etc. was also done successfully. Graphs were also comprehended successfully.

**PRECAUTIONS:**

1. Book the session on amity virtual lab beforehand
2. Make sure that you have a stable internet connection
3. Save your program before running it

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

1. In case connection is lost, refresh your page

**EXPERIMENT 5**

**AIM**

**1) Create a vector Z with elements, 𝒁=(−1)n+1(2n−1)Add up 100 elements of the vector Z. Plot Z.**

**2) Plot the functions, x, x3, ex, exp(x2) over the interval 0<x<4 (by choosing appropriate mesh values for x to obtain smooth curves), on a rectangular plot.**

**DESCRIPTION**

* **Addition (+) :** C = [A](https://in.mathworks.com/help/matlab/ref/plus.html#btx03j8-1-A) + [B](https://in.mathworks.com/help/matlab/ref/plus.html#btx03j8-1-A) adds arrays A and B by adding corresponding elements. If one input is a string array, then plus appends the corresponding elements as strings.
* **Subtraction (-):** C = [A](https://in.mathworks.com/help/matlab/ref/minus.html#btx9ad2-A) - [B](https://in.mathworks.com/help/matlab/ref/minus.html#btx9ad2-A) subtracts array B from array A by subtracting corresponding elements. The sizes of A and B must be the same or be [compatible](https://in.mathworks.com/help/matlab/matlab_prog/compatible-array-sizes-for-basic-operations.html).
* **Multiplication (\*):** [C](https://in.mathworks.com/help/matlab/ref/mtimes.html#btx9i74-C) = [A](https://in.mathworks.com/help/matlab/ref/mtimes.html#btx9i74-A)\*[B](https://in.mathworks.com/help/matlab/ref/mtimes.html#btx9i74-A) is the matrix product of A and B. If A is an m-by-p and B is a p-by-n matrix, then C is an m-by-n matrix defined by
* **Element by Element Multiplication (.\*):** C = [A](https://in.mathworks.com/help/matlab/ref/times.html#btx_6a1-A).\*[B](https://in.mathworks.com/help/matlab/ref/times.html#btx_6a1-A) multiplies arrays A and B by multiplying corresponding elements.
* **Division (/):** [x](https://in.mathworks.com/help/matlab/ref/mrdivide.html#btg5p6j-x) = [B](https://in.mathworks.com/help/matlab/ref/mrdivide.html#btg5p6j-A)/[A](https://in.mathworks.com/help/matlab/ref/mrdivide.html#btg5p6j-A) solves the system of linear equations x\*A = B for x.
* **Element by Element Division ( . /):** x = [A](https://in.mathworks.com/help/matlab/ref/rdivide.html#btg5tj1-1-A)./[B](https://in.mathworks.com/help/matlab/ref/rdivide.html#btg5tj1-1-A) divides each element of A by the corresponding element of B.
* **Power (^):** If A is a square matrix and p is a positive integer, then A^p effectively multiplies A by itself p-1 times
* **Element-wise power ( .^):** C = [A](https://in.mathworks.com/help/matlab/ref/power.html#btx_7d7-A).^[B](https://in.mathworks.com/help/matlab/ref/power.html#btx_7d7-A) raises each element of A to the corresponding powers in B. The sizes of A and B must be the same or be [compatible](https://in.mathworks.com/help/matlab/matlab_prog/compatible-array-sizes-for-basic-operations.html).
* **Exponentiation ( exp ):** [Y](https://in.mathworks.com/help/matlab/ref/exp.html#bt9_gkt-1-Y) = exp([X](https://in.mathworks.com/help/matlab/ref/exp.html#bt9_gkt-1-X)) returns the exponential ex for each element in array X.
* **Sum:**  S = sum([A](https://in.mathworks.com/help/matlab/ref/sum.html#btv6ok6-1-A)) returns the sum of the elements of A along the first array dimension whose size does not equal 1.
* **2-D Line Plot:** plot([X](https://in.mathworks.com/help/matlab/ref/plot.html#btzitot-X),[Y](https://in.mathworks.com/help/matlab/ref/plot.html#btzitot-Y)) creates a 2-D line plot of the data in Y versus the corresponding values in X.

plot([X](https://in.mathworks.com/help/matlab/ref/plot.html#btzitot-X),[Y](https://in.mathworks.com/help/matlab/ref/plot.html#btzitot-Y),[LineSpec](https://in.mathworks.com/help/matlab/ref/plot.html#btzitot_sep_mw_3a76f056-2882-44d7-8e73-c695c0c54ca8)) sets the line style, marker symbol, and color.

* xlabel([txt](https://in.mathworks.com/help/matlab/ref/xlabel.html?searchHighlight=xlabel&s_tid=srchtitle#btpmg0w-1_sep_shared-txt)) labels the x-axis of the current axes or standalone visualization.

ylabel([txt](https://in.mathworks.com/help/matlab/ref/ylabel.html?s_tid=doc_ta#buiyzbx-1_sep_shared-txt)) labels the y-axis of the current axes or standalone visualization

legend creates a legend with descriptive labels for each plotted data series.

* **Subplot:** subplot([m](https://in.mathworks.com/help/matlab/ref/subplot.html#btw1t4b-1-m),[n](https://in.mathworks.com/help/matlab/ref/subplot.html#btw1t4b-1-n),[p](https://in.mathworks.com/help/matlab/ref/subplot.html#btw1t4b-1-p)) divides the current figure into an m-by-n grid and creates axes in the position specified by p. MATLAB® numbers subplot positions by row. The first subplot is the first column of the first row, the second subplot is the second column of the first row, and so on.

Table

Description automatically generated

**PROCEDURE**

To plot the graph of a function, you need to take the following steps:

1. Define x, by specifying the range of values for the variable x, for which the function is to be plotted.

2. Define the function, z = f(x).

3. Call the plot command, as plot(x, y).

**PROGRAMS AND OUTPUTS**

**1. Creating a vector Z with elements, 𝒁=(−1)n+1(2n−1) Adding up 100 elements of the vector Z and Plotting Z.**

* **SOURCE CODE**

clc;clear all;

v=[1:100];

z=(-1).^(v+1)./(2.\*v-1);

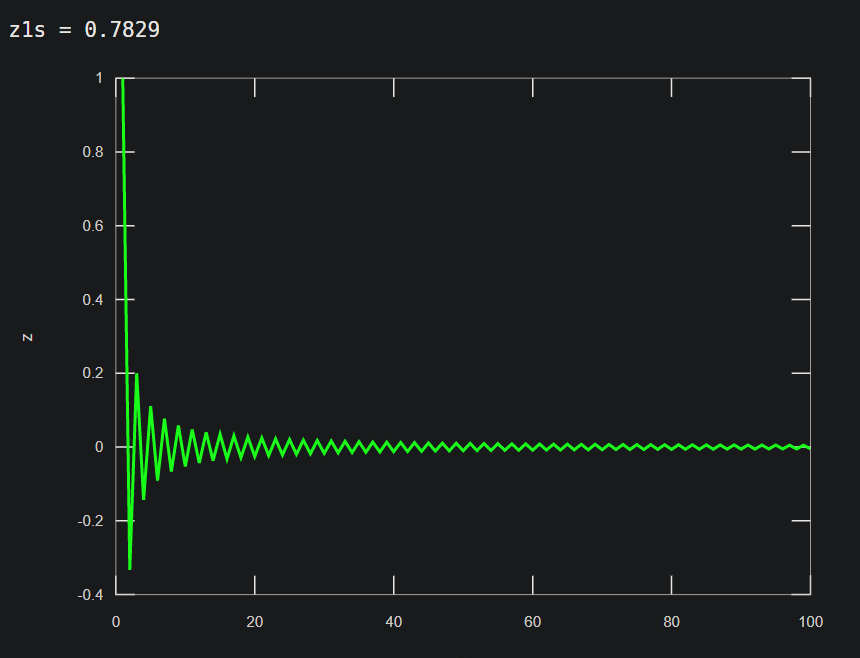
plot(z,'linewidth',2,'g')

xlabel('no of terms')

ylabel('z')

z1s=sum(z)

* **OUTPUT**

****

**2. Plotting the functions, x, x3, ex, exp(x2) over the interval 0<x<4 (by choosing appropriate mesh values for x to obtain smooth curves), on a rectangular plot.**

**PROCEDURE**

To plot the graph of a function, you need to take the following steps:

1. Define x, by specifying the range of values for the variable x, for which the function is to be plotted.

2. Define the function, z = f(x).

3. Call the plot command, as subplot(x, y).

* **SOURCE CODE**

clc;clear all;

x=[0:0.1:4];

subplot(2,2,1)

plot(x,'linewidth',2.5);

xlabel('Number of terms');

legend ('X');

y=x.^3;

subplot(2,2,2)

plot(y,'linewidth',2.5)

xlabel('Number of terms');

legend ('X^3');

z=exp(x);

subplot(2,2,3)

plot(z,'linewidth',2.5)

xlabel('Number of terms');

legend('e(x)');

d=x.^2;

w=exp(d);

subplot(2,2,4)

plot(w,'linewidth',2.5)

xlabel('Number of terms');

legend('e(x^2)');

* Graphical user interface

  Description automatically generated**OUTPUT**

**INTERPRETATION OF GRAPHS**

We find that if the values are tending to infinity, then the graph is usually blank (it doesn’t show nothing because its impossible to plot infinite values on graph). Also, the graphs of various polynomial functions differ according to the values and the equation involved. The graphs of exponential and power functions usually tend to infinity for infinite values.

**RESULTS AND DISCUSSIONS**

Plotting of various random vectors was done successfully. Moreover, graphs of various functions like polynomial functions, power function and exponential functions etc. was also done successfully. Graphs were also comprehended successfully.

**PRECAUTIONS**

1. Book the session on amity virtual lab beforehand
2. Make sure that you have a stable internet connection
3. Save your program before running it

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

1. In case connection is lost, refresh your page

**EXPERIMENT 6**

**AIM**

1) Generating a Sinusoidal Signal of a given Frequency with Titling, Labeling, Adding Text, Adding Legends, Printing Text in Greek Letters, Plotting as Multiple and Subplot.

2) Time Scale the generated signal for Different values. E.g. 2X, 4X, 0.25X, 0.0625X.

**DESCRIPTION**

* **Sin(t):** [Y](https://in.mathworks.com/help/matlab/ref/sin.html#bt5p3vk-1-Y) = sin([X](https://in.mathworks.com/help/matlab/ref/sin.html#bt5p3vk-1-X)) returns the sine of the elements of X. The sin function operates element-wise on arrays.
* **Cos(t):** [Y](https://in.mathworks.com/help/matlab/ref/double.cos.html#bt5p4al-1-Y) = cos([X](https://in.mathworks.com/help/matlab/ref/double.cos.html#bt5p4al-1-X)) returns the cosine for each element of X. The cos function operates element-wise on arrays.
* **2-D Line Plot:** plot([X](https://in.mathworks.com/help/matlab/ref/plot.html#btzitot-X),[Y](https://in.mathworks.com/help/matlab/ref/plot.html#btzitot-Y)) creates a 2-D line plot of the data in Y versus the corresponding values in X.

plot([X](https://in.mathworks.com/help/matlab/ref/plot.html#btzitot-X),[Y](https://in.mathworks.com/help/matlab/ref/plot.html#btzitot-Y),[LineSpec](https://in.mathworks.com/help/matlab/ref/plot.html#btzitot_sep_mw_3a76f056-2882-44d7-8e73-c695c0c54ca8)) sets the line style, marker symbol, and color.

* **xlabel:** xlabel([txt](https://in.mathworks.com/help/matlab/ref/xlabel.html?searchHighlight=xlabel&s_tid=srchtitle#btpmg0w-1_sep_shared-txt)) labels the x-axis of the current axes or standalone visualization.
* **ylabel:** ylabel([txt](https://in.mathworks.com/help/matlab/ref/ylabel.html?s_tid=doc_ta#buiyzbx-1_sep_shared-txt)) labels the y-axis of the current axes or standalone visualization
* **legend:** legend creates a legend with descriptive labels for each plotted data series.
* **title:** title([titletext](https://in.mathworks.com/help/matlab/ref/title.html#btpi3rq-1-txt)) adds the specified title to the current axes or standalone visualization.
* **text:** text([x](https://in.mathworks.com/help/matlab/ref/text.html#f68-481090-x),[y](https://in.mathworks.com/help/matlab/ref/text.html#f68-481090-y),[txt](https://in.mathworks.com/help/matlab/ref/text.html#f68-481090-txt)) adds a text description to one or more data points in the current axes using the text specified by txt.
* **Subplot:** subplot([m](https://in.mathworks.com/help/matlab/ref/subplot.html#btw1t4b-1-m),[n](https://in.mathworks.com/help/matlab/ref/subplot.html#btw1t4b-1-n),[p](https://in.mathworks.com/help/matlab/ref/subplot.html#btw1t4b-1-p)) divides the current figure into an m-by-n grid and creates axes in the position specified by p. The first subplot is the first column of the first row, the second subplot is the second column of the first row, and so on.

Table

Description automatically generated

**PROGRAMS AND OUTPUTS**

**1. Generating a Sinusoidal Signal of a given Frequency and time scaling the generated signal for Different values of t**

* **SOURCE CODE**

1.

clear all

t=0:0.01:0.5

a=1

f=100

x=a.\*sin(2\*pi\*f\*t)

hold all

plot(t,x)

title('plot of sinusoidal signal')

xlabel('time')

ylabel('sin function')

legend('sine function')

plot(t,x,'--+g')

title('plot of sinosoidal signal')

xlabel('time')

ylabel('sine function')

legend('sine)

plot(t,x,'--rs','linewidth',2,'Markeredgecolor','y','Markerfacecolor','b','Markersize',6)

title('plot of sinosoidal signal')

xlabel('time')

ylabel('sin(theta)')

legend('sine function','sine','sine new')

2.

t=-0.25:0.0001:0.25;

f1=3;

y1=sin(2\*pi\*f1\*t);

y2=sin(2\*pi\*f1\*2\*t);

y3=sin(2\*pi\*f1\*4\*t);

y4=sin(2\*pi\*f1\*0.25\*t);

y5=sin(2\*pi\*f1\*0.625\*t);

plot(t,y1,'k',t,y2,'g',t,y3,'b',t,y4,'m',t,y5,'r')

xlabel('Time(-0.2 < x < 0)')

ylabel(' Amplitude (sine values)')

title('Graph of sine waves having different time value')

legend('y1','y2','y3','y4','y5')

3.

clc;clear all;clear vars; clear;

subplot(4,1,1)

t = [0:0.1:4\*pi];

a = sin(t);

plot(t,a)

title(['\color{red}Sin(x)']);

subplot(4,1,2)

t = [0:0.1:4\*pi];

a = cos(t);

plot(t,a)

title(['\color{white}Cos(x)']);

subplot(4,1,3)

x1 = -pi/2+0.01:0.01:pi/2-0.01;

x2 = pi/2+0.01:0.01:(3\*pi/2)-0.01;

plot(x1,tan(x1),x2,tan(x2))

title(['\color{white}Tan(x)']);

subplot(4,1,4)

x1 = -pi+0.01:0.01:-0.01;

x2 = 0.01:0.01:pi-0.01;

plot(x1,cot(x1),x2,cot(x2))

title(['\color{white}Cot(x)']);

4.

t=0:0.002:.3

a=1

f=50

x=a.\*sin(2\*pi\*f\*t)

hold all

plot(t,x)

xlabel('time'); ylabel('sine function'); title('plot of sinosinal wave')

legend('for the sine function')

plot(t,x,'--+c')

xlabel('time'); ylabel('sine function'); title('plot of sinosinal wave')

legend('for the sine wave')

plot(t,x,'--+b')

plot(t,x,'--rs','linewidth',1,'markerfacecolor','g','markersize',5)

plot(t,x,'--ys')

plot(t,x,'linewidth',3)

plot(t,x,'markeredgecolor','b')

xlabel('time'); ylabel('sine(thetha)'); title('plot of sinosinal signal wave')

legend('sine new')

xlabel('time'); ylabel('sine new'); title('plot of sinosinal signal wave')

plot(t,x,'--+k')

5.

clc;clear all;clear vars; clear;

x = linspace(0,10,50);

y1 = sin(x); p1=plot(x,y1,'g'); title('Combine Plots'); hold on;

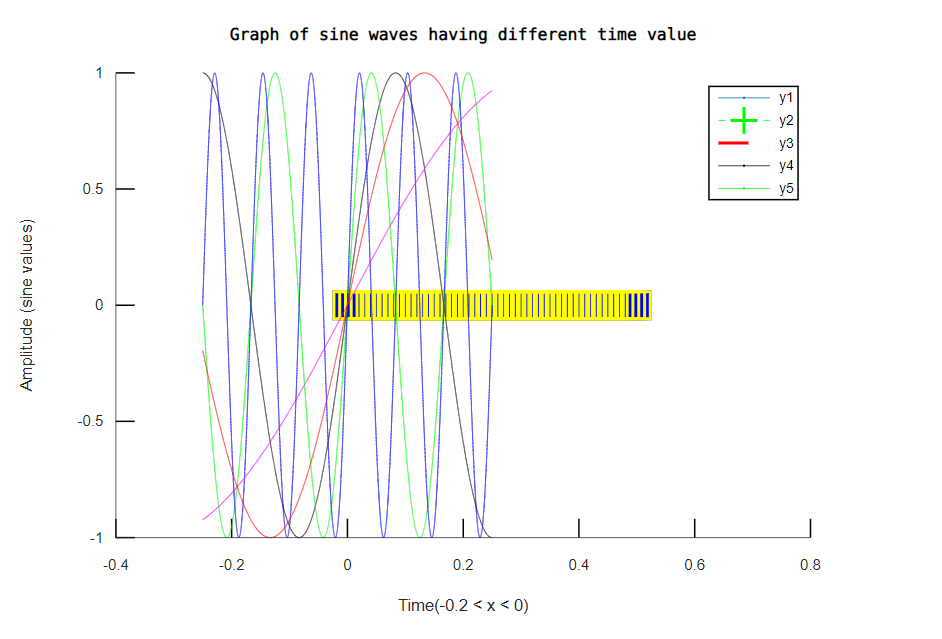
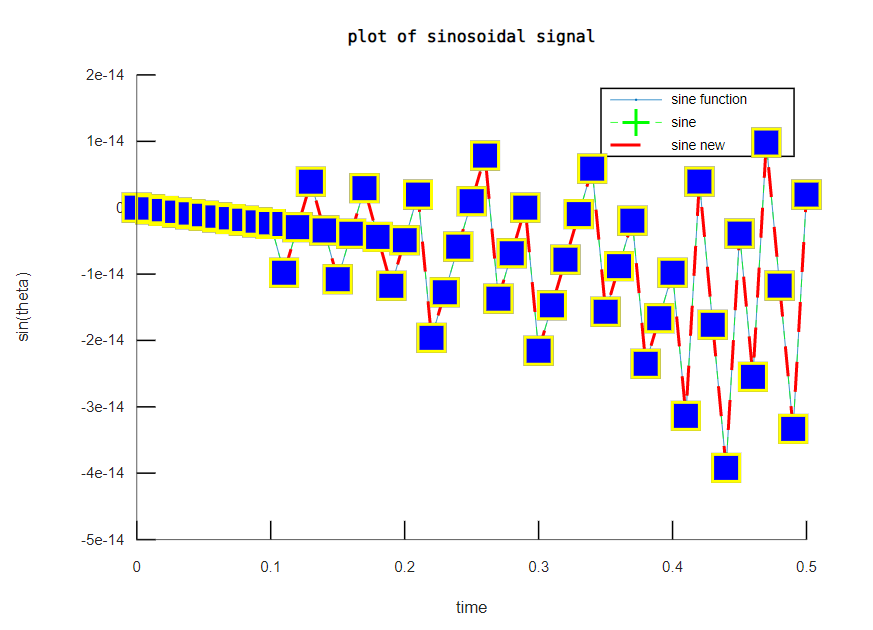
y2 = cos(x); p2=plot(x,y2,'rs');

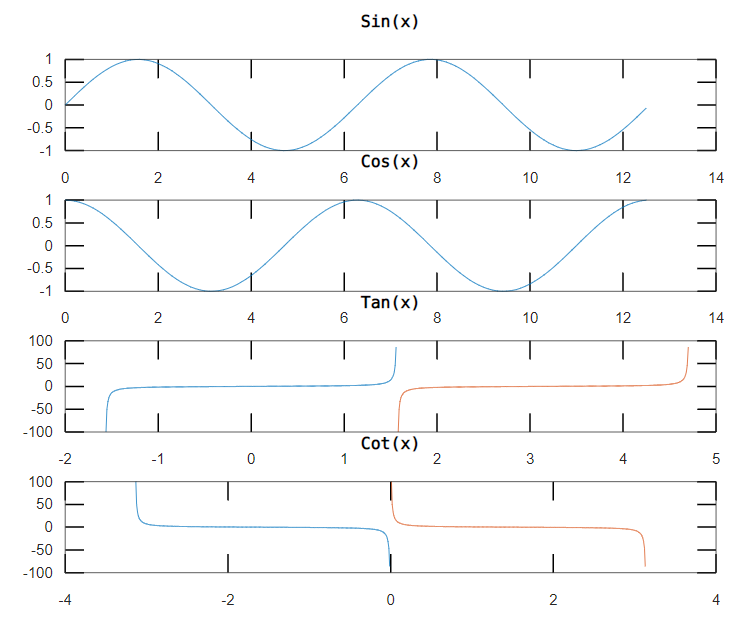
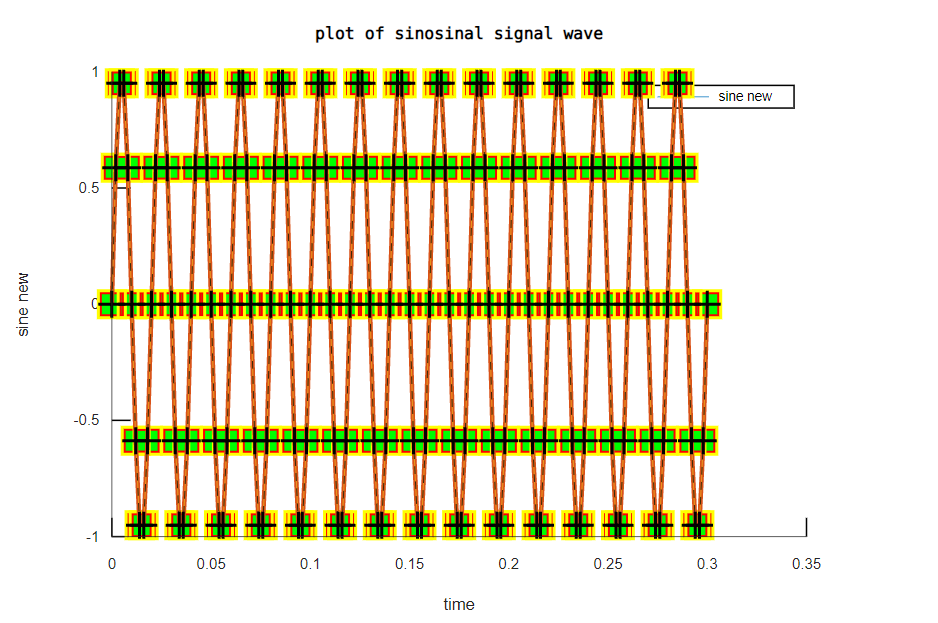
y3 = 2\*sin(x); p3=plot(x,y3','yc'); hold off;

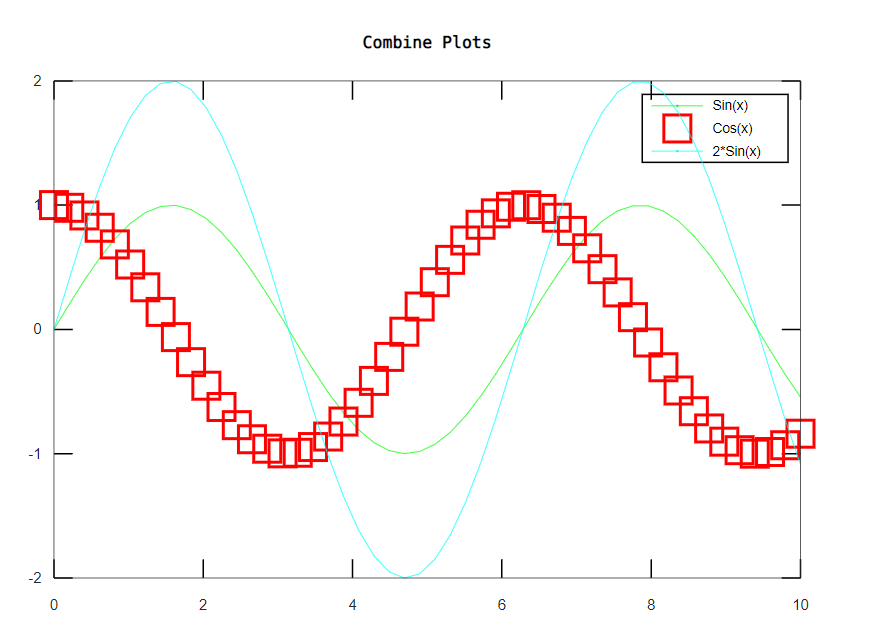
h = [p1(1);p2;p3(1)];

legend(h,'Sin(x)','Cos(x)','2\*Sin(x)');

* **OUTPUT**

****

****

****

**INTERPRETATION OF GRAPHS:**

We observe that in order to plot multiple plots in a single graph, we use hold on command so that the previous graph which was plotted will not be erased. Further, while plotting subplots, we should keep in mind the space provided in the canvas. If the space is too small, then we will not be able to plot subplots in the required canvas. For adding text, the font size should not be too large otherwise it will not be plotted in the command window.

**RESULTS AND DISCUSSIONS:**

Plotting of sinusoidal function was done successfully. Moreover, various graphical enhancements were also used to plot different types of graphs. Graphs were also comprehended successfully.

**PRECAUTIONS**

* Book the session on amity virtual lab beforehand
* Make sure that you have a stable internet connection
* Save your program before running it
* In case connection is lost, refresh your page

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

**EXPERIMENT- 7**

**AIM**

Solve Ordinary Differential Equation using Built-in Functions and plot.

➢ First Order Ordinary Differential Equation

➢ Second Order Ordinary Differential Equation, and

➢ Third Order Ordinary Differential Equation

**DESCRIPTION**

* **ODE45: [t,y] =** ode45(odefun,tspan,y0), where tspan = [t0 tf], integrates the system of differential equations y′=f(t,y) from t0 to tf with initial conditions y0. Each row in the solution array y corresponds to a value returned in column vector t.
* **@: A function handle** is a MATLAB® data type that stores an association to a function. Indirectly calling a function enables you to invoke the function regardless of where you call it from. Typical uses of function handles include:
  + Passing a function to another function (often called function functions). For example, passing a function to integration and optimization functions, such as integral and fzero.

**PROGRAMS AND OUTPUTS**

**SOURCE CODE**

clc; clear all;

tspan = [0,2];

y0 = 2;

diffeq1 = @(t,y)[5];

[T,Y] = ode45(diffeq1, tspan, y0);

figure;

plot(T,Y,'linewidth',1.6)

title('First order ODE - dy/dt = 5');

ylabel('Value of Y');

xlabel('Value of T');

# Second Order ODE

diffeq2 = @(t,y)[y(2);0];

tspan = [0,2];

y0 = [3;4];

[T,Y] = ode45(diffeq2, tspan, y0);

figure;

plot(T,Y,'linewidth',1.6)

title('Second order ODE - d^2y/dt^2 = 0');

ylabel('Value of Y');

xlabel('Value of T');

legend('y\_1' , 'y\_2');

#third order ODE

diffeq3 = @(t,y)[y(2); y(3); 0];

tspan = [0,2];

y0 = [3;4;6];

[T,Y] = ode45(diffeq3, tspan, y0);

figure;

plot(T,Y,'linewidth',1.6)

title('Third order ODE - d^3y/dt^3 = 0');

ylabel('Value of Y');

xlabel('Value of T');

legend('y\_1' ,'y\_2','y\_3');

* **OUTPUT**

**A picture containing chart

Description automatically generated**

**A picture containing line chart

Description automatically generated**

**Chart, line chart

Description automatically generated**

**RESULTS AND DISCUSSIONS**

First, Second and Third Order ODE were Solved Using Built-in Functions like

ode45 and Function handle (in this exp it is - @) and the solution of these

Equations were plotted on the graph (x vs y) with the help of plot Function.

**PRECAUTIONS**

1. Book the session on amity virtual lab beforehand
2. Make sure that you have a stable internet connection
3. Save your program before running it

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

1. In case connection is lost, refresh your page

**EXPERIMENT 8**

**AIM**

* Testing the Script

T = 5, h = -5 and T = 110, h = 949.5

Where T is an independent Variable (user’s input), and h is a function of T defined by

h(T) = (T – 10) for 0 < T < 100

h(T) = (0.45 T + 900) for T > 100.

* To design seven segement display (SSD)
* To Design Multiplexer, decoder and encoder.

**DESCRIPTION**

* **Input:** [x](https://in.mathworks.com/help/matlab/ref/input.html#btt5u2a-x) = input([prompt](https://in.mathworks.com/help/matlab/ref/input.html#btt5u2a-prompt)) displays the text in prompt and waits for the user to input a value and press the **Return** key.

[str](https://in.mathworks.com/help/matlab/ref/input.html#btt5u2a-str) = input([prompt](https://in.mathworks.com/help/matlab/ref/input.html#btt5u2a-prompt),'s') returns the entered text, without evaluating the input as an expression.

* **While loop:** while *expression*, *statements*, end evaluates an [expression](https://in.mathworks.com/help/matlab/ref/while.html#bub68r7-4), and repeats the execution of a group of statements in a loop while the expression is true.

while *expression*

*statements*

end

* **If, elseif, else:** if *expression*, *statements*, end evaluates an [expression](https://in.mathworks.com/help/matlab/ref/if.html#bt_csfy), and executes a group of statements when the expression is true.

The elseif and else blocks are optional. The statements execute only if previous expressions in the if..end block are false. An if block can include multiple elseif blocks.

if *expression*

*statements*

elseif *expression*

*statements*

else

*statements*

end

* **Continue:** continue passes control to the next iteration of a for or while loop. It skips any remaining statements in the body of the loop for the current iteration. The program continues execution from the next iteration.
* **Error:** error([msg](https://in.mathworks.com/help/matlab/ref/error.html#bumt5nd-msg)) throws an error and displays an error message.
* **7 Segment display:** it is one of the oldest methods of displaying values in electronic devices. The combination of 7 LEDs makes the whole display. Every time a single pin gets the power of a specific range it starts glowing. The pattern and drawing of LED make the decimal digit 8. Then turning on/off the specific pins make the 7-segment to show the other decimal numbers. The LED has a total of 10 input pins.

A picture containing table

Description automatically generated

**PROCEDURE**

An if statement can be followed by an optional else if...else statement, which is

very useful to test various conditions using single if...else if statement.

When using if , else if , else statements there are few points to keep in mind:

* An if can have zero or one else's and it must come after any else if's.
* An if can have zero to many else if's and they must come before the else.
* Once an else if succeeds, none of the remaining else if's or else's will be

tested.

**PROGRAMS AND OUTPUTS**

* **SOURCE CODE**

clc;clear all;

val = input("How Many Values you want to Enter");

i = 0;

while i<val

t = input("\n Enter the Value of T (>0) : ");

if t>0 && t<100

disp("\n Value of H\_a is: ");

h = t-10

elseif t>=100

disp("\n Value of H\_b is: ");

h = 0.45\*t+900

else

check = input("\nWrong Value\nRetry?\n (Y- to Continue or N- to exit with error)",'s');

if (check == 'Y' || check == 'y')

continue;

else

error('Incorrect Value');

end

end

i=i+1;

end

clc;

disp("----PRIORITY-ENCODER----") ;

D0 = input("Enter input D0 ") ;

D1 = input("Enter input D1 ") ;

D2 = input("Enter input D2 ") ;

D3 = input("Enter input D3 ") ;

D4 = input("Enter input D4 ") ;

D5 = input("Enter input D5 ") ;

D6 = input("Enter input D6 ") ;

D7 = input("Enter input D7 ") ;

if( D0 == 1)

printf(" Output: X=0 , Y=0 , Z=0 ") ;

elseif( D1 == 1)

printf(" Output: X=0 , Y=0 , Z=1 ") ;

elseif( D2 == 1)

printf(" Output: X=0 , Y=1 , Z=0") ;

elseif( D3 == 1)

printf(" Output: X=0 , Y=1 , Z=1 ") ;

elseif( D4 == 1)

printf(" Output: X=1 , Y=0 , Z=0 ") ;

elseif( D5 == 1)

printf(" Output: X=1 , Y=0 , Z=1 ") ;

elseif( D6 == 1)

printf(" Output: X=1 , Y=1 , Z=0 ") ;

else

printf(" Output: X=1 , Y=1 , Z=1 ") ;

end ;

clc;

disp(" 4:1 MUX ");

S0= input("Enter value of Select Line S0:") ;

S1= input("Enter value of Select Line S1:") ;

if( S0 == 0 && S1 == 0 )

disp("Output:I0") ;

elseif( S0 == 1 && S1 == 0 )

disp("Output:I1") ;

elseif( S0 == 0 && S1 == 1 )

disp("Output:I2") ;

else

disp("Output:I3") ;

end

clc

clear all

display('3\*8 Line Decoder')

s1=input('enter S1 value :-')

s2=input('enter S2 value :-')

s3=input('enter S3 value :-')

if (s1==0 && s2==0 && s3==0)

display("A0=1\nA1,A2,A3,A4,A5,A6,A7=0")

elseif (s1==0 && s2==0 && s3==1)

display("A1=1=\nA0,A2,A3,A4,A5,A6,A7=0")

elseif (s1==0 && s2==1 && s3==0)

display("A2=1\nA0,A1,A3,A4,A5,A6,A7=0")

elseif (s1==0 && s2==1 && s3==1)

display("A3=1\nA0,A1,A2,A4,A5,A6,A7=0")

elseif (s1==1 && s2==0 && s3==0)

display("A4=1\nA1,A2,A3,A0,A5,A6,A7=0")

elseif (s1==1 && s2==0 && s3==1)

display("A5=1\nA1,A2,A3,A4,A0,A6,A7=0")

elseif (s1==1 && s2==1 && s3==0)

display("A6=1\nA1,A2,A3,A4,A5,A0,A7=0")

elseif

display("A7=1\nA1,A2,A3,A4,A5,A6,A0=0")

else

display('Wrong input given')

end

a=0; b=0; c=0; d=0; E=0; f=0; g=0; h=0;

inp=input("Enter Value (0-9): ");

if(inp==0)

disp(" \n ");

a=0; b=0; c=0; d=0; e=0; f=0; g=0; h=0;

elseif(inp==1)

disp(" |\n |");

b=1; c=1;

elseif(inp==2)

disp(" \n \_|\n| ");

a=b=g=E=d=1;

elseif(inp==3)

disp(" \_\n \_|\n \_|");

a=b=g=c=d=1;

elseif(inp==4)

disp("|\_|\n |");

f=g=b=c=1;

elseif(inp==5)

disp(" \n|\n \_|");

a=f=g=c=d=1;

elseif(inp==6)

disp(" \_")

disp("|\n||")

a=f=g=c=d=E=1;

elseif(inp==7)

disp(" \_\n |\n |");

a=b=c=1;

elseif(inp==8)

disp(" \n||\n|\_|");

a=b=c=d=E=f=g=h=1;

elseif(inp==9)

disp(" \n||\n \_|");

a=b=c=d=f=g=1;

else

disp("Invalid Input");

end;

disp("---");

fprintf("a=%d, b= %d, c=%d, d=%d, e=%d, f=%d, g=%d, h=%d",a,b,c,d,E,f,g,h)

* Text

  Description automatically generatedGraphical user interface, text

  Description automatically generated**OUTPUT**

Text

Description automatically generated

**Text

Description automatically generated**

**Graphical user interface, application, Word

Description automatically generated**

**CONCLUSION**

Using Different Values of T, the function h(T) which has different relations with T depending on different conditions was evaluated with the help of the equation that gives us the relation b/w h and T, if, elseif, else ladder, while loop and functions like input, continue and error.

8x1 priority encoder, 4x1 mux & 3x8 Line Decoder have been designed.

**PRECAUTIONS**

1. Book the session on amity virtual lab beforehand
2. Make sure that you have a stable internet connection
3. Save your program before running it

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

1. In case connection is lost, refresh your page

**EXPERIMENT 9**

**AIM**

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

To find factorial of an input number.

To print multiplication table for input number.

**DESCRIPTION**

* **For loop:** for *index* = *values*, *statements*, end executes a group of statements in a loop for a specified number of times. *values* has one of the forms:

*valArray* — Create a column vector, *index*, from subsequent columns of array *valArray* on each iteration.

for *index* = *values*

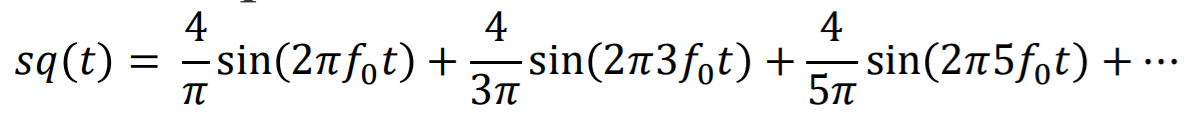
*statements*

end

* **Square wave:** SQUARE waves are mathematically equivalent to the sum of a

• sine wave at that same frequency,

• plus an infinite series of odd-multiple frequency sine waves at diminishing amplitude



A picture containing schematic

Description automatically generated

**PROGRAMS AND OUTPUTS**

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

* **SOURCE CODE**

clc;clear all;

T=1;

f0=10;

F=1000;

t = [0:1/F:T];

s1 = sin(2\*pi\*f0\*t);

#for 1...7

sq = 0;

val = [1:2:7];

for k = val

ang = sin(2\*pi\*k\*f0\*t);

num = (4/(k\*pi));

sq = sq + (num \* ang);

end

figure

plot(t, s1, t, sq,'linewidth',1.35);

title("for 1...7");

xlabel("t");

ylabel("Amplitude");

legend("Sin", "Square");

#for 1...49

sq1 = 0;

val1 = [1:2:49];

for k = val1

ang = sin(2\*pi\*k\*f0\*t);

num = (4/(k\*pi));

sq1 = sq1 + (num \* ang);

end

figure

plot(t, s1, t, sq1, "linewidth" ,1.1);

title("for 1...49");

xlabel("t");

ylabel("Amplitude");

legend("Sin", "Square");

#for 1...99

sq2 = 0;

val2 = [1:2:99];

for k = val2

ang = sin(2\*pi\*k\*f0\*t);

num = (4/(k\*pi));

sq2 = sq2 + (num \* ang);

end

figure

plot(t, s1, t, sq2, "linewidth",1.3);

title("for 1...99");

xlabel("t");

ylabel("Amplitude");

legend("Sin", "Square");

clc; clear all;

inp=input("Enter a Number ");

fact = 1;

for i = 1:inp

fact = fact\*i;

end

fprintf("Factorial: %d",fact);

clc; clear all;

n=input('Enter an integer: ');

b=input('Number of multiples: ');

i=1;

for i=i:b

fprintf('%d\*%d=%d \n',n,i,n\*i);

i=i+1;

end

* **Chart

  Description automatically generatedChart

  Description automatically generatedOUTPUT**

Graphical user interface, application

Description automatically generated**Histogram

Description automatically generated**

**Text

Description automatically generated**

**INTERPRETATION OF GRAPHS**

By varying frequencies of sine wave, different types of graphs can be plotted successfully. Colours of various graphs can also be changed according to user’s requirement. Seven segment display can be designed according to the truth table on octave. Factorial and table of a number was also successfully interpreted.

**RESULT AND DISCUSSION**

With the help of for loop we generated sum of sine waves of certain amplitude and frequency and from that sum we generated square waves. Square waves and sine wave were plotted. Increment in the Number of harmonics led to an increase in the number of curves that led to a smoother graph resulting into a smoother square wave. For loop was also used in finding the factorials and multiplication tables. Also, designing of seven segment LED display on octave was interpreted successfully.

**PRECAUTIONS**

1. Book the session on amity virtual lab beforehand
2. Make sure that you have a stable internet connection
3. Save your program before running it

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

1. In case connection is lost, refresh your page

**EXPERIMENT 10**

**AIM**

a) Basic 2D and 3D plots: parametric space curve, polygons with vertices, 3D contour lines and pie and bar charts.

b) Design Arithmetic and Logic unit

c) Designing of calculator

**TOOLS USED**

Octave

**THEORY:**

* ezplot(f) plots the curve defined by the function y = f(x) over the default interval [-2π 2π] for x.
* ezplot automatically adds a title and axis labels to the plot.  
  ezplot(f,xinterval) plots over the specified interval. Specify the interval as a two-element vector of the form [xmin xmax].
* ezplot(f2) plots the curve defined by the implicit function 0 = f2(x,y) over the default interval [-2π 2π] for x and y.
* ezplot(f2,xyinterval) plots over the specified interval. To use the same interval for both x and y, specify xyinterval as a two-element vector of the form [min max]. To use different intervals, specify a four-element vector of the form [xmin xmax ymin ymax].
* ezplot(funx,funy) plots the parametrically defined planar curve defined by x = funx(u) and y = funy(u) over the default interval [0 2π] for u.
* ezplot(funx,funy,uinterval) plots over the specified interval. Specify the interval as a two-element vector of the form [umin umax].
* fplot3([xt](https://in.mathworks.com/help/symbolic/fplot3.html#buzhvpq-xt),[yt](https://in.mathworks.com/help/symbolic/fplot3.html#buzhvpq-yt),[zt](https://in.mathworks.com/help/symbolic/fplot3.html#buzhvpq-zt)) plots the parametric curve *xt* = *x*(*t*), *yt* = *y*(*t*), and *zt* = *z*(*t*) over the default interval –5 < *t* < 5.
* fplot3([xt](https://in.mathworks.com/help/symbolic/fplot3.html#buzhvpq-xt),[yt](https://in.mathworks.com/help/symbolic/fplot3.html#buzhvpq-yt),[zt](https://in.mathworks.com/help/symbolic/fplot3.html#buzhvpq-zt),[[tmin tmax]](https://in.mathworks.com/help/symbolic/fplot3.html#buzhvpq-tmintmax)) plots *xt* = *x*(*t*), *yt* = *y*(*t*), and *zt* = *z*(*t*) over the interval *tmin* < *t* < *tmax*.
* Calculator: is a device which is used to perform different types of functions like addition, subtraction etc.

The polyshape function creates a polygon defined by 2-D vertices, and returns a polyshape object with properties describing its vertices, solid regions, and holes. For example, pgon = polyshape([0 0 1 1],[1 0 0 1]) creates the solid square defined by the four points (0,1), (0,0), (1,0), and (1,1).

>> pgon = polyshape([0 0 1 1],[1 0 0 1])

pgon =

polyshape with properties:

Vertices: [4×2 double]

NumRegions: 1

NumHoles: 0

>> plot(pgon)

* contour3(Z) creates a 3-D contour plot containing the isolines of matrix Z, where Z contains height values on the x-y plane. MATLAB® automatically selects the contour lines to display. The column and row indices of Z are the x and y coordinates in the plane, respectively.
* contour3(X,Y,Z) specifies the x and y coordinates for the values in Z.

pie(X) draws a pie chart using the data in X. Each slice of the pie chart represents an element in X.

* If sum(X) ≤ 1, then the values in X directly specify the areas of the pie slices. pie draws only a partial pie if sum(X) < 1.
* If sum(X) > 1, then pie normalizes the values by X/sum(X) to determine the area of each slice of the pie.
* If X is of data type categorical, the slices correspond to categories. The area of each slice is the number of elements in the category divided by the number of elements in X.

1. bar(y) creates a bar graph with one bar for each element in y. If y is an m-by-n matrix, then bar creates m groups of n bars.
2. bar(x,y) draws the bars at the locations specified by x.

**PROGRAM**

clc; clear all;

% 2D Parametric curves

f1 = @(x) x.^2-2;

figure;

ezplot(f1,[-2,2],100);

grid on;

xlabel('x');

ylabel('x.^2-2');

% 3D Parametric curves

fx = @(t) cos(t);

figure;

ezplot(fx,[0,6\*pi],100);

grid on;

xlabel('t');

fy = @(t) sin(t);

fz = @(t) t;

figure;

ezplot3(fx,fy,fz,[0,6\*pi],100);

xlabel('cos(t)');

ylabel('sin(t)');

zlabel('t');

% Polygon with vertices

figure;

drawPolygon([0,0; 1,0; 1,1; 0,1])

xlim([-0.1 1.1])

ylim([-0.1 1.1])

figure;

drawPolygon([0,0; 2,0; 1,2])

xlim([-0.1 2.1])

ylim([-0.1 2.1])

%Surface plot with contour

y1 = -2\*pi:pi/10:2\*pi;

y2 = y1;

[X,Y] = meshgrid(y1,y2);

R = sqrt(X.^2 + Y.^2);

Z = sin(R)./R;

figure;

surfc(Z);

figure;

contour(Z);

colorbar;

%Pie, bar chart

z1 = [1,2,3,4];

figure;pie(z1);

figure;bar(z1);

clc; clear all;

fprintf("Calculator: \n\t1.Addition\n\t2.Subtraction\n\t3.Multiplication\n\t4.Division\n\t5.Percentage\n\t6.Exponential\n\t7.Square Root\n\t8.Logrithm(10)\n\t9.Sin(x)\n\t10.Cos(x)\n");

oper=input("Operation: ");

answer=0;

if oper == 1

num1=input("Number 1 : ");

num2=input("Number 2 : ");

answer=num1+num2;

fprintf("Answer: %d",answer);

elseif oper == 2

num1=input("Number 1 : ");

num2=input("Number 2 : ");

answer=num1-num2;

fprintf("Answer: %d",answer);

elseif oper == 3

num1=input("Number 1 : ");

num2=input("Number 2 : ");

answer=num1\*num2;

fprintf("Answer: %d",answer);

elseif oper == 4

num1=input("Number 1 : ");

num2=input("Number 2 : ");

answer=num1/num2;

fprintf("Answer: %d",answer);

elseif oper == 5

num1=input("Number : ");

num2=input("Percentage(in %) : ");

answer=(num2/100)\*num1;

fprintf("Answer: %d",answer);

elseif oper == 6

num1=input("Number : ");

num2=input("Power : ");

answer=num1^num2;

fprintf("Answer: %d",answer);

elseif oper == 7

num1=input("Number : ");

answer=sqrt(num1);

fprintf("Answer: %d",answer);

elseif oper == 8

num1=input("Number : ");

answer=log10(num1);

fprintf("Answer: %d",answer);

elseif oper == 9

num1=input("Number : ");

answer=sin(num1);

fprintf("Answer: %d",answer);

elseif oper == 10

num1=input("Number : ");

answer=cos(num1);

fprintf("Answer: %d",answer);

else

print("Invalid Input!");

end;

clc; clear all;

fprintf("2-Bit ALU\n");

control=input("Control Line(1:Arithmetic;0:Logical): ");

select2=input("Select Line 1: ");

select1=input("Select Line 2: ");

if(control==0 && select2==0 && select1==0)

printf("Logical Unit\n")

printf("Selected function: OR\n")

A=input("Enter A [1 0 1 ...]: ");

B=input("Enter B [0 0 1 ...]: ");

C=or(A,B);

fprintf("A OR B: ");

display(C);

elseif(control==0 && select2==0 && select1==1)

printf("Logical Unit\n")

printf("Selected function: XNOR\n")

A=input("Enter A [1 0 1 ...]: ");

B=input("Enter B [0 0 1 ...]: ");

C=xor(A,B);

fprintf("A XNOR B: ");

display(C);

elseif(control==0 && select2==1 && select1==0)

printf("Logical Unit\n")

printf("Selected function: XOR\n")

A=input("Enter A [1 0 1 ...]: ");

B=input("Enter B [0 0 1 ...]: ");

C=xor(A,B);

fprintf("A XOR B: ");

display(C);

elseif(control==0 && select2==1 && select1==1)

printf("Logical Unit\n")

printf("Selected function: AND\n")

A=input("Enter A [1 0 1 ...]: ");

B=input("Enter B [0 0 1 ...]: ");

C=and(A,B);

fprintf("A OR B: ");

display(C);

elseif(control==1 && select2==0 && select1==0)

printf("Arithmetic Unit\n")

printf("Selected function: Increment A\n")

A=input("Number : ")

fprintf("Increment A: %d\n",A+1)

elseif(control==1 && select2==0 && select1==1)

printf("Arithmetic Unit\n")

printf("Selected function: Add B to A\n")

A=input("Enter A : ");

B=input("Enter B : ");

fprintf("A + B = %d\n", A+B);

elseif(control==1 && select2==1 && select1==0)

printf("Arithmetic Unit\n")

printf("Selected function: Decrement A\n")

A=input("Number : ")

fprintf("Increment A: %d\n",A-1);

elseif(control==1 && select2==1 && select1==1)

printf("Arithmetic Unit\n")

printf("Selected function: Subract B from A\n")

A=input("Enter A : ");

B=input("Enter B : ");

fprintf("A - B = %d\n", A-B);

end;

**OUTPUT**

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

Line chart

Description automatically generated with medium confidence

Chart

Description automatically generated

Chart, line chart

Description automatically generated

Chart, surface chart

Description automatically generated

Chart, surface chart

Description automatically generated

A picture containing chart

Description automatically generated

Chart, pie chart

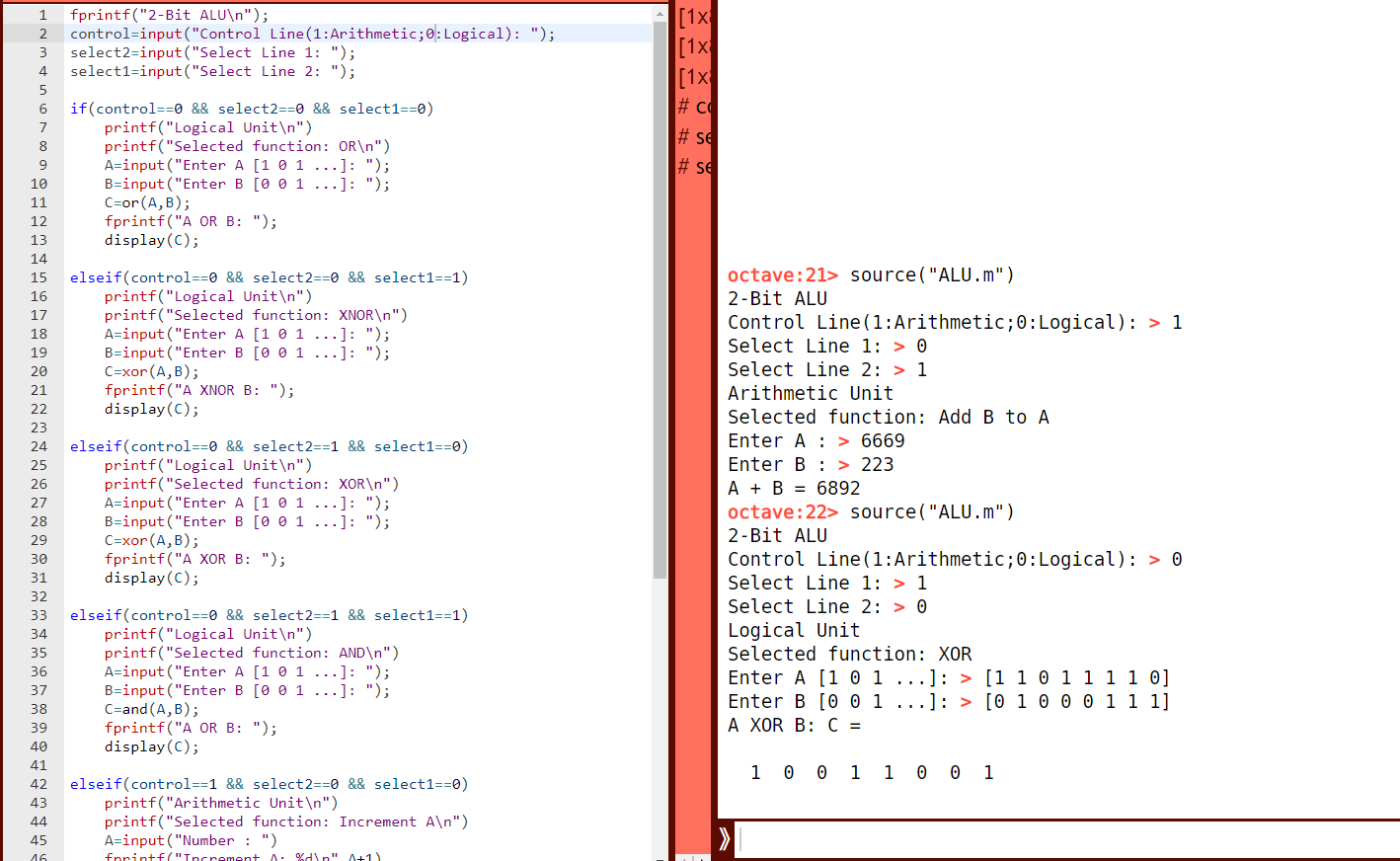
Description automatically generated

Chart, bar chart

Description automatically generated

Text

Description automatically generated



**INTERPRETATION OF GRAPHS:**

pie charts and bar charts are drawn in accordance with values given in the problem. They are also useful in comparing different values given in the text. We can also draw polygons using the drawPolygon command.

**RESULTS AND DISCUSSIONS**

Basic 2D and 3D plots, parametric space curve, polygons with vertices, 3D contour lines and pie and bar charts were implemented on online platform and were interpreted successfully. Design of ALU and calculator was also carefully designed.

**PRECAUTIONS:**

• Stable internet connection is necessary.

• Try to get as less error as possible.

• Beware of the inactivity status shown on the screen on octave software. Session can expire if you don’t use after activation for a long time.

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