

EXPERIMENT 1(A)

AIM: Creating a One-Dimensional Array(Row / Column Vector); Creating a Two-Dimensional Array (Matrix of given size)

SOFTWARE USED: MATLAB 7.12.0(R2011a)

PROCEDURE:

>> % One Dimensional Array

>>A=[10 20 30 40

50]A=

10 20 30 40 50

>> % Two Dimensional Array

>>B=[10 20 30;40 50 60;70 80

90]B=

10 20 30

40 50 60

70 80 90

DISCUSSION:

Creating a Matrix:

In MATLAB, we use the matrix constructor operator[] or with colon operator.

Creating one dimensional matrix:

Simply, enter elements separated by a comma or

space. Row= [E1, E2,...,Em]

Creating two dimensional matrix:

To start a new row, terminate the current row with a

semicolon: A= [row1; row2;... row n]

RESULT:

One- Dimensional and Two- Dimensional Matrix were successfully created.

CONCLUSION:

One Dimensional and two dimensional matrices were created using constructor operator [] and colon operator (first:step:last).

EXPERIMENT – 1(B)

AIM: To perform Arithmetic Operations: Addition, Subtraction, Multiplication and Exponentiation.

SOFTWARE USED: MATLAB 7.12.0(R2011a)

PROCEDURE:

```
>>A=[ 10 30 50;20 40 60; 70 90
```

```
110]A=
```

```
10 30 50
```

```
20 40 60
```

```
70 90 110
```

```
>>B=[20 40 60;30 50 70;80 100
```

```
120]B=
```

```
20 40 60
```

```
30 50 70
```

```
80 100 120
```

```
>>% Addition
```

```
>> C=
```

```
A+BC =
```

```
30 70 110
```

```
50 90 130
```

```
150 190 230
```

```
>>%
```

```
Subtraction
```

```
>> C= A-
```

```
BC =
```

```
-10 -10 -10
```

```
-10 -10 -10
```

```
-10 -10 -10
```

```
>> % Multiplication
```

```
>> C= A-
```

```
BC =
```

5100	6900	8700
6400	8800	11200
12900	18300	23700

>>% **Exponentiation**

>>

C=exp(A)

C=

1.0e+047 *

0.0000 0.0000 0.0000

0.0000 0.0000 0.0000

0.0000 0.0000 5.9210

DISCUSSION:

Matrix Addition (+):

A+B adds A and B. A and B must have the same dimensions, unless one is scalar.

Matrix Subtraction (-):

A-B subtracts B from A. A and B must have the same dimensions, unless one is scalar.

Matrix Multiplication (*):

A*B is the linear algebraic product of A and B. The number of columns of A must equal the number of rows of B, unless one is scalar.

Exponentiation:

$Y = \exp(X)$

It returns the exponential for each element of X.

RESULT:

Two- Dimensional Matrix were created and Arithmetic Operations: +,-,*,exp()

Were successfully performed on them.

CONCLUSION:

Various Arithmetic Operations were successfully operated and results were obtained.

EXPERIMENT 1(C)

AIM: To perform Matrix Operations – Inverse, Transpose, Rank.

SOFTWARE USED: MATLAB 7.12.0 (R2011a)

PROCEDURE:

```
>> A=[50 20 30; 10 90 40;100 70
```

```
80]A=
```

```
50 20 30
```

```
10 90 40
```

```
100 70 80
```

```
>> %Inverse
```

```
>>B=inv(
```

```
A)B=
```

```
0.1257 0.0143 -0.0543
```

```
0.0914 0.0286 -0.0486
```

```
-0.2371 -0.0429 0.1229
```

```
>>%Transpose
```

```
>> B=
```

```
transpose(A)B=
```

```
50 10 100
```

```
20 90 70
```

```
30 40 80
```

```
>>%Rank
```

```
>>
```

```
B=rank(A)
```

```
B=
```

```
3
```

DISCUSSION:

Inverse Operation: $Y = \text{inv}(X)$ returns the onverse of square matrix X .

Transpose Operation:

$B = \text{transpose}(A)$ returns the transpose of the matrix.

Rank Operation:

$K = \text{rank}(A)$

The rank function provides an estimate of the number of linearly independent rows or columns of a full matrix.

RESULT:

Two- Dimensional Matrix is created and matrix Operations: inverse, Transpose, and Rank were performed on it.

CONCLUSION:

Various Matrix Operations were successfully operated and results were obtained.

Internal Assessment (Mandatory Experiment) Sheet for Lab Experiment Department of Mechanical & Automation Engg. Amity University, Noida (U.P)			
Programme	B.Tech CSE	Course Name	Basic Simulation
Course Code	ES204	Semester	4
Student Name	Nandini Sain	Enrollment No.	A2305221060
Marking Criteria			
Criteria	Total Marks	Marks Obtained	Comments
Concept	2		
Implementation	2		
Performance	2		
Total	6		

PRACTICAL 2(B)

AIM: Creating Arrays X and Y of given size (1XN) and Performing Relational Operations - >, <, ==, <=, >=, ~=

SOFTWARE USED: MATLAB 7.12.0(R2022a)

PROCEDURE:

```
%Relational Operators
```

```
A=[10 30;50 70];  
display(A);
```

```
A = 2×2  
    10    30  
    50    70
```

```
B=[80 60;40 20];  
display(B);
```

```
B = 2×2  
    80    60  
    40    20
```

```
Equality=eq(A,B);  
display(Equality);
```

```
Equality = 2×2 logical array  
     0     0  
     0     0
```

```
Lessthan=lt(A,B);  
display(Lessthan);
```

```
Lessthan = 2×2 logical array  
     1     1  
     0     0
```

```
Greaterthan=gt(A,B);  
display(Greaterthan);
```

```
Greaterthan = 2×2 logical array  
     0     0  
     1     1
```

```
LessthanEqualto=le(A,B);  
display(LessthanEqualto);
```

```
LessthanEqualto = 2×2 logical array  
     1     1  
     0     0
```

```
GreaterthanEqualto=ge(A,B);  
display(GreaterthanEqualto);
```

```
GreaterthanEqualto = 2×2 logical array  
     0     0  
     1     1
```

```
NotEqualto=ne(A,B);  
display(NotEqualto);
```

```
NotEqualto = 2x2 logical array
```

```
1 1  
1 1
```

DISCUSSION:

Relational Operations:

Test for equality/greater than or equal/greater than/less than or equal/ less than/inequality is performed using eq/ ge/ gt/ le /lt/ ne respectively.

The relational operators are <, >, <=, >=, == and ~=. Relational operators perform element-by-element comparisons between two arrays. They return a logical array of the same size, with elements set to logical 1 (true) where the relation is true, and elements set to logical 0 (false) where it is not.

RESULT:

Matrices were created and Relational Operations (are <, >, <=, >=, ==, ~=) were performed on them.

CONCLUSION:

Various Matrix Relational Operations were successfully operated and results were obtained.

PRACTICAL 2(C)

AIM: Creating Arrays X and Y of given size (1 X N) and Performing Logical Operations - , ~ , & , |, XOR.

SOFTWARE USED: MATLAB 7.12.0(R2022a)

PROCEDURE:

```
%Logical Operators
```

```
A=[1 0 1 0 1];  
display(A);
```

```
A = 1x5  
    1    0    1    0    1
```

```
B=[1 0 0 0 1];  
display(B);
```

```
B = 1x5  
    1    0    0    0    1
```

```
NOT=~A;  
display(NOT);
```

```
NOT = 1x5 logical array  
    0    1    0    1    0
```

```
NOT2=~B;  
display(NOT2);
```

```
NOT2 = 1x5 logical array  
    0    1    1    1    0
```

```
AND=(A&B);  
display(AND);
```

```
AND = 1x5 logical array  
    1    0    0    0    1
```

```
OR=A|B
```

```
OR = 1x5 logical array  
    1    0    1    0    1
```

```
display(OR);
```

```
OR = 1x5 logical array  
    1    0    1    0    1
```

```
XOR=xor(A,B);  
display(XOR);
```

```
XOR = 1x5 logical array  
    0    0    1    0    0
```

DISCUSSION:

Logical Operations:

The symbols &, |, ~, xor are the logical array operators AND, OR, NOT and exclusive OR. Logical Operations return a logical array with elements set to 1 (true) or 0 (false), as appropriate.

Logical AND (&):

Expr1 & Expr2 represents a logical AND operation between values, arrays, or expressions expr1 and expr2. In an AND operation, if expr1 is true and expr2 is true, then the AND of those inputs is true. If either expression is false, the result is false.

Logical OR (|):

Expr1 | Expr2 represents a logical OR operation between values, arrays, or expressions expr1 and expr2. In an OR operation, if expr1 is true or expr2 is true, then the OR of those inputs is true. If both expressions are false, the result is false.

Logical NOT(~):

~Expr represents a logical NOT operation applied to expression expr. In a NOT operation, if expr is false, then the result of the operation is true. If expr is true, then the result is false.

Exclusive OR(xor):

C = xor(A,B) performs an exclusive OR operation on the corresponding elements of arrays A and B. The resulting element C(I,j,...) is logical true (1) if A(I,j,...) or B(I,j,...), but not both, is nonzero.

RESULT:

Matrices were created and Logical Operations (~, &, |, XOR) were performed on them.

CONCLUSION:

Various Logical Operations were successfully operated and results were obtained.

PRACTICAL 2(A)

AIM: Performing Matrix Manipulations – Concatenating, Indexing, Sorting, Shifting , Reshaping, Resizing and Flipping about a Vertical Axis/ Horizontal Axis.

SOFTWARE USED: MATLAB 7.12.0(R2022a)

PROCEDURE:

```
A= [70 50 30;10 100 90;40 60 20];  
display(A);
```

```
A    3x3  
    70    50    30  
    10   100    90  
    40    60    20
```

```
B= [20 100 70;40 30 50;10 90 60];  
display(B);
```

```
B    3x3  
    20   100    70  
    40    30    50  
    10    90    60
```

```
HorzCat=horzcat(A,B);  
display(HorzCat);
```

```
HorzCat = 3x6  
    70    50    30    20   100    70  
    10   100    90    40    30    50  
    40    60    20    10    90    60
```

```
VerCat=vertcat(A,B);  
display(VerCat);
```

```
VerCat = 6x3  
    70    50    30  
    10   100    90  
    40    60    20  
    20   100    70  
    40    30    50  
    10    90    60
```

```
Indexing=A(:,2);  
display(Indexing);
```

```
Indexing = 3x1  
    50  
   100  
    60
```

```
Index1=A(3,:);  
display(Index1);
```

```
Index1 = 1x3  
    40    60    20
```

```
Index2=A(2,2);  
display(Index2);
```

```
Index2 = 100
```

```
SortA=sort(A);  
display(SortA);
```

```
SortA = 3x3  
    10    50    20  
    40    60    30  
    70   100    90
```

```
SortB=sort(B);  
display(SortB);
```

```
SortB = 3x3  
    10    30    50  
    20    90    60  
    40   100    70
```

```
SortRows=sortrows(A);  
display(SortRows);
```

```
SortRows = 3x3  
    10   100    90  
    40    60    20  
    70    50    30
```

```
Sort1=sort(sort(A,2),1);  
display(Sort1);
```

```
Sort1 = 3x3  
    10    40    60  
    20    50    70  
    30    90   100
```

```
Cirshifting=circshift(A,2);  
display(Cirshifting);
```

```
Cirshifting = 3x3  
    10   100    90  
    40    60    20  
    70    50    30
```

```
Cirshifting2=circshift(A,[2,-1]);  
display(Cirshifting2);
```

```
Cirshifting2 = 3x3  
   100    90    10  
    60    20    40  
    50    30    70
```

```
A=[20 60 30 40 50;10 40 80 30 70;80 60 10 70 90];  
display(A);
```

```
A    3x5  
    20    60    30    40    50  
    10    40    80    30    70  
    80    60    10    70    90
```

```
Reshaping=reshape(A,5,3,[]);  
display(Reshaping);
```

```
Reshaping = 5x3  
    20    60    30  
    10    30    70  
    80    80    50  
    60    10    70  
    40    40    90
```

```
Horiflipping=flipud(A);  
display(Horiflipping);
```

```
Horiflipping = 3x5  
    80    60    10    70    90  
    10    40    80    30    70  
    20    60    30    40    50
```

```
Verflipping=fliplr(A);  
display(Verflipping);
```

```
Verflipping = 3x5  
    50    40    30    60    20  
    70    30    80    40    10  
    90    70    10    60    80
```

DISCUSSION:

Concatenation: It is the process of joining one or more matrices to make a new matrix using `horzcat` (Horizontally concatenate matrices) or `vertcat` (Vertically concatenate matrices).

Indexing: To reference a particular element in a matrix, specify its row and column number using the following syntax : `A(row, column)`.

Sorting: The `sort` function sorts matrix elements along a specified dimension. Syntax for the function is: `sort (matrix, dimension, 'mode')`.

Shifting: The `circshift` function shifts the elements of a matrix in a circular manner along one or more dimensions. Rows or columns that shifted out of the matrix circulate back into the opposite end.

Flipping: `B = fliplr(A)` returns A with columns flipped in the left-right direction and `B = flipud(a)` returns A with rows flipped in the up-down direction.

RESULT:

Matrices were created and Matrix manipulations (Concatenating, Indexing, Sorting, Shifting, Reshaping, Resizing and Flipping) were performed on them.

CONCLUSION:

Various Matrix manipulations were successfully operated and results were obtained.

EXPERIMENT 3

AIM: Generating a set of Commands on a given Vector (Example: X= [1 8 3 9 0 1]) to add up the values of the elements(Check with sum)

- A) where Running Sum for element j = the sum of the elements from 1 to j, inclusive.
- B) Generating a Random Sequence using rand() / randn() functions and plotting them.

SOFTWARE USED: MATLAB 7.12.0(R2022a)

PROCEDURE:

```
%Generating a set of commands on a given Vector to add up to the values of  
%elements(Check with sum)  
X=[ 1 8 3 9 0 1];  
display(X);
```

```
X = 1×6  
    1     8     3     9     0     1
```

```
Y=sum(X);  
display(Y);
```

```
Y = 22
```

```
Y=sum(X(3:6))
```

```
Y = 13
```

```
display(Y);
```

```
Y = 13
```

```
Z=cumsum(X)
```

```
Z = 1×6  
    1     9    12    21    21    22
```

```
display(Z);
```

```
Z = 1×6  
    1     9    12    21    21    22
```

```
%Compute running sum where running sum for element j= sum of elements from  
%1 to j,inclusive  
Length =length(X);  
SUM=0
```

```
SUM = 0
```

```
for i = 1:Length  
    SUM = SUM+X(i)  
end
```

```
SUM = 1  
SUM = 9  
SUM = 12  
SUM = 21  
SUM = 21  
SUM = 22
```

`%Generating Random Sequence using rand() / randn() functions and plotting.`

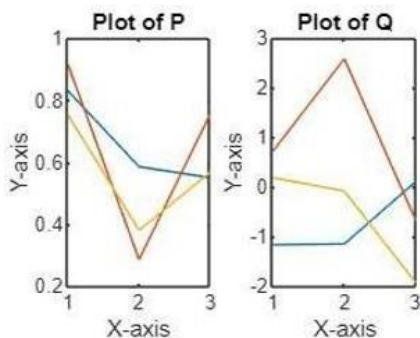
`P=rand(3)`

```
P = 3x3
    0.8308    0.9172    0.7537
    0.5853    0.2858    0.3804
    0.5497    0.7572    0.5678
```

`Q=randn(3)`

```
Q = 3x3
   -1.1658    0.7223    0.1873
   -1.1480    2.5855   -0.0825
    0.1049   -0.6669   -1.9330
```

```
subplot(1,2,1)
plot(P)
title('Plot of P')
xlabel('X-axis')
ylabel('Y-axis')
subplot(1,2,2)
plot(Q)
title('Plot of Q')
xlabel('X-axis')
ylabel('Y-axis')
```



DISCUSSION:

Sum of all/specific elements of a vector were calculated using

`sum()`. `Sum(A)` returns the sum of all the elements of vector.

`Sum(A(i:j))` returns the sum of *i*th to *j*th elements of vector.

For Loop is used here to calculate the running sum of all elements of the vector. If the matrix is of two dimensions then we have to use two for loops.

Generating Random Sequence:

`r = randn(n)` returns an *n*-by-*n* matrix containing pseudorandom values drawn from the standard normal distribution while `r = rand(n)` returns by *n*-by-*n* matrix containing pseudorandom values drawn from the standard uniform distribution on the open interval (0,1).

Plotting using plot():

Plot(Y) plots the columns of Y versus the index of each value when Y is real number and plot(X1,Y1,...,Xn,Yn) plot each vector Yn versus vector Xn on the same axes. Functions xlabel and ylabel were used to label x and y axis.

Plotting Using stem():

It plots discrete sequence data. Stem (X,Y) plot X versus the columns of Y, X and Y must be vectors or matrices of the same size.

Subplot():

Subplot divides the current figure into rectangular panes that are numbered row-wise. h=subplot (m,n,p) or subplot (mnp) breaks the figure window into an m-by-m matrix of small axes, selects the pth axes object for the current plot, and returns the axes handle.

RESULT:

1. Matrices were created and sum of all as well as specific elements were calculated using sum().
2. Vector was created and its running sum was calculated using for loops.
3. Random sequence was generated using rand() and randn() and plotting was done using plot, subplot and stem functions.

CONCLUSION:

Vector was created and sum of all as well as specific elements were successfully calculated. Running sum of a matrix was successfully calculated. Functions rand, randn, plot, subplot and stem were successfully performed.

EXPERIMENT – 5

Aim: Creating a vector X with elements $X_n = (-1)^{n+1}/(2n-1)$ and adding up 100 elements of the vector X and plotting the functions x , x^3 , e^x , $\exp(x^2)$ over the interval $0 < x < 4$ (by choosing appropriate mesh values for x to obtain smooth curves) on a Rectangular Plot.

SOFTWARE USED: MATLAB 7.12.0(R2022a)

PROCEDURE:

$X(n)=0$

X = 1×100

1.0000	-0.3333	0.2000	-0.1429	0.1111	-0.0909	0.0769	-0.0667	0.0588	-0.05
--------	---------	--------	---------	--------	---------	--------	---------	--------	-------

for n=1:100

$X(n)=((-1)^{(n+1)})/(2*n-1)$

end

X = 1×100

1.0000	-0.3333	0.2000	-0.1429	0.1111	-0.0909	0.0769	-0.0667	0.0588	-0.05
--------	---------	--------	---------	--------	---------	--------	---------	--------	-------

X = 1×100

1.0000	-0.3333	0.2000	-0.1429	0.1111	-0.0909	0.0769	-0.0667	0.0588	-0.05
--------	---------	--------	---------	--------	---------	--------	---------	--------	-------

X = 1×100

1.0000	-0.3333	0.2000	-0.1429	0.1111	-0.0909	0.0769	-0.0667	0.0588	-0.05
--------	---------	--------	---------	--------	---------	--------	---------	--------	-------

X = 1×100

1.0000	-0.3333	0.2000	-0.1429	0.1111	-0.0909	0.0769	-0.0667	0.0588	-0.05
--------	---------	--------	---------	--------	---------	--------	---------	--------	-------

X = 1×100

1.0000	-0.3333	0.2000	-0.1429	0.1111	-0.0909	0.0769	-0.0667	0.0588	-0.05
--------	---------	--------	---------	--------	---------	--------	---------	--------	-------

X = 1×100

1.0000	-0.3333	0.2000	-0.1429	0.1111	-0.0909	0.0769	-0.0667	0.0588	-0.05
--------	---------	--------	---------	--------	---------	--------	---------	--------	-------

X = 1×100

1.0000	-0.3333	0.2000	-0.1429	0.1111	-0.0909	0.0769	-0.0667	0.0588	-0.05
--------	---------	--------	---------	--------	---------	--------	---------	--------	-------

X = 1×100

1.0000	-0.3333	0.2000	-0.1429	0.1111	-0.0909	0.0769	-0.0667	0.0588	-0.05
--------	---------	--------	---------	--------	---------	--------	---------	--------	-------

X = 1×100

1.0000	-0.3333	0.2000	-0.1429	0.1111	-0.0909	0.0769	-0.0667	0.0588	-0.05
--------	---------	--------	---------	--------	---------	--------	---------	--------	-------

X = 1×100

1.0000	-0.3333	0.2000	-0.1429	0.1111	-0.0909	0.0769	-0.0667	0.0588	-0.05
--------	---------	--------	---------	--------	---------	--------	---------	--------	-------

X = 1×100

1.0000	-0.3333	0.2000	-0.1429	0.1111	-0.0909	0.0769	-0.0667	0.0588	-0.05
--------	---------	--------	---------	--------	---------	--------	---------	--------	-------

X = 1×100

1.0000	-0.3333	0.2000	-0.1429	0.1111	-0.0909	0.0769	-0.0667	0.0588	-0.05
--------	---------	--------	---------	--------	---------	--------	---------	--------	-------

X = 1×100

1.0000	-0.3333	0.2000	-0.1429	0.1111	-0.0909	0.0769	-0.0667	0.0588	-0.05
--------	---------	--------	---------	--------	---------	--------	---------	--------	-------

1.0000	-0.3333	0.2000	-0.1429	0.1111	-0.0909	0.0769	-0.0667	0.0588	-0.05
--------	---------	--------	---------	--------	---------	--------	---------	--------	-------

X = 1×100

1.0000	-0.3333	0.2000	-0.1429	0.1111	-0.0909	0.0769	-0.0667	0.0588	-0.05
--------	---------	--------	---------	--------	---------	--------	---------	--------	-------

X = 1×100

1.0000	-0.3333	0.2000	-0.1429	0.1111	-0.0909	0.0769	-0.0667	0.0588	-0.05
--------	---------	--------	---------	--------	---------	--------	---------	--------	-------

X = 1×100

1.0000	-0.3333	0.2000	-0.1429	0.1111	-0.0909	0.0769	-0.0667	0.0588	-0.05
--------	---------	--------	---------	--------	---------	--------	---------	--------	-------

X = 1×100


1.0000	-0.3333	0.2000	-0.1429	0.1111	-0.0909	0.0769	-0.0667	0.0588	-0.05
--------	---------	--------	---------	--------	---------	--------	---------	--------	-------

X = 1×100

1.0000	-0.3333	0.2000	-0.1429	0.1111	-0.0909	0.0769	-0.0667	0.0588	-0.05
--------	---------	--------	---------	--------	---------	--------	---------	--------	-------

X = 1×100

1.0000	-0.3333	0.2000	-0.1429	0.1111	-0.0909	0.0769	-0.0667	0.0588	-0.05
--------	---------	--------	---------	--------	---------	--------	---------	--------	-------

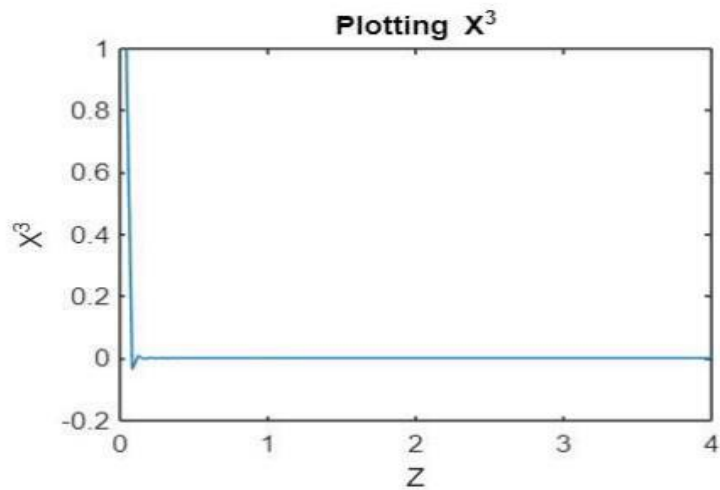


A line plot titled "Plot of X" showing the variable $X(n)$ on the vertical axis versus Z on the horizontal axis. The vertical axis ranges from -0.4 to 0.8 with major ticks every 0.2. The horizontal axis ranges from 0 to 4 with major ticks every 1. The plot shows a blue line that starts at $Z=0$ with a value of approximately 0.9. It exhibits damped oscillations, with the amplitude decreasing as Z increases. The oscillations cross the $X(n)=0$ line at regular intervals. By $Z=4$, the amplitude of the oscillations is very small, and the line appears to be approaching zero.

```
P=X.^3
```

```
P = 1×100  
1.0000 -0.0370 0.0080 -0.0029 0.0014 -0.0008 0.0005 -0.0003
```

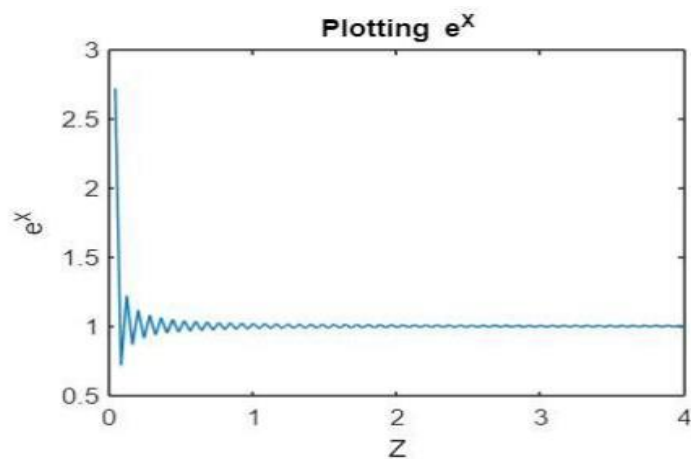
```
%subplot(2,2,2)  
plot(Z,P)  
xlabel('Z')  
ylabel('X^3')  
title('Plotting X^3')
```



```
Q=exp(X)
```

```
Q = 1×100  
2.7183 0.7165 1.2214 0.8669 1.1175 0.9131 1.0800 0.9355
```

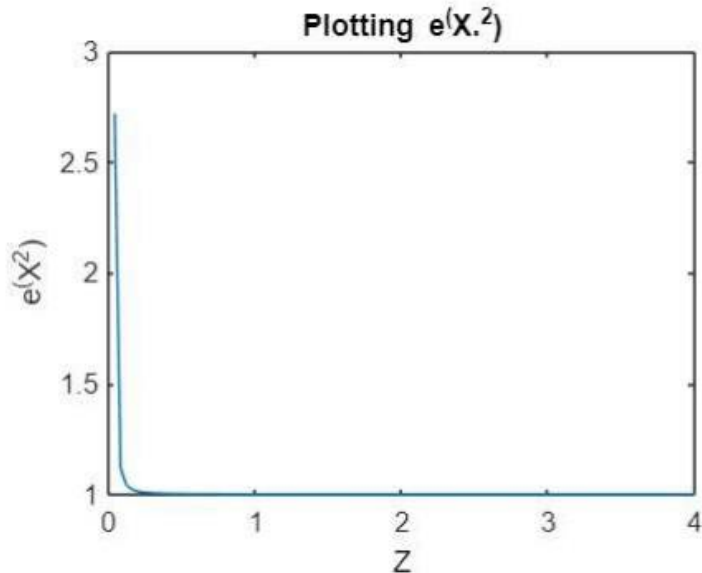
```
%subplot(2,2,3)  
plot(Z,Q)  
xlabel('Z')  
ylabel('e^X')  
title('Plotting e^X')
```



```
R=exp(X.^2)
```

```
R = 1×100  
2.7183 1.1175 1.0408 1.0206 1.0124 1.0083 1.0059 1.0045
```

```
%subplot(2,2,4)
plot(Z,R)
xlabel('Z')
ylabel('e^(X^2)')
title('Plotting e^(X.^2)')
```



ans =

Text (Plotting e^(X.^2)) with properties:

```
String: 'Plotting e^(X.^2)'
FontSize: 11
FontWeight: 'bold'
FontName: 'Helvetica'
Color: [0 0 0]
HorizontalAlignment: 'center'
Position: [2.0000 2.8144 0]
Units: 'data'
```

Show all properties

DISCUSSION:

The vector was constructed using for loop and the given formula. And sum of all elements was calculated using sum(). Then functions x , x^3 , e^x , $\exp(x^2)$ were plotted over the interval $0 < x < 4$ using plot().

RESULT:

Vector x was created and subsequently x , x^3 , e^x , $\exp(x^2)$ were plotted over the interval $0 < x < 4$.

CONCLUSION:

Several function(x , x^3 , e^x , $\exp(x^2)$) were successfully plotted for a vector X .

EXPERIMENT 4(A)

AIM: Evaluating a given expression and rounding it to the nearest integer value using Round, Floor, Ceil and Fix Functions.

SOFTWARE USED: MATLAB 7.12.0(R2022a)

PROCEDURE:

%Round

```
a = round(99.9678)
```

```
a = 100
```

```
b = round(-79.6578346)
```

```
b = -80
```

%Ceil

```
p = ceil(99.678)
```

```
p = 100
```

```
q = ceil(67.11111)
```

```
q = 68
```

%Floor

```
c = floor(99.67676)
```

```
c = 99
```

```
d = floor(3.999999)
```

```
d = 3
```

%Fix

```
e = fix(1.9999996)
```

```
e = 1
```

```
f = fix(-0.999999)
```

```
f = 0
```

DISCUSSION:

Rounding:

$Y = \text{round}(X)$ rounds the elements of X to the nearest integers. Positive Elements with a fractional part of 0.5 round up to the nearest positive integer. Negative elements with a fractional part of -0.5 round down to the nearest negative integer.

Flooring:

$B = \text{floor}(A)$ rounds the elements of A to the nearest Integers less than or equal to A .

Ceiling:

$B = \text{ceil}(A)$ rounds the elements of A to the nearest integers greater than or equal to A .

RESULT:

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

CONCLUSION:

Various Rounding Functions were successfully implemented.

EXPERIMENT 4(B)

AIM: Generating and Plots of Trigonometric Functions – $\sin(t)$, $\cos(t)$, $\tan(t)$, $\sec(t)$, $\operatorname{cosec}(t)$ and $\cot(t)$ for a given duration, T .

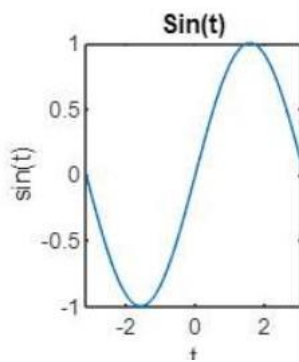
SOFTWARE USED: MATLAB 7.12.0(R2022a)

PROCEDURE:

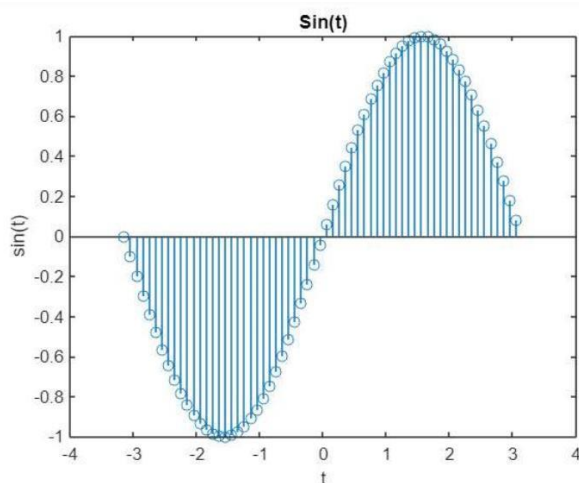
```
%Generating plotting sin(t)
X = sin(t)
```

```
X = 1×63
    -0.0000    -0.0998    -0.1987    -0.2955    -0.3894    -0.4794    -0.5646    -0.6442    -0.7174    -0.78
```

```
subplot(2,3,1)
plot(t,X)
xlabel('t')
ylabel('sin(t)')
title('Sin(t)')
```



```
subplot(1,1,1)
stem(t,X)
xlabel('t')
ylabel('sin(t)')
title('Sin(t)')
```

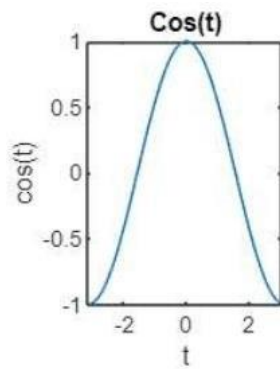


```
% for cos(t)
Y = cos(t)
```

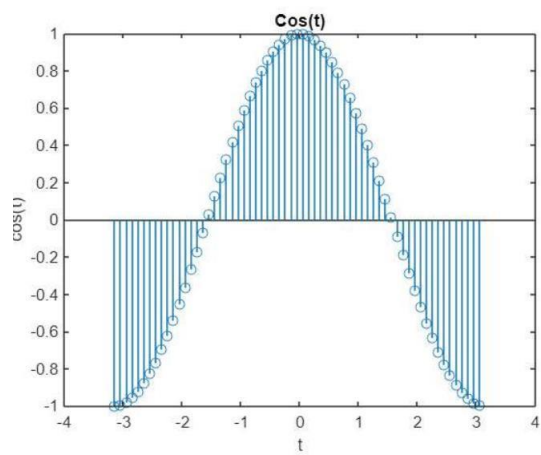
```
Y = 1×63
```

```
-1.0000 -0.9950 -0.9801 -0.9553 -0.9211 -0.8776 -0.8253 -0.7648 -0.6967 -0.62
```

```
subplot(2,3,2)
plot(t,Y)
xlabel('t')
ylabel('cos(t)')
title('Cos(t)')
```



```
subplot(1,1,1)
stem(t,Y)
xlabel('t')
ylabel('cos(t)')
title('Cos(t)')
```



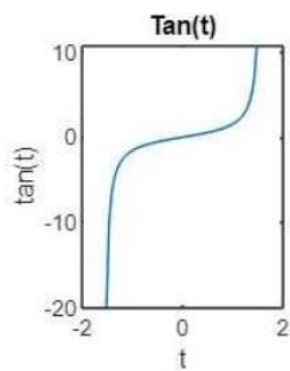

```
% for tan(t)
t=(-pi/2)+0.05:0.05:(pi/2)-0.05
```

```
t = 1×61
    -1.5208    -1.4708    -1.4208    -1.3708    -1.3208    -1.2708    -1.2208    -1.1708    -1.1208    -1.07
```

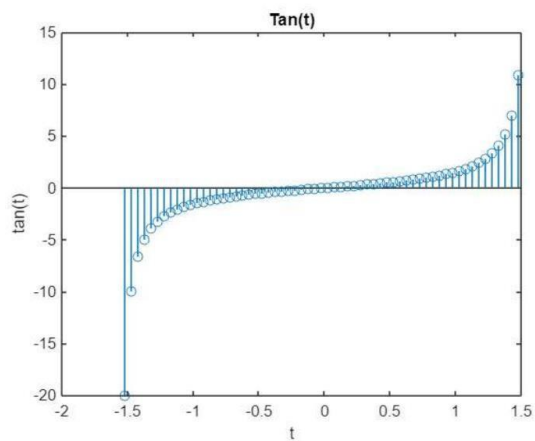
```
Z = tan(t)
```

```
Z = 1×61
   -19.9833    -9.9666    -6.6166    -4.9332    -3.9163    -3.2327    -2.7395    -2.3652    -2.0702    -1.83
```

```
subplot(2,3,3)
plot(t,Z)
xlabel('t')
ylabel('tan(t)')
title('Tan(t)')
```



```
subplot(1,1,1)
stem(t,Z)
xlabel('t')
ylabel('tan(t)')
title('Tan(t)')
```



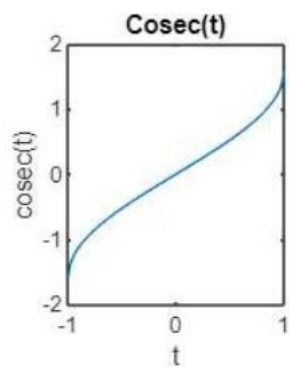
```
% for asin(t)
t=-1:0.02:1
```

```
t = 1×101
    -1.0000    -0.9800    -0.9600    -0.9400    -0.9200    -0.9000    -0.8800    -0.8600    -0.8400    -0.82
```

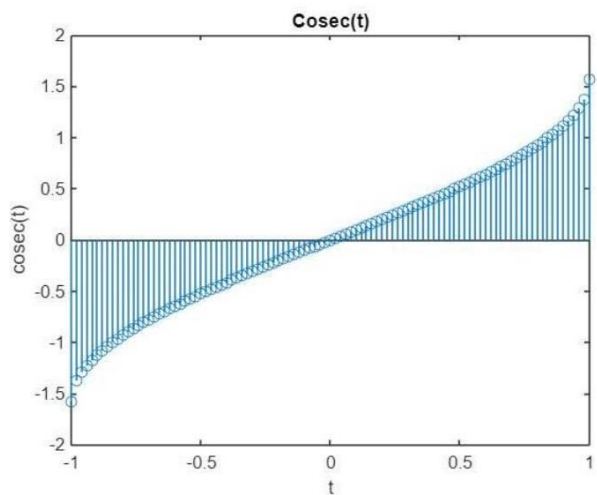
```
A = asin(t)
```

```
A = 1×101
    -1.5708    -1.3705    -1.2870    -1.2226    -1.1681    -1.1198    -1.0759    -1.0353    -0.9973    -0.96
```

```
subplot(2,3,4)
plot(t,A)
xlabel('t')
ylabel('cosec(t)')
title('Cosec(t)')
```



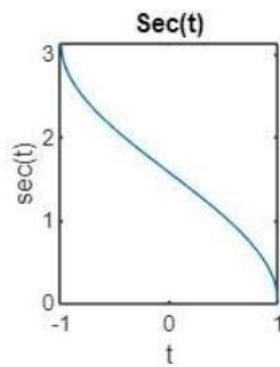
```
subplot(1,1,1)
stem(t,A)
xlabel('t')
ylabel('cosec(t)')
title('Cosec(t)')
```



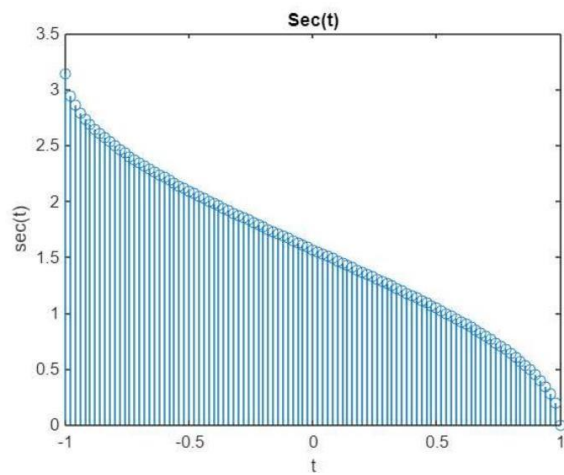
```
% for acos(t)
S = acos(t)
```

```
S = 1×101
    3.1416    2.9413    2.8578    2.7934    2.7389    2.6906    2.6467    2.6061    2.5681    2.53
```

```
subplot(2,3,5)
plot(t,S)
xlabel('t')
ylabel('sec(t)')
title('Sec(t)')
```



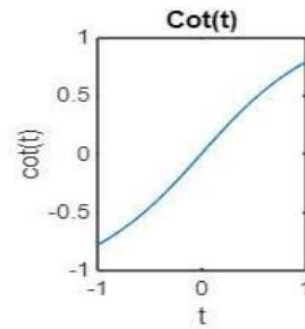
```
subplot(1,1,1)
stem(t,S)
xlabel('t')
ylabel('sec(t)')
title('Sec(t)')
```



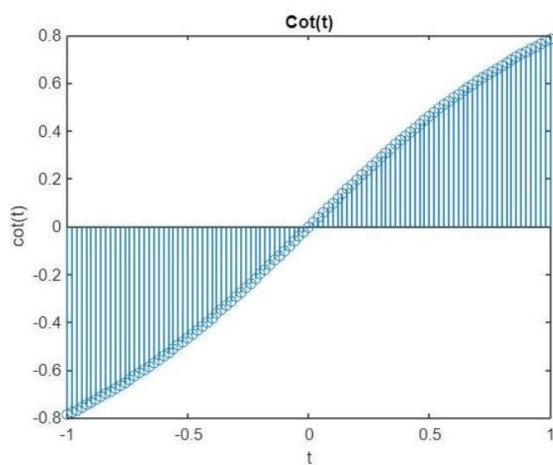
```
% for atan(t)
C = atan(t)
```

```
C = 1×101
   -0.7854   -0.7753   -0.7650   -0.7545   -0.7438   -0.7328   -0.7217   -0.7103   -0.6987   -0.68
```

```
subplot(2,3,6)
plot(t,C)
xlabel('t')
ylabel('cot(t)')
title('Cot(t)')
```



```
subplot(1,1,1)
stem(t,C)
xlabel('t')
ylabel('cot(t)')
title('Cot(t)')
```



DISCUSSION:

Trigonometric Functions:

$Y = \sin(x)$ returns the circular sine of the elements of X /sine of argument in radians, Similarly, $\cos, \tan, \operatorname{acos}, \operatorname{asin}, \operatorname{atan}$ perform.

Plotting using plot():

$\operatorname{Plot}(Y)$ plots the columns of Y versus the index of each value when Y is a real number and $\operatorname{plot}(X1, Y1, \dots, Xn, Yn)$ plots each vector Yn versus Xn on the same axes. Functions xlabel and ylabel were used to label x and y axis.

Plotting using stem():

It plots discrete sequence data. $\operatorname{Stem}(X, Y)$ plots X versus the columns of Y , X and Y must be the vectors or matrices of the same size.

Subplot():

$\operatorname{Subplot}$ divides the current figure into rectangular panes that are numbered rowwise. $H = \operatorname{subplot}(m, n, p)$ or $\operatorname{subplot}(mnp)$ breaks the figure window into an m -by- n matrix of small axes, selects the p th axes object for the current plot, and returns the axes handle.

RESULT:

Matrices were created and trigonometric functions ($\sin(t)$, $\cos(t)$, $\tan(t)$, $\arccos(t)$, $\arcsin(t)$, $\arctan(t)$) were generated and plotted. And Plotting was done using plot, subplot and stem functions.

CONCLUSION:

Various trigonometric functions were generated and plotted.

EXPERIMENT 4(c)

AIM: Generating and Plots of Logarithmic and other Functions – $\log(A)$, $\log_{10}(A)$, Square root of A, Real nth root of A.

SOFTWARE USED: MATLAB 7.12.0(R2022a)

PROCEDURE:

```
X = 1:1:20
```

```
X = 1×20
```

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
```

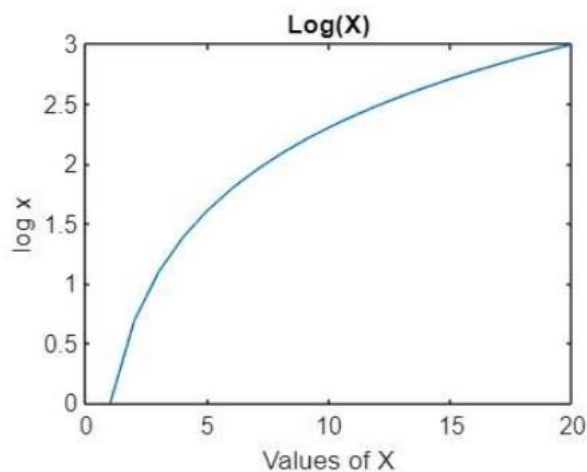
```
%Log(X)
```

```
Y = log(X)
```

```
Y = 1×20
```

```
0 0.6931 1.0986 1.3863 1.6094 1.7918 1.9459 2.0794 2.1972 2.30
```

```
plot(X,Y)  
title('Log(X)')  
xlabel('Values of X')  
ylabel('log x')
```



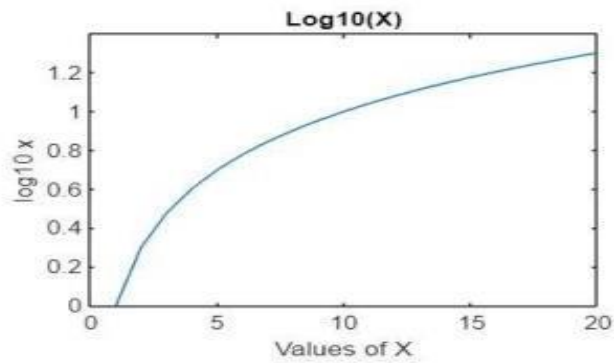
```
%Log10(X)
```

```
Y = log10(X)
```

```
Y = 1×20
```

```
0 0.3010 0.4771 0.6021 0.6990 0.7782 0.8451 0.9031 0.9542 1.00
```

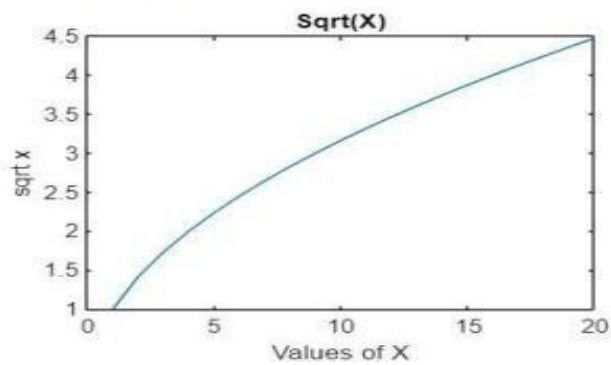
```
plot(X,Y)  
title('Log10(X)')  
xlabel('Values of X')  
ylabel('log10 x')
```



```
%Sqrt(X)
Y = sqrt(X)
```

```
Y = 1×20
    1.0000    1.4142    1.7321    2.0000    2.2361    2.4495    2.6458    2.8284    3.0000    3.16
```

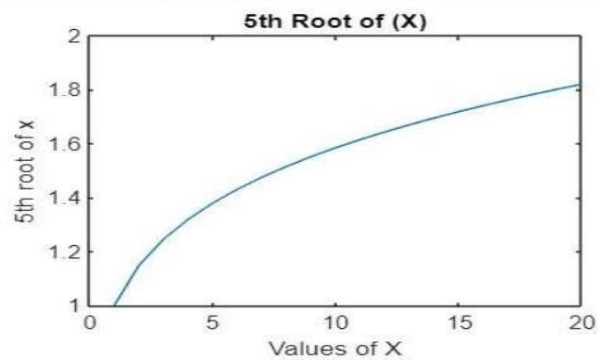
```
plot(X,Y)
title('Sqrt(X)')
xlabel('Values of X')
ylabel('sqrt x')
```



```
%nthroot(X)
Y = nthroot(X,5)
```

```
Y = 1×20
    1.0000    1.1487    1.2457    1.3195    1.3797    1.4310    1.4758    1.5157    1.5518    1.58
```

```
plot(X,Y)
title('5th Root of (X)')
xlabel('Values of X')
ylabel('5th root of x')
```



DISCUSSION:

Logarithmic Functions:

$Y = \log(X)$ returns the natural logarithm of the elements of X.

$Y = \log_{10}(X)$ returns the base 10 logarithmic of the elements of X.

Square Root:

$B = \text{sqrt}(X)$ returns the square root of each element of the array X.

Real nth root of A:

$Y = \text{nthroot}(X,n)$ returns the real nth root of the elements of X. Both X and n must be real and n must be scalar. If X has negative entries, n must be an odd integer.

RESULT:

Matrix was created and Logarithmic Functions(log and log10) along with other functions (sqrt and nthroot) were implemented and results were obtained. And plotting was done using plot, subplot and stem Functions.

CONCLUSION:

Various Logarithmic and other Functions were successfully implemented and plotted.

EXPERIMENT-6

Aim: Generating a sinusoidal signal of given frequency with titling, labelling, adding text , adding legends , print text in Greek letters , plotting as multiple and plots and subplots using marker edge colour and marker shape.

SOFTWARE USED: MATLAB 7.12.0(R2022a)

PROCEDURE:

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

```
t = 1x41
    0    0.1571    0.3142    0.4712    0.6283    0.7854    0.9425    1.0996    1.2566    1.41
```

```
X=sin(t)
```

```
X = 1x41
    0    0.1564    0.3090    0.4540    0.5878    0.7071    0.8090    0.8910    0.9511    0.98
```

```
subplot(3,1,1)
plot(t,X,'-', 'Color',[1,0.753,0.796],'LineStyle','-','LineWidth',3)
xlabel('T')

ylabel('sin(t)')
title('Plotting sin(t)')
legend('sin(t)')
```

```
hold on
Y=sin(3*t)
```

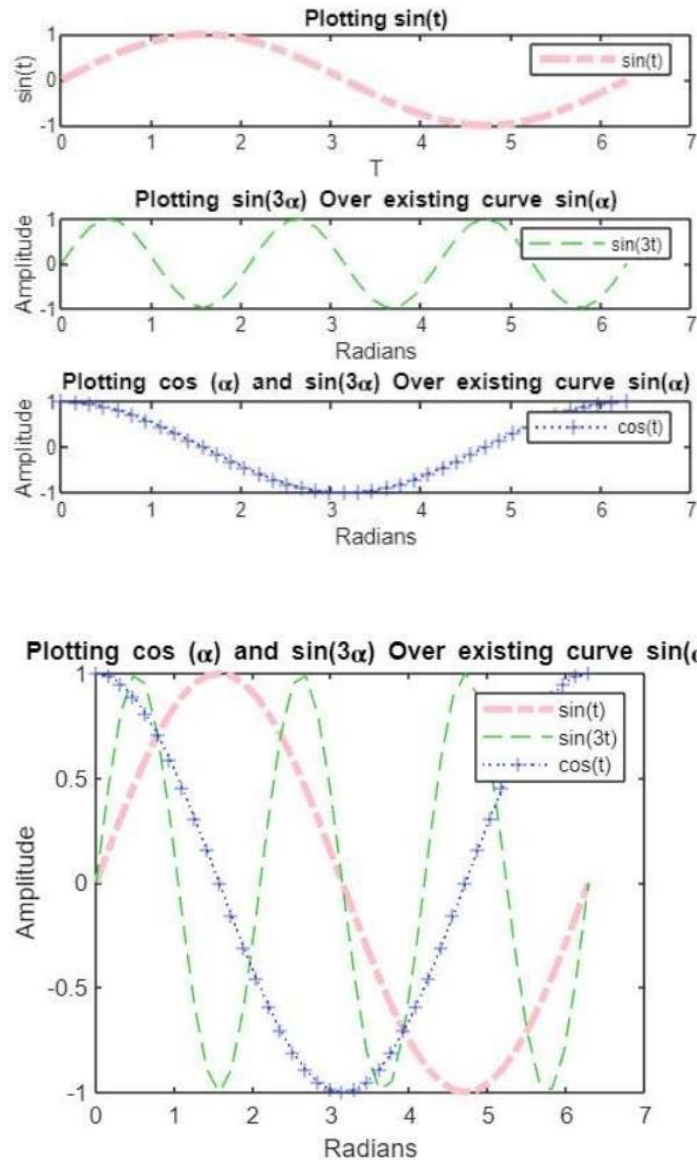
```
Y = 1x41
    0    0.4540    0.8090    0.9877    0.9511    0.7071    0.3090   -0.1564   -0.5878   -0.89
```

```
subplot(3,1,2)
plot(t,Y,'Color',[0.196,0.804,0.196],'LineStyle','--')
xlabel('Radians')
ylabel('Amplitude')
title('Plotting sin(3{\alpha}) Over existing curve sin({\alpha})')
legend('sin(3t)')
```

```
hold on
Z=cos(t)
```

```
Z = 1x41
    1.0000    0.9877    0.9511    0.8910    0.8090    0.7071    0.5878    0.4540    0.3090    0.15
```

```
subplot(3,1,3)
plot(t,Z,"Color",[0.135,0.206,0.85],"LineStyle",":","Marker","+")
xlabel('Radians')
ylabel('Amplitude')
title('Plotting cos ({\alpha}) and sin(3{\alpha}) Over existing curve sin({\alpha})')
legend('cos(t)')
```



DISCUSSION:

Generating and plotting a Sinusoidal Signal: Input parameter t was generated using colon operator (start:step:stop). Sinusoidal Signals was generated using $\sin()$ and plotted using $\text{plot}()$.

Plotting with Graphical Enhancements:

Titling: title('string') outputs the string at top and in the center of the current axes.

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

Adding Text: $\text{text}(x,y,z,'string',\text{'Property Name'}, \text{Property Value..})$: adds the string in quotes to the location defined by the coordinates and uses the value for the specified text properties.

Adding Legends: legend('string1', 'string2'): displays a legend in the current axes using specified strings to label each set of data.

Adding New Plots to Existing Plot: hold on: It retains the current plot and certain axes properties so that subsequent graphing commands add to the existing graphs.

Printing Text In Greek Letters: You can define text that includes symbols and Greek letters using text function, assigning the character sequence to the String Property of text objects. You can also include these character sequence in the string arguments of the title, xlabel, ylabel and zlabel functions.

RESULT:

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

CONCLUSION: Sinusoidal Signals were successfully generated and plotted with several graphical enhancements.

EXPERIMENT – 8

AIM: Solving the First Order Ordinary Differential Equation using Built in Functions.

Consider the following Ordinary Differential Equations: $x(dy/dx) + 2y = x^3$ where $dy/dx = (x^3 - 2y)/x$, $1 < x < 3$ and $y = 4.2$

SOFTWARE USED: MATLAB 7.12.0(R2022a)

PROCEDURE:

```
ode1 = @(x,y)(x^3-2*y) / x
```

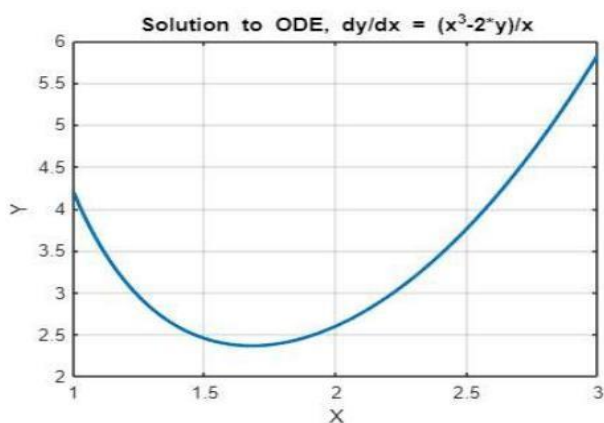
```
ode1 = function_handle with value:  
@(x,y)(x^3-2*y)/x
```

```
[x,y]= ode45(ode1,[1:0.01:3],4.2)
```

```
x = 201x1  
1.0000  
1.0100  
1.0200  
1.0300  
1.0400  
1.0500  
1.0600  
1.0700  
1.0800  
1.0900
```

```
y = 201x1  
4.2000  
4.1272  
4.0569  
3.9889  
3.9232  
3.8596  
3.7982  
3.7388  
3.6813  
3.6257
```

```
plot(x,y, 'linewidth',2)  
xlabel('X'),ylabel('Y'),grid on  
title('Solution to ODE, dy/dx = (x^3-2*y)/x')
```



DISCUSSION:

- **Ode45:** ode45 is based on an explicit Runge Kutta (4,5) formula, the Dormand-Prince pair. It is one step solver – in computing $y(t_n)$, it needs only the solution at the immediate preceding time point, $y(t_{n-1})$.

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

EXPERIMENT – 9

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

SOFTWARE USED: MATLAB 7.12.0(R2022a)

PROCEDURE:

A = 3										
A = 3										
t = 0:0.05*pi:2*pi										
t = 1×41	0	0.1571	0.3142	0.4712	0.6283	0.7854	0.9425	1.0996	1.2566	1.41
sum = 0										
sum = 0										
for n=1:2:100 Y = (sin(n*t))/n sum = sum+Y end										
Y = 1×41	0	0.1564	0.3090	0.4540	0.5878	0.7071	0.8090	0.8910	0.9511	0.98
sum = 1×41	0	0.1564	0.3090	0.4540	0.5878	0.7071	0.8090	0.8910	0.9511	0.98
Y = 1×41	0	0.1513	0.2697	0.3292	0.3170	0.2357	0.1030	-0.0521	-0.1959	-0.29
sum = 1×41	0	0.3078	0.5787	0.7832	0.9048	0.9428	0.9120	0.8389	0.7551	0.69
Y = 1×41	0	0.1414	0.2000	0.1414	0.0000	-0.1414	-0.2000	-0.1414	-0.0000	0.14
sum = 1×41	0	0.4492	0.7787	0.9246	0.9048	0.8014	0.7120	0.6974	0.7551	0.83
Y = 1×41	0	0.1273	0.1156	-0.0223	-0.1359	-0.1010	0.0441	0.1411	0.0840	-0.06
sum = 1×41	0	0.5765	0.8943	0.9023	0.7689	0.7004	0.7562	0.8385	0.8391	0.76
Y = 1×41	0	0.1097	0.0343	-0.0990	-0.0653	0.0786	0.0899	-0.0504	-0.1057	0.01
sum = 1×41	0	0.6862	0.9286	0.8033	0.7036	0.7789	0.8461	0.7881	0.7334	0.78
Y = 1×41	0	0.0898	-0.0281	-0.0810	0.0534	0.0643	-0.0735	-0.0413	0.0865	0.01
Y = 1×41	0	0.0471	-0.0667	0.0471	0.0000	-0.0471	0.0667	-0.0471	-0.0000	0.04
sum = 1×41	0	0.8917	0.7716	0.7574	0.8302	0.7417	0.8154	0.7757	0.7747	0.81
Y = 1×41	0	0.0267	-0.0476	0.0581	-0.0559	0.0416	-0.0182	-0.0092	0.0346	-0.05
sum = 1×41	0	0.9184	0.7240	0.8155	0.7743	0.7833	0.7972	0.7665	0.8092	0.75
Y = 1×41	0	0.0082	-0.0163	0.0239	-0.0309	0.0372	-0.0426	0.0469	-0.0501	0.05
sum = 1×41	0	0.9266	0.7078	0.8394	0.7433	0.8205	0.7547	0.8134	0.7592	0.81
Y = 1×41	0	-0.0074	0.0147	-0.0216	0.0280	-0.0337	0.0385	-0.0424	0.0453	-0.04

```
C=(4*A*sum)/pi
```

```
C = 1×41
      0      3.1211      2.9383      3.0420      2.9675      3.0270      2.9764      3.0214
```

```
hold on
t=0:0.1:10
```

```
t = 1×101
      0      0.1000      0.2000      0.3000      0.4000      0.5000      0.6000      0.7000
```

```
D=sin(t)
```

```
D = 1×101
      0      0.0998      0.1987      0.2955      0.3894      0.4794      0.5646      0.6442
```

```
subplot(2,2,2)
plot(t,D,'R+-')
xlabel('Radians')
ylabel('Amplitude')
title('Generating sin wave')
```

```
subplot(2,2,1)
hold on
```

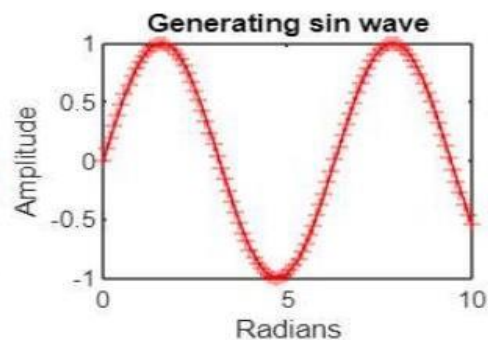
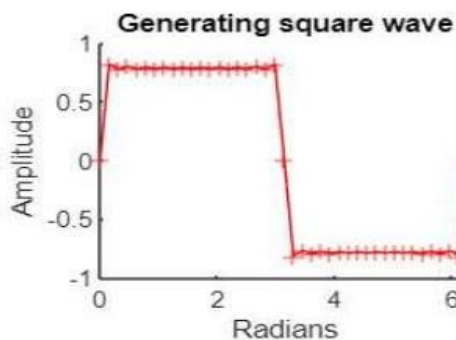
```
t= 0:0.05*pi:2*pi
```

```
t = 1×41
      0      0.1571      0.3142      0.4712      0.6283      0.7854      0.9425      1.0996
```

```
D=sin(t)
```

```
D = 1×41
      0      0.1564      0.3090      0.4540      0.5878      0.7071      0.8090      0.8910
```

```
plot(t,sum,'R+-')
xlabel('Radians')
ylabel('Amplitude')
title('Generating square wave')
```



DISCUSSION:

- Input parameter t was generated using colon operator(start:step:stop).
- **Generating square wave using Fourier series expansion:**
 - The fourier series expansion for a square-wave is made up of a sum of odd harmonics.
 - The more waves you add more smooth the square wave will become.
 - $X(t)=4*a/pi[\sin(w*t)+(1/3)*\sin(3*w*t)+(1/5)*\sin(5*w*t)+...]$
- **Adding new plots to existing plot:**
 - Hold on: it retains the current plot and certain axes properties so that subsequent graphing commands add to the existing graph.

RESULT:

Square wave was generated from sum of sine waves of certain amplitude and frequencies using Fourier Series expansion.

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

EXPERIMENT-10

AIM: Basic 2D and 3D plots: parametric space curve . Polygons with vertices. 3D contour lines, pie and bar charts.

SOFTWARE USED: MATLAB(R7.0)

PROCEDURE:

```
>> ode1=@(x,y)(x^3-2*y)/x
```

ode1 =

function_handle with value:

$\text{@(x,y)(x}^3\text{-2*y)/x}$

```
>> [x,y]=ode45(ode1,[1:0.01:3],4.2)
```

x =

1.0000
1.0100
1.0200
1.0300
1.0400
1.0500
1.0600
1.0700
1.0800
1.0900
1.1000
1.1100
1.1200
1.1300
1.1400
1.1500
1.1600
1.1700
1.1800
1.1900
1.2000
1.2100
1.2200
1.2300
1.2400
1.2500
1.2600
1.2700
1.2800
1.2900
1.3000
1.3100

1.3200
1.3300
1.3400
3.0000

y =

4.2000
4.1272
4.0569
3.9889
3.9232
3.8596
3.7982
3.7388
3.6813
3.6257
3.5720
3.5200
3.4697
3.4211
3.3741
3.3287
3.2848
3.2424
3.2013
3.1617
3.1234
3.0864
3.0506
3.0161
2.9828
2.9506
2.9196
2.8897
2.8608
2.8330
2.8063
2.7805
2.7557
2.7318
2.7089
2.6869
4.6883
4.7300
4.7721
4.8146
4.8574
4.9006
4.9442
4.9881
5.0325
5.0772

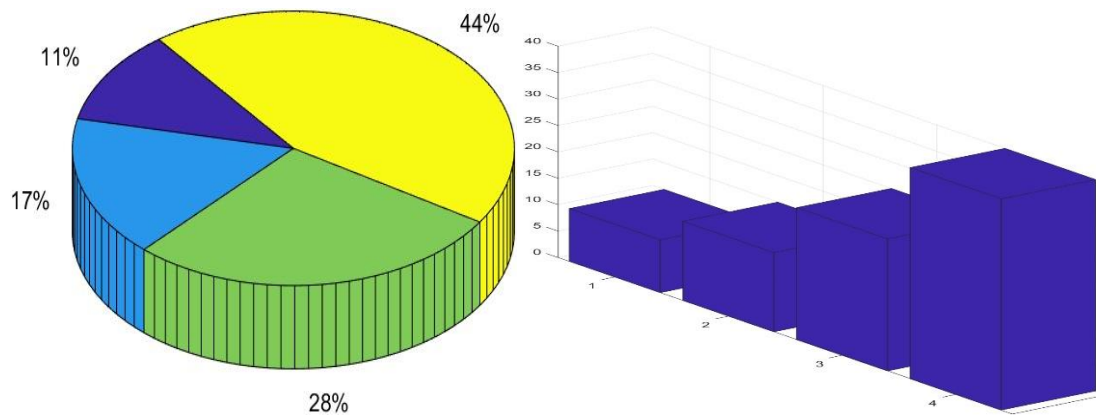
5.1223
5.1678
5.2136
5.2598
5.3064
5.3534
5.4008
5.4485
5.4967
5.5452
5.5941
5.6434
5.6931
5.7431
5.7936
5.8444

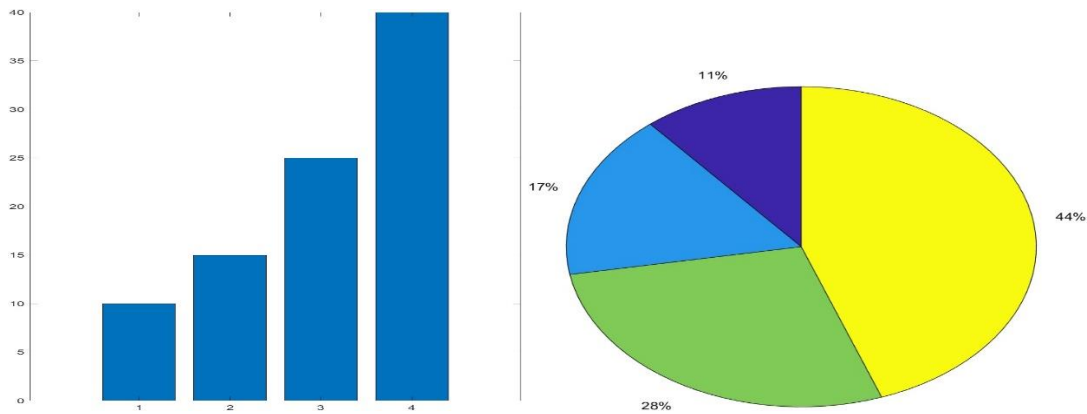
```
>> plot(x,y)
>> plot(x,y,'linewidth',2)
>> xlabel('x')
>> ylabel('y')
>> title('Solution to ODE')
>> x=[10,15,25,40]
```

x =

10 15 25 40

```
>> bar(x)
>> bar3(x)
>> pie(x)
>> pie3(x)
```





DISCUSSION:

PIE: The ratio of the circumference of a circle to its diameter.

CONCLUSION:

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

EXPERIMENT 7

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

$$h(T)=(T-10) \quad ,\text{for } 0<T<100 \\ = (0.45T+900), \text{ for } T>100$$

SOFTWARE USED: MATLAB 7.12.0(R2011A)

PROCEDURE:

The Script

```
H=0;
T=input("Enter value of T:");
if(T==0)
    disp("Enter the value greater than 0");
else if (0<T && T<100)
    fprintf("For T=%d",T);
    H=(T-10);
else if(T>100)
    fprintf("For T=%d",T);
    H=((0.45*T)+900);
end
end
end
fprintf("H=%d",H);
```

DISCUSSION:

- **If-else statements:** If expression 1 evaluates as false and expression 2 as true ,MATLAB executes the one or more commands denoted here as statements2.A true expression has either a logical true or nonzero value.

RESULT:

```
H = 0
Enter value of T:5
T = 5
For T=5
H = -5
H=-5untitled3
H = 0
Enter value of T:110
T = 110
For T=110
H = 949.5000
H=9.495000e+002
```

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

Index

S.No	Name of the Experiment	Date Allotted	Signature
1.	(A) Creating a One-Dimensional Array (Row/Column Vector); Creating a Two-Dimensional Array (Matrix of a given size) (B) Performing Arithmetic Operations - Addition, Subtraction, Multiplication and Exponentiation. (C) Performing Matrix operations - Inverse, Transpose, Rank with plots.	5/01/2023	
2.	(A) Performing Matrix Manipulations- Concatenating, Indexing, Sorting, Shifting, Reshaping, Resizing and Flipping about a Vertical/Horizontal Axis. (B) Creating Arrays X & Y of given size (1 x N) and performing Relational Operations - >, <=, >=, ~= (C) Creating Arrays X & Y of given size (1 x N) and performing Logical Operations - ~, &, , XOR	12/01/2023	
3.	(A) Generating a set of Commands on a given Vector (Example=[1 8 3 9 0 1]) to add up the values of the elements(Check with sum). (B) Generating a set of Commands on a given Vector (Example= [1 8 3 9 0 1]) to compute the Running Sum (Check with sum), where Running Sum for element j=the sum of the elements from 1 to j, inclusive. (C) Generating a Random Sequence using rand() / randn() functions and plotting them.	2/02/2023	
4.	(A) Evaluating a given expression and rounding it to the nearest integer value using Round, Floor, Ceil and Fix functions. (B) Generating and Plots of Trigonometric Functions – sin(t), cos(t), tan(t), sec(t), cosec(t) and cot(t) for a given duration ‘t’. (C) Generating and Plots of Logarithmic and other Functions – log(a), log10 (a), Square root of a and Real nth root of a.	9/02/2023	
5.	Creating a vector X with elements $X_n = (-1)^{(n+1)}/(2n-1)$ and Adding up 100 elements of the vector X and plotting the functions	16/02/2023	

	$x, x^3, \exp(x), \exp(x^2)$ over the interval $0 < x < 4$ (by choosing appropriate mesh values for x to obtain smooth curves on a Rectangular Plot.		
6.	Generating a Sinusoidal Signal of a given frequency (say, 100Hz) and Plotting with Graphical Enhancement: Titling, Labeling, Adding Text, Adding Legends, Adding New Plots to Existing Plot, Printing Text in Greek Letters, Plotting as Multiple and Subplot	23/02/2023	
7.	Writing brief Scripts starting each Script with a request for input (using input) the function $h(T)$ using if else statement, where $h(T) = (T-10)$, for $0 < T < 100$ $= (0.45T + 900)$, for $T > 100$	2/03/2023	
8.	Solving First Order Ordinary Differential Equation using Built-in Functions.	9/03/2023	
9.	Generating a square wave from sum of sine waves of certain amplitude and frequencies.	6/04/2023	
10.	Basic 2D and 3D plots: parametric space curve . Polygons with vertices. 3D contour lines, pie and bar charts.	13/04/2023	

AMITY SCHOOL OF ENGINEERING AND TECHNOLOGY

AMITY UNIVERSITY CAMPUS, SECTOR 125, NOIDA-201303



BASIC SIMULATION LAB
PRACTICAL FILE COURSE
CODE: ES204

SUBMITTED TO:
DR. LALA BHASKAR

SUBMITTED BY:
ANANYA SINGH
A2305221332
4CSE6(X)