

INDEX

S.No.	Category of Assignment (_) Mandatory Experiment*	Code	Name of Experiment	Date of experiment	Date of Evaluation	Max. Marks	Marks obtained	Signature of Faculty
1.		LR (15)	CREATING A ONE AND TWO-DIMENSIONAL ARRAY (ROW / COLUMN VECTOR) (MATRIX OF GIVEN SIZE) THEN, (A). PERFORM ARITHMETIC OPERATIONS - ADDITION, SUBTRACTION, MULTIPLICATION AND EXPONENTIATION. (B). PERFORM MATRIX OPERATIONS - INVERSE, TRANSPOSE, RANK WITH PLOTS.			1.5		
2.			PERFORMING MATRIX MANIPULATIONS - CONCATENATING, INDEXING, SORTING, SHIFTING, RESHAPING, RESIZING AND FLIPPING ABOUT A VERTICAL / HORIZONTAL AXIS; CREATING ARRAYS X & Y OF SIZE (1 X N) AND PERFORMING (A). RELATIONAL OPERATIONS - >, <=, >=, ~= (B). LOGICAL OPERATIONS - ~, &, , XOR.			1.5		
3.			GENERATING A SET OF COMMANDS ON A GIVEN VECTOR (EXAMPLE: X = [1 8 3 9 0 1]) TO (A). ADD UP THE VALUES OF THE ELEMENTS (CHECK WITH SUM) (B). COMPUTE THE RUNNING SUM (CHECK WITH SUM), WHERE RUNNING SUM FOR ELEMENT J = THE SUM OF THE ELEMENTS FROM 1 TO J, INCLUSIVE. (C) GENERATE A RANDOM SEQUENCE USING RAND() / RANDN() FUNCTIONS AND PLOT THEM.			1.5		
4.			EVALUATING A GIVEN EXPRESSION AND ROUNDING IT TO THE NEAREST INTEGER VALUE USING ROUND, FLOOR, CEIL AND FIX FUNCTIONS; ALSO, GENERATING AND PLOTS OF (A) TRIGONOMETRIC FUNCTIONS - SIN(T), COS(T), TAN(T), SEC(T), COSEC(T) AND COT(T) FOR A GIVEN DURATION, 'T'. (B) LOGARITHMIC AND OTHER FUNCTIONS – LOG(A), LOG10(A),			1.5		

			SQUARE ROOT OF A, REAL NTH ROOT OF A.					
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, X, X^3 , EX, $\exp(X^2)$ OVER THE INTERVAL $0 < X < 4$ (BY CHOOSING APPROPRIATE MESH VALUES FOR X TO OBTAIN SMOOTH CURVES), ON A RECTANGULAR PLOT.			1.5		
6.			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. TIME SCALE THE GENERATED SIGNAL FOR DIFFERENT VALUES. E.G. 2X, 4X, 0.25X, 0.0625X.			1.5		
7.			SOLVING FIRST, SECOND AND THIRD ORDER ORDINARY DIFFERENTIAL EQUATION USING BUILT-IN FUNCTIONS AND PLOT.			1.5		
8.			WRITING BRIEF SCRIPTS STARTING EACH SCRIPT WITH A REQUEST FOR INPUT (USING INPUT) TO EVALUATE THE FUNCTION H(T) USING IF-ELSE STATEMENT, WHERE $H(T) = (T - 10)$ FOR $0 < T < 100$ $H(T) = (0.45 T + 900)$ FOR $T > 100$. EXERCISE: TESTING THE SCRIPTS WRITTEN USING A). $T = 5$, $H = -5$ AND B). $T = 110$, $H = 949.5$			1.5		
9.			GENERATING A SQUARE WAVE FROM SUM OF SINE WAVES OF CERTAIN AMPLITUDE AND FREQUENCIES.			1.5		
10.			BASIC 2D AND 3D PLOTS: PARAMETRIC SPACE CURVE, POLYGONS WITH VERTICES, 3D CONTOUR LINES AND PIE AND BAR CHARTS.			1.5		
11.	Design Based Open Ended experiment **	PR (10)				10		
12.	Viva	Viva (10)				10		

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 the colon operator (first: step: last).

Creating one dimensional matrix:

Simply enter elements separated by comma or space.

Row = [E₁, E₂,...E_m]

Creating two dimensional matrixes:

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

$$A = [\text{row}_1; \text{row}_2; \dots; \text{row}_n]$$

RESULT:

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

CONCLUSION:

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

EXPERIMENT – 1 (B)

AIM:

Performing 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+B
```

```
C =
```

```
30  70 110
```

```
50  90 130
```

```
150 190 230
```

```
>> % Subtraction
```

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

C =

-10 -10 -10

-10 -10 -10

-10 -10 -10

>> % Multiplication

>> C=A*B

C =

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 a 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:

Performing Matrix operations - Inverse, Transpose and 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 inverse of the 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.

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

EXPERIMENT – 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 (R2011a)

PROCEDURE:

```
>> A= [70 50 30; 10 100 90; 40 60 20]
```

```
A =
```

```
70  50  30
10 100  90
40  60  20
```

```
>> B= [20 100 70; 40 30 50; 10 90 60]
```

```
B =
```

```
20 100  70
40  30  50
10  90  60
```

```
>> % Concatenation
```

```
>> % 1) Horizontal Concatenation
```

```
>> horzcat(A,B)
```

```
Ans =
```

```
70  50  30  20 100  70
10 100  90  40  30  50
40  60  20  10  90  60
```

```
>> % 2) Vertical Concatenation
```

```
>> vertcat(A,B)
```

Ans =

70 50 30

10 100 90

40 60 20

20 100 70

40 30 50

10 90 60

>> % Indexing

>> A(:,2)

Ans =

50

100

60

>> A(1,:)

Ans =

70 50 30

>> A(2,2)

Ans =

100

>> % Sorting

>> sort(A)

Ans =

10 50 20

40 60 30

70 100 90

```
>> sortrows(A)
```

```
ans =
```

```
10 100 90
```

```
40 60 20
```

```
70 50 30
```

```
>> sort(sort(A,2),1)
```

```
Ans =
```

```
10 40 60
```

```
20 50 70
```

```
30 90 100
```

```
>> % Shifting
```

```
>> circshift(A,2)
```

```
Ans =
```

```
10 100 90
```

```
40 60 20
```

```
70 50 30
```

```
>> circshift(A,[2,-1])
```

```
Ans =
```

```
100 90 10
```

```
60 20 40
```

```
50 30 70
```

```
>> % Reshaping
```

```
>> A= [20 60 30 40 50; 10 40 80 30 70; 80 60 10 70 90]
```

A =

20	60	30	40	50
10	40	80	30	70
80	60	10	70	90

>> reshape(A,5,3,[])

Ans =

20	60	30
10	30	70
80	80	50
60	10	70
40	40	90

>> **% Flipping**

>> **% Flipping Matrix about Vertical Axis**

>> fliplr(A)

Ans =

30	50	70
90	100	10
20	60	40

>> **% Flipping Matrix about Horizontal Axis**

>> flipud(A)

Ans =

40	60	20
10	100	90
70	50	30

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 are 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 – 2 (B)

AIM:

Creating Arrays X & Y of given size (1 x N) and Performing Relational Operations - >, <, ==, <=, >=, ~=

SOFTWARE USED: MATLAB 7.12.0 (R2011a)

PROCEDURE:

```
>> A= [10 30; 50 70]
```

A =

```
10 30
```

```
50 70
```

```
>> B= [80 60; 40 20]
```

B =

```
80 60
```

```
40 20
```

```
>> % Equal To
```

```
>> eq(A,B)
```

Ans =

```
0 0
```

```
0 0
```

```
>> % Less Than
```

```
>> lt(A,B)
```

Ans =

```
1 1
```

```
0 0
```

```
>> % Greater Than
```

```
>> gt(A,B)
```

ns =

```
0 0
```



```
1 1
```

```
>> % Less Than Equal To
```

```
>> le(A,B)
```

```
Ans =
```

```
1 1
```

```
0 0
```

```
>> % Greater Than Equal To
```

```
>> ge(A,B)
```

```
Ans =
```

```
0 0
```

```
1 1
```

```
>> % Not Equal To
```

```
>> ne(A,B)
```

```
Ans =
```

```
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 (>, <, ==, <=, >=, ~=) were performed on them.

CONCLUSION:

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

EXPERIMENT – 2 (C)

AIM:

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

SOFTWARE USED: MATLAB 7.12.0 (R2011a)

PROCEDURE:

```
>> A= [1 0 1 0 1]
```

A =

```
1 0 1 0 1
```

```
>> B= [1 0 0 0 1]
```

B =

```
1 0 0 0 1
```

```
>> % NOT
```

```
>> ~A
```

Ans =

```
0 1 0 1 0
```

```
>> ~B
```

Ans =

```
0 1 1 1 0
```

```
>> % AND
```

```
>> A&B
```

Ans =

```
1 0 0 0 1
```

```
>> % OR
```

```
>> A|B
```

ns =

```
1 0 1 0 1
```

```
>> % XOR
```

```
>> xor(A,B)
```

```
Ans =
```

```
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, 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.

Internal Assessment (Mandatory Experiment) Sheet for Lab Experiment
Department of Computer Science & Engineering
Amity University, Noida (UP)

Marking Criteria

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

EXPERIMENT – 3 (A)

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)

SOFTWARE USED: MATLAB 7.12.0 (R2011a)

PROCEDURE:

```
>> X=[1 3 5 7 9 2 4 6 8 10]
```

```
X =
```

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

```
>> % SUM
```

```
>> Y=sum(X)
```

```
Y =
```

```
55
```

```
>> Y=sum(X(3:7))
```

```
Y =
```

```
27
```

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

```
Z =
```

```
1    4    9   16   25   27   31   37   45   55
```

DISCUSSION:

Sum of all/specific elements of a vector were calculated using sum ().

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

sum (A(i:j)) returns sum of ith to jth elements of the vector.

RESULT:

Matrices was created and sum of all as well as specific elements was calculated using sum ().

CONCLUSION:

Vector was created and sum of all as well as specific elements was successfully calculated.

EXPERIMENT – 3 (B)

AIM:

Generating a set of Commands on a given Vector (Example: $X = [1 \ 8 \ 3 \ 9 \ 0 \ 1]$) to compute the Running Sum (Check with sum), where Running Sum for element j = the sum of the elements from 1 to j , inclusive.

SOFTWARE USED: MATLAB 7.12.0 (R2011a)

PROCEDURE:

```
>> X=[1 3 5 7 9 2 4 6 8 10]
```

```
X =
```

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

```
>> Len=length(X)
```

```
Len =
```

```
    10
```

```
>> Sum=0
```

```
Sum =
```

```
     0
```

```
>> for i=1:Len
```

```
    Sum=Sum+X(i)
```

```
end
```

```
Sum =1
```

```
Sum = 4
```

```
Sum=9
```

```
Sum = 16
```

```
Sum =25
```

```
Sum =27
```

```
Sum =31
```

```
Sum =37
```

```
Sum = 45
```

Sum =55

DISCUSSION:

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.

RESULT:

Vector was created and its running sum was calculated using for loops.

CONCLUSION:

Running sum of a matrix was successfully calculated.

EXPERIMENT – 3 (C)

AIM:

Generating a Random Sequence using rand() / randn() functions and plotting them.

SOFTWARE USED: MATLAB 7.12.0 (R2011a)

PROCEDURE:

```
>> X=rand(3)
```

```
X =
```

```
    0.7922    0.0357    0.6787
```

```
    0.9595    0.8491    0.7577
```

```
    0.6557    0.9340    0.7431
```

```
>> Y=randn(3)
```

```
Y =
```

```
   -0.3034    0.8884   -0.8095
```

```
    0.2939   -1.1471   -2.9443
```

```
   -0.7873   -1.0689    1.4384
```

```
>> subplot(1,2,1)
```

```
>> plot(X)
```

```
>> title('Plot of X')
```

```
>> xlabel('X-axis')
```

```
>> ylabel('Y-axis')
```

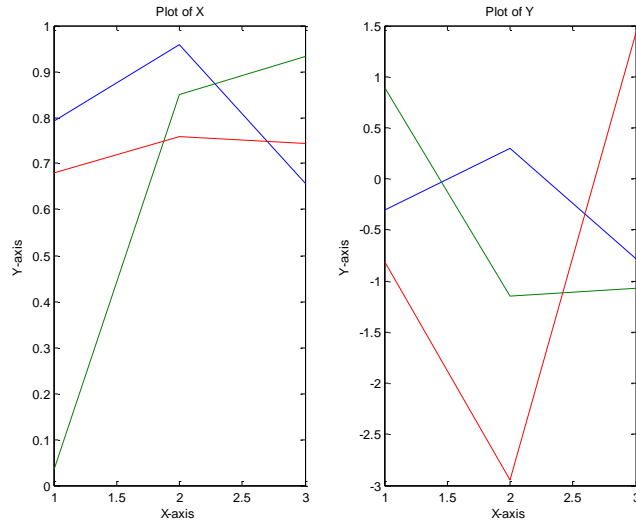
```
>> subplot(1,2,2)
```

```
>> plot(Y)
```

```
>> title('Plot of Y')
```

```
>> xlabel('X-axis')
```

```
>> ylabel('Y-axis')
```

DISCUSSION:

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 an 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 a real number and `plot(X1,Y1,...,Xn,Yn)` plots each vector Y_n versus vector X_n 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)` plots 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-n matrix of small axes, selects the pth axes object for the current plot, and returns the axes handle.

RESULT:

Random sequence was generated using `rand ()` and `randn ()`. And plotting was done using `plot`, `subplot` and `stem` functions.

CONCLUSION:

Functions `rand`, `randn`, `plot`, `subplot` and `stem` were successfully performed.

Internal Assessment (Mandatory Experiment) Sheet for Lab Experiment
Department of Computer Science & Engineering
Amity University, Noida (UP)

Marking Criteria

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

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 (R2011a)

PROCEDURE:

>> % Round

>> round(-99.97855)

Ans =

-100

>> round(99.97855)

Ans =

100

>> % Ceil

>> ceil(99.97855)

Ans =

100

>> ceil(-99.97855)

Ans =

-99

>> % Floor

>> floor(99.97855)

Ans =

99

>> floor(-99.97855)

Ans =

-100

>> % Fix

```
>> fix(50.55555)
```

```
Ans = 50
```

```
>> fix(-50.55555)
```

```
Ans = -50
```

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 .

Fixing:

$B = \text{fix}(A)$ rounds the elements of A toward zero.

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 (R2011a)

PROCEDURE:

>> % Generating and Plotting Trigonometric functions

>> t=pi:0.1:pi

t =

Columns 1 through 9

-3.1416 -3.0416 -2.9416 -2.8416 -2.7416 -2.6416 -2.5416 -2.4416 -2.3416

Columns 10 through 18

-2.2416 -2.1416 -2.0416 -1.9416 -1.8416 -1.7416 -1.6416 -1.5416 -1.4416

Columns 19 through 27

-1.3416 -1.2416 -1.1416 -1.0416 -0.9416 -0.8416 -0.7416 -0.6416 -0.5416

Columns 28 through 36

-0.4416 -0.3416 -0.2416 -0.1416 -0.0416 0.0584 0.1584 0.2584 0.3584

>> % Generating and Plotting sin (t)

>> X=sin(t)

X =

Columns 1 through 9

-0.0000 -0.0998 -0.1987 -0.2955 -0.3894 -0.4794 -0.5646 -0.6442 -0.7174

Columns 10 through 18

-0.7833 -0.8415 -0.8912 -0.9320 -0.9636 -0.9854 -0.9975 -0.9996 -0.9917

Columns 19 through 27

-0.9738 -0.9463 -0.9093 -0.8632 -0.8085 -0.7457 -0.6755 -0.5985 -0.5155

>> subplot(1,2,1)

>> plot(t,X)

>> xlabel('t')

>> ylabel('sin(t)')

>> title('Sin(t)')

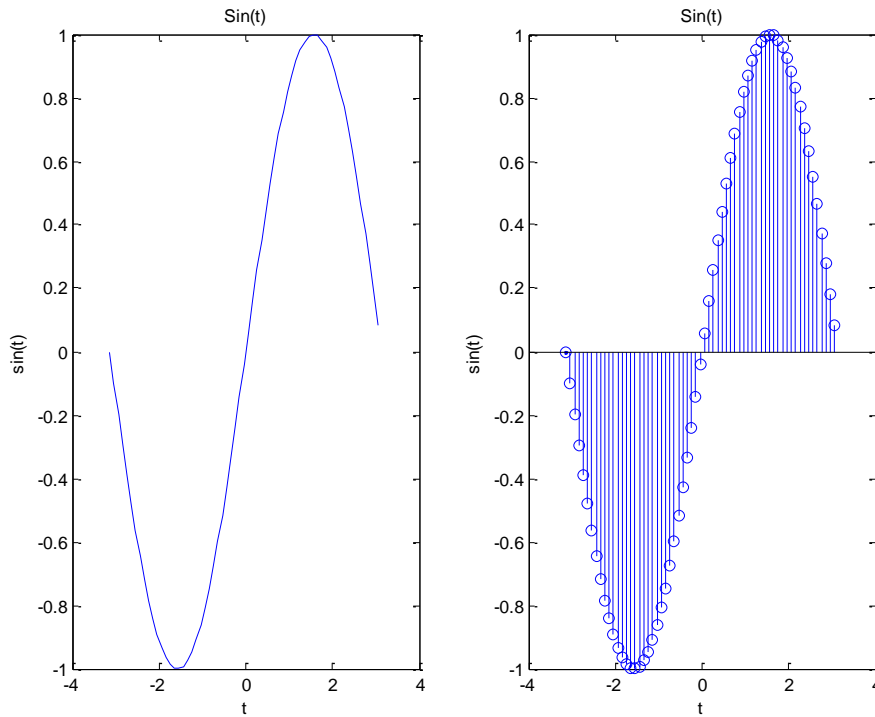
>> subplot(1,2,2)

>> stem(t,X)

>> xlabel('t')

>> ylabel('sin(t)')

>> title('Sin(t)')



```
>> % Generating and Plotting cos(t)
```

```
>> X=cos(t)
```

```
X =
```

```
Columns 1 through 9
```

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

```
Columns 10 through 18
```

```
-0.6216 -0.5403 -0.4536 -0.3624 -0.2675 -0.1700 -0.0707 0.0292 0.1288
```

```
Columns 19 through 27
```

```
0.2272 0.3233 0.4161 0.5048 0.5885 0.6663 0.7374 0.8011 0.8569
```

```
>> subplot(1,2,1)
```

```
>> plot(t,X)
```

```
>> xlabel('t')
```

```
>> ylabel('cos(t)')
```

```
>> title('Cos(t)')
```

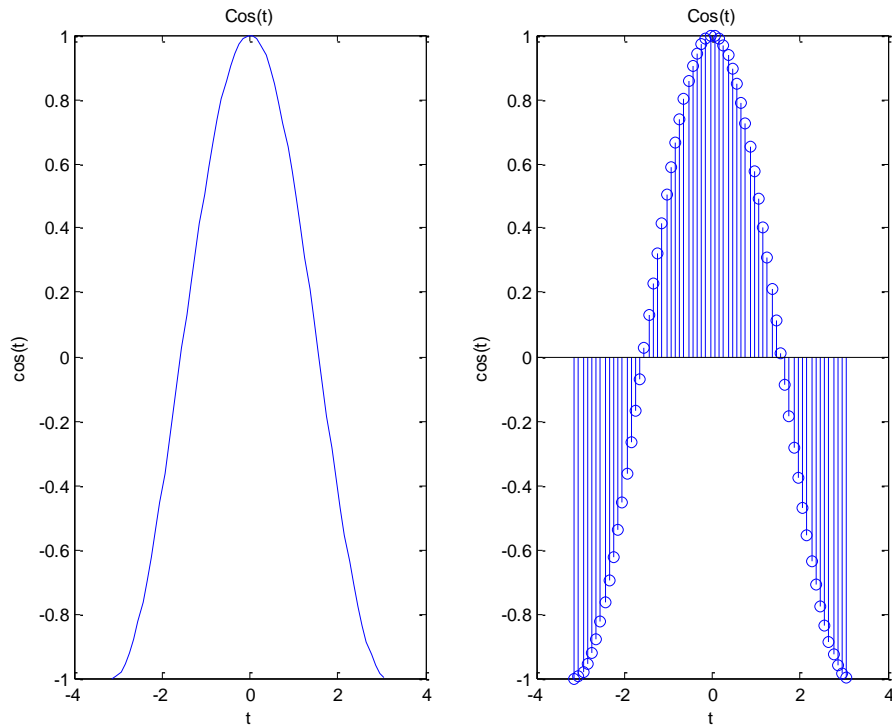
```
>> subplot(1,2,2)
```

```
>> stem(t,X)
```

```
>> xlabel('t')
```

```
>> ylabel('cos(t)')
```

```
>> title('Cos(t)')
```



>> % Generating and Plotting tan(t)

>> t=(-pi/2)+0.05:0.05:(pi/2)-0.05

t =

Columns 1 through 9

-1.5208 -1.4708 -1.4208 -1.3708 -1.3208 -1.2708 -1.2208 -1.1708 -1.1208

Columns 10 through 18

-1.0708 -1.0208 -0.9708 -0.9208 -0.8708 -0.8208 -0.7708 -0.7208 -0.6708

>> X=tan(t)

X =

Columns 1 through 9

-19.9833 -9.9666 -6.6166 -4.9332 -3.9163 -3.2327 -2.7395 -2.3652 -2.0702

Columns 10 through 18

-1.8305 -1.6310 -1.4617 -1.3154 -1.1872 -1.0734 -0.9712 -0.8785 -0.7936

Columns 19 through 27

-0.7151 -0.6421 -0.5736 -0.5090 -0.4475 -0.3888 -0.3323 -0.2776 -0.2245

Columns 28 through 36

-0.1725 -0.1214 -0.0709 -0.0208 0.0292 0.0794 0.1299 0.1811 0.2333

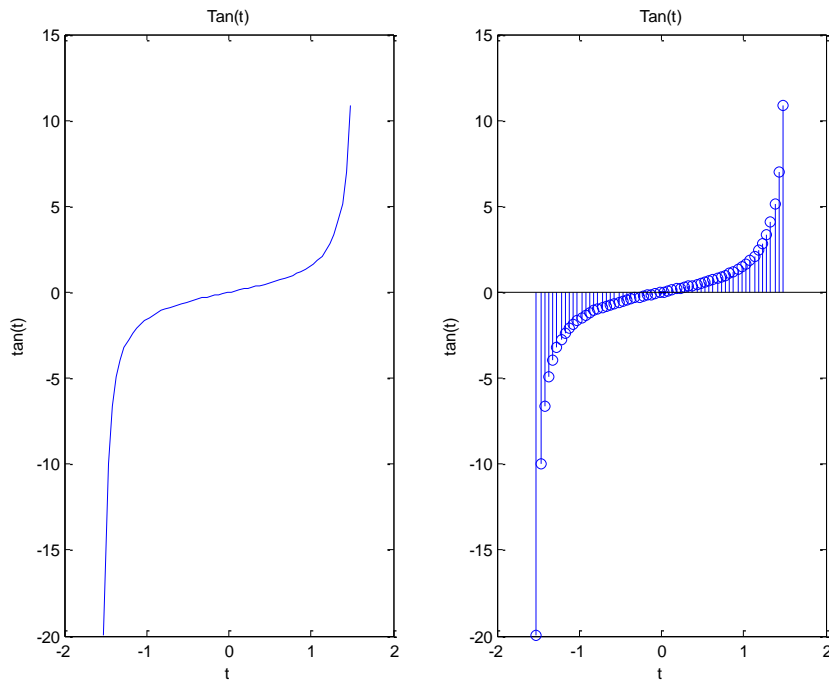
Columns 37 through 45 0.2867 0.3416 0.3985 0.4577 0.5196 0.5848 0.6540

0.7279 0.8073

```

>> subplot(1,2,1)
>> plot(t,X)
>> xlabel('t')
>> ylabel('tan(t)')
>> title('Tan(t)')
>> subplot(1,2,2)
>> stem(t,X)
>> xlabel('t')
>> ylabel('tan(t)')
>> title('Tan(t)')

```



>> % Generating and Plotting asin(t)

```
>> t=-1:0.02:1
```

```
t =
```

```
Columns 1 through 9
```

```
-1.0000 -0.9800 -0.9600 -0.9400 -0.9200 -0.9000 -0.8800 -0.8600 -0.8400
```

```
Columns 10 through 18
```

```
-0.8200 -0.8000 -0.7800 -0.7600 -0.7400 -0.7200 -0.7000 -0.6800 -0.6600
```

```
Columns 19 through 27
```

```
-0.6400 -0.6200 -0.6000 -0.5800 -0.5600 -0.5400 -0.5200 -0.5000 -0.4800
```

```
Columns 28 through 36
```

```
-0.4600 -0.4400 -0.4200 -0.4000 -0.3800 -0.3600 -0.3400 -0.3200 -0.3000
```

```
Columns 37 through 4
```

```
>> X=asin(t)
```


X =

Columns 1 through 9

-1.5708 -1.3705 -1.2870 -1.2226 -1.1681 -1.1198 -1.0759 -1.0353 -0.9973

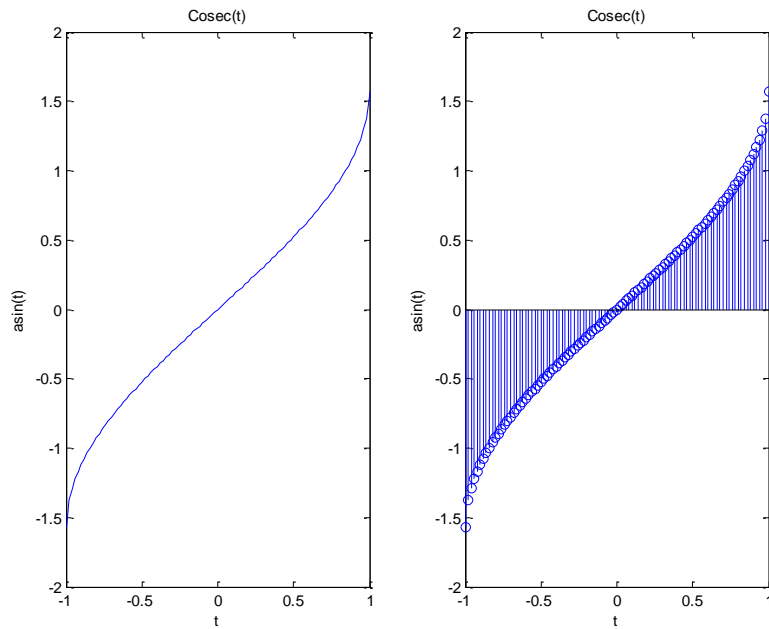
Columns 10 through 18

-0.9614 -0.9273 -0.8947 -0.8633 -0.8331 -0.8038 -0.7754 -0.7478 -0.7208

Columns 19 through 27

-0.6945 -0.6687 -0.6435 -0.6187 -0.5944 -0.5704 -0.5469 -0.5236 -0.5007

```
>> subplot(1,2,1)
>> plot(t,X)
>> xlabel('t')
>> ylabel('asin(t)')
>> title('Cosec(t)')
>> subplot(1,2,2)
>> stem(t,X)
>> xlabel('t')
>> ylabel('asin(t)')
>> title('Cosec(t)')
```



>> % Generating and Plotting acos(t)

>> X=acos(t)

X =

Columns 1 through 9

3.1416	2.9413	2.8578	2.7934	2.7389	2.6906	2.6467	2.6061	2.5681
--------	--------	--------	--------	--------	--------	--------	--------	--------

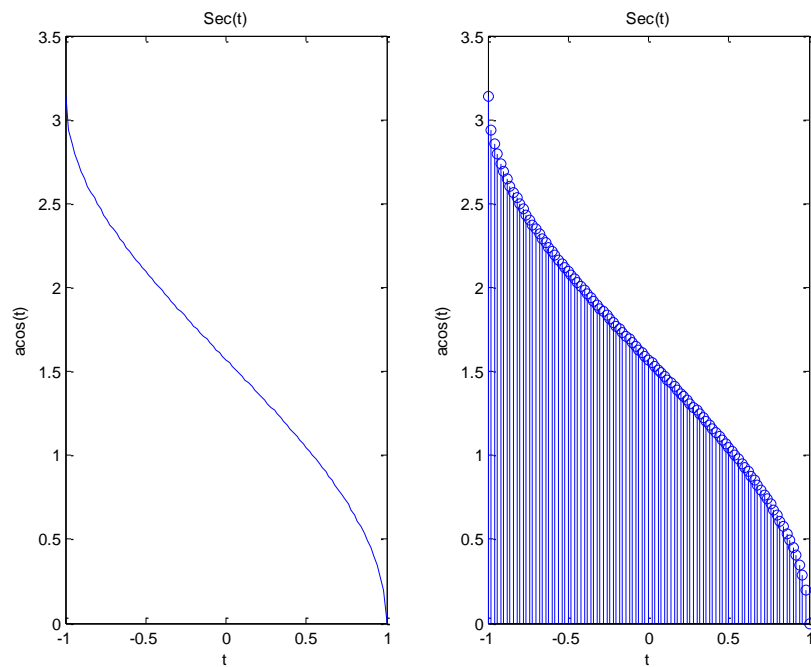
Columns 10 through 18

2.5322	2.4981	2.4655	2.4341	2.4039	2.3746	2.3462	2.3186	2.2916
--------	--------	--------	--------	--------	--------	--------	--------	--------

Columns 19 through 27

2.2653	2.2395	2.2143	2.1895	2.1652	2.1412	2.1176	2.0944	2.0715
--------	--------	--------	--------	--------	--------	--------	--------	--------

```
>> subplot(1,2,1)
>> plot(t,X)
>> xlabel('t')
>> ylabel('acos(t)')
>> title('Sec(t)')
>> subplot(1,2,2)
>> stem(t,X)
>> xlabel('t')
>> ylabel('acos(t)')
>> title('Sec(t)')
```



>> % Generating and Plotting atan(t)

```
>> t=-5:0.1:5
```

t =

Columns 1 through 9

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

Columns 10 through 18

```

-4.1000 -4.0000 -3.9000 -3.8000 -3.7000 -3.6000 -3.5000 -3.4000 -3.3000
Columns 19 through 27
-3.2000 -3.1000 -3.0000 -2.9000 -2.8000 -2.7000 -2.6000 -2.5000 -2.4000

```

```
>> X= tan(t)
```

```
X =
```

```

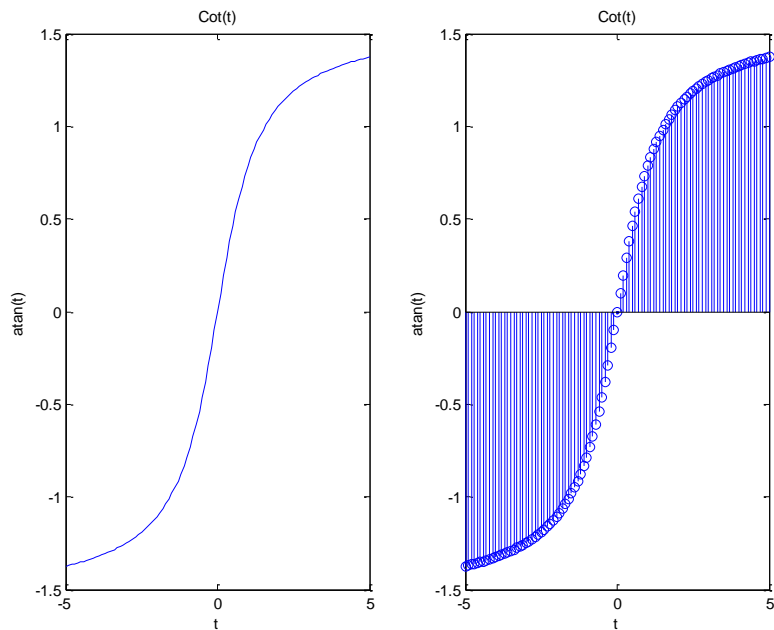
Columns 1 through 9
-1.3734 -1.3695 -1.3654 -1.3612 -1.3567 -1.3521 -1.3473 -1.3423 -1.3371
Columns 10 through 18
-1.3316 -1.3258 -1.3198 -1.3135 -1.3068 -1.2998 -1.2925 -1.2847 -1.2766
Columns 19 through 27
-1.2679 -1.2588 -1.2490 -1.2387 -1.2278 -1.2161 -1.2036 -1.1903 -1.1760
Columns 28 through 36
-1.1607 -1.1442 -1.1264 -1.1071 -1.0863 -1.0637 -1.0391 -1.0122 -0.9828

```

```

>> subplot(1,2,1)
>> plot(t,X)
>> xlabel('t')
>> ylabel('atan(t)')
>> title('Cot(t)')
>> subplot(1,2,2)
>> stem(t,X)
>> xlabel('t')
>> ylabel('atan(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 , \acos , \asin , \atan perform.

Plotting using plot ():

`Plot (Y)` plots the columns of Y versus the index of each value when Y is a real number and `plot(X1,Y1,...,Xn,Yn)` plots each vector Y_n versus vector X_n 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)` plots 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 rowwise. `h = subplot (m,n,p)` or `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 ($\sin(t)$, $\cos(t)$, $\tan(t)$, $\acos(t)$, $\asin(t)$, $\atan(t)$) 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 n^{th} root of A.

SOFTWARE USED: MATLAB 7.12.0 (R2011a)

PROCEDURE:

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

```
X =
```

```
Columns 1 through 15
```

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

```
Columns 16 through 20
```

```
16 17 18 19 20
```

```
>> % Log(X)
```

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

```
Y =
```

```
Columns 1 through 9
```

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

```
Columns 10 through 18
```

```
2.3026  2.3979  2.4849  2.5649  2.6391  2.7081  2.7726  2.8332  2.8904
```

```
Columns 19 through 20
```

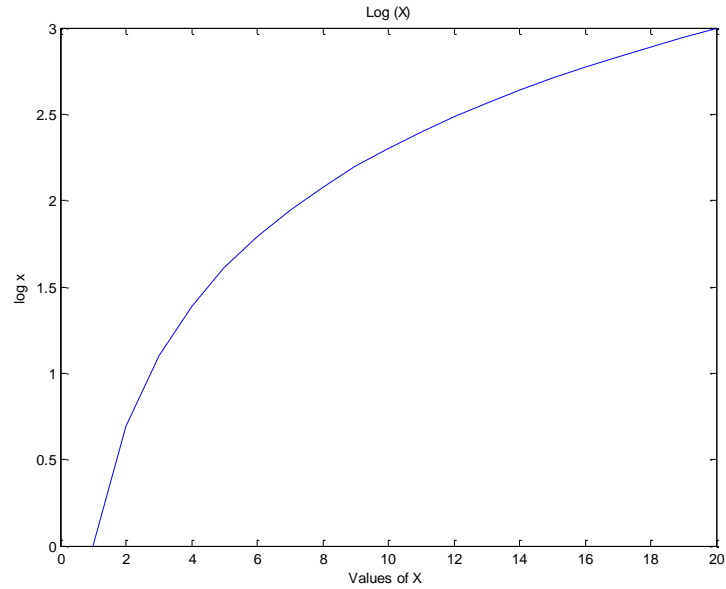
```
2.9444  2.9957
```

```
>> plot(X,Y)
```

```
title('Log (X)')
```

```
xlabel('Values of X ')
```

```
ylabel('log x')
```



```
>> % Log10(X)
```

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

```
Y =
```

```
Columns 1 through 9
```

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

```
Columns 10 through 18
```

```
    1.0000    1.0414    1.0792    1.1139    1.1461    1.1761    1.2041    1.2304    1.2553
```

```
Columns 19 through 20
```

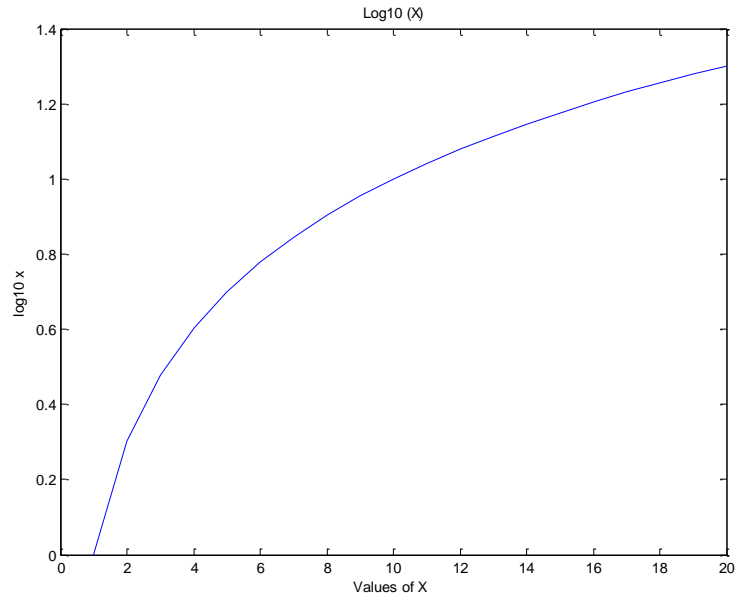
```
    1.2788    1.3010
```

```
>> plot(X,Y)
```

```
title('Log10 (X)')
```

```
xlabel('Values of X ')
```

```
ylabel('log10 x')
```



```
>> % Sqrt(X)
```

```
>> Y=sqrt(X)
```

```
Y =
```

```
Columns 1 through 9
```

```
1.0000 1.4142 1.7321 2.0000 2.2361 2.4495 2.6458 2.8284 3.0000
```

```
Columns 10 through 18
```

```
3.1623 3.3166 3.4641 3.6056 3.7417 3.8730 4.0000 4.1231 4.2426
```

```
Columns 19 through 20
```

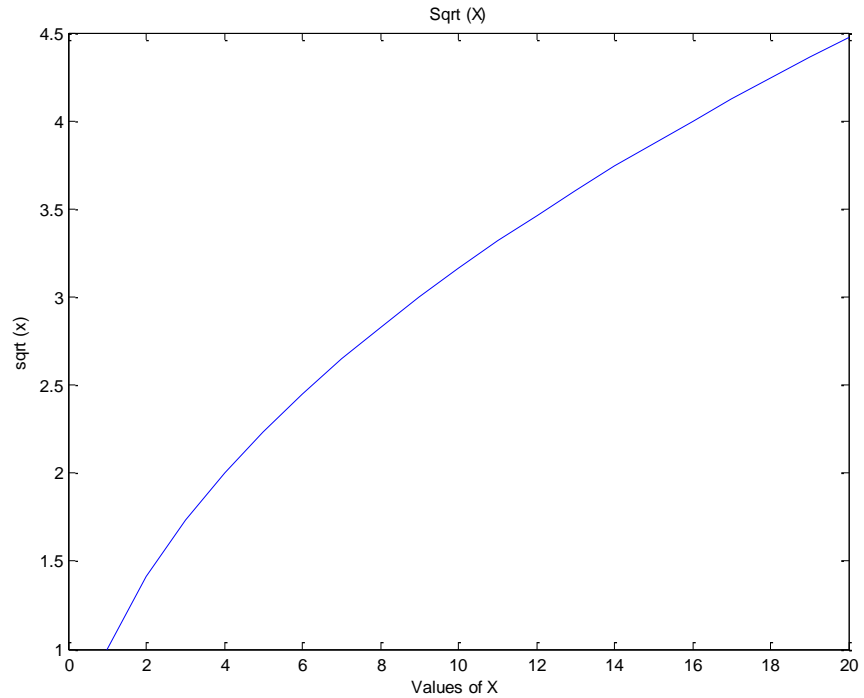
```
4.3589 4.4721
```

```
>> plot(X,Y)
```

```
title('Sqrt (X)')
```

```
xlabel('Values of X ')
```

```
ylabel('sqrt (x)')
```



```
>> % nthroot(X)
```

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

```
Y =
```

```
Columns 1 through 9
```

```
1.0000 1.1487 1.2457 1.3195 1.3797 1.4310 1.4758 1.5157 1.5518
```

```
Columns 10 through 18
```

```
1.5849 1.6154 1.6438 1.6703 1.6952 1.7188 1.7411 1.7623 1.7826
```

```
Columns 19 through 20
```

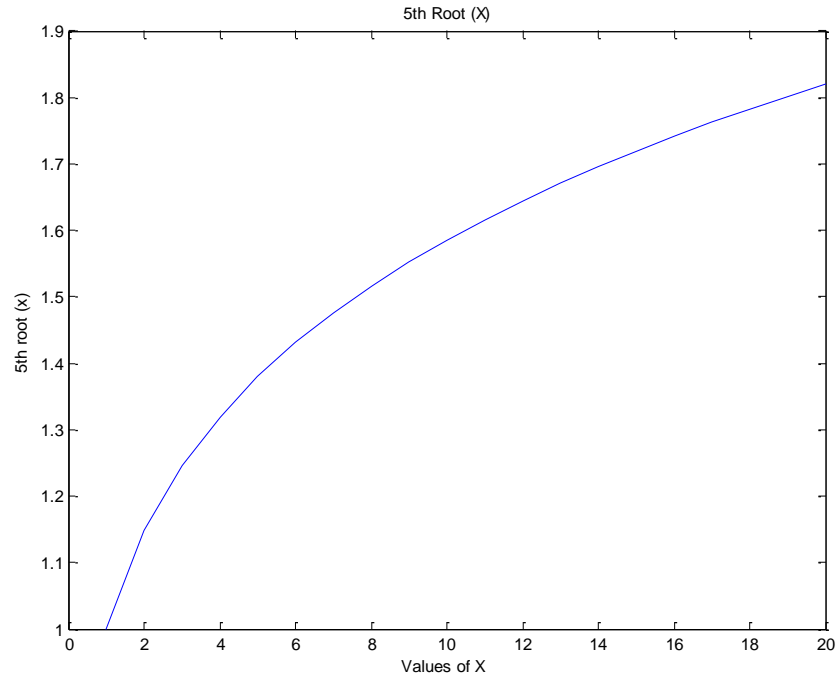
```
1.8020 1.8206
```

```
>> plot(X,Y)
```

```
title('5th Root (X)')
```

```
xlabel('Values of X ')
```

```
ylabel('5th root (x)')
```

DISCUSSION:

Logarithmic Functions:

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

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

Square root:

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

Real n^{th} root of A :

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

RESULT:

Matrix was created and Logarithmic Functions (\log and \log_{10}) 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.

Internal Assessment (Mandatory Experiment) Sheet for Lab Experiment
Department of Computer Science & Engineering
Amity University, Noida (UP)

Marking Criteria

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

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(R2011a)

PROCEDURE:

>> % Creating Vector X

```
>> for n=1:100
X(n)=((-1)^(n+1))/(2*n-1)
end
```

X =

Columns 1 through 9

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

Columns 10 through 18

-0.0526	0.0476	-0.0435	0.0400	-0.0370	0.0345	-0.0323	0.0303	-0.0286
---------	--------	---------	--------	---------	--------	---------	--------	---------

Columns 19 through 27

0.0270	-0.0256	0.0244	-0.0233	0.0222	-0.0213	0.0204	-0.0196	0.0189
--------	---------	--------	---------	--------	---------	--------	---------	--------

Columns 28 through 36

-0.0182	0.0175	-0.0169	0.0164	-0.0159	0.0154	-0.0149	0.0145	-0.0141
---------	--------	---------	--------	---------	--------	---------	--------	---------

Columns 37 through 45

0.0137	-0.0133	0.0130	-0.0127	0.0123	-0.0120	0.0118	-0.0115	0.0112
--------	---------	--------	---------	--------	---------	--------	---------	--------

>> % Adding elements of vector X

```
>> Y=sum(X(n))
```

Y =

-0.0050

>> % Creating interval vector Z

```
>> Z=0.04:0.04:4
```

Z =

Columns 1 through 9

0.0400	0.0800	0.1200	0.1600	0.2000	0.2400	0.2800	0.3200	0.3600
--------	--------	--------	--------	--------	--------	--------	--------	--------

Columns 10 through 18

0.4000	0.4400	0.4800	0.5200	0.5600	0.6000	0.6400	0.6800	0.7200
--------	--------	--------	--------	--------	--------	--------	--------	--------

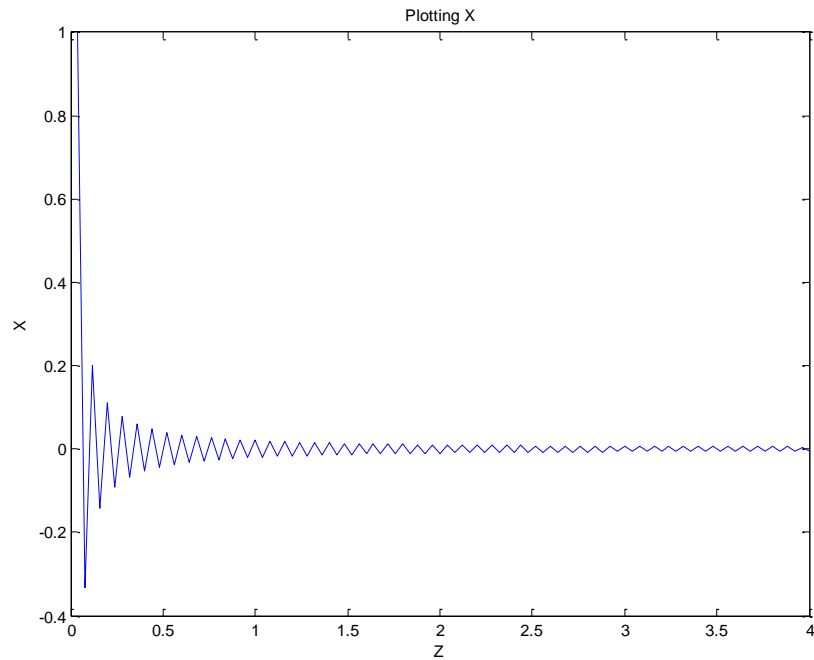
Columns 19 through 27

0.7600	0.8000	0.8400	0.8800	0.9200	0.9600	1.0000	1.0400	1.0800
--------	--------	--------	--------	--------	--------	--------	--------	--------

Columns 28 through 36

1.1200	1.1600	1.2000	1.2400	1.2800	1.3200	1.3600	1.4000	1.4400
--------	--------	--------	--------	--------	--------	--------	--------	--------

```
>> % Plotting Functions
>> % Plotting X
>> plot(Z,X)
>> xlabel('Z')
>> ylabel('X')
>> title('Plotting X')
```



```
>> % Plotting X^3
>> A=X.^3
A =
```

Columns 1 through 9

1.0000	-0.0370	0.0080	-0.0029	0.0014	-0.0008	0.0005	-0.0003	0.0002
--------	---------	--------	---------	--------	---------	--------	---------	--------

Columns 10 through 18

-0.0001	0.0001	-0.0001	0.0001	-0.0001	0.0000	-0.0000	0.0000	-0.0000
---------	--------	---------	--------	---------	--------	---------	--------	---------

Columns 19 through 27

0.0000	-0.0000	0.0000	-0.0000	0.0000	-0.0000	0.0000	-0.0000	0.0000
--------	---------	--------	---------	--------	---------	--------	---------	--------

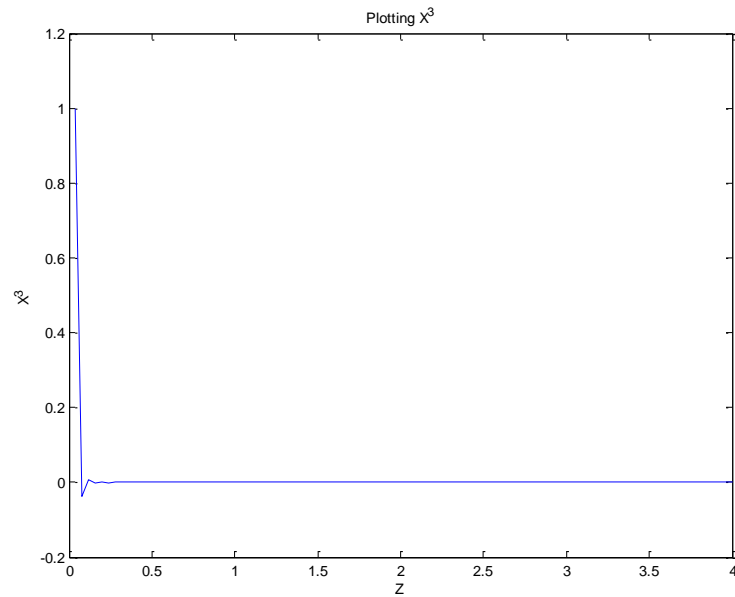
Columns 28 through 36

-0.0000	0.0000	-0.0000	0.0000	-0.0000	0.0000	-0.0000	0.0000	-0.0000
---------	--------	---------	--------	---------	--------	---------	--------	---------

Columns 37 through 45

0.0000	-0.0000	0.0000	-0.0000	0.0000	-0.0000	0.0000	-0.0000	0.0000
--------	---------	--------	---------	--------	---------	--------	---------	--------

```
>> plot(Z,A)
>> xlabel('Z')
>> ylabel('X^3')
>> title('Plotting X^3')
```



```
>> % Plotting e^X
```

```
>> A=exp(X)
```

```
A =
```

```
Columns 1 through 9
```

```
2.7183 0.7165 1.2214 0.8669 1.1175 0.9131 1.0800 0.9355 1.0606
```

```
Columns 10 through 18
```

```
0.9487 1.0488 0.9575 1.0408 0.9636 1.0351 0.9683 1.0308 0.9718
```

```
Columns 19 through 27
```

```
1.0274 0.9747 1.0247 0.9770 1.0225 0.9789 1.0206 0.9806 1.0190
```

```
Columns 28 through 36
```

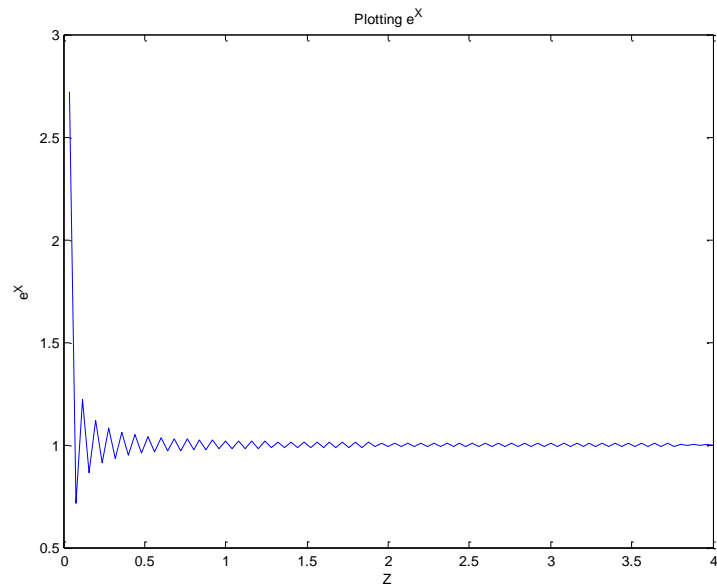
```
0.9820 1.0177 0.9832 1.0165 0.9843 1.0155 0.9852 1.0146 0.9860
```

```
>> plot(Z,A)
```

```
>> xlabel('Z')
```

```
>> ylabel('e^X')
```

```
>> title('Plotting e^X')
```



```
>> % Plotting  $e^{(X^2)}$ 
```

```
>> A=exp(X.^2)
```

```
A =
```

```
Columns 1 through 9
```

```
2.7183 1.1175 1.0408 1.0206 1.0124 1.0083 1.0059 1.0045 1.0035
```

```
Columns 10 through 18
```

```
1.0028 1.0023 1.0019 1.0016 1.0014 1.0012 1.0010 1.0009 1.0008
```

```
Columns 19 through 27
```

```
1.0007 1.0007 1.0006 1.0005 1.0005 1.0005 1.0004 1.0004 1.0004
```

```
Columns 28 through 36
```

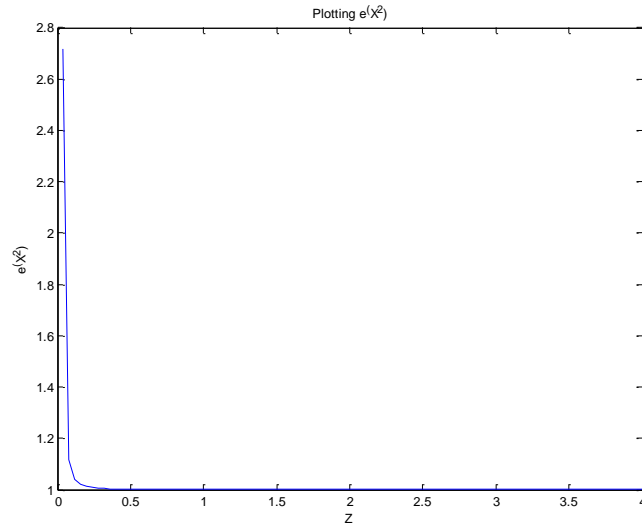
```
1.0003 1.0003 1.0003 1.0003 1.0003 1.0002 1.0002 1.0002 1.0002
```

```
>> plot(Z,A)
```

```
>> xlabel('Z')
```

```
>> ylabel('e^(X^2)')
```

```
>> title('Plotting  $e^{(X^2)}$ ')
```



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

EXPERIMENT - 6

AIM:

Generating a Sinusoidal Signal of a given frequency (say, 100Hz) and Plotting with Graphical Enhancements: Titling, Labeling, Adding Text, Adding Legends, Adding New Plots to Existing Plot, Printing Text in Greek Letters, Plotting as Multiple and Subplot.

SOFTWARE USED: MATLAB 7.12.0(R2011a)

PROCEDURE:

```
>> % Generating parameter t
```

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

```
t =
```

```
Columns 1 through 9
```

```
    0    0.1571    0.3142    0.4712    0.6283    0.7854    0.9425    1.0996    1.2566
```

```
Columns 10 through 18
```

```
    1.4137    1.5708    1.7279    1.8850    2.0420    2.1991    2.3562    2.5133    2.6704
```

```
Columns 19 through 27
```

```
    2.8274    2.9845    3.1416    3.2987    3.4558    3.6128    3.7699    3.9270    4.0841
```

```
Columns 28 through 36
```

```
    4.2412    4.3982    4.5553    4.7124    4.8695    5.0265    5.1836    5.3407    5.4978
```

```
Columns 37 through 41
```

```
    5.6549    5.8119    5.9690    6.1261    6.2832
```

```
>> % Generating sin(t)
```

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

```
X =
```

```
Columns 1 through 9
```

```
    0    0.1564    0.3090    0.4540    0.5878    0.7071    0.8090    0.8910    0.9511
```

```
Columns 10 through 18
```

```
    0.9877    1.0000    0.9877    0.9511    0.8910    0.8090    0.7071    0.5878    0.4540
```

```
Columns 19 through 27
```

```
    0.3090    0.1564    0.0000   -0.1564   -0.3090   -0.4540   -0.5878   -0.7071   -0.8090
```

```
Columns 28 through 36
```

```
   -0.8910   -0.9511   -0.9877   -1.0000   -0.9877   -0.9511   -0.8910   -0.8090   -0.7071
```

```
Columns 37 through 41
```

```
   -0.5878   -0.4540   -0.3090   -0.1564   -0.0000
```

```
>> % Plotting with graphical enhancements
```

```
>> % Plotting sin(t) (With Titling, Labeling, Adding Text and Legends)
```

```
>> plot(t,X,'r-')
```

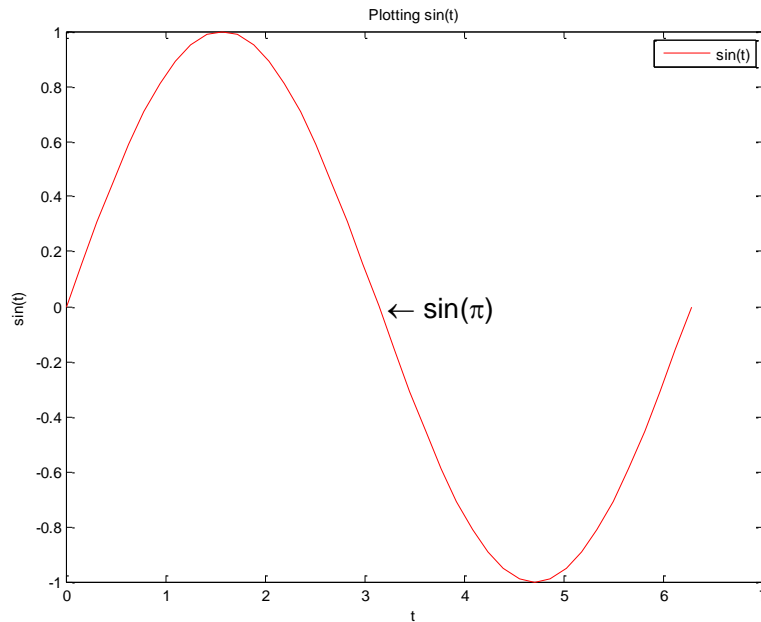
```
>> xlabel('t')
```

```
>> ylabel('sin(t)')
```

```
>> title('Plotting sin(t)')
```



```
>> legend('sin(t)')
>> text(pi,0,'← sin(π)',FontSize,18)
```



```
>> % Adding new plots to existing plots,Printing Text in Greek Letters
>> % Adding sin(3*t) to sin(t)
```

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

Columns 1 through 9

```
0 0.4540 0.8090 0.9877 0.9511 0.7071 0.3090 -0.1564 -0.5878
```

Columns 10 through 18

```
-0.8910 -1.0000 -0.8910 -0.5878 -0.1564 0.3090 0.7071 0.9511 0.9877
```

Columns 19 through 27

```
0.8090 0.4540 0.0000 -0.4540 -0.8090 -0.9877 -0.9511 -0.7071 -0.3090
```

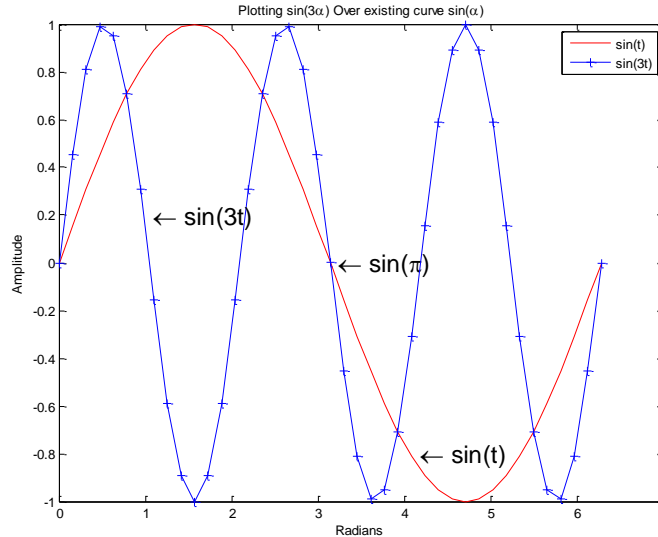
Columns 28 through 36

```
0.1564 0.5878 0.8910 1.0000 0.8910 0.5878 0.1564 -0.3090 -0.7071
```

Columns 37 through 41

```
-0.9511 -0.9877 -0.8090 -0.4540 -0.0000
```

```
>> plot(t,Y,'B +-')
>> xlabel('Radians')
>> ylabel('Amplitude')
>> legend('sin(t)','sin(3t)')
>> title('Plotting sin(3{α}) Over existing curve sin({α})')
>> text(4.1,-0.8,'← sin(t)',FontSize,18)
>> text(1,0.2,'← sin(3t)',FontSize,18)
```



DISCUSSION:

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

Plotting with Graphical Enhancements:

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

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

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

Adding Legends: $\text{legend('string1','string2',..)}$: displays a legend in the current axes using the 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 graph.

Printing Text in Greek Letters: You can define text that includes symbols and Greek letters using the text function, assigning the character sequence to the String property of text objects. You can also include these character sequences in the string arguments of the title, xlabel, ylabel, and zlabel functions. Example: $\text{xlabel('Time \backslash \mu\text{sec}')}$ displays Time μsec on graph.

RESULT:

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

CONCLUSION:

Sinusoidal Signal were successfully generated and plotted with several graphical enhancements.

Internal Assessment (Mandatory Experiment) Sheet for Lab Experiment
Department of Computer Science & Engineering
Amity University, Noida (UP)

Marking Criteria

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

EXPERIMENT - 9

AIM:

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

SOFTWARE USED: MATLAB 7.12.0(R2011a)

PROCEDURE:

>> % Defining Constant A

>> A=3

A =

3

>> % Generating parameter t

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

t =

Columns 1 through 9

0 0.1571 0.3142 0.4712 0.6283 0.7854 0.9425 1.0996 1.2566

Columns 10 through 18

1.4137 1.5708 1.7279 1.8850 2.0420 2.1991 2.3562 2.5133 2.6704

Columns 19 through 27

2.8274 2.9845 3.1416 3.2987 3.4558 3.6128 3.7699 3.9270 4.0841

Columns 28 through 36

4.2412 4.3982 4.5553 4.7124 4.8695 5.0265 5.1836 5.3407 5.4978

Columns 37 through 41

5.6549 5.8119 5.9690 6.1261 6.2832

>> % Generating & Plotting Square wave using Fourier Series Expansion

>> sum=0

sum =

0

>> for n=1:2:100

Y=(sin(n*t))/n

sum=sum+Y

end

Y =

Columns 1 through 9

0 0.0016 -0.0031 0.0046 -0.0059 0.0071 -0.0082 0.0090 -0.0096

Columns 10 through 18

0.0100 -0.0101 0.0100 -0.0096 0.0090 -0.0082 0.0071 -0.0059 0.0046

Columns 19 through 27

-0.0031 0.0016 -0.0000 -0.0016 0.0031 -0.0046 0.0059 -0.0071 0.0082

Columns 28 through 36

-0.0090 0.0096 -0.0100 0.0101 -0.0100 0.0096 -0.0090 0.0082 -0.0071

```
sum =
Columns 1 through 9
    0  0.8171  0.7692  0.7964  0.7769  0.7925  0.7792  0.7910  0.7801
Columns 10 through 18
    0.7905  0.7804  0.7905  0.7801  0.7910  0.7792  0.7925  0.7769  0.7964
Columns 19 through 27
    0.7692  0.8171  0.0000 -0.8171 -0.7692 -0.7964 -0.7769 -0.7925 -0.7792
```

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

```
C =
Columns 1 through 9
    0  3.1211  2.9383  3.0420  2.9675  3.0270  2.9764  3.0214  2.9799
Columns 10 through 18
    3.0193  2.9809  3.0193  2.9799  3.0214  2.9764  3.0270  2.9675  3.0420
Columns 19 through 27
    2.9383  3.1211  0.0000 -3.1211 -2.9383 -3.0420 -2.9675 -3.0270 -2.9764
>> plot(t,C)
```

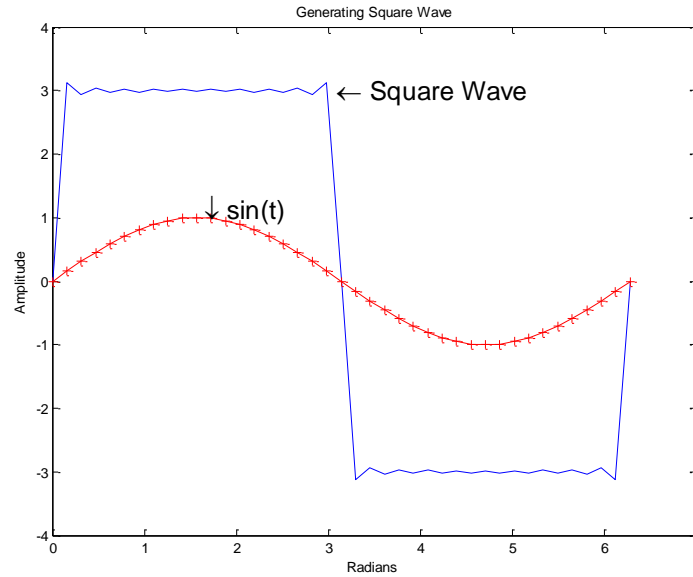
```
>> % Generating and Plotting sine wave over the existing curve
```

```
>> hold on
```

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

```
D =
```

```
Columns 1 through 9
    0  0.1564  0.3090  0.4540  0.5878  0.7071  0.8090  0.8910  0.9511
Columns 10 through 18
    0.9877  1.0000  0.9877  0.9511  0.8910  0.8090  0.7071  0.5878  0.4540
Columns 19 through 27
    0.3090  0.1564  0.0000 -0.1564 -0.3090 -0.4540 -0.5878 -0.7071 -0.8090
Columns 28 through 36
    -0.8910 -0.9511 -0.9877 -1.0000 -0.9877 -0.9511 -0.8910 -0.8090 -0.7071
Columns 37 through 41
    -0.5878 -0.4540 -0.3090 -0.1564 -0.0000
>> plot(t,D,'R+-')
>> xlabel('Radians')
>> ylabel('Amplitude')
>> title('Generating Square Wave')
>> text(pi/2,1.125,' \downarrow sin(t)','FontSize',18)
>> text(3,3,' \leftarrow Square Wave','FontSize',18)
```



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.

$$X(t) = \frac{4A}{\pi} \left[\sin(w*t) + \left(\frac{1}{3}\right)*\sin(3*w*t) + \left(\frac{1}{5}\right)*\sin(5*w*t) + .. \right]$$

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:

Square Wave was successfully generated and plotted.

Internal Assessment (Mandatory Experiment) Sheet for Lab Experiment
Department of Computer Science & Engineering
Amity University, Noida (UP)

Marking Criteria

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

EXPERIMENT – 7

AIM:

Solving First Order Ordinary Differential Equation using Built-in Functions.

Consider the following Ordinary Differential Equation:

$$x(dy/dx) + 2y = x^3$$

where, $dy/dx = (x^3 - 2y) / x$ $1 < x < 3$ and $y = 4.2$

SOFTWARE USED: MATLAB 7.12.0 (R2011a)

PROCEDURE:

>>% Clearing Everything

>> clc;clear all;clf

>> % Creating an Ordinary Differential Equation

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

ode1 =

@(x,y)(x^3-2*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	1.3500
1.3600	1.3700	1.3800	1.3900	1.4000	1.4100
1.4200	1.4300	1.4400	1.4500	1.4600	1.4700
1.4800	1.4900	1.5000	1.5100	1.5200	1.5300
1.5400	1.5500	1.5600	1.5700	1.5800	1.5900

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
2.6657	2.6454	2.6260	2.6074	2.5896	2.5726
2.5564	2.5409	2.5262	2.5122	2.4990	2.4864
3.9658	4.0005	4.0356	4.0711	4.1069	4.1431
4.1797	4.2166	4.2539	4.2915	4.3295	4.3679
4.4067	4.4458	4.4853	4.5252	4.5654	4.6060
4.6470	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

>>% Plotting the ordinary differential equation

>> plot(x,y,'linewidth',2)

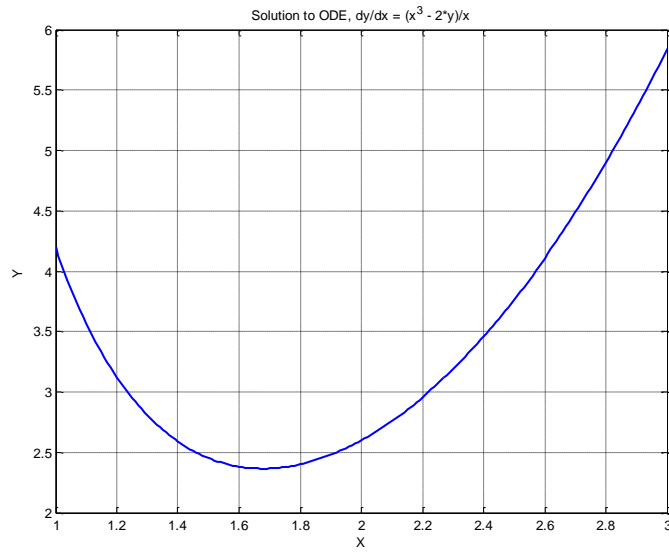
>> xlabel('X'),ylabel('Y'),grid on

>> 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 a one-step solver - in computing $y(tn)$, it needs only the solution at the immediately preceding time point, $y(tn-1)$.

RESULT:



CONCLUSION:

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

Internal Assessment (Mandatory Experiment) Sheet for Lab Experiment
Department of Computer Science & Engineering
Amity University, Noida (UP)

Marking Criteria

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

EXPERIMENT – 10

AIM: Basic 2D and 3D plots:

- parametric space curve.
- polygons with vertices.
- 3D contour lines, pie and bar charts.

SOFTWARE USED: MATLAB R2019b

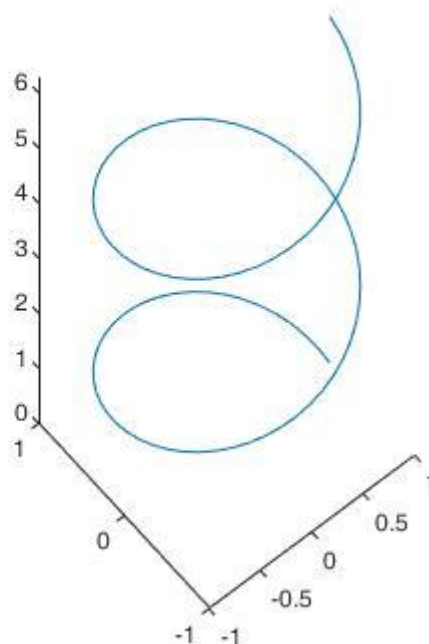
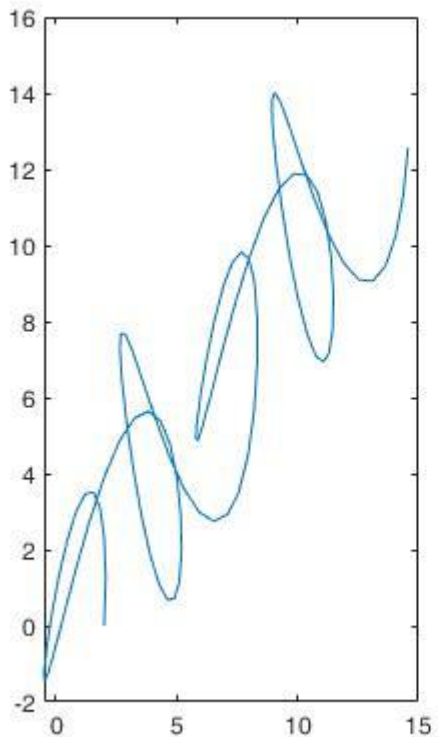
PROCEDURE:

%2D PLOT

```
clc;  
clear all;  
t=linspace(0,4*pi);  
x=t+2*cos(2*t);  
y=t+3*sin(3*t);  
subplot(1,2,1)  
plot(x,y);
```

%3D PLOT

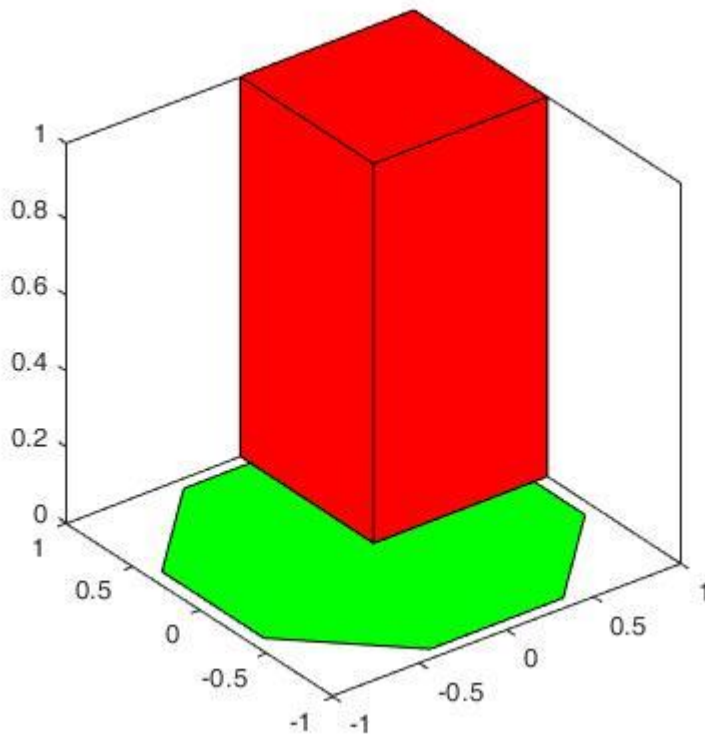
```
t=linspace(0,4*pi);  
x=cos(t);  
y=sin(t);  
z=t/2;  
subplot(1,2,2)  
plot3(x,y,z);
```



```

% polygon with vertices
t = (1/16:1/8:1)*2*pi;
x = cos(t);
y = sin(t);
fill(x,y,'g')
axis square
%3d
patch([0 0 1 1],[0 1 1 0],[1 1 1 1],'r')
patch([0 1 1 0],[0 0 0 0],[0 0 1 1],'r')
patch([0 0 0 0],[0 1 1 0],[0 0 1 1],'r')
view(-37.5, 30)
axis square

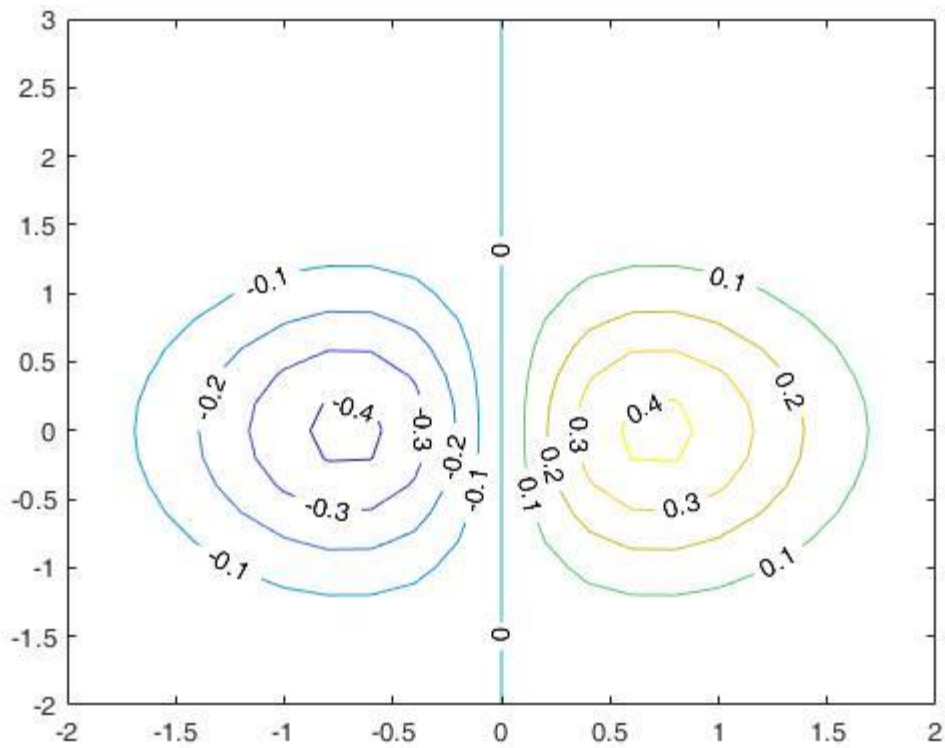
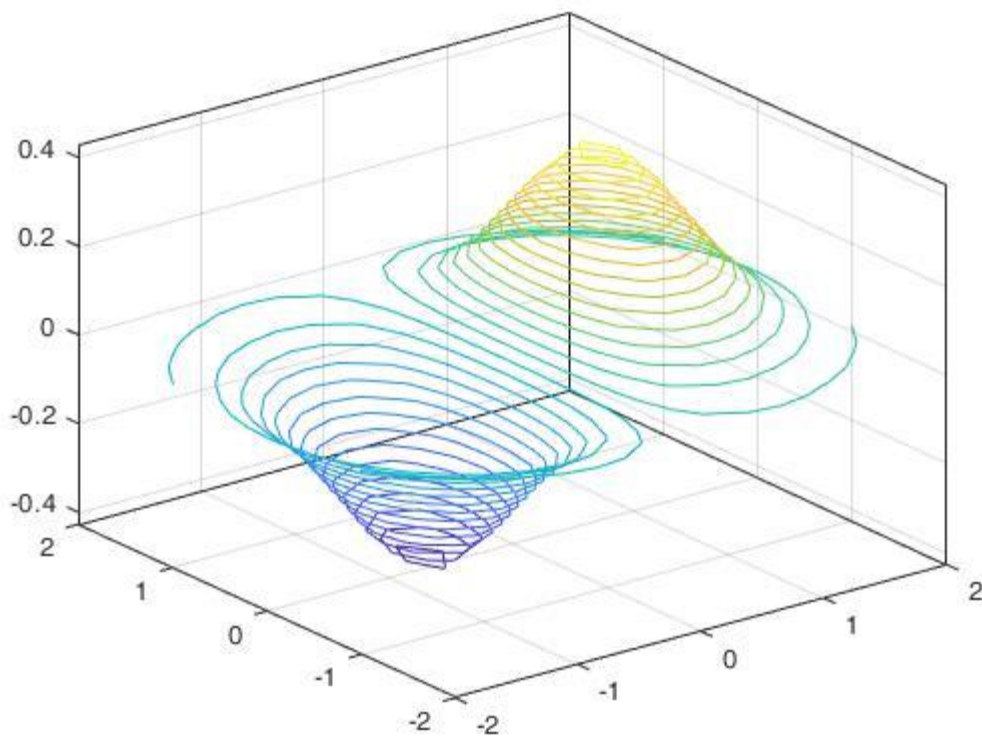
```



```

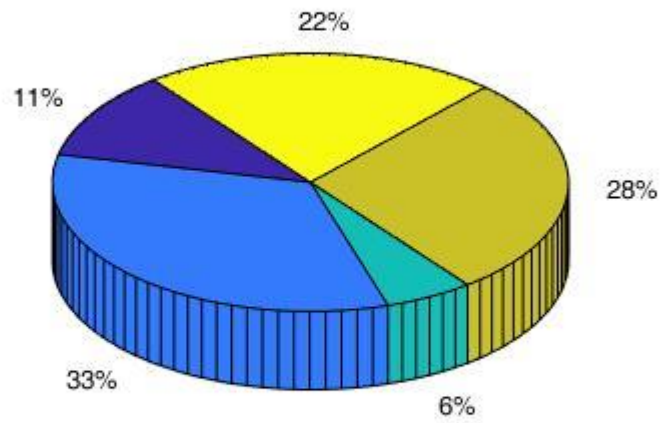
%3-D Contour Lines
x = -2:0.25:2;
[X,Y] = meshgrid(x);
Z = X.*exp(-X.^2-Y.^2);
contour3(X,Y,Z,30)
% 2d
x = -2:0.2:2;
y = -2:0.2:3;
[X,Y] = meshgrid(x,y);
Z = X.*exp(-X.^2-Y.^2);
figure
contour(X,Y,Z,'ShowText','on')

```



%3D Pie Chart

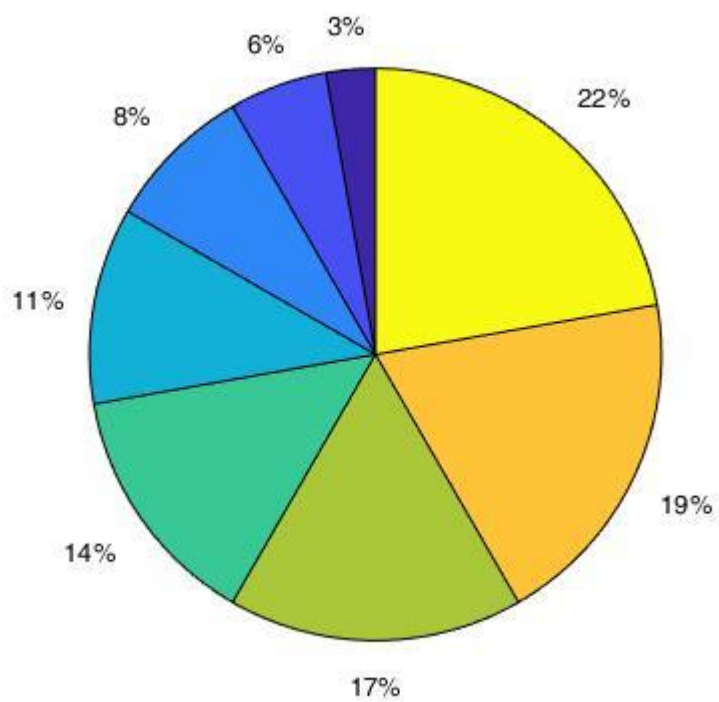
```
n=[1,3,0.5,2.5,2];
figure;
pie3(n);
```



%2d pie

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

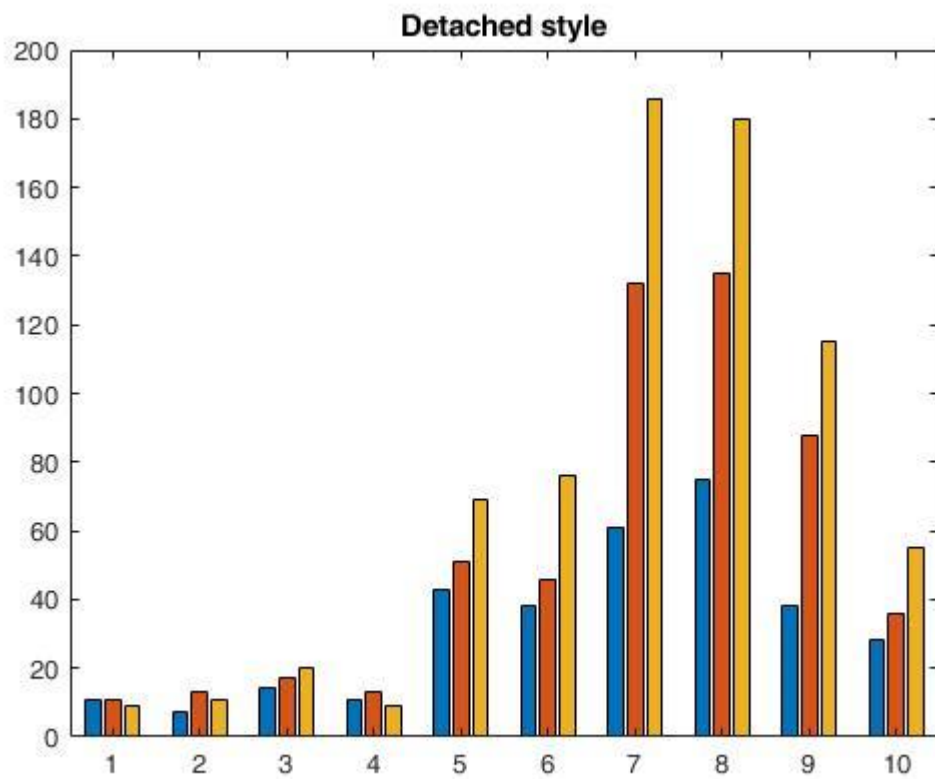
pie(x);



```

%2d bar
load count.dat
z=count(1:10,:);
figure;
bar(z);
title('Detached style');

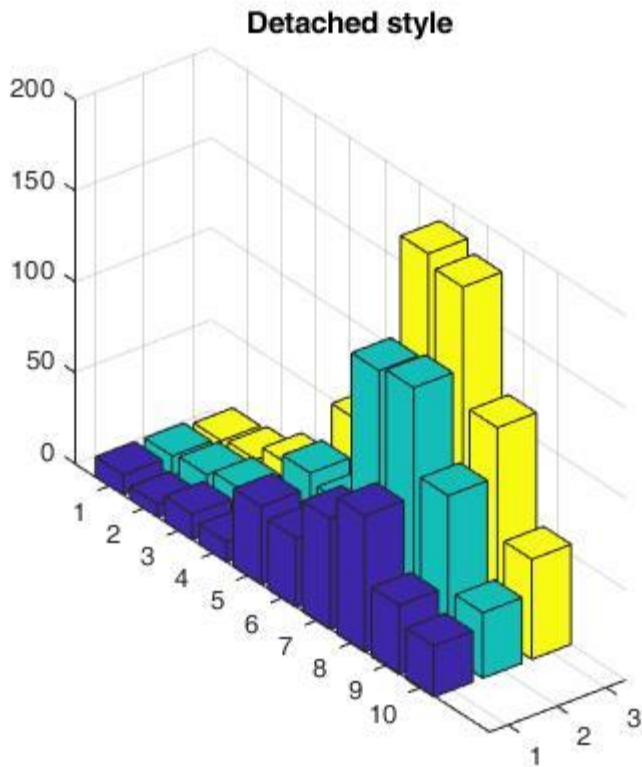
```



```

% 3d bar
load count.dat
z=count(1:10,:);
figure;
bar3(z);
title('Detached style');

```

DISSCUSION:

Contour () and pie () can be used as functions to design various charts.

RESULT:

The result is displayed above.

CONCLUSION:

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

Internal Assessment (Mandatory Experiment) Sheet for Lab Experiment
Department of Computer Science & Engineering
Amity University, Noida (UP)

Marking Criteria

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

EXPERIMENT – 8

AIM:

Writing brief Scripts starting each Script with a request for input (using input) to Evaluate the function $h(T)$ using if-else statement, where

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

Exercise: Testing the Scripts written using A). $T = 5$, $h = -5$ and B). $T = 110$, $h = 949.5$

SOFTWARE USED: MATLAB 7.12.0 (R2011a)

PROCEDURE:

```
>> % The Script
H=0
T=input('Enter the value of T: ')
if(T==0)
    disp('Enter a 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 expression1 evaluates as false and expression2 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 the value of T: 5

T =

5

For $T = 5$

$H =$

-5

$H = -5$

$H =$

0

Enter the 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.