

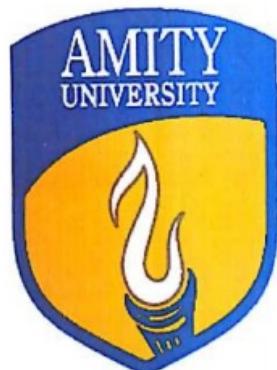
**AMITY SCHOOL OF ENGINEERING  
AND TECHNOLOGY**

**AMITY UNIVERSITY**

**----- UTTAR PRADESH -----**

**BASIC SIMULATION LAB**

**LAB MANUAL**



**Made By:**  
**Nikhita Wadhawan**  
**4MAE1**  
**13102**



Scanned with OKEN Scanner

S.No	Category of Assignment	Code	Experiment No.	Name of Experiment	Date of Allotment of experiment	Date of Evaluation	Max. Marks	Marks obtained	Signature of Faculty
1.	Mandatory Experiment*	LR (10)	1	Creating a One-Dimensional Array (Row / Column Vector) ; Creating a Two-Dimensional Array (Matrix of given size) and (A). Performing Arithmetic Operations - Addition, Subtraction, Multiplication and Exponentiation. (B). Performing Matrix operations - Inverse, Transpose, Rank.	17/12/2014	24/12/2014	1		<i>✓ 10/10</i>
2.			2	Performing Matrix Manipulations - Concatenating, Indexing, Sorting, Shifting, Reshaping, Resizing and Flipping about a Vertical Axis / Horizontal Axis; Creating Arrays X & Y of given size (1 x N) and Performing (A). Relational Operations - >, <, ==, <=, >=, ~= (B). Logical Operations - ~, &,  , XOR	24/12/14	07/01/15	1		<i>✓ 10/10</i>
3.			3	Generating a set of Commands on a given Vector (Example: X = [1 8	07/01/15	14/01/15	1		<i>✓ 10/10</i>



4.	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. Also, Generating a Random Sequence using rand() / randn() functions and plotting them.				
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 - $\log(A), \log_{10}(A)$ , Square root of A, Real nth root of A.	21/01/15	28/01/15	1	<i>Upcoming</i>
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	04/02/15	11/02/15	1	<i>Upcoming</i>

		functions, $x$ , $x^3$ , $\exp(x)$ over the interval $0 < x < 4$ (by choosing appropriate mesh values for $x$ to obtain smooth curves), on A Rectangular Plot					
6.		6 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.	11/02/15	18/01/15	1		<del>CB 18/01/15</del>
7.		7 Solving First Order Ordinary Differential Equation using Built-in Functions.	04/03/15	11/03/15	1		<del>CB 11/03/15</del>
8.		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 $0 < T < 100$ = $(0.45 T + 900)$ for $T > 100$ . Exercise : Testing the Scripts	11/03/15	18/03/15	1		<del>CB 18/03/15</del>



			written using A). T = 5, h = -5 and B). T = 110, h = 949.5 Also, Creating a Graphical User Interface (GUI) and calling back the script from GUI.					
9.			9. Generating a Square Wave from sum of Sine Waves of certain Amplitude and Frequencies.	18/03/15	25/03/15	1		<del>Up To Date</del>
10.			10. Basic 2D and 3D plots: parametric space curve. polygons with vertices. 3D contour lines, pie and bar ch.	25/03/15	01/04/15	1		<del>Up To Date</del>
11.	Viva	Viva (5)		08/04/15		5		<del>Up To Date</del>



## 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 = [ $E_1, E_2, \dots, E_m$ ]



Scanned with OKEN Scanner

**Creating two dimensional matrix:**

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 were successfully created.

**CONCLUSION:**

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



Scanned with OKEN Scanner

## **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
```



Scanned with OKEN Scanner

```
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 (\*):**



Scanned with OKEN Scanner

$A \cdot B$  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.



Scanned with OKEN Scanner

### EXPERIMENT – 1 (C)

#### AIM:

Performing 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
```



Scanned with OKEN Scanner

```
>> 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.



Scanned with OKEN Scanner

Internal Assessment(Mandatory Experiment) Sheet for Lab Experiment Department of Mechanical & Automation Engg. Amity University , Noida (U.P)			
Programme	Btech MAE	Course Name	Basic Simulation
Course Code	ES204	Semester	4
Student Name	NIKHITA WADHAWAN	Enrollment No.	A2305413102
Marking Criteria			
Criteria	Total Marks	Marks Obtained	Comments
Concept	2		
Implementation	2		
Performance	2		
Total	6		



Scanned with OKEN Scanner

## 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
```



Scanned with OKEN Scanner

```
>> 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
```



Scanned with OKEN Scanner

```
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 =
```



Scanned with OKEN Scanner

```
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:**



Scanned with OKEN Scanner

**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.



Scanned with OKEN Scanner

## **EXPERIMENT – 2 (B)**

### **AIM:**

Creating Arrays X & Y of given size ( $1 \times N$ ) and Performing Relational Operations -  $>$ ,  $<$ ,  
 $==$ ,  $<=$ ,  $>=$ ,  $\sim=$

**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
```



Scanned with OKEN Scanner

**>> % Greater Than**

>> gt(A,B)

ans =

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



Scanned with OKEN Scanner

---

## **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.



Scanned with OKEN Scanner

## EXPERIMENT – 2 (C)

### AIM:

Creating Arrays X & Y of given size ( $1 \times N$ ) and Performing Logical Operations -  $\sim$ ,  $\&$ ,  $\mid$ , 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
```

```
ans =
```



Scanned with OKEN Scanner

```
1 0 1 0 1  
>> % XOR  
>> xor(A,B)  
ans =  
0 0 1 0 0
```

## DISCUSSION:

### Logical Operations:

The symbols  $\&$ ,  $|$ ,  $\sim$ ,  $xor$  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 ( $\sim$ ):

$\sim expr$  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  $C(i,j,\dots)$  is logical true (1) if  $A(i,j,\dots)$  or  $B(i,j,\dots)$ , but not both, is nonzero.

## RESULT:

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

## CONCLUSION:

Various Logical Operations were successfully operated and results were obtained.



Scanned with OKEN Scanner

Internal Assessment(Mandatory Experiment) Sheet for Lab Experiment  
Department of Mechanical & Automation Engg.  
Amity University , Noida (U.P)

Programme	Btech MAE	Course Name	Basic Simulation
Course Code	ES204	Semester	4
Student Name	NIKHITA WADHAWAN	Enrollment No.	A2305413102
Marking Criteria			
Criteria	Total Marks	Marks Obtained	Comments
Concept	2		
Implementation	2		
Performance	2		
Total	6		



Scanned with OKEN Scanner

## **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
```

```
>> % Sum of some elements
>> Y=sum(X(3:7))
Y =
27
```

```
>> % Cumulative Sum
>> Z=cumsum(X)
Z =
1   4   9   16  25  27  31  37  45  55
```

### **DISCUSSION:**



Scanned with OKEN Scanner

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.



Scanned with OKEN Scanner

---

### 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 =
```



Scanned with OKEN Scanner

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.



Scanned with OKEN Scanner

### 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
```

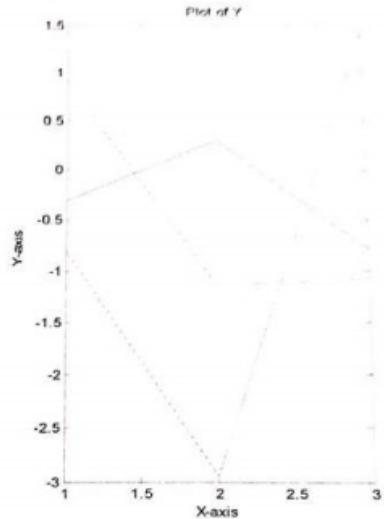
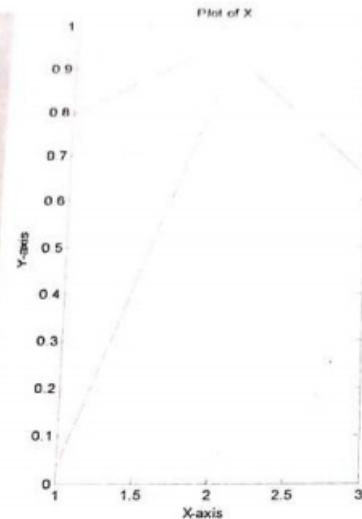
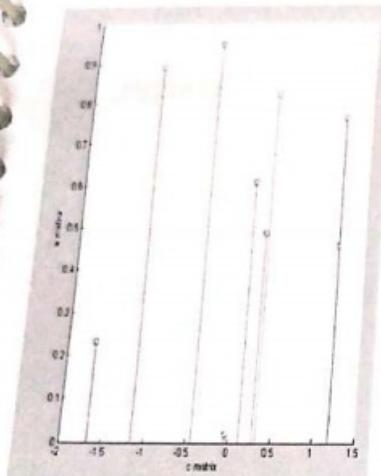
```
>> Y=randn(3)  
Y =  
-0.3034 0.8884 -0.8095  
0.2939 -1.1471 -2.9443  
-0.7873 -1.0689 1.4384
```

```
>> 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')
```



Scanned with OKEN Scanner

>> ylabel('Y-axis')



## DISCUSSION:

### Generating Random Sequence:

$r = \text{randn}(n)$  returns an  $n$ -by- $n$  matrix containing pseudorandom values drawn from the standard normal distribution while  $r = \text{rand}(n)$  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 ():



Scanned with OKEN Scanner

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.



Scanned with OKEN Scanner

**Internal Assessment(Mandatory Experiment) Sheet for Lab Experiment**  
**Department of Mechanical & Automation Engg.**  
**Amity University , Noida (U.P)**

Programme	Btech MAE	Course Name	Basic Simulation
Course Code	ES204	Semester	4
Student Name	NIKHITA WADHAWAN	Enrollment No.	A2305413102
Marking Criteria			
Criteria	Total Marks	Marks Obtained	Comments
Concept	2		
Implementation	2		
Performance	2		
Total	6		



Scanned with OKEN Scanner

## **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)
```



Scanned with OKEN Scanner

```
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.



Scanned with OKEN Scanner

## EXPERIMENT – 4 (B)

### AIM:

Generating and Plots of Trigonometric Functions -  $\sin(t), \cos(t), \tan(t), \sec(t), \cosec(t)$  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  
Columns 37 through 45  
0.4584 0.5584 0.6584 0.7584 0.8584 0.9584 1.0584 1.1584 1.2584  
Columns 46 through 54  
1.3584 1.4584 1.5584 1.6584 1.7584 1.8584 1.9584 2.0584 2.1584  
Columns 55 through 63  
2.2584 2.3584 2.4584 2.5584 2.6584 2.7584 2.8584 2.9584 3.0584
```

```
>> % 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  
Columns 28 through 36  
-0.4274 -0.3350 -0.2392 -0.1411 -0.0416 0.0584 0.1577 0.2555 0.3508
```



Scanned with OKEN Scanner

```
Columns 37 through 45
0.4425 0.5298 0.6119 0.6878 0.7568 0.8183 0.8716 0.9162 0.9516
Columns 46 through 54
0.9775 0.9937 0.9999 0.9962 0.9825 0.9589 0.9258 0.8835 0.8323
Columns 55 through 63
0.7728 0.7055 0.6313 0.5507 0.4646 0.3739 0.2794 0.1822 0.0831
```

```
>> 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
Columns 28 through 36
0.9041 0.9422 0.9710 0.9900 0.9991 0.9983 0.9875 0.9668 0.9365
Columns 37 through 45
0.8968 0.8481 0.7910 0.7259 0.6536 0.5748 0.4903 0.4008 0.3073
Columns 46 through 54
0.2108 0.1122 0.0124 -0.0875 -0.1865 -0.2837 -0.3780 -0.4685 -0.5544
Columns 55 through 63
-0.6347 -0.7087 -0.7756 -0.8347 -0.8855 -0.9275 -0.9602 -0.9833 -0.9965
```

```
>> subplot(1,2,1)
>> plot(t,X)
>> xlabel('t')
>> ylabel('cos(t)')
```



Scanned with OKEN Scanner

```
>> 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

Columns 19 through 27

-0.6208 -0.5708 -0.5208 -0.4708 -0.4208 -0.3708 -0.3208 -0.2708 -0.2208

Columns 28 through 36

-0.1708 -0.1208 -0.0708 -0.0208 0.0292 0.0792 0.1292 0.1792 0.2292

Columns 37 through 45

0.2792 0.3292 0.3792 0.4292 0.4792 0.5292 0.5792 0.6292 0.6792

Columns 46 through 54

0.7292 0.7792 0.8292 0.8792 0.9292 0.9792 1.0292 1.0792 1.1292

Columns 55 through 61

1.1792 1.2292 1.2792 1.3292 1.3792 1.4292 1.4792

>> 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



Scanned with OKEN Scanner

```
-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
Columns 46 through 54
 0.8935 0.9877 1.0917 1.2077 1.3386 1.4884 1.6622 1.8676 2.1154
Columns 55 through 61
 2.4218 2.8127 3.3317 4.0584 5.1554 7.0153 10.8874
```

```
>> 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



Scanned with OKEN Scanner

-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 45  
 -0.2800 -0.2600 -0.2400 -0.2200 -0.2000 -0.1800 -0.1600 -0.1400 -0.1200  
 Columns 46 through 54  
 -0.1000 -0.0800 -0.0600 -0.0400 -0.0200 0 0.0200 0.0400 0.0600  
 Columns 55 through 63  
 0.0800 0.1000 0.1200 0.1400 0.1600 0.1800 0.2000 0.2200 0.2400  
 Columns 64 through 72  
 0.2600 0.2800 0.3000 0.3200 0.3400 0.3600 0.3800 0.4000 0.4200  
 Columns 73 through 81  
 0.4400 0.4600 0.4800 0.5000 0.5200 0.5400 0.5600 0.5800 0.6000  
 Columns 82 through 90  
 0.6200 0.6400 0.6600 0.6800 0.7000 0.7200 0.7400 0.7600 0.7800  
 Columns 91 through 99  
 0.8000 0.8200 0.8400 0.8600 0.8800 0.9000 0.9200 0.9400 0.9600  
 0.9800 1.0000

>> 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  
 Columns 28 through 36  
 -0.4780 -0.4556 -0.4334 -0.4115 -0.3898 -0.3683 -0.3469 -0.3257 -0.3047  
 Columns 37 through 45  
 -0.2838 -0.2630 -0.2424 -0.2218 -0.2014 -0.1810 -0.1607 -0.1405 -0.1203  
 Columns 46 through 54  
 -0.1002 -0.0801 -0.0600 -0.0400 -0.0200 0 0.0200 0.0400 0.0600  
 Columns 55 through 63  
 0.0801 0.1002 0.1203 0.1405 0.1607 0.1810 0.2014 0.2218 0.2424  
 Columns 64 through 72  
 0.2630 0.2838 0.3047 0.3257 0.3469 0.3683 0.3898 0.4115 0.4334  
 Columns 73 through 81  
 0.4556 0.4780 0.5007 0.5236 0.5469 0.5704 0.5944 0.6187 0.6435  
 Columns 82 through 90  
 0.6687 0.6945 0.7208 0.7478 0.7754 0.8038 0.8331 0.8633 0.8947



Scanned with OKEN Scanner

Columns 91 through 99  
0.9273 0.9614 0.9973 1.0353 1.0759 1.1198 1.1681 1.2226 1.2870  
Columns 100 through 101  
1.3705 1.5708

```
>> 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  $\cos(t)$

>> X=cos(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  
Columns 28 through 36  
2.0488 2.0264 2.0042 1.9823 1.9606 1.9391 1.9177 1.8965 1.8755  
Columns 37 through 45  
1.8546 1.8338 1.8132 1.7926 1.7722 1.7518 1.7315 1.7113 1.6911  
Columns 46 through 54  
1.6710 1.6509 1.6308 1.6108 1.5908 1.5708 1.5508 1.5308 1.5108  
Columns 55 through 63  
1.4907 1.4706 1.4505 1.4303 1.4101 1.3898 1.3694 1.3490 1.3284  
Columns 64 through 72  
1.3078 1.2870 1.2661 1.2451 1.2239 1.2025 1.1810 1.1593 1.1374  
Columns 73 through 81  
1.1152 1.0928 1.0701 1.0472 1.0239 1.0004 0.9764 0.9521 0.9273  
Columns 82 through 90  
0.9021 0.8763 0.8500 0.8230 0.7954 0.7670 0.7377 0.7075 0.6761  
Columns 91 through 99  
0.6435 0.6094 0.5735 0.5355 0.4949 0.4510 0.4027 0.3482 0.2838



Scanned with OKEN Scanner

Columns 100 through 101  
0.2003 0

```
>> 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



Scanned with OKEN Scanner

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  
 Columns 28 through 36  
 -2.3000 -2.2000 -2.1000 -2.0000 -1.9000 -1.8000 -1.7000 -1.6000 -1.5000  
 Columns 37 through 45  
 -1.4000 -1.3000 -1.2000 -1.1000 -1.0000 -0.9000 -0.8000 -0.7000 -0.6000  
 Columns 46 through 54  
 -0.5000 -0.4000 -0.3000 -0.2000 -0.1000 0 0.1000 0.2000 0.3000  
 Columns 55 through 63  
 0.4000 0.5000 0.6000 0.7000 0.8000 0.9000 1.0000 1.1000 1.2000  
 Columns 64 through 72  
 1.3000 1.4000 1.5000 1.6000 1.7000 1.8000 1.9000 2.0000 2.1000  
 Columns 73 through 81  
 2.2000 2.3000 2.4000 2.5000 2.6000 2.7000 2.8000 2.9000 3.0000  
 Columns 82 through 90  
 3.1000 3.2000 3.3000 3.4000 3.5000 3.6000 3.7000 3.8000 3.9000  
 Columns 91 through 99  
 4.0000 4.1000 4.2000 4.3000 4.4000 4.5000 4.6000 4.7000 4.8000  
 Columns 100 through 101  
 4.9000 5.0000

$\gg X = \text{atan}(t)$

$X =$

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  
 Columns 37 through 45  
 -0.9505 -0.9151 -0.8761 -0.8330 -0.7854 -0.7328 -0.6747 -0.6107 -0.5404  
 Columns 46 through 54  
 -0.4636 -0.3805 -0.2915 -0.1974 -0.0997 0 0.0997 0.1974 0.2915  
 Columns 55 through 63  
 0.3805 0.4636 0.5404 0.6107 0.6747 0.7328 0.7854 0.8330 0.8761  
 Columns 64 through 72  
 0.9151 0.9505 0.9828 1.0122 1.0391 1.0637 1.0863 1.1071 1.1264  
 Columns 73 through 81  
 1.1442 1.1607 1.1760 1.1903 1.2036 1.2161 1.2278 1.2387 1.2490



Scanned with OKEN Scanner

```

1.2588 1.2679 1.2766 1.2847 1.2925 1.2998 1.3068 1.3135 1.3198
Columns 91 through 99
1.3258 1.3316 1.3371 1.3423 1.3473 1.3521 1.3567 1.3612 1.3654
Columns 100 through 101
1.3695 1.3734

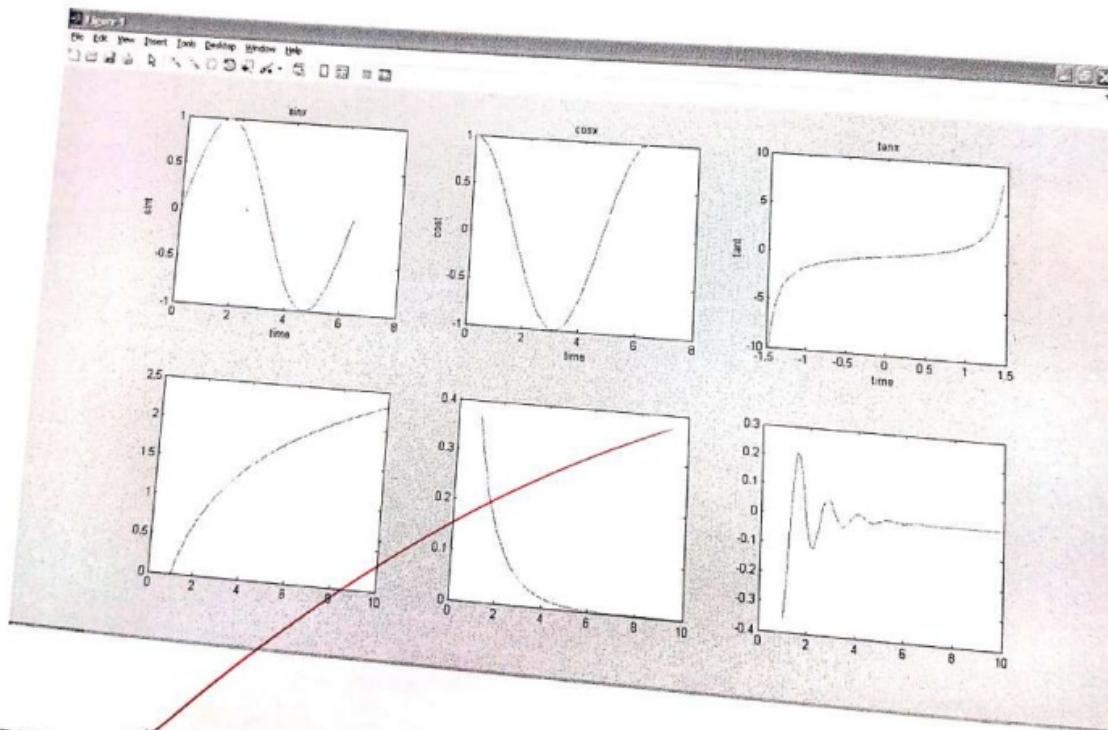
```

```

>> 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)')

```

### PLOTS:



### DISCUSSION:

#### Trigonometric Functions:



Scanned with OKEN Scanner

$Y = \sin(X)$  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( $X_1, Y_1, \dots, X_n, Y_n$ ) 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 ( $\sin(t)$ ,  $\cos(t)$ ,  $\tan(t)$ ,  $\text{acos}(t)$ ,  $\text{asin}(t)$ ,  
 $\text{atan}(t)$ ) were generated and plotted. And plotting was done using plot, subplot and  
stem functions.

**CONCLUSION:**

Various trigonometric functions were generated and plotted.



Scanned with OKEN Scanner

## EXPERIMENT - 4 (C)

### AIM:

Generating and Plots of Logarithmic and other Functions –  $\log(A)$ ,  $\log_{10}(A)$ , Square root of  $A$ , Real  $n^{\text{th}}$  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

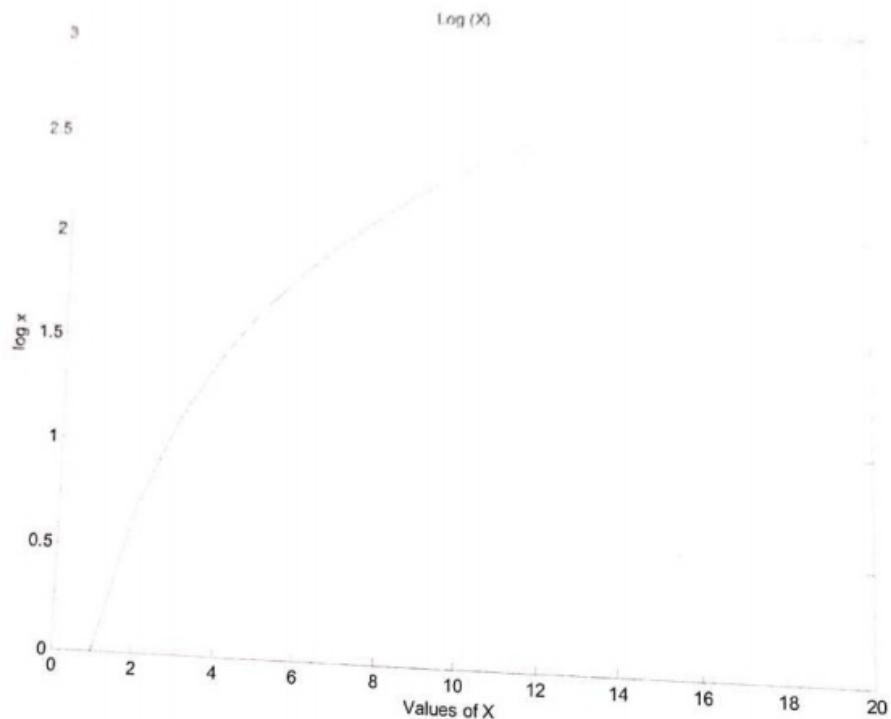
>> % Log(X)
>> Y=log(X)
Y =
Columns 1 through 9
0 0.6931 1.0986 1.3863 1.6094 1.7918 1.9459 2.0794 2.1972
Columns 10 through 18
2.3026 2.3979 2.4849 2.5649 2.6391 2.7081 2.7726 2.8332 2.8904
Columns 19 through 20
2.9444 2.9957

>> plot(X,Y)
title('Log (X)')
xlabel('Values of X ')
```



Scanned with OKEN Scanner

```
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

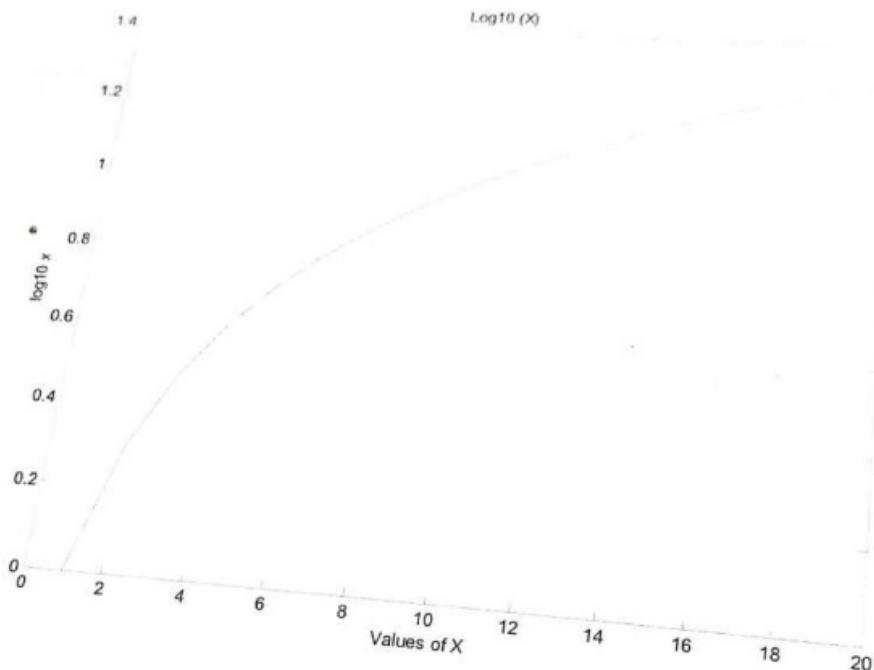
```
1.2788 1.3010
```

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



Scanned with OKEN Scanner

```
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

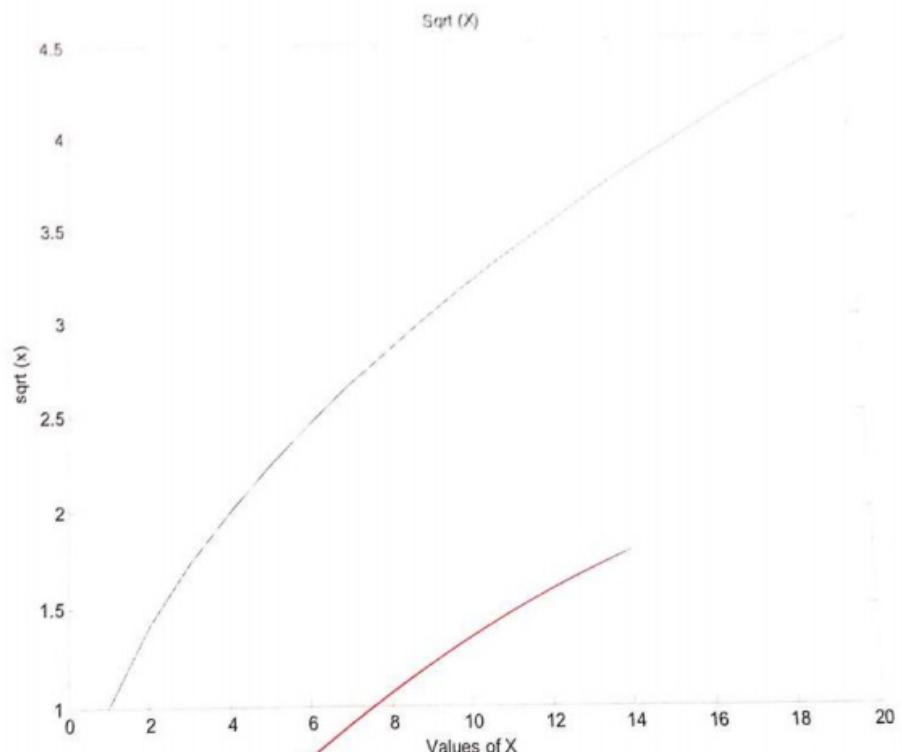


Scanned with OKEN Scanner

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

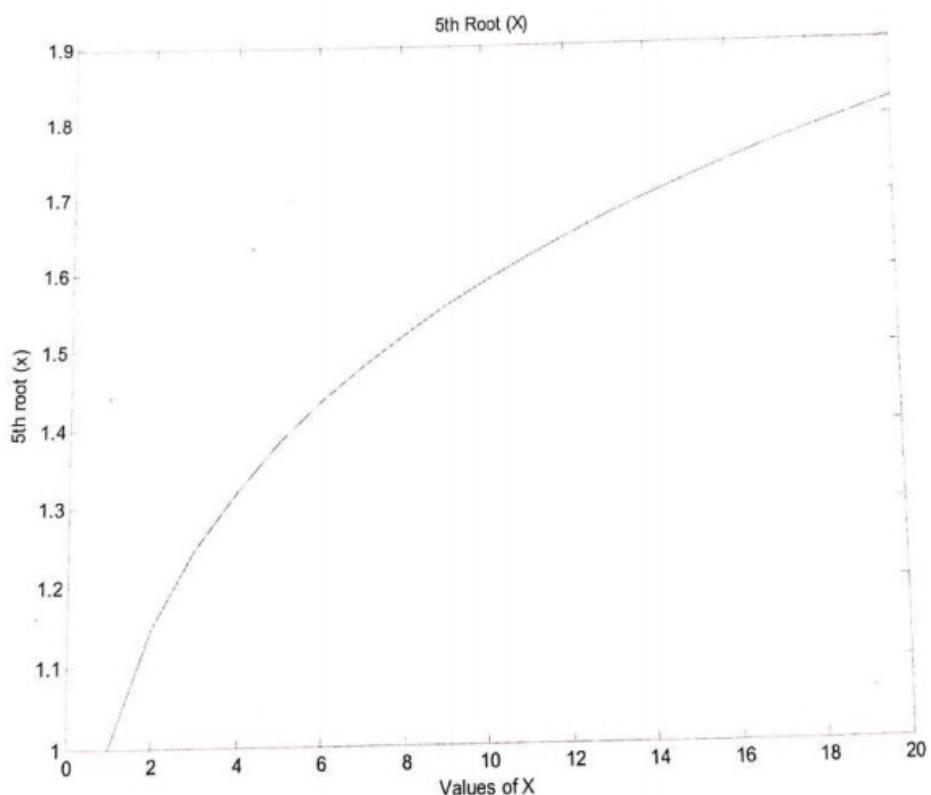


Scanned with OKEN Scanner

```

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)')

```



Scanned with OKEN Scanner

**DISCUSSION:****Logarithmic Functions:**

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

$Y = \log10(X)$  returns the base 10 logarithm of the elements of  $X$ .

**Square root:**

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

**Real  $n^{\text{th}}$  root of A:**

$y = \text{nthroot}(X, n)$  returns the real  $n^{\text{th}}$  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 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.



Scanned with OKEN Scanner

Internal Assessment(Mandatory Experiment) Sheet for Lab Experiment Department of Mechanical & Automation Engg. Amity University , Noida (U.P)			
Programme	Btech MAE	Course Name	Basic Simulation
Course Code	ES204	Semester	4
Student Name	NIKHITA WADHAWAN	Enrollment No.	A2305413102
Marking Criteria			
Criteria	Total Marks	Marks Obtained	Comments
Concept	2		
Implementation	2		
Performance	2		
Total	6		



Scanned with OKEN Scanner

## 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  
Columns 46 through 54  
-0.0110 0.0108 -0.0105 0.0103 -0.0101 0.0099 -0.0097 0.0095 -0.0093  
Columns 55 through 63  
0.0092 -0.0090 0.0088 -0.0087 0.0085 -0.0084 0.0083 -0.0081 0.0080  
Columns 64 through 72  
-0.0079 0.0078 -0.0076 0.0075 -0.0074 0.0073 -0.0072 0.0071 -0.0070  
Columns 73 through 81  
0.0069 -0.0068 0.0067 -0.0066 0.0065 -0.0065 0.0064 -0.0063 0.0062  
Columns 82 through 90  
-0.0061 0.0061 -0.0060 0.0059 -0.0058 0.0058 -0.0057 0.0056 -0.0056  
Columns 91 through 99  
0.0055 -0.0055 0.0054 -0.0053 0.0053 -0.0052 0.0052 -0.0051 0.0051  
Column 100  
-0.0050
```

```
>> % Adding elements of vector X
```



Scanned with OKEN Scanner

```
>> 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
Columns 37 through 45
1.4800 1.5200 1.5600 1.6000 1.6400 1.6800 1.7200 1.7600 1.8000
Columns 46 through 54
1.8400 1.8800 1.9200 1.9600 2.0000 2.0400 2.0800 2.1200 2.1600
Columns 55 through 63
2.2000 2.2400 2.2800 2.3200 2.3600 2.4000 2.4400 2.4800 2.5200
Columns 64 through 72
2.5600 2.6000 2.6400 2.6800 2.7200 2.7600 2.8000 2.8400 2.8800
Columns 73 through 81
2.9200 2.9600 3.0000 3.0400 3.0800 3.1200 3.1600 3.2000 3.2400
Columns 82 through 90
3.2800 3.3200 3.3600 3.4000 3.4400 3.4800 3.5200 3.5600 3.6000
Columns 91 through 99
3.6400 3.6800 3.7200 3.7600 3.8000 3.8400 3.8800 3.9200 3.9600
Column 100
4.0000
```

>> % Plotting Functions

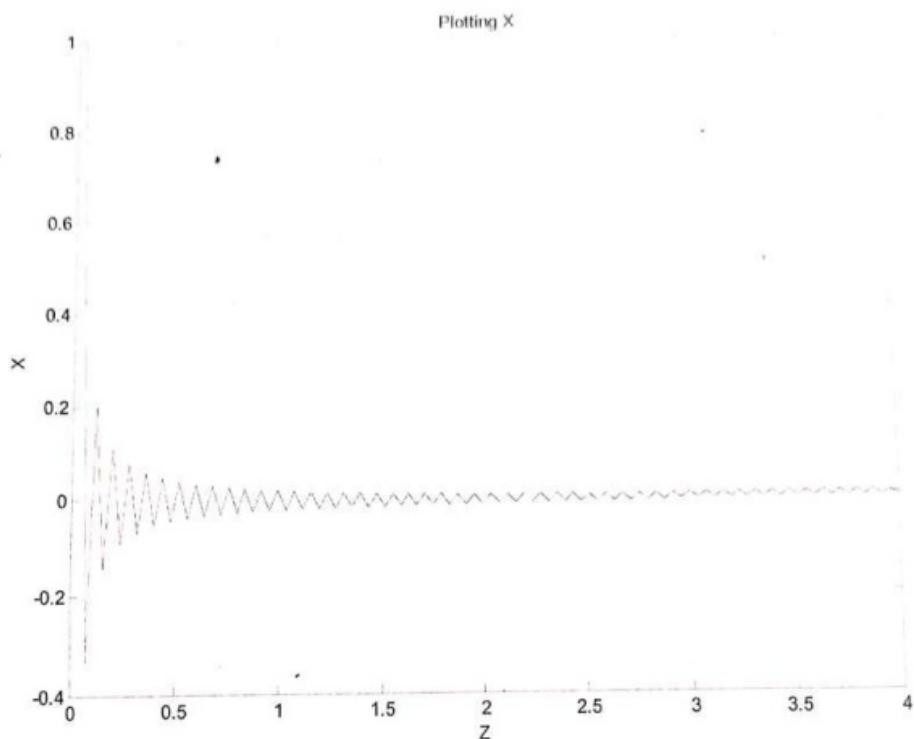
>> % Plotting X

```
>> plot(Z,X)
>> xlabel('Z')
>> ylabel('X')
```



Scanned with OKEN Scanner

```
>> 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

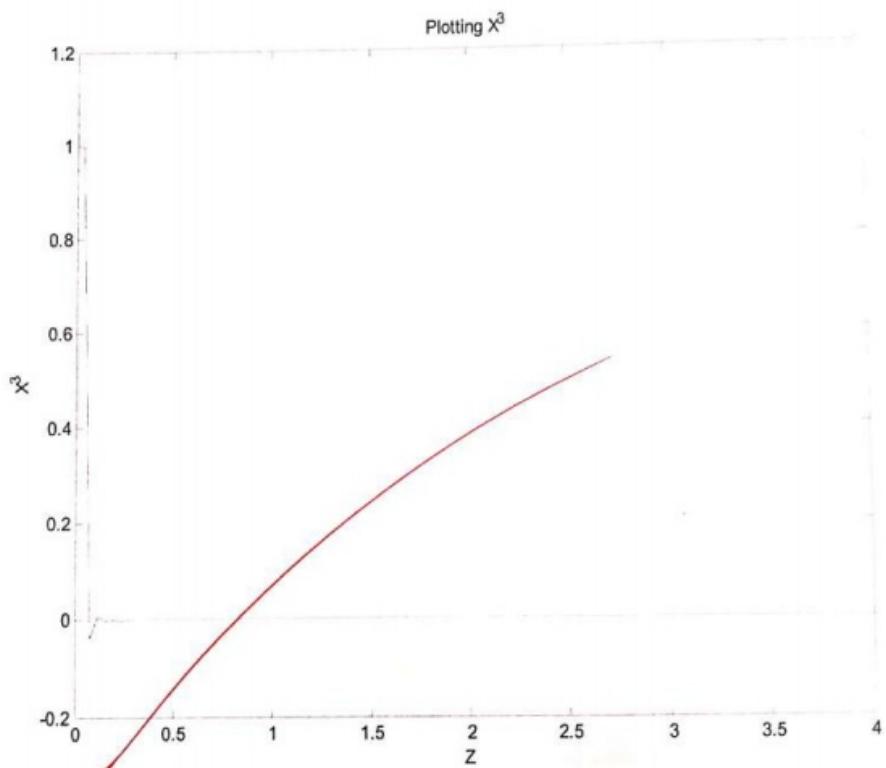
Columns 46 through 54



Scanned with OKEN Scanner

```
-0.0000  0.0000 -0.0000  0.0000 -0.0000  0.0000 -0.0000  0.0000 -0.0000  
Columns 55 through 63  
 0.0000 -0.0000  0.0000 -0.0000  0.0000 -0.0000  0.0000 -0.0000  0.0000  
Columns 64 through 72  
-0.0000  0.0000 -0.0000  0.0000 -0.0000  0.0000 -0.0000  0.0000 -0.0000  
Columns 73 through 81  
 0.0000 -0.0000  0.0000 -0.0000  0.0000 -0.0000  0.0000 -0.0000  0.0000  
Columns 82 through 90  
-0.0000  0.0000 -0.0000  0.0000 -0.0000  0.0000 -0.0000  0.0000 -0.0000  
Columns 91 through 99  
 0.0000 -0.0000  0.0000 -0.0000  0.0000 -0.0000  0.0000 -0.0000  0.0000  
Column 100  
-0.0000
```

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



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



Scanned with OKEN Scanner

```
>> 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
--------	--------	--------	--------	--------	--------	--------	--------	--------

Columns 37 through 45

1.0138	0.9868	1.0131	0.9874	1.0124	0.9880	1.0118	0.9886	1.0113
--------	--------	--------	--------	--------	--------	--------	--------	--------

Columns 46 through 54

0.9891	1.0108	0.9895	1.0104	0.9899	1.0100	0.9903	1.0096	0.9907
--------	--------	--------	--------	--------	--------	--------	--------	--------

Columns 55 through 63

1.0092	0.9910	1.0089	0.9913	1.0086	0.9916	1.0083	0.9919	1.0080
--------	--------	--------	--------	--------	--------	--------	--------	--------

Columns 64 through 72

0.9922	1.0078	0.9924	1.0075	0.9926	1.0073	0.9928	1.0071	0.9930
--------	--------	--------	--------	--------	--------	--------	--------	--------

Columns 73 through 81

1.0069	0.9932	1.0067	0.9934	1.0066	0.9936	1.0064	0.9937	1.0062
--------	--------	--------	--------	--------	--------	--------	--------	--------

Columns 82 through 90

0.9939	1.0061	0.9940	1.0059	0.9942	1.0058	0.9943	1.0057	0.9944
--------	--------	--------	--------	--------	--------	--------	--------	--------

Columns 91 through 99

1.0055	0.9946	1.0054	0.9947	1.0053	0.9948	1.0052	0.9949	1.0051
--------	--------	--------	--------	--------	--------	--------	--------	--------

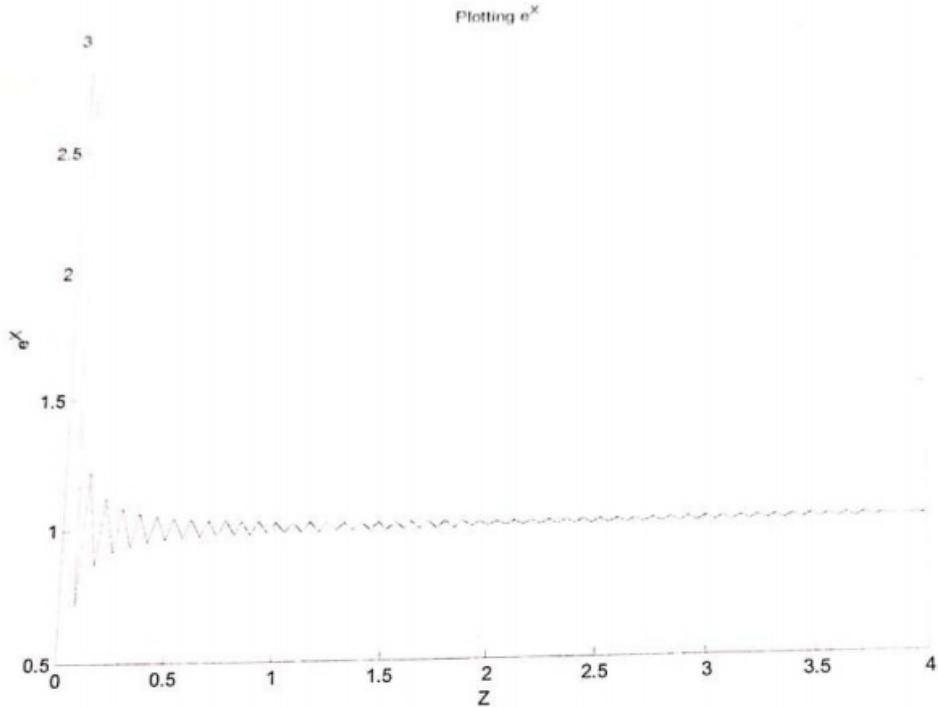
Column 100

0.9950
--------

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



Scanned with OKEN Scanner



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

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

A =

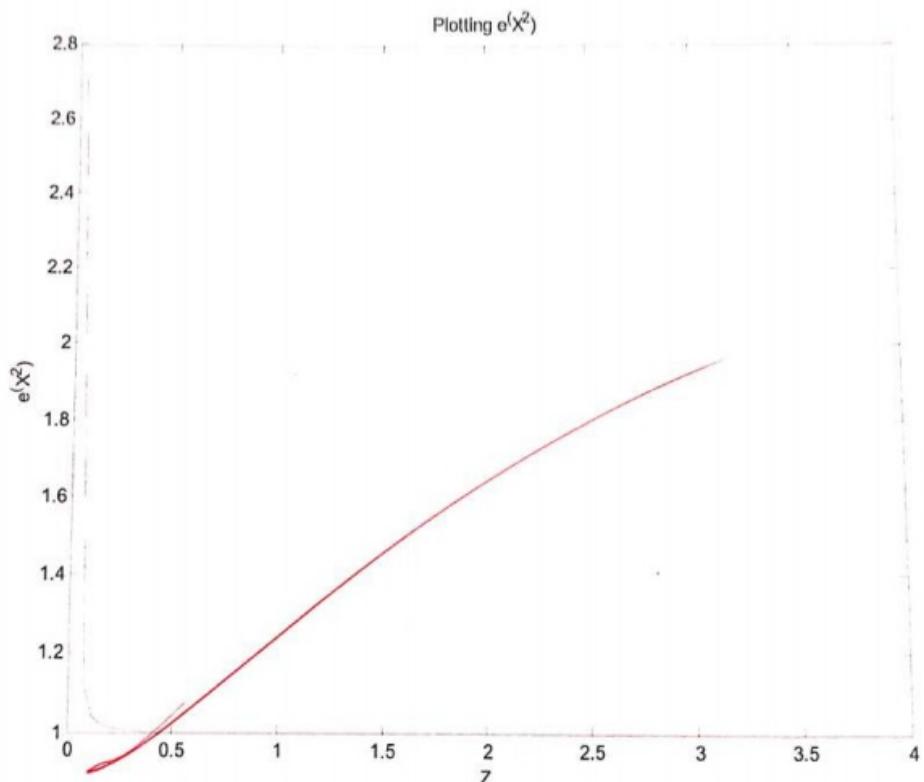
Columns 1 through 9	2.7183	1.1175	1.0408	1.0206	1.0124	1.0083	1.0059	1.0045	1.0035
Columns 10 through 18	1.0028	1.0023	1.0019	1.0016	1.0014	1.0012	1.0010	1.0009	1.0008
Columns 19 through 27	1.0007	1.0007	1.0006	1.0005	1.0005	1.0005	1.0004	1.0004	1.0004
Columns 28 through 36	1.0003	1.0003	1.0003	1.0003	1.0003	1.0002	1.0002	1.0002	1.0002
Columns 37 through 45	1.0002	1.0002	1.0002	1.0002	1.0002	1.0001	1.0001	1.0001	1.0001
Columns 46 through 54	1.0001	1.0001	1.0001	1.0001	1.0001	1.0001	1.0001	1.0001	1.0001



Scanned with OKEN Scanner

Columns 55 through 63  
1.0001 1.0001 1.0001 1.0001 1.0001 1.0001 1.0001 1.0001 1.0001  
Columns 64 through 72  
1.0001 1.0001 1.0001 1.0001 1.0001 1.0001 1.0001 1.0001 1.0000  
Columns 73 through 81  
1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000  
Columns 82 through 90  
1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000  
Columns 91 through 99  
1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000  
Column 100  
1.0000

```
>> plot(Z,A)
>> xlabel('Z')
>> ylabel('e^(X^2)')
>> title('Plotting e^(X^2)')
```



Scanned with OKEN Scanner

---

**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  $X$ .



Internal Assessment(Mandatory Experiment) Sheet for Lab Experiment Department of Mechanical & Automation Engg. Amity University , Noida (U.P)			
Programme	Btech MAE	Course Name	Basic Simulation
Course Code	ES204	Semester	4
Student Name	NIKHITA WADHAWAN	Enrollment No.	A2305413102
Marking Criteria			
Criteria	Total Marks	Marks Obtained	Comments
Concept	2		
Implementation	2		
Performance	2		
<b>Total</b>	<b>6</b>		



Scanned with OKEN Scanner

## EXPERIMENT - 6

### AIM:

Generating a Sinusoidal Signal of a given frequency (say, 100Hz) and Plotting with Graphical Enhancements: Tiling, 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

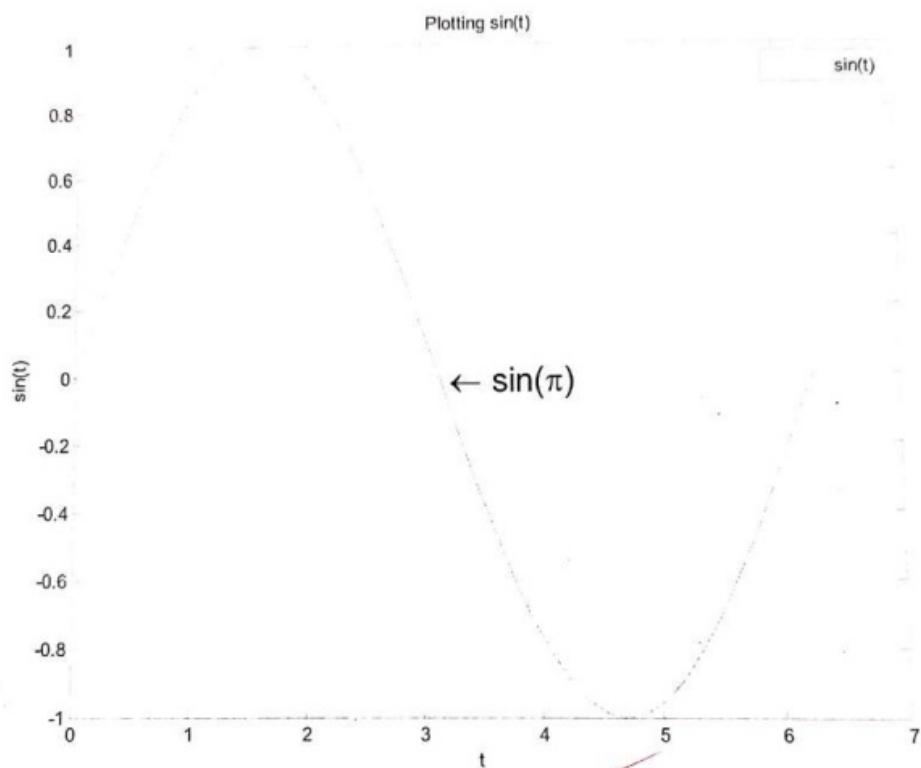
>> % Plotting with graphical enhancements



Scanned with OKEN Scanner

```
>> % 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,' \leftarrow sin(\pi)', '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
```



Scanned with OKEN Scanner

```

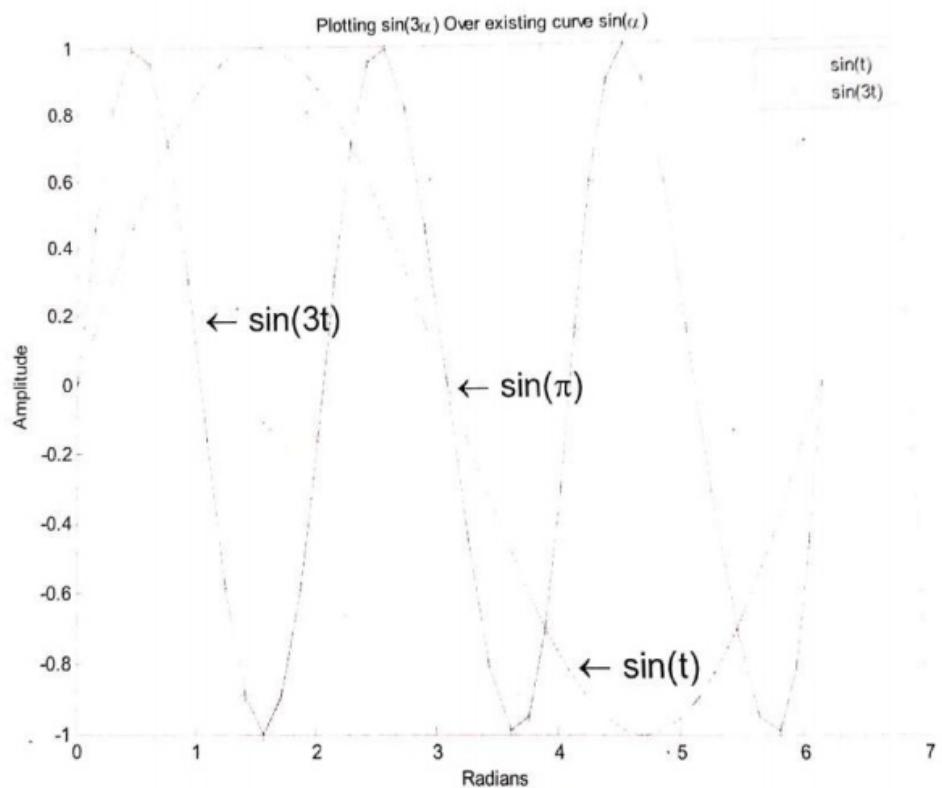
-0.8910 -1.0000 -0.8910 -0.5878 -0.1564 0.3090 0.7071 0.9511 0.9877
Columns 19 through 27
 0.8090 0.4540 0.0000 -0.4540 -0.8090 -0.9877 -0.9511 -0.7071 -0.3090
Columns 28 through 36
 0.1564 0.5878 0.8910 1.0000 0.8910 0.5878 0.1564 -0.3090 -0.7071
Columns 37 through 41
-0.9511 -0.9877 -0.8090 -0.4540 -0.0000

```

```

>> plot(t,Y,'B +-')
>> xlabel('Radians')
>> ylabel('Amplitude')
>> legend('sin(t)', 'sin(3t)')
>> title('Plotting sin(3t) Over existing curve sin(\alpha)')
>> text(4.1,-0.8, '\leftarrow sin(t)', 'FontSize', 18)
>> text(1,0.2, '\leftarrow 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 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(x,y,z,'string','PropertyName',PropertyValue.): adds the string in quotes to the location defined by the coordinates and uses the values for the specified text properties.

**Adding Legends:** 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: xlabel('Time \musec.') displays Time  $\mu$ sec 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.



Scanned with OKEN Scanner

**Internal Assessment(Mandatory Experiment) Sheet for Lab Experiment**  
Department of Mechanical & Automation Engg.  
Amity University, Noida (U.P.)

Programme	Btech MAE	Course Name	Basic Simulation
Course Code	ES204	Semester	4
Student Name	NUCKELIA WATKISSAN	Enrollment No.	A2305413102
Marking Criteria			
Criteria	Total Marks	Marks Obtained	Comments
Concept	2		
Implementation	2		
Performance	2		
Total	6		



Scanned with OKEN Scanner

## EXPERIMENT - 7

### 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
```



Scanned with OKEN Scanner

```

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
Columns 37 through 41
0.0059 -0.0046 0.0031 -0.0016 0.0000

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
Columns 28 through 36
-0.7910 -0.7801 -0.7905 -0.7804 -0.7905 -0.7801 -0.7910 -0.7792 -0.7925
Columns 37 through 41
-0.7769 -0.7964 -0.7692 -0.8171 -0.0000

>> 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
Columns 28 through 36
-3.0214 -2.9799 -3.0193 -2.9809 -3.0193 -2.9799 -3.0214 -2.9764 -3.0270
Columns 37 through 41
-2.9675 -3.0420 -2.9383 -3.1211 -0.0000
>> 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

```

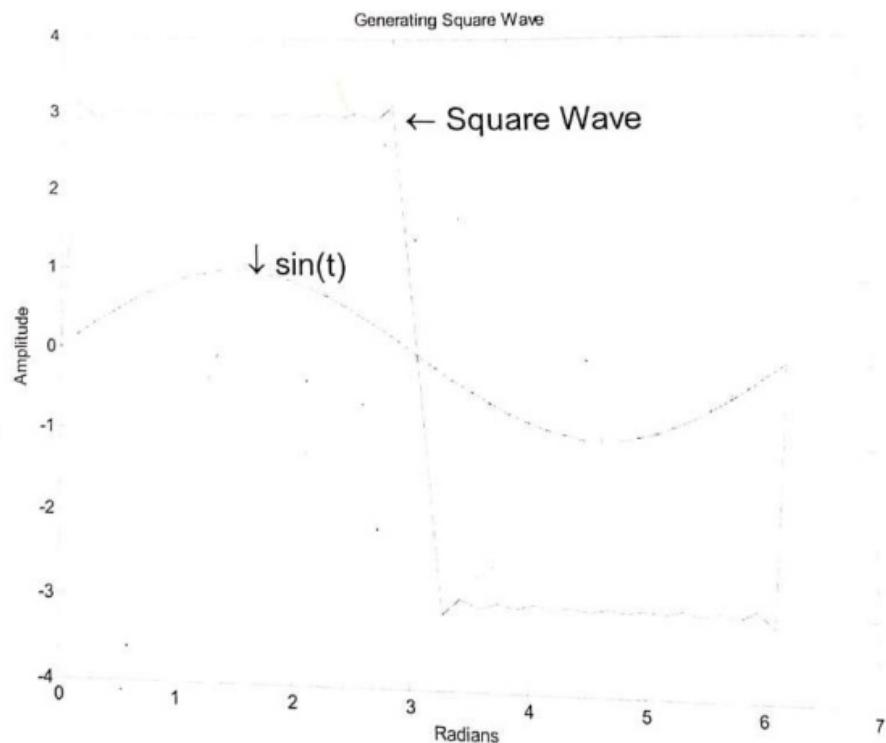


Scanned with OKEN Scanner

```

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,'↓sin(t)','FontSize',18)
>> text(3,3,'← Square Wave','FontSize',18)

```



## DISCUSSION:



Scanned with OKEN Scanner

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.

$$X(t) = 4 \cdot A / \pi [ \sin(w \cdot t) + (1/3) \cdot \sin(3 \cdot w \cdot t) + (1/5) \cdot \sin(5 \cdot w \cdot t) + \dots ]$$

#### **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.



Scanned with OKEN Scanner

Internal Assessment(Mandatory Experiment) Sheet for Lab Experiment  
Department of Mechanical & Automation Engg.  
Amity University, Noida (U.P.)

Programme	Btech - MAE	Course Name	Basic Simulation
Course Code	ES204	Semester	4
Student Name	KIRANITA KUMARI ANANDA	Enrollment No.	A2305413102-
Marking Criteria			
Criteria	Total Marks	Marks Obtained	Comments
Concept	2		
Implementation	2		
Performance	2		
Total	6		



Scanned with OKEN Scanner

## 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



Scanned with OKEN Scanner

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
1.6000	1.6100	1.6200	1.6300	1.6400	1.6500
1.6600	1.6700	1.6800	1.6900	1.7000	1.7100
1.7200	1.7300	1.7400	1.7500	1.7600	1.7700
1.7800	1.7900	1.8000	1.8100	1.8200	1.8300
1.8400	1.8500	1.8600	1.8700	1.8800	1.8900
1.9000	1.9100	1.9200	1.9300	1.9400	1.9500
1.9600	1.9700	1.9800	1.9900	2.0000	2.0100
2.0200	2.0300	2.0400	2.0500	2.0600	2.0700
2.0800	2.0900	2.1000	2.1100	2.1200	2.1300
2.1400	2.1500	2.1600	2.1700	2.1800	2.1900
2.2000	2.2100	2.2200	2.2300	2.2400	2.2500
2.2600	2.2700	2.2800	2.2900	2.3000	2.3100
2.3200	2.3300	2.3400	2.3500	2.3600	2.3700
2.3800	2.3900	2.4000	2.4100	2.4200	2.4300
2.4400	2.4500	2.4600	2.4700	2.4800	2.4900
2.5000	2.5100	2.5200	2.5300	2.5400	2.5500
2.5600	2.5700	2.5800	2.5900	2.6000	2.6100
2.6200	2.6300	2.6400	2.6500	2.6600	2.6700
2.6800	2.6900	2.7000	2.7100	2.7200	2.7300



Scanned with OKEN Scanner

2.7400	2.7500	2.7600	2.7700	2.7800	2.7900
2.8000	2.8100	2.8200	2.8300	2.8400	2.8500
2.8600	2.8700	2.8800	2.8900	2.9000	2.9100
2.9200	2.9300	2.9400	2.9500	2.9600	2.9700
2.9800	2.9900	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
2.6657	2.6454	2.6260	2.6074	2.5896	2.5726
2.5564	2.5409	2.5262	2.5122	2.4990	2.4864
2.4745	2.4633	2.4528	2.4429	2.4337	2.4251
2.4171	2.4097	2.4029	2.3968	2.3912	2.3862
2.3817	2.3778	2.3745	2.3717	2.3694	2.3677
2.3665	2.3658	2.3656	2.3659	2.3667	2.3680
2.3698	2.3720	2.3748	2.3780	2.3817	2.3858
2.3904	2.3955	2.4010	2.4069	2.4133	2.4201
2.4274	2.4351	2.4432	2.4517	2.4607	2.4700
2.4798	2.4900	2.5006	2.5117	2.5231	2.5349
2.5471	2.5598	2.5728	2.5862	2.6000	2.6142
2.6288	2.6437	2.6591	2.6748	2.6910	2.7075



Scanned with OKEN Scanner

2.7243	2.7416	2.7592	2.7772	2.7956	2.8144
2.8335	2.8530	2.8729	2.8931	2.9137	2.9347
2.9560	2.9778	2.9998	3.0223	3.0451	3.0682
3.0918	3.1157	3.1399	3.1646	3.1895	3.2149
3.2406	3.2667	3.2931	3.3199	3.3470	3.3745
3.4024	3.4307	3.4592	3.4882	3.5175	3.5472
3.5772	3.6076	3.6384	3.6695	3.7010	3.7328
3.7650	3.7976	3.8305	3.8638	3.8974	3.9314
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
5.7431	5.7936	5.8444			

>>% 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:

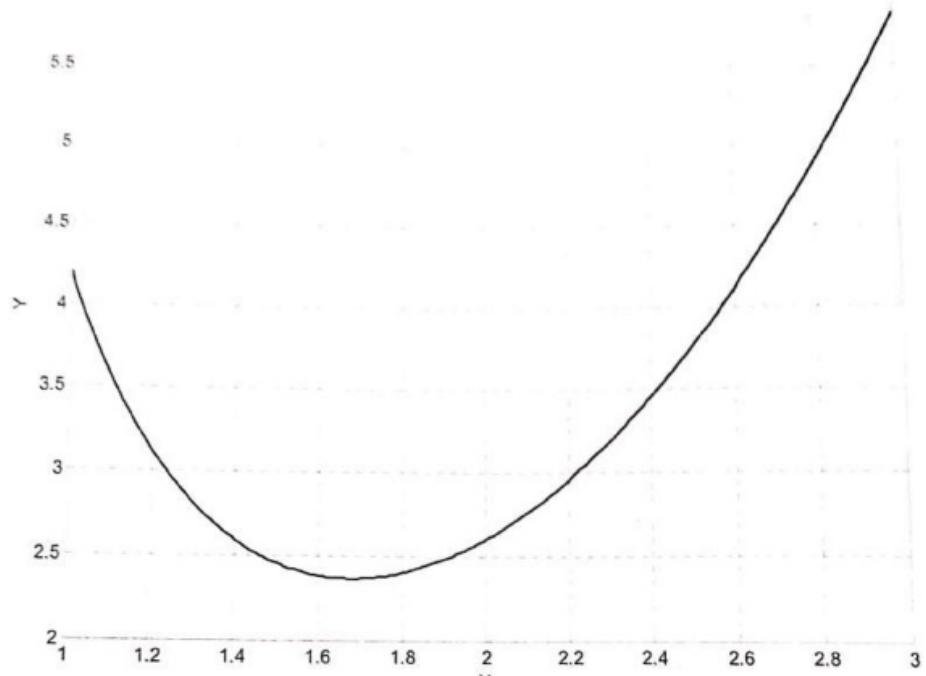
**Ode45:** ode45 is based on an explicit Runge-Kutta (4,5) formula, the Dormand-Prince pair. It is a one-step solver - in computing  $y(t_n)$ , it needs only the solution at the immediately preceding time point,  $y(t_{n-1})$ .

#### RESULT:



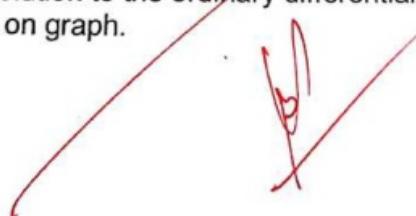
Scanned with OKEN Scanner

Solution to ODE,  $dy/dx = (x^3 - 2^*y)/x$



### CONCLUSION:

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



Scanned with OKEN Scanner

Internal Assessment (Mandatory Experiment) Sheet for Lab Experiment

Department of Mechanical & Automation Engg.

Sathyabama University, Chennai (U.P.)

Programme	B.Tech M.E.	Course Name	Basic Simulation
Course Code	ES2006	Student Name	A
Student Name	Shreyas T S	Enrollment No.	A2305613102
Marking Criteria			
Criteria	Total Marks	Marks Obtained	Comments
Concept	2		
Implementation	2		
Performance	2		
Total	6		



Scanned with OKEN Scanner

## 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

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

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.



Scanned with OKEN Scanner

**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.



Scanned with OKEN Scanner

**Internal Assessment(Mandatory Experiment) Sheet for Lab Experiment**  
**Department of Mechanical & Automation Engg.**  
**Amity University , Noida (U.P)**

Programme	Btech MAE	Course Name	Basic Simulation
Course Code	ES204	Semester	4
Student Name	Abhishek Singh	Enrollment No.	A230541302
Marking Criteria			
Criteria	Total Marks	Marks Obtained	Comments
Concept	2		
Implementation	2		
Performance	2		
Total	6		



Scanned with OKEN Scanner

## 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
```

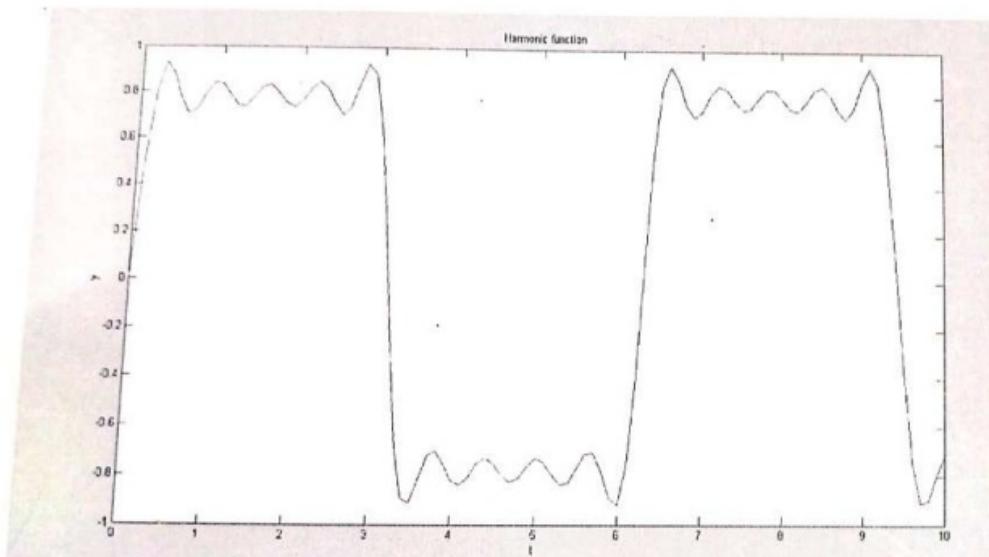
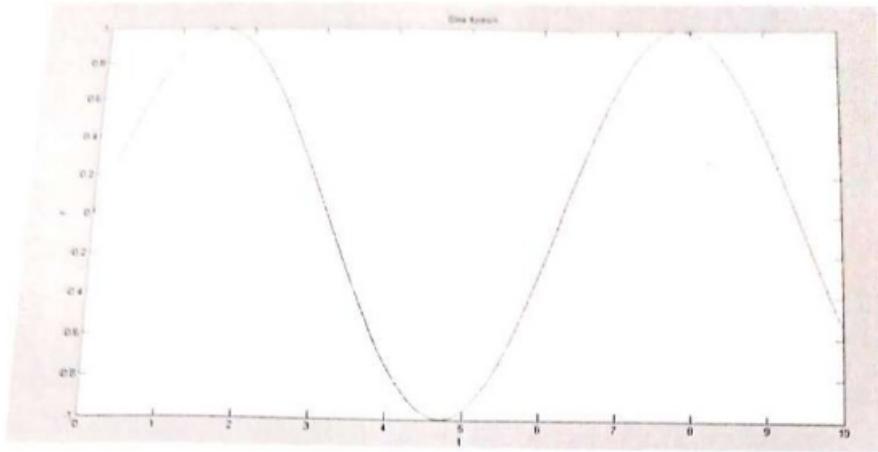
>> % Generating & Plotting Square wave using Fourier Series Expansion

```
>> sum=0  
  
sum =  
    0  
  
>> for n=1:2:100  
Y=(sin(n*t))/n  
sum=sum+Y  
end
```

### PLOTS:



Scanned with OKEN Scanner



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

Experiment no : 10



Scanned with OKEN Scanner

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

Software Used: Matlab R 7.0

Procedure:

```
>> 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)
```

```
>> plot(x,y)
```

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

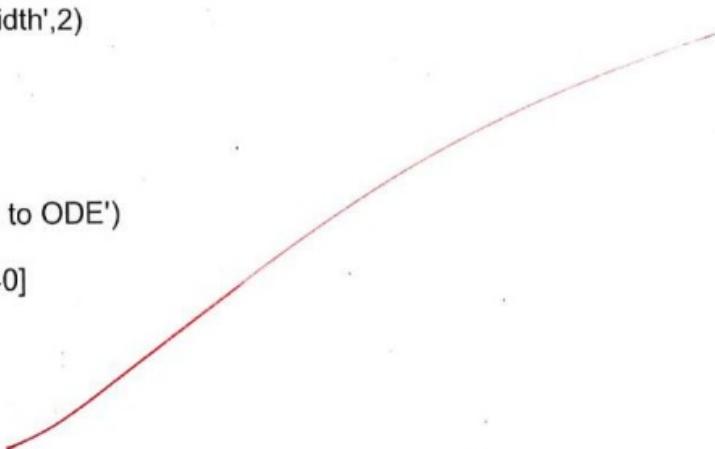
```
>> xlabel('x')
```

```
>> ylabel('y')
```

```
>> Title('Solution to ODE')
```

```
>> x=[10,15,25,40]
```

```
x =
```



```
10 15 25 40
```

```
>> bar(x)
```

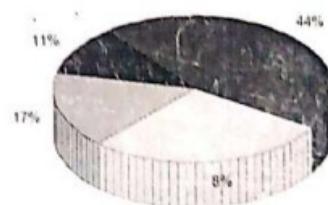
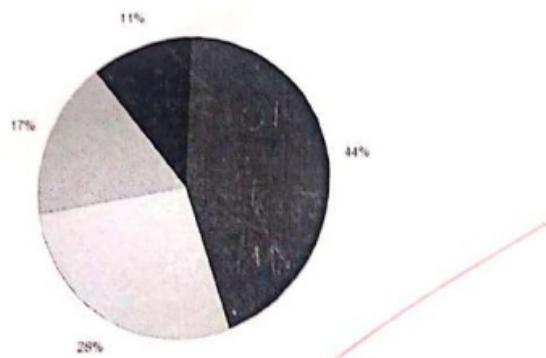
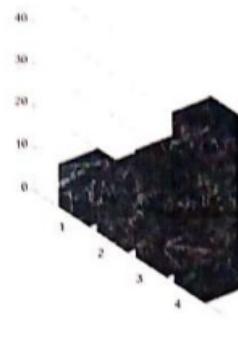
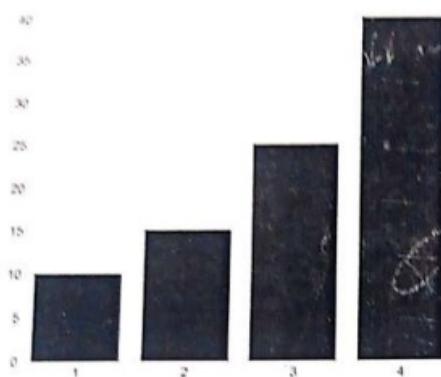
```
>> bar3(x)
```



Scanned with OKEN Scanner

```
>> pie(x)  
>> pie3(x)
```

Plots:



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



Scanned with OKEN Scanner

Internal Assessment(Mandatory Experiment) Sheet for Lab Experiment			
Department of Mechanical & Automation Engg.			
Amity University , Noida (U.P)			
Programme	Btech MAE	Course Name	Basic Simulation
Course Code	ES204	Semester	4
Student Name	NIKHITA WADHAWAN	Enrollment No.	A2305413102
Marking Criteria			
Criteria	Total Marks	Marks Obtained	Comments
Concept	2		
Implementation	2		
Performance	2		
Total	6		



Scanned with OKEN Scanner

Experiment no : 10(b)

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

Software Used: Matlab R 7.0

Procedure:

[ x=[0 0 0; 1 1 -1; 1 -1 -1]

x =

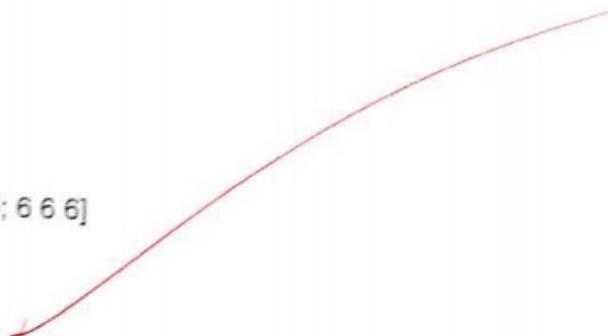
0 0 0

1 1 -1

1 -1 -1

>> y=[0 0 0; 5 5 5; 6 6 6]

y =



0 0 0

5 5 5

6 6 6



Scanned with OKEN Scanner

```
>> z=[0 0 0; 1 1 -1; 1 1 1]
```

*z =*

```
0 0 0  
1 1 -1  
1 1 1
```

```
>> v=rand(3)
```

*v =*

```
0.9501 0.4860 0.4565  
0.2311 0.8913 0.0185  
0.6068 0.7621 0.8214
```

```
>> fill(x,y,z)
```

```
>> figure(2)
```

```
>> fill3(x,y,z,v)
```

```
>> gridon
```

```
??? Undefined function or variable 'gridon'.
```

```
>> grid on
```

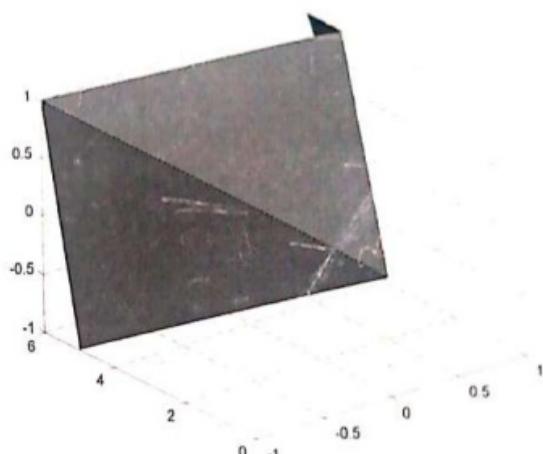
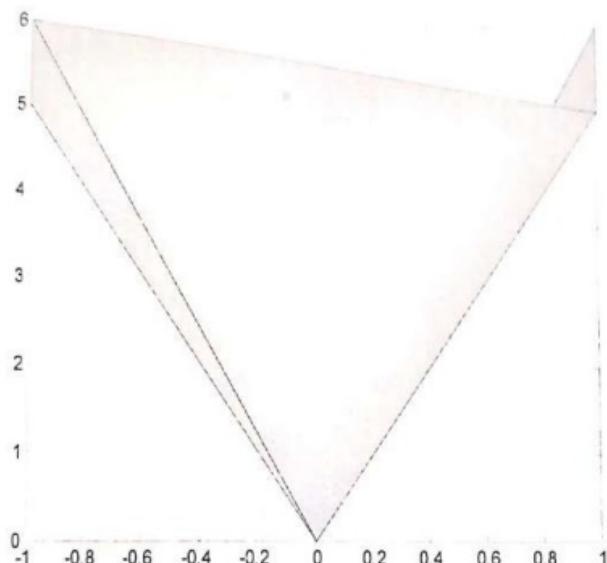
grid on



Scanned with OKEN Scanner

6244  
6239

PLOT:



**CONCLUSION:** Basic 2D and 3D plots have been plotted , bar and pie charts have also been plotted along with polygons



Scanned with OKEN Scanner



**CONCLUSION:** Basic 2D and 3D plots have been plotted , bar and pie charts have also been plotted along with polygons and contours.



Scanned with OKEN Scanner