



NEHRU COLLEGE OF ENGINEERING AND RESEARCH CENTRE
(NAAC Accredited)
(Approved by AICTE, Affiliated to APJ Abdul Kalam Technological University, Kerala)



DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

LAB MANUAL



ECL 201 SCIENTIFIC COMPUTING LAB

VISION OF THE INSTITUTION

To mould true citizens who are millennium leaders and catalysts of change through excellence in education.

MISSION OF THE INSTITUTION

NCERC is committed to transform itself into a center of excellence in Learning and Research in Engineering and Frontier Technology and to impart quality education to mould technically competent citizens with moral integrity, social commitment and ethical values.

We intend to facilitate our students to assimilate the latest technological know-how and to imbibe discipline, culture and spiritually, and to mould them in to technological giants, dedicated research scientists and intellectual leaders of the country who can spread the beams of light and happiness among the poor and the underprivileged.

ABOUT DEPARTMENT

- ◆ Established in: 2002
- ◆ Course offered : B.Tech in Electronics and Communication Engineering
M.Tech in VLSI
- ◆ Approved by AICTE New Delhi and Accredited by NAAC
- ◆ Affiliated to the University of Dr. A P J Abdul Kalam Technological University.

DEPARTMENT VISION

Providing Universal Communicative Electronics Engineers with corporate and social relevance towards sustainable developments through quality education.

DEPARTMENT MISSION

- 1) Imparting Quality education by providing excellent teaching, learning environment.
- 2) Transforming and adopting students in this knowledgeable era, where the electronic gadgets (things) are getting obsolete in short span.
- 3) To initiate multi-disciplinary activities to students at earliest and apply in their respective fields of interest later.
- 4) Promoting leading edge Research & Development through collaboration with academia & industry.

PROGRAMME EDUCATIONAL OBJECTIVES

PEO1. To prepare students to excel in postgraduate programmes or to succeed in industry / technical profession through global, rigorous education and prepare the students to practice and innovate recent fields in the specified program/ industry environment.

PEO2. To provide students with a solid foundation in mathematical, Scientific and engineering fundamentals required to solve engineering problems and to have strong practical knowledge required to design and test the system.

PEO3. To train students with good scientific and engineering breadth so as to comprehend, analyze, design, and create novel products and solutions for the real life problems.

PEO4. To provide student with an academic environment aware of excellence, effective communication skills, leadership, multidisciplinary approach, written ethical codes and the life-long learning needed for a successful professional career.

PROGRAM OUTCOMES (POS)

Engineering Graduates will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSO)

PSO1: Ability to Formulate and Simulate Innovative Ideas to provide software solutions for Real-time Problems and to investigate for its future scope.

PSO2: Ability to learn and apply various methodologies for facilitating development of high quality System Software Tools and Efficient Web Design Models with a focus on performance optimization.

PSO3: Ability to inculcate the Knowledge for developing Codes and integrating hardware/software products in the domains of Big Data Analytics, Web Applications and Mobile Apps to create innovative career path and for the socially relevant issues.

SYLLABUS

Experiment 1. Familiarization of the Computing Tool

1. Needs and requirements in scientific computing
2. Familiarization of a programming language like Python/R/ MATLAB/SCILAB/LabVIEW for scientific computing
3. Familiarization of data types in the language used.
4. Familiarization of the syntax of *while*, *for*, *if* statements.
5. Basic syntax and execution of small scripts.

Experiment 2. Familiarization of Scientific Computing

1. Functions with examples
2. Basic arithmetic functions such as *abs*, *sine*, *real*, *imag*, *complex*, *sinc* etc. using built in modules.
3. Vectorized computing without loops for fast scientific applications.

Experiment 3. Realization of Arrays and Matrices

1. Realize one dimensional array of real and complex numbers
2. stem and continuous plots of real arrays using *matplotlib/GUIs/charts*.
3. Realization of two dimensional arrays and matrices and their visualizations with *imshow/matshow/charts*
4. Inverse of a square matrix and the solution of the matrix equation

$$[\mathbf{A}][\mathbf{X}] = [\mathbf{b}]$$

where \mathbf{A} is an $N \times N$ matrix and \mathbf{X} and \mathbf{b} are $N \times 1$ vectors.

5. Computation of the rank(ρ) and eigen values (λ_i) of \mathbf{A}
6. Approximate \mathbf{A} for $N = 1000$ with the help of singular value decomposition of \mathbf{A} as

$$\tilde{\mathbf{A}} = \sum_{i=0}^r \lambda_i U_i V_i^T$$

where U_i and V_i are the singular vectors and λ_i are the eigen values with $\lambda_i < \lambda_j$ for $i > j$. One may use the built-in functions for singular value decomposition.

7. Plot the absolute error(ζ) between \mathbf{A} and $\tilde{\mathbf{A}}$ as $\zeta = \sum_{i=1}^N \sum_{j=1}^N |a_{i,j} - \tilde{a}_{i,j}|^2$ against r for $r = 10, 50, 75, 100, 250, 500, 750$ and appreciate the plot

Experiment 4. Numerical Differentiation and Integration

1. Realize the functions $\sin t$, $\cos t$, $\sinh t$ and $\cosh t$ for the vector $t = [0, 10]$ with increment 0.01
2. Compute the first and second derivatives of these functions using built in tools such as *grad*.
3. Plot the derivatives over the respective functions and appreciate.
4. Familiarize the numerical integration tools in the language you use.
5. Realize the function

$$f(t) = 4t^2 + 3$$

and plot it for the vector $t = [-5, 5]$ with increment 0.01

6. Use general integration tool to compute

$$\int_{-2}^2 f(t) dt$$

7. Repeat the above steps with trapezoidal and Simpson method and compare the results.
8. Compute

$$\frac{1}{\sqrt{2\pi}} \int_0^{\infty} e^{-\frac{x^2}{2}} dx$$

using the above three methods.

Experiment 5. Solution of Ordinary Differential Equations

1. Solve the first order differential equation

$$\frac{dx}{dt} + 2x = 0$$

with the initial condition $x(0) = 1$

2. Solve for the current transient through an RC network (with $RC = 3$) that is driven by

- 5 V DC
- the signal $5e^{-t}U(t)$

and plot the solutions.

3. Solve the second order differential equation

$$\frac{d^2x}{dt^2} + 2\frac{dx}{dt} + 2x = e^{-t}$$

4. Solve the current transient through a series RLC circuit with $R = 1\Omega$, $L = 1mH$ and $C = 1\mu F$ that is driven by

- 5 V DC
- the signal $5e^{-t}U(t)$

Experiment 6. Simple Data Visualization

1. Draw stem plots, line plots, box plots, bar plots and scatter plots with random data.
2. plot the histogram of a random data.
3. create legends in plots.
4. Realize a vector $t = [-10, 10]$ with increment 0.01 as an array.
5. Implement and plot the functions
 - $f(t) = \cos t$
 - $f(t) = \cos t \cos 5t + \cos 5t$

Experiment 7. Simple Data Analysis with Spreadsheets

1. Display an electrical signal on DSO and export it as a *.csv* file.
2. Read this *.csv* or *.xls* file as an array and plot it.
3. Compute the mean and standard deviation of the signal. Plot its histogram with an appropriate bin size.

Experiment 8. Convergence of Fourier Series

1. The experiment aims to understand the lack of convergence of Fourier series

2. Realize the Fourier series

$$f(t) = \frac{4}{\pi} \left[1 - \frac{1}{3} \cos \frac{2\pi 3t}{T} + \frac{1}{5} \cos \frac{2\pi 5t}{T} - \frac{1}{7} \cos \frac{2\pi 7t}{T} + \dots \right]$$

3. Realize the vector $t = [0, 100]$ with an increment of 0.01 and keep $T = 20$.

4. Plot the first 3 or 4 terms on the same graphic window and understand how the smooth sinusoids add up to a discontinuous square function.

5. Compute and plot the series for the first 10, 20, 50 and 100 terms of the and understand the lack of convergence at the points of discontinuity.

6. With t made a zero vector, $f(0) = 1$, resulting in the *Madhava* series for π as

$$\pi = 4 \left[1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots \right]$$

7. Use this to compute π for the first 10, 20, 50 and 100 terms.

Experiment 9: Coin Toss and the Level Crossing Problem

1. Simulate a coin toss that maps a head as 1 and tail as 0.

2. Toss the coin $N = 100, 500, 1000, 5000$ and 500000 times and compute the probability (p) of head in each case.

3. Compute the absolute error $|0.5 - p|$ in each case and plot against N and understand the law of large numbers.

4. Create a uniform random vector with maximum magnitude 10, plot and observe.

5. Set a threshold ($V_T = 2$) and count how many times the random function has crossed V_T .

6. Count how many times the function has gone above and below the threshold.

COURSE OUTCOMES

The student will be able to

CO 1	Describe the needs and requirements of scientific computing and to familiarize one programming language for scientific computing and data visualization.
CO 2	Approximate an array/matrix with matrix decomposition.
CO 3	Implement numerical integration and differentiation.
CO 4	Solve ordinary differential equations for engineering applications
CO 5	Realize how periodic functions are constituted by sinusoids
CO 6	Realize how periodic functions are constituted by sinusoids
CO 7	Simulate random processes and understand their statistics.

MAPPING OF COURSE OUTCOMES WITH PROGRAM OUTCOMES

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	3	3	3	2	3	0	0	0	3	1	0	3
CO2	3	3	1	2	3	0	0	0	3	0	0	1
CO3	3	3	1	1	3	0	0	0	0	0	0	1
CO4	3	3	1	1	3	0	0	0	0	0	0	1
CO5	3	3	1	3	0	0	0	0	3	3	0	0
CO6	3	3	2	2	3	0	0	0	3	1	0	0
CO7	3	3	2	2	3	0	0	0	3	1	0	1

LIST OF EXPERIMENTS

EXPERIMENT NUMBER	NAME OF EXPERIMENT
1.	Familiarization of Scientific computing
2.	Familiarization of Computing tool
3.	Realization of Arrays and Matrices
4.	Numerical Differentiation and Integration
5.	Solution of Ordinary Differential Equations
6.	Simple Data Visualization
7.	Simple Data Analysis with Spreadsheets
8.	Convergence of Fourier Series
9.	Coin Toss and the Level Crossing Problem

INDEX

EXPT. NO	NAME OF EXPERIMENT	PAGE NO.
1.	Familiarization of Scientific computing	01
2.	Familiarization of Computing tool	13
3.	Realization of Arrays and Matrices	25
4.	Numerical Differentiation and Integration	38
5.	Solution of Ordinary Differential Equations	50
6.	Simple Data Visualization	60
7.	Simple Data Analysis with Spreadsheets	60
8.	Convergence of Fourier Series	67
9.	Coin Toss and the Level Crossing Problem	69

EXP NO: 1

DATE:

FAMILIARIZATION OF THE COMPUTING TOOL

OBJECTIVE

- To familiarize with the scientific computing tool

LEARNING OUTCOMES

- After the completion of this experiment students will be able to compile and run a MATLAB[®] code.
- Execute small scripts using if, for and while statements

SOFTWARE USED:

MATLAB[®] R2013

THEORY

MATLAB is a programming language developed by MathWorks. It started out as a matrix programming language where linear algebra programming was simple. It can be run both under interactive sessions and as a batch job. This experiment gives you a gentle introduction of MATLAB programming language. It is designed to give students fluency in MATLAB programming language. MATLAB (matrix laboratory) is a fourth-generation high-level programming language and interactive environment for numerical computation, visualization and programming.

It allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, interfacing with programs written in other languages, including C, C++, Java, and FORTRAN, analyze data, develop algorithms and create models and applications.

It has numerous built-in commands and math functions that help you in mathematical calculations, generating plots, and performing numerical methods.

MATLAB's Power of Computational Mathematics

MATLAB is used in every facet of computational mathematics. Following are some commonly used mathematical calculations where it is used most commonly.

- Dealing with Matrices and Arrays
- 2-D and 3-D Plotting and graphics
- Linear Algebra
- Algebraic Equations
- Non-linear Functions
- Statistics
- Data Analysis
- Calculus and Differential Equations
- Numerical Calculations
- Integration
- Transforms
- Curve Fitting
- Various other special functions

Features of MATLAB

Following are the basic features of MATLAB

- It is a high-level language for numerical computation, visualization and application development.
- It also provides an interactive environment for iterative exploration, design and problem solving.
- It provides vast library of mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, numerical integration and solving ordinary differential equations.
- It provides built-in graphics for visualizing data and tools for creating custom plots.
- MATLAB's programming interface gives development tools for improving code quality maintainability and maximizing performance.
- It provides tools for building applications with custom graphical interfaces.

- It provides functions for integrating MATLAB based algorithms with external applications and languages such as C, Java, .NET and Microsoft Excel.

Uses of MATLAB

MATLAB is widely used as a computational tool in science and engineering encompassing the fields of physics, chemistry, math and all engineering streams. It is used in a range of applications including

- Signal Processing and Communications
- Image and Video Processing
- Control Systems
- Test and Measurement
- Computational Finance
- Computational Biology

LAB EXERCISE

(a) Needs and requirements in scientific computing

Scientific Computing is the collection of tools, techniques, and theories required to solve on a computer mathematical models of problems in Science and Engineering.

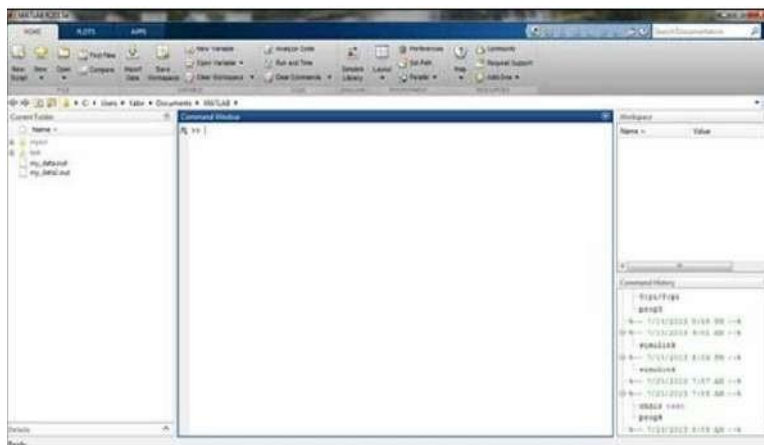
A majority of these tools, techniques, and theories originally developed in Mathematics, many of them having their genesis long before the advent of electronic computers. This set of mathematical theories and techniques is called Numerical Analysis (or Numerical Mathematics) and constitutes a major part of scientific computing.

The development of the electronic computer, however, signaled a new era in the approach to the solution of scientific problems. Many of the numerical methods that had been developed for the purpose of hand calculation (including the use of desk calculators for the actual arithmetic) had to be revised and sometimes abandoned. Many of these considerations – programming languages, operating systems, management of large quantities of data, correctness of programs – were subsumed under the new discipline of Computer Science, on which scientific computing now depends heavily. But mathematics itself continues to play a major role in scientific computing: it provides the language of the mathematical models that are to be solved and information about the suitability of a model. In summary, then, scientific computing draws on mathematics and computer

science to develop the best way to use computer systems to solve problems from science and engineering.

(b) Familiarization of MATLAB

MATLAB development IDE can be launched from the icon created on the desktop. The main working window in MATLAB is called the desktop. When MATLAB is started, the desktop appears in its default layout.

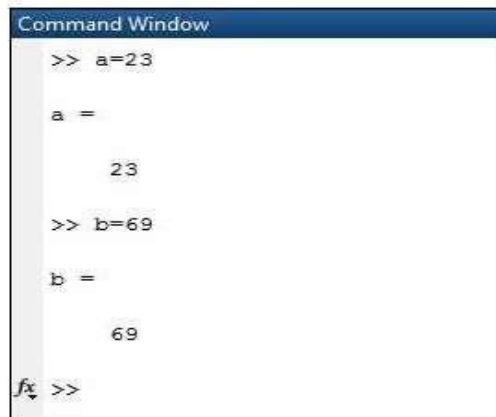


The desktop has the following panels –

- **Current Folder** – This panel allows you to access the project folders and files.

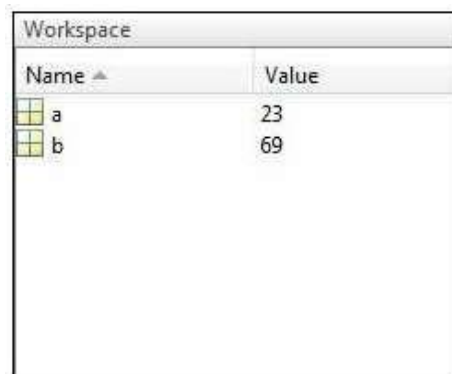


Command Window – This is the main area where commands can be entered at the command line. It is indicated by the command prompt (>>).



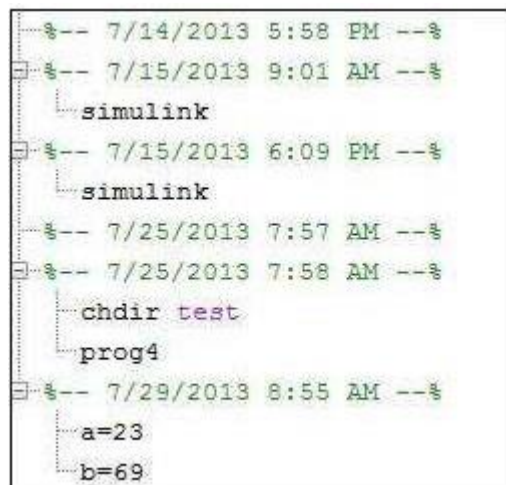
```
Command Window
>> a=23
a =
    23
>> b=69
b =
    69
fx >>
```

Workspace – The workspace shows all the variables created and/or imported from files.



Name ^	Value
a	23
b	69

Command History – This panel shows or return commands that are entered at the command line.



```
7/14/2013 5:58 PM --%
7/15/2013 9:01 AM --%
  simulink
7/15/2013 6:09 PM --%
  simulink
7/25/2013 7:57 AM --%
7/25/2013 7:58 AM --%
  chdir test
  prog4
7/29/2013 8:55 AM --%
  a=23
  b=69
```

I.Type the following in the command window and write down the results

```
(1) x=3
(2) x=7*8;
y=x*1.28
(3)clear all
(4) initial_velocity = 0;
acceleration = 9.8;
time = 20;
```

```
final_velocity = initial_velocity + acceleration * time
```

```
(5) x = sqrt(16)
(6) clc
(7) help sqrt
(8) disp a
(9) a=2;
disp(a)
```

(c) Familiarization of data types in the language used

MATLAB does not require any type declaration or dimension statements. Whenever MATLAB encounters a new variable name, it creates the variable and allocates appropriate memory space.

Data Types Available in MATLAB

MATLAB provides 15 fundamental data types. Every data type stores data that is in the form of a matrix or array. The size of this matrix or array is a minimum of 0-by-0 and this can grow up to a matrix or array of any size.

Sr.No.	Data Type & Description
1	int8 8-bit signed integer
2	uint8 8-bit unsigned integer

3	int16 16-bit signed integer
4	uint16 16-bit unsigned integer
5	int32 32-bit signed integer
6	uint32 32-bit unsigned integer
7	int64 64-bit signed integer
8	uint64 64-bit unsigned integer
9	single single precision numerical data
10	double double precision numerical data
11	logical logical values of 1 or 0, represent true and false respectively
12	char character data (strings are stored as vector of characters)

13	cell array array of indexed cells, each capable of storing an array of a different dimension and data type
14	structure C-like structures, each structure having named fields capable of storing an array of a different dimension and data type
15	function handle pointer to a function
16	user classes objects constructed from a user-defined class
17	java classes objects constructed from a Java class

(d) Familiarization of the syntax of while, for, if statements.**II (1)** Execute a script (m file) to check whether a number is divisible by 3**PROCEDURE**

1. Open MATLAB[®]
2. Open new M-file
3. Type the program
4. Save in current directory
5. Compile and Run the program
6. For the output see command window\ Figure window

PROGRAM

```
clc;  
clear all;  
close all;  
a=input('enter value of a');  
if mod(a,3)==0  
disp('divisible by 3')  
else
```

```
disp ('not divisible by 3')
end
```

INPUT

enter value of a 12

OUTPUT

divisible by 3

(2) Execute a script (m file) to check whether a number is prime or not using while loop

PROGRAM

```
clc;
clear all;
close all;
n=input('enter number');
i=2;
while i<=n/2
    if mod(n,i)==0
        flag=1;
    break;
end
i=i+1;
end
if n==1
    disp('neither prime nor composite');
else
    if flag==1
        disp('not prime');
    else
        disp('prime');
    end
end
```

(3) Execute a script (m file) to check whether a number is prime or not using for loop

PROGRAM

```
clc;
```

```
clear all;
close all;
n=input('enter number');
for i=2:n/2
    if mod(n,i)==0
        flag=1;
    break;
end
end

if n==1
    disp('neither prime nor composite')
else
    if flag==1
        disp('not prime')
    else
        disp('prime')
    end
end
end
```

INPUT

enter number 12

OUTPUT

not prime

(e) Basic syntax and execution of small scripts.

III1. Execute a script (m file) to create an array whose starting value is 100 and ends at 50 with a decrement of 7

PROGRAM

```
clc;
clear all;
```

```
close all;  
a=[100:-7:50];  
disp(a)
```

OUTPUT

```
[100 93 86 79 72 65 58 51]
```

2. Execute a script (m file) to add two matrices

PROGRAM

```
clc;  
clear all;  
close all;  
a=[1 3;4 5;7 9];  
b=[3 4;4 3;5 6];  
c=a+b;  
disp (c)
```

OUTPUT

```
4 7  
8 8  
12 15
```

3. Execute a script (m file) to find transpose of a matrix

PROGRAM

```
clc;  
clear all;  
close all;  
a=[1 3;4 5;7 9];  
c=a';  
disp (c)
```

OUTPUT

```
1  4  7
   3  5  9
```

4. Execute a script (m file) to display the second row of a 3x3 matrix as the output

PROGRAM

```
clc;
clear all;
close all;
a=[1 3 3;4 5 6;7 8 9];
a(2,:)
```

OUTPUT

```
[4 5 6]
```

5. Execute a script (m file) to display the second row third column element of a 3x3 matrix as the output

PROGRAM

```
clc;
clear all;
close all;
a=[1 3 3;4 5 6;7 8 9];
a(2,3)
```

OUTPUT

```
ans =6
```

6. Execute a script (m file) to do element by element multiplication of 2 matrices

```
clc;
clear all;
close all;
a=[1 3 3;4 5 6;7 8 9];
b=[1 5 2;2 2 3; 1 2 3];
a.*b
```

OUTPUT

1 15 6

8 10 18

7 16 27

INFERENCE:

Familiarized with MATLAB which is basically a scientific computing tool based on simulations

EXP NO: 2

DATE:

FAMILIARIZATION OF SCIENTIFIC COMPUTING

OBJECTIVE

- To familiarize different functions for scientific computing with examples

LEARNING OUTCOMES

- After the completion of this experiment students will be able to execute small scripts using different arithmetic functions

SOFTWARE USED:

MATLAB[®] R2013

THEORY

The different arithmetic functions of MATLAB are listed below.

$Y = \text{abs}(X)$ returns the absolute value of each element in array X. If X is complex, $\text{abs}(X)$ returns the complex magnitude.

abs - Absolute value and complex magnitude

angle- Phase angle

complex- Create complex array

isreal- whether array is real

real- Real part of complex number

sinc – returns an array, y, whose elements are the sinc of the elements of the input, x.

imag - Imaginary Part of Complex Number

sin – Sine of argument in radians

cos - cos of argument in radians

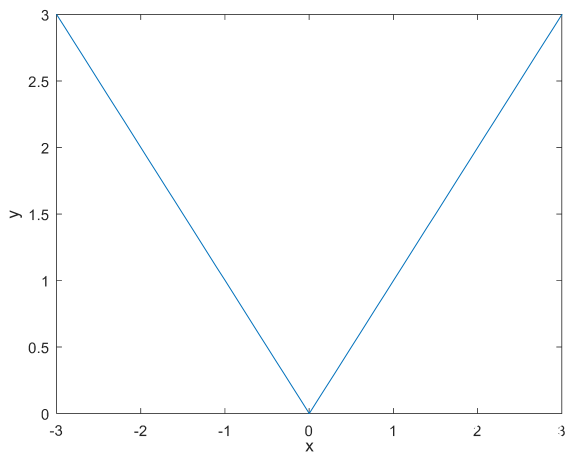
1. Type the following in the command window, write down the results and the functions of these built in modules

(a) $2.15e-3$

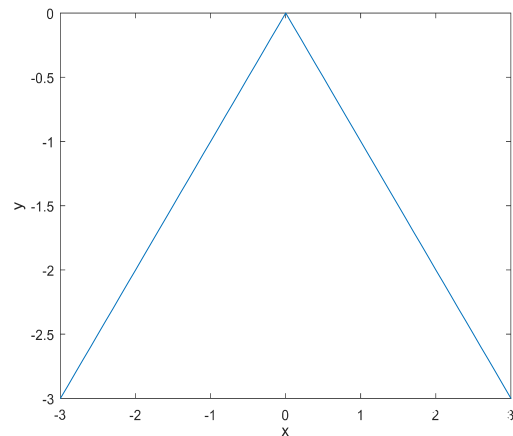
- (b) `mod(5,3)`
- (c) `floor(3.5)`
- (d) `ceil(3.6)`
- (e) `round(3.4)`
- (f) Create a one dimensional array of ones having size 5
- (g) Create a one dimensional array of zeros having size 5
- (h) Create a 3x2 matrix 'K' having only zeros

2. Generate the following output using the function 'abs'

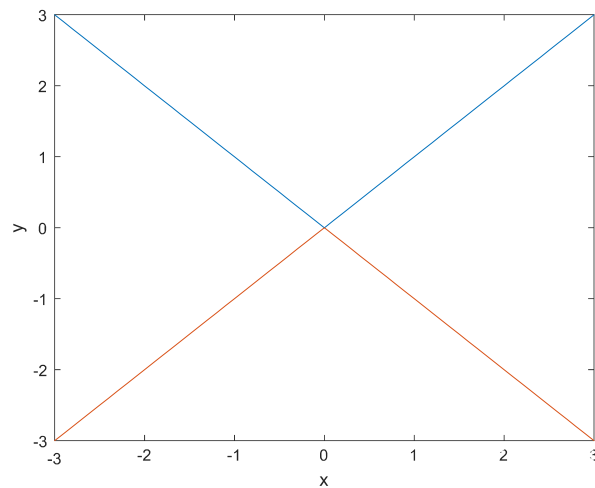
(a)



(b)



(c)



PROGRAM

(a)

```
x=[-3:0.1:3];
```

```
y=abs(x);
```

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

```
xlabel('x');
```

```
ylabel('y');
```

(b)

```
x=[-3:0.1:3];
```

```
y=-abs(x);
```

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

```
xlabel('x');
```

```
ylabel('y');
```

(c)

```
x=[-3:0.1:3];
```

```
y=abs(x);
```

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

```
hold on;
```

```
y=-abs(x);
```

```
plot(x,y);
```

```
hold off;
```

```
xlabel('x');
```

```
ylabel('y');
```

3. Create a complex number $5+4i$, extract the real and imaginary parts and compute the magnitude of the vector using built in functions

PROGRAM

```
z=complex(5,4);
```

```
disp('Real Part');
```

```
a=real(z)
```

```
disp('Imaginary Part');
```

```
b=imag(z)
```

```
disp('Magnitude');  
abs(z)
```

OUTPUT

Real Part

a = 5

Imaginary Part

b = 4

Magnitude

ans =6.4031

4. Represent the complex exponential form $e^{\pi*i}$ as rectangular form and find the real part, magnitude and the angle of the vector (both in degree and radians).

PROGRAM

```
z=exp(pi*i);  
disp('Complex number in rectangular form');  
z  
disp('Real Part');  
a=real(z)  
disp('Magnitude');  
mag=abs(z)  
disp('Angle in radians');  
ang=angle(z)  
disp('Angle in degree');  
ang_deg=ang*(180/pi)
```

OUTPUT

Complex number in rectangular form

z =-1.0000 + 0.0000i

Real Part

a = -1

Magnitude

mag =1

Angle in radians

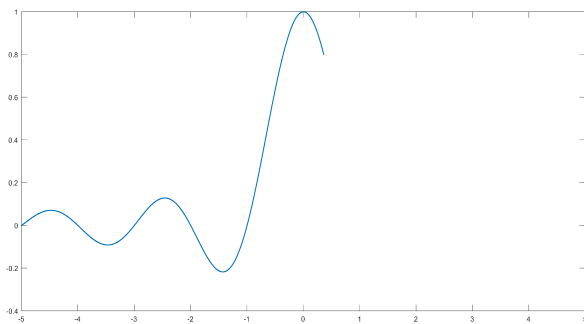
ang =3.1416

Angle in degree

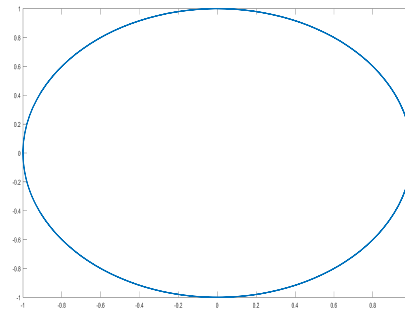
ang_deg =180

5. Plot the following

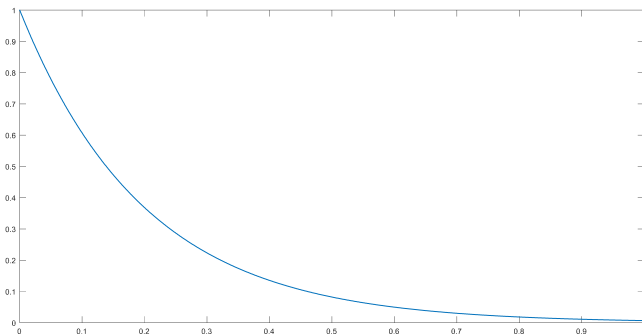
(a)



(b)



(c)



PROGRAM

(a)

```
x=[-5:0.001:5];  
y=sinc(x);  
plot(x,y)
```

(b)

```
t=-9:0.01:9;  
y=sin(t);  
x=cos(t);  
plot(x,y)
```

(c)

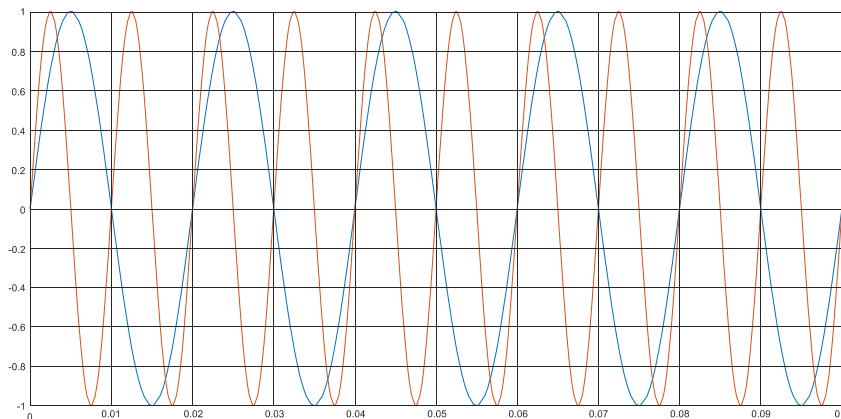
```
t=0:0.001:1  
y=exp(-5*t);  
plot(t,y);
```

5. Plot sinewaves of frequency 50 and 100 Hz on the same figure window.

PROGRAM

```
f=50;  
t=0:0.0002:0.1;  
y=sin(2*pi*f*t);  
plot(t,y);  
grid on;  
hold on;  
f=100;  
t=0:0.0002:0.1;  
y=sin(2*pi*f*t);  
plot(t,y);  
hold off;
```

OUTPUT



6. Replace the given code with vectorized code for fast computing

(a)

This code computes the sine of 1,001 values ranging from 0 to 10:

```
clc;
clear all;
close all;
i = 0;
for t = 0:.01:10
    i = i + 1;
    y(i) = sin(t);
end
disp(y);
```

(b)

```
clc;
clear all;
close all;
r=[1:4];
h=[1:4];
for i=1:4
```



```
volume(i) = pi*r(i)*r(i)*h(i);  
end  
disp(volume);
```

PROGRAM

(a)

```
t = 0:.01:10;  
y = sin(t);  
disp(y);
```

(b)

```
clc;  
clear all;  
close all;  
r=[1:4];  
h=[1:4];  
volume = pi*r.^2.*h;  
disp(volume);
```

OUTPUT

```
[3.1416 25.1327 84.8230 201.0619]
```

7. Execute a script (.m file) to obtain the dot product and the cross product of two vectors a and b, where a = (1 5 6) and b = (2 3 8).

PROGRAM

```
a = [1 5 6];  
b = [2 3 8];  
d=dot(a,b)  
c=cross(a,b)
```

OUTPUT

```
d =65  
c = 22 4 -7
```

8. Simplify the expression and express the complex number in rectangular and polar form

(a) $y = 0.5 + j6 + 3.5e^{j0.6} + (3 + j6)e^{j0.3\pi}$

$$z = \frac{(3 + j4)(5 + j2)(2 \angle 60^\circ)}{(3 + j6)(1 + j2)}$$

(b)

(a) PROGRAM

```
clc;
clear all;
close all;
Z1 = 0.5;
Z2 = 6*j;
Z3 = 3.5*exp(j*0.6);
Z4 = 3+6*j;
Z5 = exp(j*0.3*pi);
disp('Z in rectangular form is');
Z_rect = Z1+Z2+Z3+(Z4*Z5);
Z_rect
Z_mag = abs (Z_rect); % magnitude of Z
Z_angle = angle(Z_rect)*(180/pi); % Angle in degrees
disp('complex number Z in polar form, mag, phase');
Z_polar = [Z_mag, Z_angle]
```

OUTPUT

```
Z in rectangular form is
Z_rect =0.2979 +13.9300i
complex number Z in polar form, mag, phase
Z_polar =13.9332 88.7748
```

(b)PROGRAM

```
clc;
clear all;
```

```
close all;
Z1 = 3+4*j;
Z2 = 5+2*j;
theta = 60*(pi/180); % angle in radians
Z3 = 2*exp(j*theta);
Z4 = 3+6*j;
Z5 = 1+2*j;
disp('Z in rectangular form is');
Z_rect = Z1*Z2*Z3/(Z4*Z5);
Z_rect
Z_mag = abs (Z_rect); % magnitude of Z
Z_angle = angle(Z_rect)*(180/pi); % Angle in degrees
disp('complex number Z in polar form, mag, phase');
Z_polar = [Z_mag, Z_angle]
```

OUTPUT

```
Z in rectangular form is
Z_rect =3.5546 + 0.5035i
complex number Z in polar form, mag, phase
Z_polar = 3.5901  8.0616
```

9. The voltage across a capacitor is

$$v(t) = 10(1 - e^{-0.2t})$$

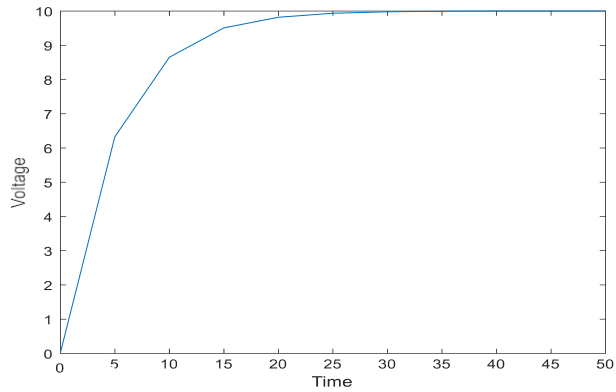
Plot voltage v (t), versus time, t, for t = 0 to 50 seconds with increment of 5 s. Do not use loops.

PROGRAM

```
clc;
clear all;
close all;
t=0:5:50
v=10*(1-exp(-0.2*t));
plot(t,v)
xlabel('Time')
```

```
ylabel('Voltage')
```

OUTPUT



INFERENCE:

Familiarized with basic arithmetic functions for scientific computing and used vectorized computing for fast scientific applications.

EXP NO: 3

DATE:

REALIZATION OF ARRAYS AND MATRICES

OBJECTIVE

- To familiarize with the realization of arrays and matrixes and their visualization using plotting functions and GUI

LEARNING OUTCOMES

- After the completion of this experiment students will be able to approximate an array/matrix with matrix decomposition.

SOFTWARE USED:

MATLAB® R2013

1. The voltage, v , across a resistance is given as (Ohm's Law), $v=iR$, where i is the current and R the resistance. The power dissipated in resistor R is given by the expression

$$P = i^2R.$$

If $R = 10$ Ohms and the current is increased from 0 to 10 A with increments of 2A, write a MATLAB program to generate a table of current, voltage and power dissipation.

PROGRAM

```
clc;
clear all;
close all;
R=10; % Resistance value
i=[0:2:10]; % Generate current values
v=i.*R; % array multiplication to obtain voltage
p=(i.^2)*R; % power calculation
sol=[i;v;p] % current, voltage and power values are printed
```

OUTPUT

sol =

0	2	4	6	8	10
0	20	40	60	80	100
0	40	160	360	640	1000

2. Find the roots of the following quadratic equation

$$x^2 - 2x + 3 = 0$$

PROGRAM

```
clc;
clear all;
close all;
a=input('enter a');
b=input('enter b');
c=input('enter c');
i = b^2 - 4*a*c;
if i > 0
    srint = sqrt(i);
    x1 = (-b + srint)/(2*a);
    x2 = (-b - srint)/(2*a);
elseif i == 0
    x1 = -b/(2*a);
    x2 = x1;
elseif i < 0
    srint = sqrt(-i);
    p1 = -b/(2*a);
    p2 = srint/(2*a);
    x1 = p1 + p2*j;
    x2 = p1 - p2*j;
end
rt = [x1;x2];
```

OUTPUT

```
rt =  
    1.0000 + 1.4142i  
    1.0000 - 1.4142i
```

3. Create two separate row vectors(arrays) a and b that contains elements from 1 to 10. Create an array of complex numbers z with a as the real part and b as the imaginary part. Find the sum and complex conjugate of the array z

PROGRAM

```
clc;  
clear all;  
close all;  
a = [1:10];  
b = [1:10];  
z = complex(a,b);  
A = sum(z);  
complex_conjugate = conj(z);
```

OUTPUT

```
A =55.0000 +55.0000i  
complex_conjugate =  
Columns 1 through 4  
1.0000 - 1.0000i 2.0000 - 2.0000i 3.0000 - 3.0000i 4.0000 - 4.0000i  
Columns 5 through 8  
5.0000 - 5.0000i 6.0000 - 6.0000i 7.0000 - 7.0000i 8.0000 - 8.0000i  
Columns 9 through 10  
9.0000 - 9.0000i 10.0000 -10.0000i
```

4. If w is a complex matrix given as

$$w = \begin{bmatrix} 1 + j1 & 2 - j2 \\ 3 + j2 & 4 + j3 \end{bmatrix}$$

Find the sum of all the elements of the matrix, conjugate transpose and un-conjugate transpose of the matrix

PROGRAM

```
clc;
clear all;
close all;
c=[1+j 2-2*j; 3+2*j 4+3*j]
s=sum(c);
S1=sum(s);
trans = c.';
conjugate_trans=c';
```

OUTPUT

```
S1 =10.0000 + 4.0000i
trans =
    1.0000 + 1.0000i    3.0000 + 2.0000i
    2.0000 - 2.0000i    4.0000 + 3.0000i
conjugate_trans =
    1.0000 - 1.0000i    3.0000 - 2.0000i
    2.0000 + 2.0000i    4.0000 - 3.0000i
```

5. Create an image that consists of alternate rows of black and white pixels without using the inbuilt function 'image'.

PROGRAM

```
clc;
clear all;
close all;
row = 126;
col = 126;
img = zeros(row, col);
i=[1:2:125];
img(i, :) = 0;

k=[2:2:126];
```

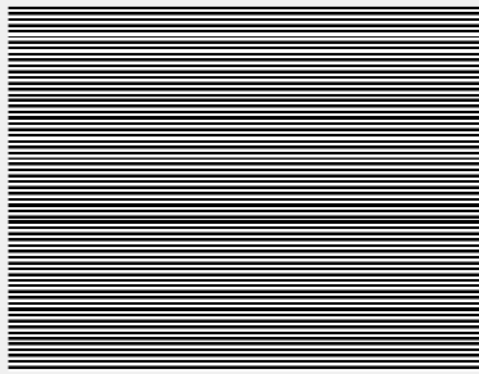


```
img(k, :) = 1;
```

```
figure;
```

```
imshow(img);
```

OUTPUT



6. Create a matrix of order 256 x 256 with some random values in the range [1, 80]. Display the corresponding image on the screen with colorbar.

PROGRAM

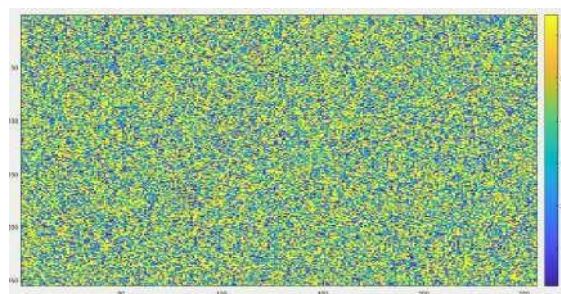
```
C = randi(80,256,256);
```

```
image(C);
```

```
colormap('default')
```

```
colorbar
```

OUTPUT



7. Check whether a matrix inverse exists, and if exists, find the inverse.

PROGRAM

```
clc;
clear all;
close all;
A=input('enter the matrix');
%inverse of A exists only if matrix is square and non singular
[m n]= size(A);%check the no. of rows and columns

if m==n
ifdet(A)==0 %matrix is singular
disp('Inverse does not exist');
else
inv_A=inv(A);
disp('Inverse is')
disp(inv_A);
end
else
disp('Matrix is not a square matrix');
end
```

INPUT

```
[0 1 2;1 2 3; 3 1 1]
```

OUTPUT

Inverse is

```
0.5000 -0.5000 0.5000
-4.0000 3.0000 -1.0000
2.5000 -1.5000 0.5000
```

8. Solve the system of equations

(a) $2x - y + 3z = 5$; $4x + 5z = 12$; $x + y + 2z = -3$.

(b) $x + y + z = 6$; $x + 2y + 3z = 10$; $2x + 4y + 6z = 20$

(c) $x_1 + 3x_2 + 2x_3 = 0$; $2x_1 - x_2 + 3x_3 = 0$; $3x_1 - 5x_2 + 4x_3 = 0$; $x_1 + 17x_2 + 4x_3 = 0$

Verify your answer using 'linsolve' function

PROGRAM

```
clc;
clear all;
close all;
A=input('enter the A matrix');
B=input('enter the B matrix');
[m n]=size(A); %m rows and n columns

C=[A B];
if rank(A)==rank(C)
if rank(A)==n %n is the no. of unknowns
disp('The system has unique solution');
    z1=inv(A)*B
else
if rank(A)<n %n is the no. of unknowns
disp('The system has infinite no.of solutions');
end
end
else
disp('The system has no solution');
end

if B==0 %Homogeneous system of equations, AX=0
disp('The trivial solution is');
    z1=0
end
```

OUTPUT

(a)

The system has unique solution $z1 =$

10.0000

-1.8000

-5.6000

(b)

The system has infinite no.of solutions

(c)The system has infinite no.of solutions

The trivial solution is

$z1 = 0$

9. Show that the sum of the eigen values is equal to the trace of the matrix and the product of the eigen values gives the determinant of the matrix.

PROGRAM

```
clc;
```

```
clear all;
```

```
close all;
```

```
A=input('enter the matrix');
```

```
Eig_values=eig(A);
```

```
Detr=det(A)
```

```
p=prod(Eig_values)
```

```
T=trace(A) %sum of diagonal elements of A
```

```
S=sum(Eig_values)
```

INPUT

```
[-2 2 -3;2 1 -6;-1 -2 0]
```

OUTPUT

```
Detr =45
```

```
p =45.0000
```

```
T =-1
```

```
S =-1.0000
```

10. Show that $AV=VD$, where D is the eigen values and V is the eigen vectors of the square matrix A . From this relation, represent A matrix using eigen value decomposition.

PROGRAM

```
clc;
clear all;
close all;
A=input('enter the matrix');
[V D]=eig(A);
Eig_values=diag(D);

%verifying AV = VD
LHS=A*V;
RHS=V*D;
Difference=LHS-RHS;
A_approximate= V*D*inv(V);
Diff=A-A_approximate;
```

INPUT

```
[-2 2 -3;2 1 -6;-1 -2 0]
```

OUTPUT

```
A_approximate =
-2.0000  2.0000 -3.0000
 2.0000  1.0000 -6.0000
-1.0000 -2.0000 -0.0000
```

11. For the matrix $A =$

```
-2  2  -3
    21-6
-1-2  0
```

Show that the eigen values are the roots of the characteristic equation

PROGRAM

```
A = [-2 2 -3; 2 1 -6; -1 -2 0];
p=poly(A);
Root = roots(p);
Eig_values=eig(A);
```

OUTPUT

```
Root =
    5.0000 + 0.0000i
   -3.0000 + 0.0000i
   -3.0000 - 0.0000i
Eig_values =
   -3.0000
    5.0000
   -3.0000
```

12. Approximate the matrix A for N = 1000 with the help of singular value decomposition of A as

$$A^{\wedge} = \sum_{i=0}^r \lambda_i U_i V_i^T$$

where U_i and V_i are the singular vectors and λ_i are the eigen values with $\lambda_i < \lambda_j$ for $i > j$. Plot the absolute error (ζ) between A and A^{\wedge} as $\zeta = \sum_{i=1}^N \sum_{j=1}^N |a_{i,j} - a^{\wedge}_{i,j}|^2$ against r for r = 10, 50, 75, 100, 250, 500, 750 and appreciate the plot.

PROGRAM

```
clc;
clear all;
close all;
M=1050;
N=1000;
A=randi(10,M,N);
[U,S,V] = svd(A);
% U is M x M matrix , V is N x N matrix and S is M x N Diagonal matrix

M1=U*S*V';
vtrans = V';

r=[10 50 75 100 250 500 750 1000];

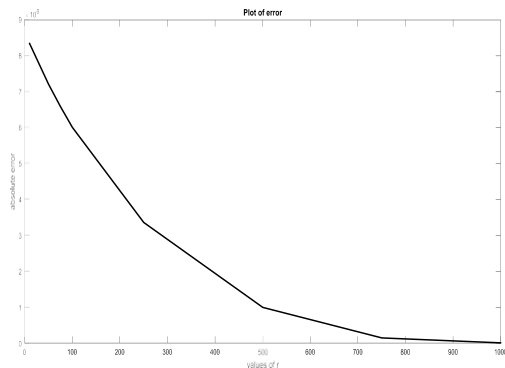
for i=1:8
    U1=U(:,[1:r(i)]);
    V1trans=vtrans([1:r(i)],:);
    S1=S([1:r(i)],[1:r(i)]);
    M2=U1*S1*V1trans;

    sum1=0;
    for j=1:M
        for k=1:N
            sum1 =sum1+(abs(A(j,k)-M2(j,k)))^2;
        end
    end

    errors(i)=sum1;
end
```

```
plot(r, errors);  
title('Plot of error');  
xlabel('values of r')  
ylabel('absolute error');
```

OUTPUT



13. Plot sine and cosine waves (both continuous and discrete plots) using GUI.

PROCEDURE

1. Type GUIDE in the command window
2. In the GUIDE Quick Start dialog box, select the **Blank GUI (Default)** template, and then click **OK**.
3. Display the names of the components in the component palette:
 - a. Select **File > Preferences > GUIDE**.
 - b. Select **Show names in component palette**.
 - c. Click **OK**.
4. Add the two push buttons to the UI. Select the push button tool from the component palette at the left side of the Layout Editor and drag it into the layout area. Create three buttons

5. Add the remaining components to the UI.

A static text area

A pop-up menu

An axes

6. Label the Push Buttons

(a) In the layout area, click the top push button.

(b) In the Property Inspector, select the String property, and then replace the existing value with the word Continuous.

(c) The push button label changes to Continuous

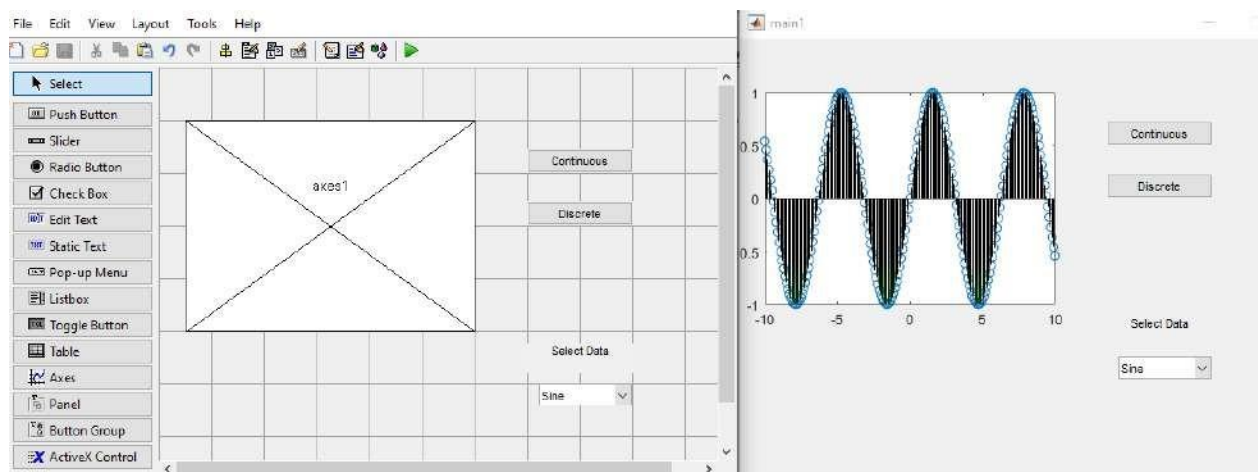
7. In the layout area, click the pop-up menu. In the Property Inspector, click the button next to String. The String dialog box displays. Replace the existing text with the names: sine and cosine

8. The static text serves as a label for the pop-up menu. In the layout area, click the static text. In the Property Inspector, click the button next to String. In the String dialog box that displays, replace the existing text with the phrase Select Data.

9. Save the Layout: When you save a layout, GUIDE creates two files, a FIG-file and a code file.

10. Code the Behavior of the App

OUTPUT



EXP NO: 4

DATE:

NUMERICAL DIFFERENTIATION AND INTEGRATION

OBJECTIVE

- To perform numerical differentiation and integration

LEARNING OUTCOMES

After the completion of this experiment students will be able to Implement numerical integration and differentiation.

SOFTWARE USED:

MATLAB[®] R2013

THEORY

The first derivative of the function $f(x)$, which we write as df/dx , is the slope of the tangent line to the function at the point x . To put this in non-graphical terms, the first derivative tells us how whether a function is increasing or decreasing, and by how much it is increasing or decreasing.

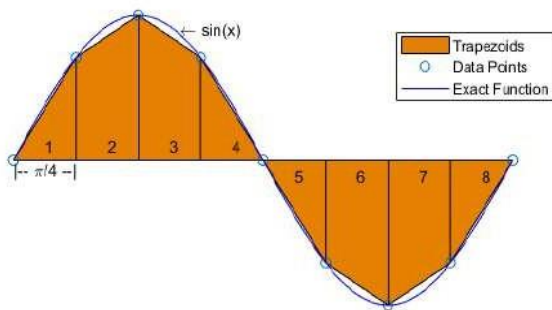
The second derivative of a function is the derivative of the derivative of that function. We write it as $d^2 f / dx^2$. While the first derivative can tell us if the function is increasing or decreasing, the second derivative tells us if the first derivative is increasing or decreasing. If the second derivative is positive, then the first derivative is increasing, so that the slope of the tangent line to the function is increasing as x increases

sin – Sine of argument in radians

cos - cos of argument in radians

✓ **Trapezoidal Method**

trap: performs numerical integration via the trapezoidal method. This method approximates the integration over an interval by breaking the area down into trapezoids with more easily computable areas. For example, here is a trapezoidal integration of the sine function using eight evenly-spaced trapezoids:



For an integration with N+1 evenly spaced points, the approximation is

$$\int_a^b f(x) dx \approx \frac{b-a}{2N} \sum_{n=1}^N (f(x_n) + f(x_{n+1}))$$

$$= \frac{b-a}{2N} [f(x_1) + 2f(x_2) + \dots + 2f(x_N) + f(x_{N+1})],$$

where the spacing between each point is equal to the scalar value $\frac{b-a}{N}$. By default MATLAB® uses a spacing of 1.

If the spacing between the N+1 points is not constant, then the formula generalizes to

$$\int_a^b f(x) dx \approx \frac{1}{2} \sum_{n=1}^N (x_{n+1} - x_n) [f(x_n) + f(x_{n+1})],$$

where $a = x_1 < x_2 < \dots < x_N < x_{N+1} = b$, and $(x_{n+1} - x_n)$ is the spacing between each consecutive pair of points.

Simpsons rule

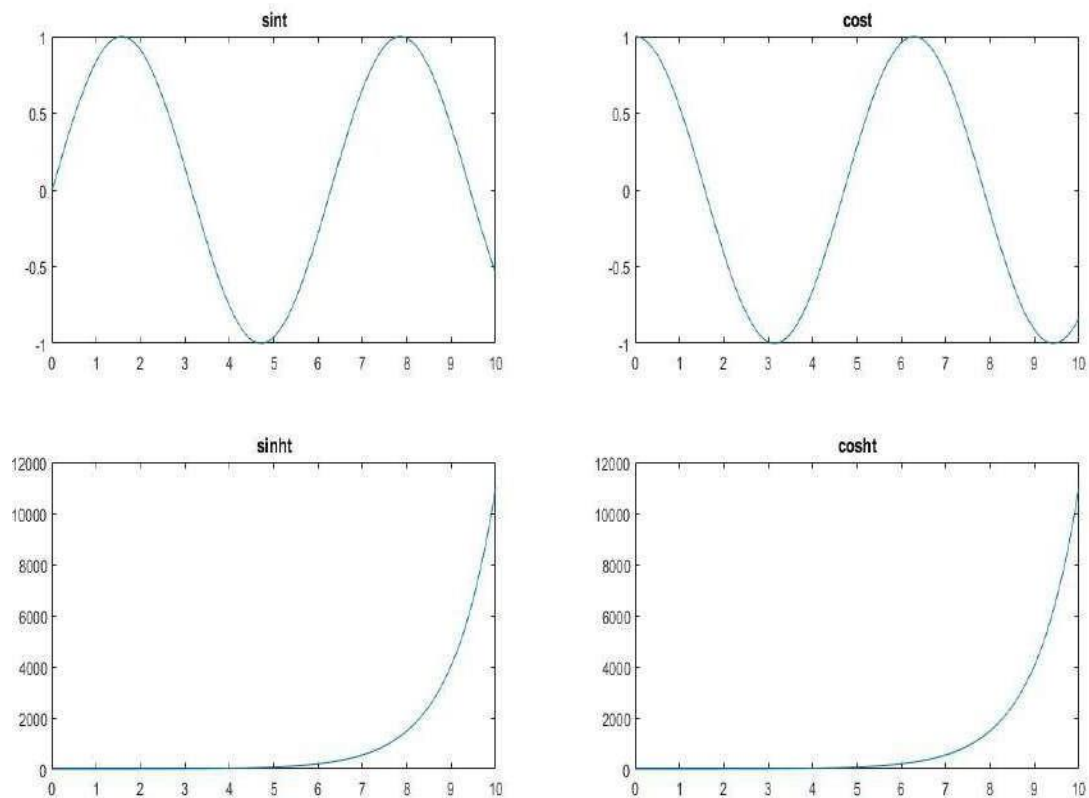
Formula: $(h/3) * [(y_0 + y_n) + 2 * (y_3 + y_5 + \dots \text{odd term}) + 4 * (y_2 + y_4 + y_6 + \dots \text{even terms})]$

$$h = (b-a)/n$$

b is upper limit, a is lower limit and n is number of sub intervals. y_0 and y_n are first and last term.

1. Realize the functions sin t, cos t, sinht and cosht for the vector $t = [-0, 10]$ with increment 0:01

```
t=-0:0.01:10;
a=sin (t);
b=cos (t);
c= sinh (t);
d=cosh (t);
subplot(2,2,1)
plot(t,a)
subplot(2,2,2)
plot(t,b)
subplot(2,2,3)
plot(t,c)
subplot(2,2,4)
plot(t,d)
```



2. Compute the first and second derivatives of these functions using built in tools such as `grad`

```
t=0:0.01:10;
```

```
a=sin (t);
```

```
subplot(3,3,1)
```

```
plot(t,a)
```

```
title('sint')
```

```
b=cos (t);
```

```
c= sinh (t);
```

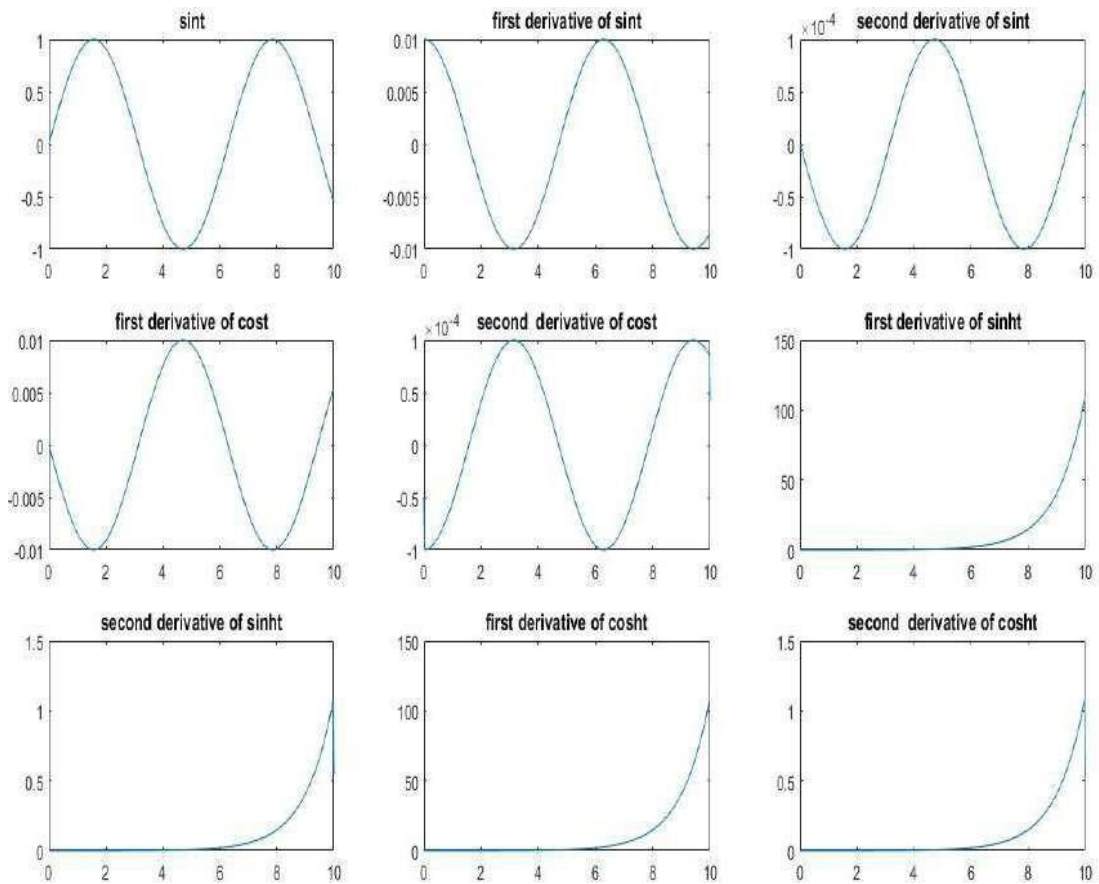
```
d=cosh (t);
```

```
da=gradient(a);
```

```
subplot(3,3,2)
```

```
plot(t,da)
```

```
title('first derivative of sint')
d2a=gradient(da);
subplot(3,3,3)
plot(t,d2a)
title('second derivative of sint')
db=gradient(b);
subplot(3,3,4)
plot(t,db)
title('first derivative of cost')
d2b=gradient(db);
subplot(3,3,5)
plot(t,d2b)
title('second derivative of cost')
dc=gradient(c);
subplot(3,3,6)
plot(t,dc)
title('first derivative of sinht')
d2c=gradient(dc);
subplot(3,3,7)
plot(t,d2c)
title('second derivative of sinht')
dd=gradient(d);
subplot(3,3,8)
plot(t,dd)
title('first derivative of cosht')
d2d=gradient(dd);
subplot(3,3,9)
plot(t,d2d)
title('second derivative of cosht')
```



3. Compute the first and second derivatives of sint, cost, sinht, cosht functions using built in tools such as grad and plot the derivatives over the respective functions for the vector $t = [-5, 5]$ with increment 0:01

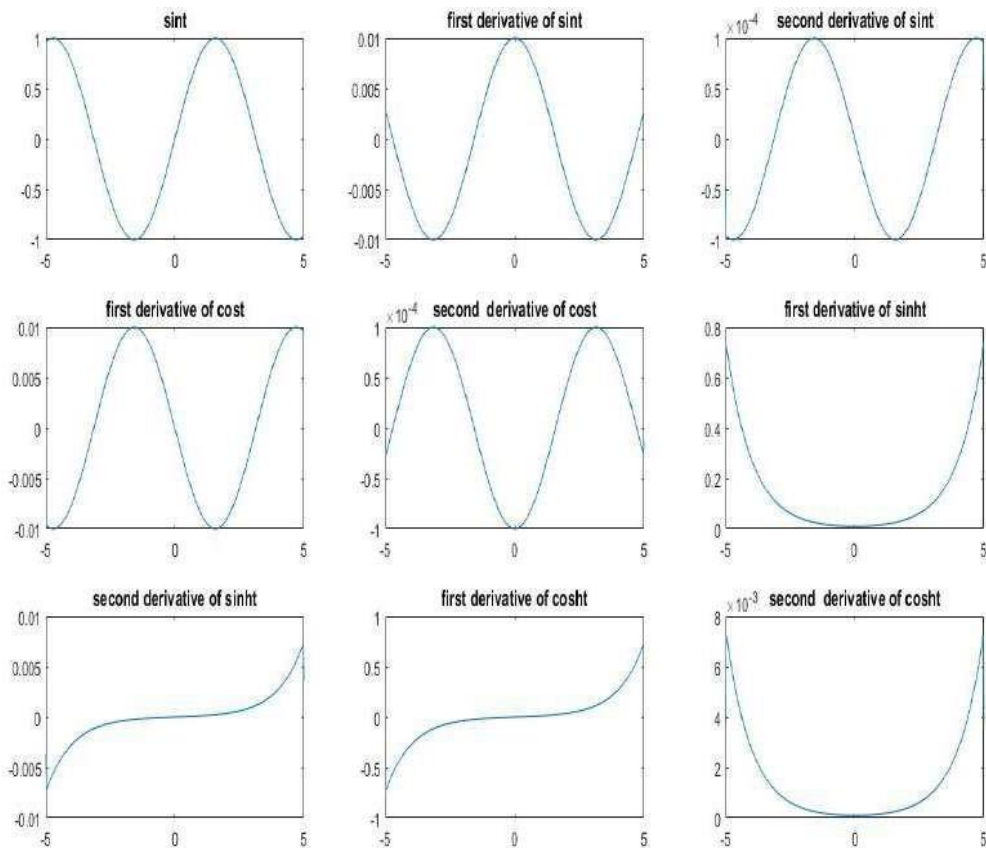
```
t=-5:0.01:5;  
a=sin (t);  
subplot(3,3,1)  
plot(t,a)  
title('sint')  
b=cos (t);  
c= sinh (t);  
d=cosh (t);
```

```
da=gradient(a);
subplot(3,3,2)
plot(t,da)
title('first derivative of sint')
d2a=gradient(da);
subplot(3,3,3)
plot(t,d2a)
title('second derivative of sint')
db=gradient(b);
subplot(3,3,4)
plot(t,db)
title('first derivative of cost')
d2b=gradient(db);
subplot(3,3,5)
plot(t,d2b)
title('second derivative of cost')
dc=gradient(c);
subplot(3,3,6)
plot(t,dc)
title('first derivative of sinht')
d2c=gradient(dc);
subplot(3,3,7)
plot(t,d2c)
title('second derivative of sinht')
dd=gradient(d);
subplot(3,3,8)
plot(t,dd)
title('first derivative of cosht')
d2d=gradient(dd);
subplot(3,3,9)
```



```
plot(t,d2d)
```

```
title('second derivative of cosht')
```



4. Familiarise numerical integration tools used in matlab

a. Create the function $f(x)=e^{-x^2}(\ln x)^2$. Evaluate integral from 0 to infinity

```
fun = @(x) exp(-x.^2).*log(x).^2;
```

```
q = integral(fun,0,Inf)
```

```
q = 1.9475
```

b. Create the function $f(x)=1/(x^3-2x-c)$ with one parameter, c . Evaluate the integral from $x=0$ to $x=2$ at $c=5$.

```
fun = @(x,c) 1./(x.^3-2*x-c);
```

```
q = integral(@(x) fun(x,5),0,2)
```

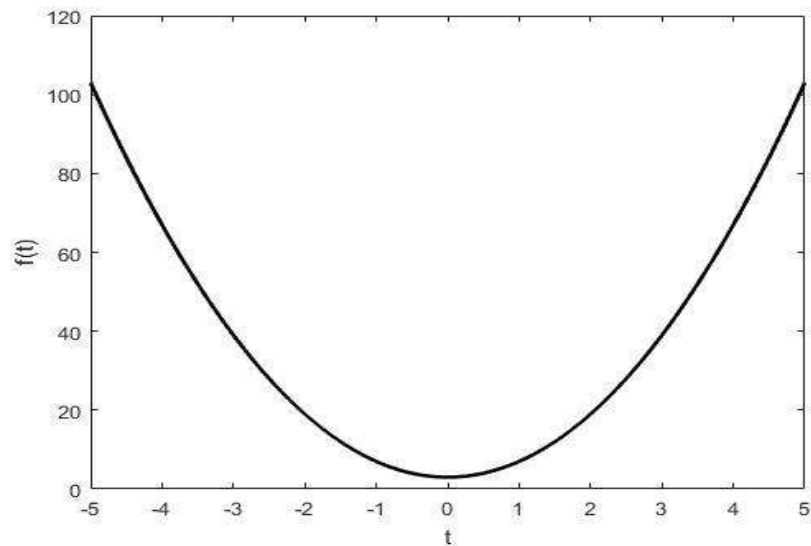
$$q = -0.4605$$

c. Create the function $f(x)=\ln(x)$. Evaluate the integral from $x=0$ to $x=1$

```
fun = @(x)log(x);  
q1 = integral(fun,0,1)  
q1 = -1.000000
```

5. Realize the function $f(t) = 4t^2 + 3$ and plot it for the vector $[-5,5]$ with increment 0.01

```
t=-5:0.01:5;  
y=4*(t.^2)+3;  
plot(t,y,'k','linewidth',2)  
xlabel't'  
ylabel'f(t)'
```



6. Use general integration tool to compute

$$\int_{-2}^2 (t + 2) dt$$

General integration tool

```
clc;
clear all;
fun = @(t) t + 2;
q = integral(fun,-2,2)
ans q=8
```

7. Repeat using trapezoidal method and Simpsons method

```
t=-2:.5:2;
y=t+2;
q=trapz(t,y)

ans q=8
```

TRAPEZOIDAL USING LOOP

```
clc;
clear all;
f=@(x)x+2; %Change here for different function
a=input('Enter lower limit a: '); % exmple a=1
b=input('Enter upper limit b: '); % exmple b=2
n=input('Enter the no. of subinterval: '); % exmple n=10
h=(b-a)/n;
sum=0;
for k=1:1:n-1
x(k)=a+k*h;
y(k)=f(x(k));
sum=sum+y(k);
end
% Formula: (h/2)*[(y0+yn)+2*(y2+y3+..+yn-1)]
```

```
answer=h/2*(f(a)+f(b)+2*sum);
fprintf("\n The value of integration is %f",answer);
```

```
answer=8
```

SIMPSON USING LOOP

```
clc;
clear all;
f=@(x)x+2; %Change here for different function
a=input('Enter lower limit a: '); % exmple a=-2
b=input('Enter upper limit b: '); % exmple b=2
n=input('Enter the number of sub-intervals n: '); % exmple n=16
h=(b-a)/n;

for k=1:1:n
x(k)=a+k*h;
y(k)=f(x(k));
end
so=0;se=0;
for k=1:1:n-1
if rem(k,2)==1
    so=so+y(k);%sum of odd terms
else
    se=se+y(k); %sum of even terms
end
end
% Formula: (h/3)*[(y0+yn)+4*(y3+y5+..odd term)+2*(y2+y4+y6+...even terms)]
answer=h/3*(f(a)+f(b)+4*so+2*se);
fprintf("\n The value of integration is %f",answer);
```

8. Compute $\frac{1}{\sqrt{2\pi}} \int_0^{\infty} e^{-x^2/2} dx$ using above three methods

```
i)    t=0:.5:1000;
      yi=@(t)exp((-t.^2)/2);
      qi = integral(yi,0,1000)
      answeri=(1/sqrt(2*pi))*qi
```

answeri= 0.500

```
ii)   t=0:.5:1000;
      y=exp((-t.^2)/2);
      q=trapz(t,y);
      answer= (1/sqrt(2*pi))*q
```

answer= 0.500

```
iii)

      clc;
      clear all;
      f=@(x)exp((-x.^2)/2); %Change here for different function
      a=input('Enter lower limit a: '); % exmple a=1
      b=input('Enter upper limit b: '); % exmple b=2
      n=input('Enter the number of sub-intervals n: '); % exmple n=16
      h=(b-a)/n;

      for k=1:1:n
          x(k)=a+k*h;
          y(k)=f(x(k));
      end
      so=0;se=0;
      for k=1:1:n-1
          if rem(k,2)==1
              so=so+y(k);%sum of odd terms
          else
              se=se+y(k); %sum of even terms
```

```
end
end
% Formula: (h/3)*[(y0+yn)+2*(y3+y5+..odd term)+4*(y2+y4+y6+...even terms)]
q=h/3*(f(a)+f(b)+4*so+2*se);
answer= (1/sqrt(2*pi))*q
fprintf('\n The value of integration is %f',answer);
```

Answer:

Enter lower limit a: 0

Enter upper limit b: 1000

Enter the number of sub-intervals n: 1000

answer =

0.4976

The value of integration is 0.497603

EXP NO: 5

DATE:

SOLUTION OF ORDINARY DIFFERENTIAL EQUATION

OBJECTIVE

- To solve ordinary differential Equation

LEARNING OUTCOMES

- After the completion of this experiment students will be able to solve ordinary differential Equation

SOFTWARE USED:

MATLAB[®] R2013

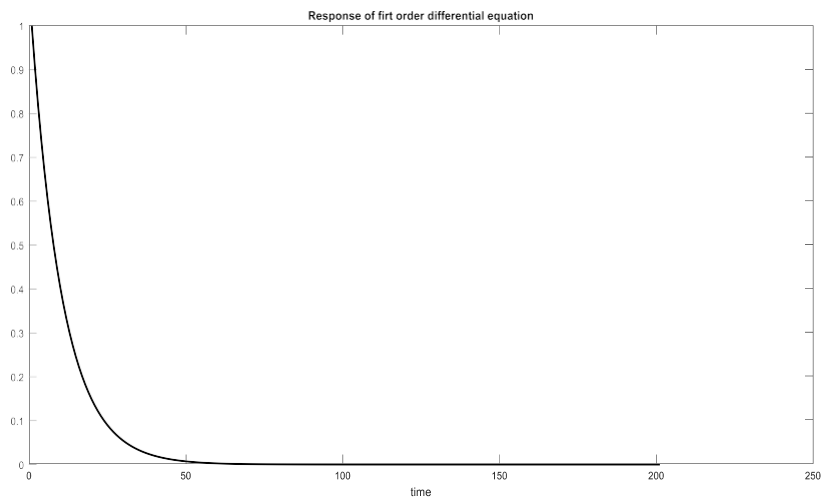
1. Solve the first order differential equation

$$\frac{dx}{dt} + 2x = 0$$

With initial condition $x(0)=1$

PROGRAM

```
syms x(t);  
eqn= diff(x,t)+ 2*x ==0;  
cond= x(0)==1;  
soln= dsolve(eqn,cond)  
t=0:0.05:10  
s=subs(soln) plot(s)  
title('Response of first order  
differential equation')  
xlabel('time')
```

OUTPUT

soln =exp (-2*t)

2. Solve second order differential equation

$$\frac{d^2x}{dt^2} + 2\frac{dx}{dt} + 2x = e^{-t}$$

PROGRAM

```
syms x(t) ;  
eqn= diff(x,t,2) + 2*diff(x,t) + 2*x == exp(-t);  
soln= dsolve(eqn);  
soln=simplify(soln)
```

OUTPUT

```
soln =  
exp(-t)*(C4*cos(t) + C5*sin(t) + 1)
```


3. SOLVE THE DIFFERENTIAL EQUATION

$$\frac{d^2y}{dx^2} = \cos(2*x) - y$$

$$y(0) = 1$$

$$y'(0) = 0$$

PROGRAM

```
syms y(x)
Dy = diff(y);
ode = diff(y,x,2) == cos(2*x)-y;
cond1 = y(0) == 1;
cond2 = Dy(0) == 0;
conds = [cond1 cond2];
ySol = dsolve(ode,conds);
ySol = simplify(ySol)
```

OUTPUT

1 - (8*sin(x/2)^4)/3

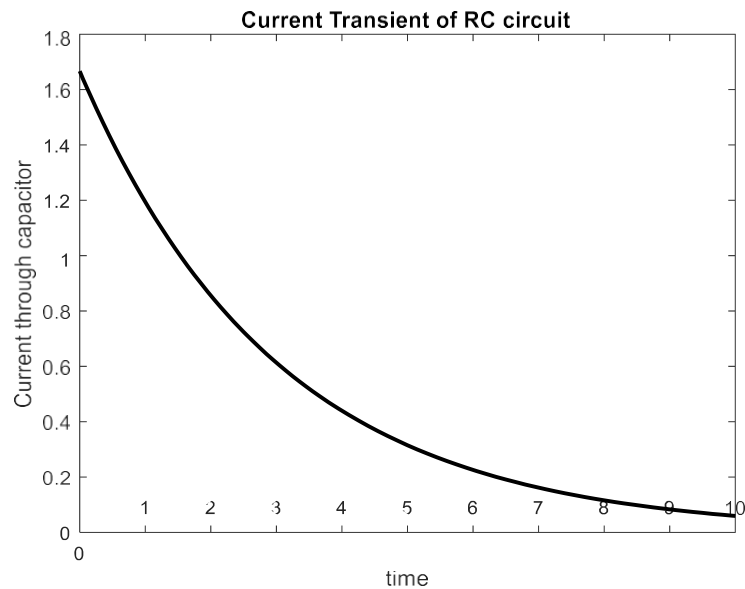
4. Solve for the current transient through an RC network (with $RC = 3$) that is driven by
- 5V DC
 - The signal $5e^{-t}U(t)$ and plot the solutions.

(a) PROGRAM

```
clc;
clear all;
close all;
symsi(t);

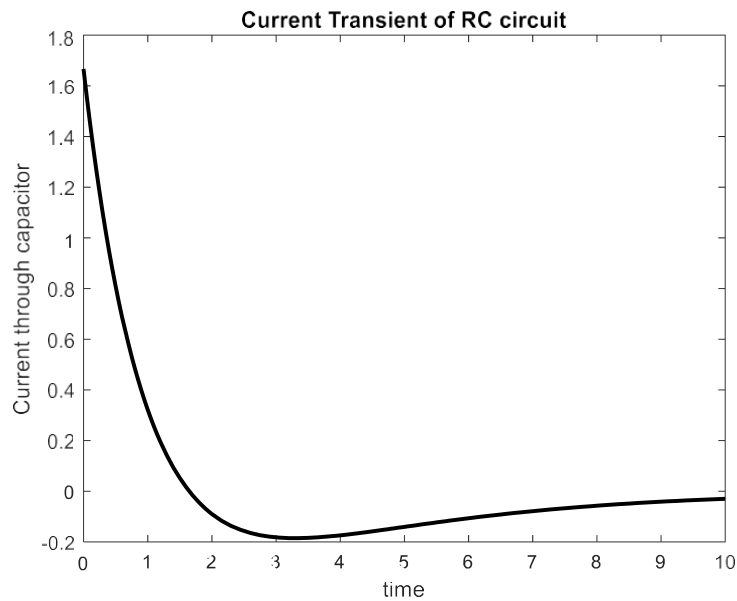
V=5;
R=3;
C=1;

eqn= diff(i,t)+ i/(R*C) ==0;
cond= i(0)==V/R;
soln= dsolve(eqn,cond)
t=0:.05:10;
s=subs(soln);
plot(t,s,'k', 'linewidth',2)
title('Current Transient of RC circuit');
xlabel('time');
ylabel('Current through capacitor');
```

OUTPUT**(b)PROGRAM**

```
clc;
clear all;
close all;
syms i(t);
V=5*exp(-t);

R=3;
C=1;
eqn= diff(i,t)+ i/(R*C) ==diff(V,t)/R;
cond= i(0)==5/R;
soln= dsolve(eqn,cond)
t=0:.05:10;
s=subs(soln);
plot(t,s,'k', 'linewidth',2)
title('Current Transient of RC circuit');
xlabel('time');
ylabel('Current through capacitor');
```

OUTPUT

5. Solve for the voltage across the capacitor of an RC network that is driven by 5V DC, with three different time constants.

PROGRAM

```
clc;
clear all;
close all;
symsvc(t);
V=5;
R1=3; %RC time constant = 3
C1=1;
TC1=R1*C1;%Time constant
eqn= diff(vc,t)==(V-vc)/(TC1);
cond= vc(0)==0;
soln1= dsolve(eqn,cond)
```

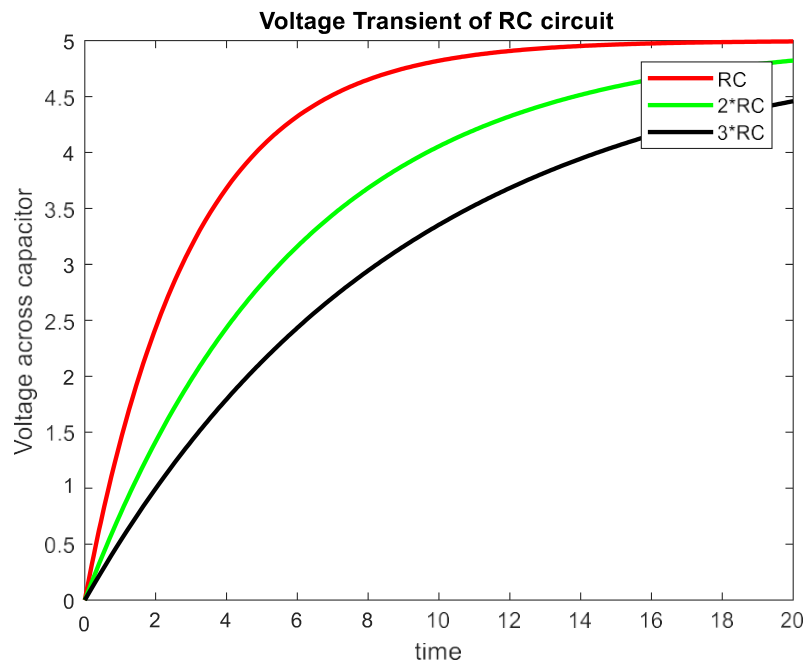
```
TC2=2*TC1;%2 times the Time constant
eqn= diff(vc,t)==(V-vc)/(TC2);
soln2= dsolve(eqn,cond)
```

```
TC3=3*TC1;%3 times the Time constant
eqn= diff(vc,t)==(V-vc)/(TC3);
soln3= dsolve(eqn,cond)
```

```
t=0:.05:20;
s1=subs(soln1);
s2=subs(soln2);
s3=subs(soln3);
plot(t,s1, 'r', 'linewidth',2);
hold on;
plot(t,s2, 'g', 'linewidth',2);
hold on;

plot(t,s3, 'k', 'linewidth',2);
legend('RC','2*RC','3*RC');
title('Voltage Transient of RC circuit');
xlabel('time');
ylabel('Voltage across capacitor');
```

OUTPUT



6. Solve the current transient through a series RLC circuit with $R = 9$, $L = 1\text{H}$ and $C = 0.05\text{ F}$ that is driven by

(a) 20 V DC

(b) The signal $20e^{-t}U(t)$ and plot the solutions

(a) PROGRAM

```
clc;
```

```
clear all;
```

```
close all;
```

```
symsi(t);
```

```
V=20;
```

```
R1=9;
```

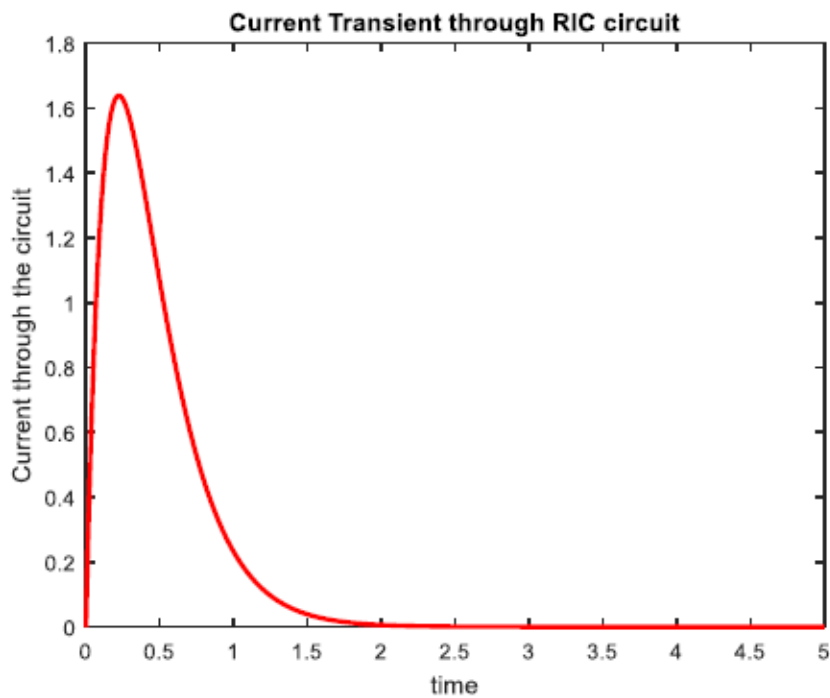
```
L1=1;
```

```
C1=0.05;
```

0.5 1 1.5 2 2.5 3 3.5 4 4.5 5

```
Di=diff(i);  
eqn1= diff(i,t,2)+(R1/L1)*diff(i,t)+(1/(L1*C1))* i ==0;  
cond1=[i(0)==0, Di(0)==20];  
soln1= dsolve(eqn1, cond1);  
t=0:.005:5;  
s1=subs(soln1);  
plot(t,s1,'r', 'linewidth',2);  
title('Current Transient through RIC circuit');  
xlabel('time');  
ylabel('Current through the circuit');
```

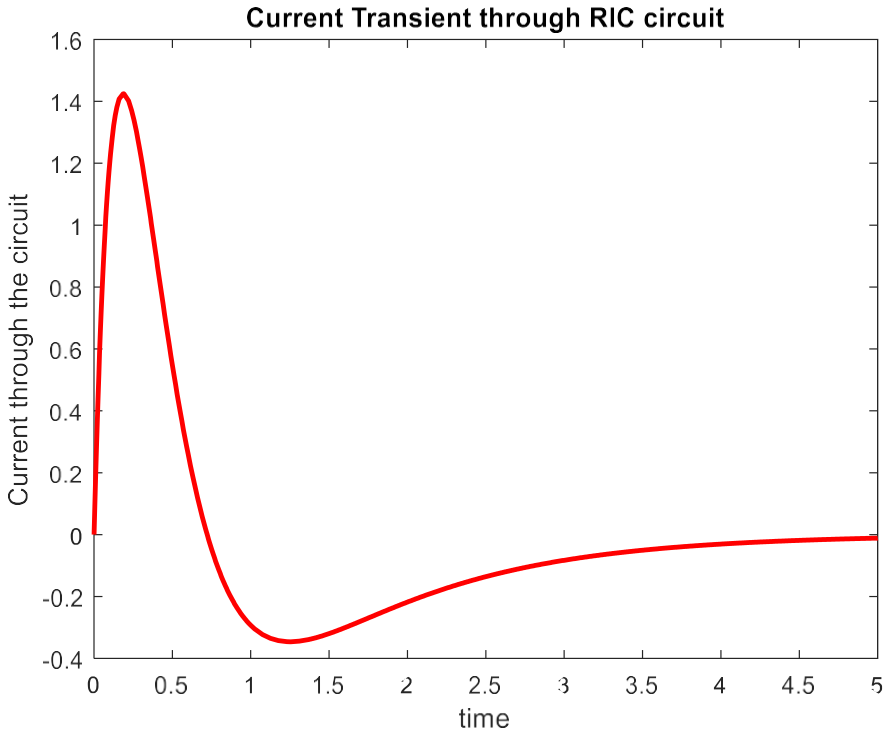
OUTPUT



(b) PROGRAM

```
clc;
clear all;
close all;
symsi(t);
V=20*exp(-t);
R1=9;
L1=1;
C1=0.05;
Di=diff(i);
eqn1= diff(i,t,2)+(R1/L1)*diff(i,t)+(1/(L1*C1))* i ==(1/L1)*diff(V,t);
cond1=[i(0)==0, Di(0)==20];
soln1= dsolve(eqn1, cond1);
t=0:.005:5;
s1=subs(soln1);
plot(t,s1,'r', 'linewidth',2);
title('Current Transient through RIC circuit');
xlabel('time');
ylabel('Current through the circuit');
```

OUTPUT



SIMPLE DATA VISUALISATION

OBJECTIVE

- To visualize the data in different ways

LEARNING OUTCOMES

- After the completion of this experiment students will be able to plot data using scatter, boxplot, histogram, bar functions

SOFTWARE USED:

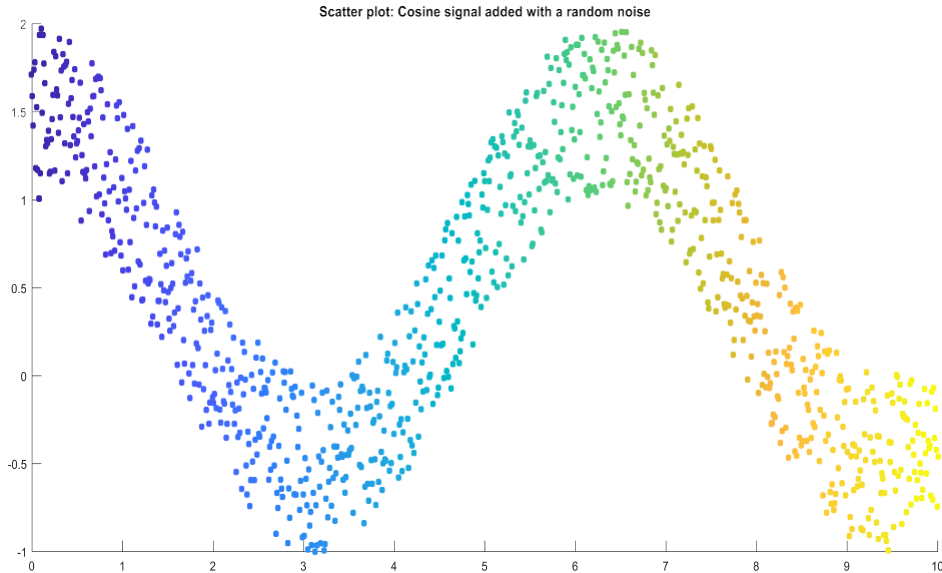
MATLAB[®] R2013

1. Plot a cosine signal added with random noise using scatter plot

PROGRAM

```
x = 0:0.01:10;  
len=length(x);  
y = cos(x) + rand(1,len);  
sz = 25;  
c = 0:0.01:10;  
scatter(x,y,sz,c,'filled')  
title('Scatter plot: Cosine signal added with a random noise')
```

OUTPUT

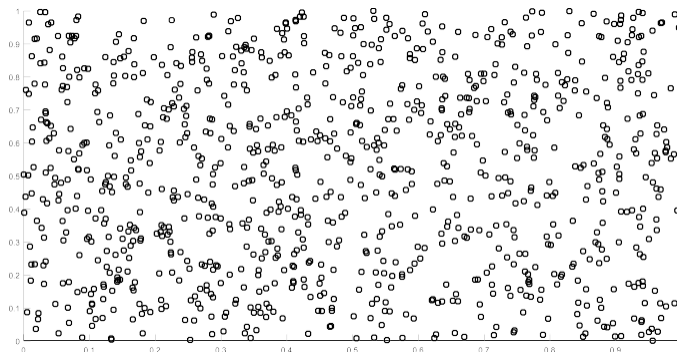


2. Plot some random data using scatter plot

PROGRAM

```
x = rand(1000,1);  
y = rand(1000,1);  
s = scatter(x,y,[],'k');
```

OUTPUT



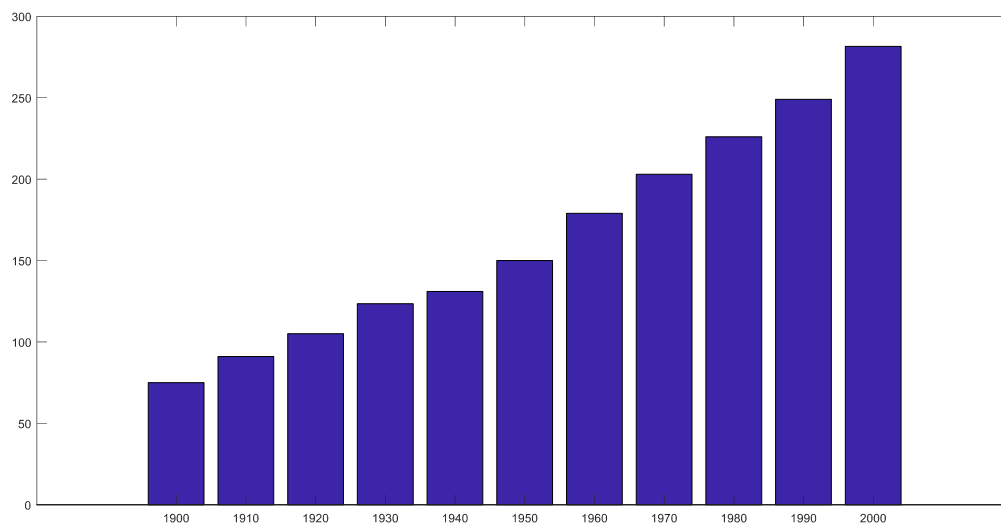
3. Plot the bar plot of the data $y = [75 \ 91 \ 105 \ 123.5 \ 131 \ 150 \ 179 \ 203 \ 226 \ 249 \ 281.5]$ having values

on the x axis as [1900:10:2000].

PROGRAM

```
x = 1900:10:2000;  
y = [75 91 105 123.5 131 150 179 203 226 249 281.5];  
bar(x,y)
```

OUTPUT



4. Plot the bar plot of the matrix

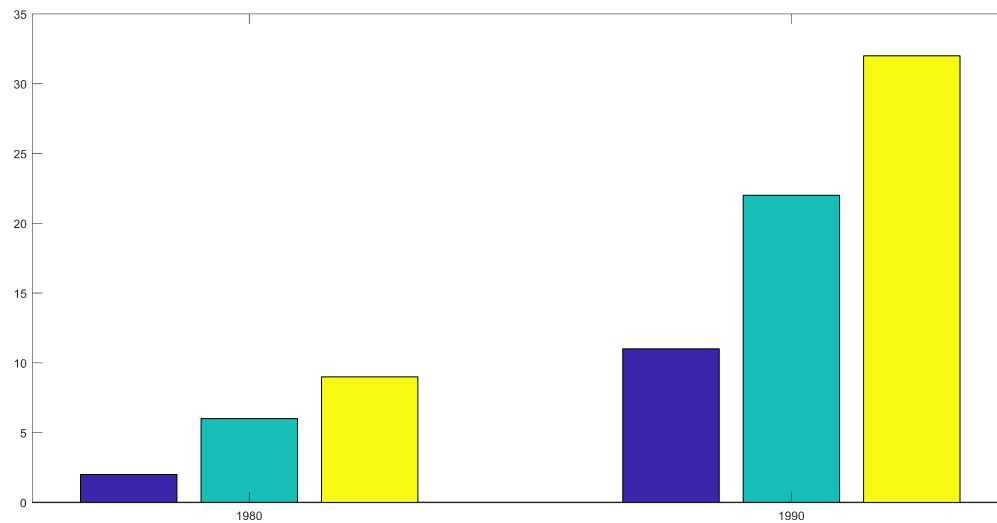
```
      2      6      9  
11  22  32
```

Against x as [1980 1990]

PROGRAM

```
x = [1980 1990];  
y = [2 6 9;11 22 32];  
bar(x,y)
```

OUTPUT



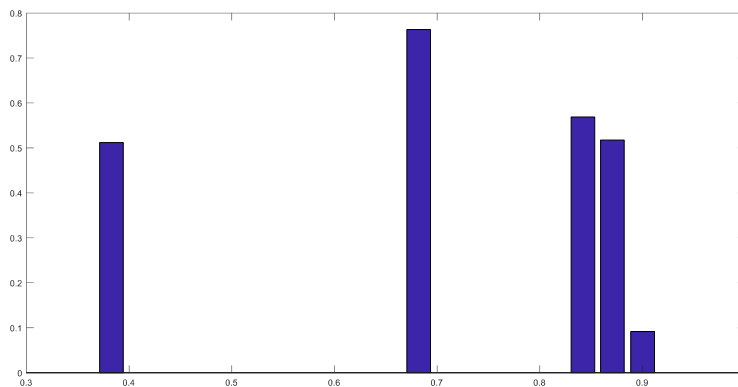
5. Plot the bar plot of some random data

PROGRAM

```

clc ;
clear all;
close all;
x = rand (1,5)
y = rand (1,5)
bar (x, y);
    
```

OUTPUT

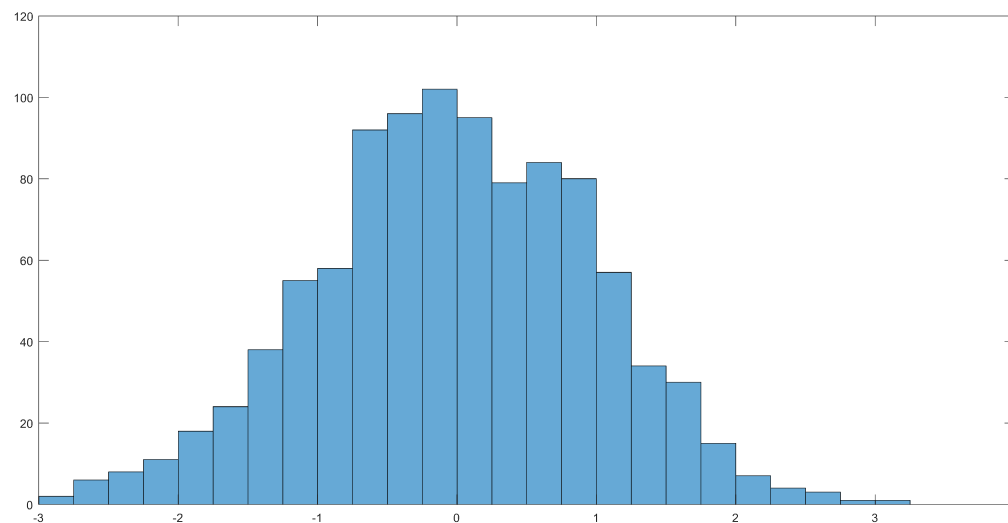


6. Plot the histogram of some random data, with the number of bins specified

PROGRAM

```
x = randn(1000,1);  
nbins = 25;  
h = histogram(x,nbins)
```

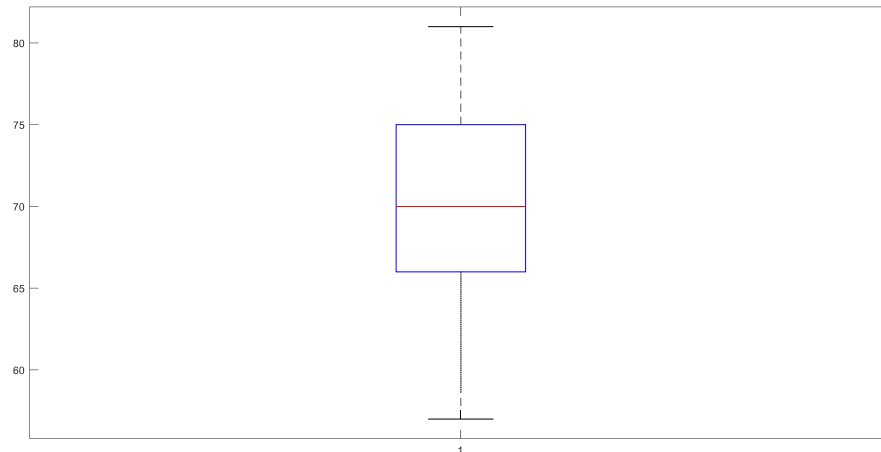
OUTPUT



7. Plot the data [57, 57, 57, 58, 63, 66, 66, 67, 67, 68, 69, 70, 70, 70, 70, 72, 73, 75, 75, 76, 76, 78, 79, 81] using box plot.

PROGRAM

```
x=[57, 57, 57, 58, 63, 66, 66, 67, 67, 68, 69, 70, 70, 70, 70, 72, 73, 75, 75, 76, 76, 78, 79, 81];  
boxplot(x)
```

OUTPUT

8. Define 't' as an array (-10, 10) with an increment of 0.01. Plot

(i)cos(t)

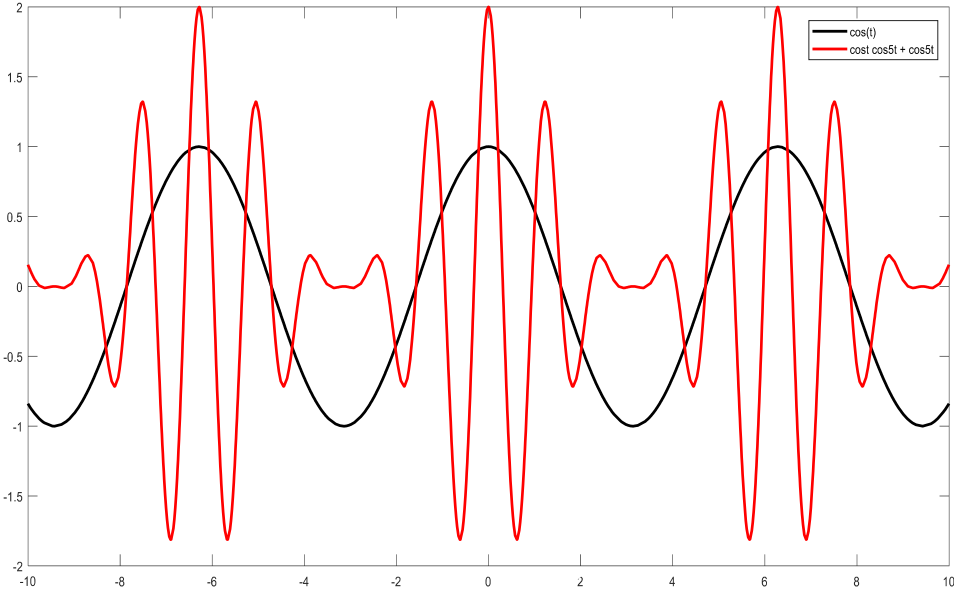
(ii) cost cos5t + cos5t

On the same graph. Create legends in plots

PROGRAM

```
clc;
clear all;
close all;
t=-10:0.01:10;
y=cos(t);
plot(t,y,'k','linewidth',2)
hold on;
y=(cos(t).*cos(5*t))+cos(5*t);
plot(t,y,'r','linewidth',2)
hold on;
legend('cos(t)','cost cos5t + cos5t');
```

OUTPUT



EXP NO: 7

DATE:

SIMPLE DATA ANALYSIS

OBJECTIVE

Simple Data Analysis

1 LEARNING OUTCOMES

After the completion of this experiment students will be able to Implement Simple Data Analysis

2 SOFTWARE USED:

MATLAB[®] R2013

3 THEORY

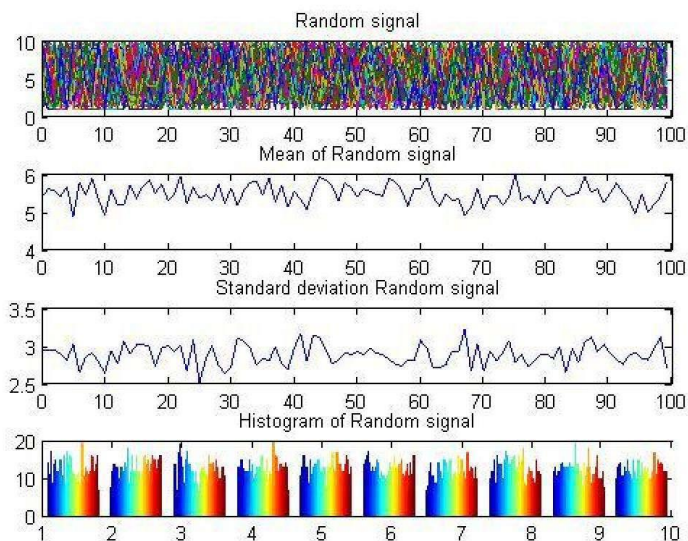
MATLAB provides wide Variety of tools to find the statistical properties of a random signal. A **mean** is the simple mathematical average of a set of two or more numbers. The standard deviation is a statistic that measures the dispersion of a dataset relative to its mean calculated as the square root of the

1. To plot a find mean and variance of random signal and Plot its histogram with an appropriate bin size.

PROGRAM

```
clc
clear all
close all
t=0:1:99;
x=randi(10,100);
subplot(4,1,1) plot(t,x)
title('Random signal')
mu=mean(x); stan=std(x);
subplot(4,1,2) plot(t,mu)
title('Mean of Random signal')
subplot(4,1,3)
plot(t,stan)
title('Standard deviation Random signal')
subplot(4,1,4)
hist(x)
title('Histogram of Random signal')
```

OUTPUT



EXP NO: 9

DATE:

COIN TOSS AND THE LEVEL CROSSING PROBLEM

OBJECTIVE

To study the coin tossing problem and level crossing problem

4 LEARNING OUTCOMES

After the completion of this experiment students will be able to perform coin tossing problem and level crossing problem

5 SOFTWARE USED:

MATLAB[®] R2013

6 THEORY

Study of experiments like coin tossing and level crossing problem can be easily done using MATLAB coding. Random numbers can be easily generated and evaluated in MATLAB using commands like randi, randn etc.

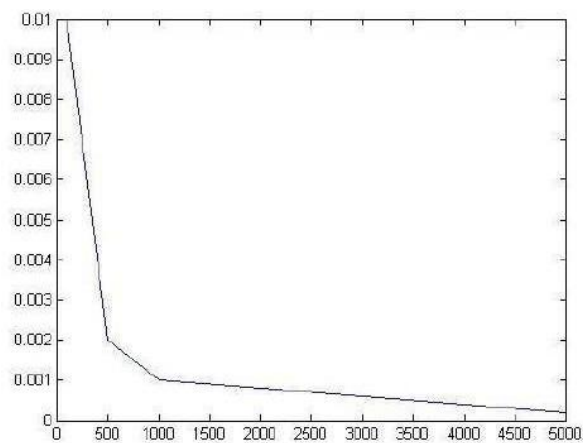
7 PROGRAM

1. MATLAB code for Coin Tossing Experiments

```
clc  
clear all  
close all
```

```
N=[100, 500,1000, 5000]
for i=1:4 exp=randi(1,N(i));
head=0;
tail=0;
for j=1:N(i)
    if(exp(j)==1)
        head=head+1;
    else
        tail=tail+1;
    end
end
p(i)=head/N(i);
abs_error(i)=p(i)/N(i); end
plot(N,abs_error)
```

OUTPUT



2. MATLAB code for level crossing Experiments

```
clc
clear all
close all
exp=randi(10,100);
abov_threshold=0; for
i=1:100
    if(exp(i)>2)
        abov_threshold=abov_threshold+1;
    end
end
below_threshold=100-abov_threshold;
disp('Number of points above threshold=')
disp(abov_threshold)
disp('Number of points below threshold')
disp(below_threshold)
```

8 OUTPUT

```
Number of points above
threshold= 85
Number of points below
threshold 15
```
