LABORATORY MANUAL

of

ECL 201: SCIENTIFIC COMPUTING LABORATORY (S3)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

COLLEGE OF ENGINEERING

TRIVANDRUM

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

COLLEGE OF ENGINEERING

TRIVANDRUM

ECL 201	SCIENTIFIC COMPUTING LABORATORY	CATEGORY	L	T	P	CREDIT
		PCC	0	0	3	2

Preamble

- The following experiments are designed to translate the mathematical concepts into system design.
- The students shall use Python for realization of experiments. Other software's such as R/MATLAB/SCILAB/LabVIEW can also be used.
- The experiments will lay the foundation for future labs such as DSP lab.
- The first two experiments are mandatory and any six of the rest should be done.

Prerequisites

- MAT 101 Linear Algebra and Calculus
- MAT 102 Vector Calculus, Differential Equations and Transforms

(i)Course Outcomes: The student will be able to

CO1	To make use of data types, control structures, and scripting techniques for solving problems in scientific computing.
C02	To analyze the relationship between eigenvalues and matrix rank, and to analyze the reconstruction error in approximating a matrix using reduced-rank approximations based on singular value decomposition.
CO3	To evaluate the performance of numerical approximation methods, such as the trapezoidal rule and Simpson's method, by comparing their accuracy against exact analytical solutions for integration and differentiation.
CO4	To analyze transient behaviors in electronic circuits by solving ordinary differential equations (ODEs)
CO5	To analyze the convergence of Fourier series in approximating periodic functions by comparing approximation errors for different numbers of terms

(ii) CO-PO/PSO matrix showing level of correlation (1-Low, 2-Medium, and 3-high)

Course: ECL 201 Scientific computing lab															
Course outcome designation	**CO-PO/PSO matrix showing level of correlation (1-Low, 2-Medium, and 3-high)**														
	PO 1	**PO 2**	**PO 3**	**PO 4**	**PO 5**	**PO 6**	**PO 7**	**PO 8**	**PO 9**	**PO 10**	**PO 11**	**PO 12**	**PS01**	**PS02**	**PS03**
ECL 201/CO1	2	1			3			1	2	2		2	2	1	1
ECL 201/CO2	3	3	1	1	2			1	2	2		2	3	2	1
ECL 201/CO3	3	2			3			1	2	2		2	2	1	2
ECL 201/CO4	3	3	2	1	3			1	2	2		2	3	2	1
ECL 201/CO5	3	3	1	1	3			1	2	2		2	3	2	1

Assessment Pattern

Mark Distribution

Total Mark	**CIE**	**ESE**
150	75	75

Continuous Internal Evaluation Pattern

Attribute	Mark
Attendance Continuous assessment Internal Test (Immediately before the second series test)	15 30 30

End Semester Examination Pattern: The following guidelines should be followed regarding award of marks.

Attribute	Mark
Preliminary work	15
Implementing the work/Conducting the experiment	10
Performance, result and inference (usage of equipment's and troubleshooting)	25
Viva voce	20
Record	5

SCIENTIFIC COMPUTING LAB

INTRODUCTION

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

- 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

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.

Experiment I
Familiarization of MATLAB: Part A

AIM:

Write a MATLAB/PYTHON 2.7 script to compute and print the first N Fibonacci numbers, where N is a positive integer entered by the user.

a) Using for loop
b) Using while loop

SOFTWARE REQUIRED:

MATLAB 2017 / PYTHON 2.7 or higher.

ALGORITHM

STEP 1 Input the value of N from the user.

STEP 2 Initialize the first two values of the series as 0 and 1.

STEP 3 Using either a *for* loop or a *while* loop, generate the n^{th} term as the sum of the two preceding terms. (Refer Appendix for syntax of *for* and *while*)

STEP 4 Display the generated numbers.

SAMPLE INPUT/OUTPUT

```
Enter value of N,to get N fibonacci numbers
8
     0

     1

     1

     2

     3

     5

     8

    13

>>
```

RESULT

The script written successfully printed the Fibonacci series till N terms where N was inputted by the user.

Familiarization of MATLAB: Part B

AIM:

Write a MATLAB/PYTHON 2.7 script for a calculator that supports the following operations:

a. i) Addition ii) Subtraction iii) Multiplication iv) Division v) Logical operations (AND, OR, NOT, EX-OR)

b. Repeat (a) with vectored computations. In addition to the five sub-sections of (a), do the following operations: vi) Element wise multiplication vii) Element wise division viii) Element wise power operations.

SOFTWARE REQUIRED:

MATLAB 2017 / PYTHON 2.7 or higher.

ALGORITHM

STEP 1 Display the operators available on the Command Window. The operators available for scalar inputs (Part (a)) is a subset of the operators available for vector inputs (Part (b)).

STEP 2 Input the preferred operation from the user.

STEP 3 Input the required number of scalar (Part (a)) or vector inputs (Part (b)) from the user. Note that not all operations require two operands.

STEP 4 Display the results.

SAMPLE INPUT/OUTPUT

```
COMMAND WINDOW
CALCULATOR
Enter first operand
8
Enter choice of operator,
 +,-,*,/,&,|,~,e (ex-or)
e
Enter second operand
1
ANSWER
   0

>>
```

```
COMMAND WINDOW
CALCULATOR
Enter first operand
5
Enter choice of operator,
 +,-,*,/,&,|,~,e (ex-or)
&
Enter second operand
0
ANSWER
   0

>>
```

```
COMMAND WINDOW
CALCULATOR
Enter first operand
30
Enter choice of operator,
 +,-,*,/,&,|,~,e (ex-or)
/
Enter second operand
3
ANSWER
    10
```

```
COMMAND WINDOW
CALCULATOR
Enter first operand
7
Enter choice of operator,
 +,-,*,/,&,|,~,e (ex-or)
)
Enter second operand
7
ANSWER
operator not supported
>>
```

```
COMMAND WINDOW
VECTOR CALCULATOR
Enter length of vector
2
Enter elements of first vector
1
2
Enter choice of operator,
 +,-,*(element wise),/(element wise),&,|,~,e (ex-or),^(element wise)
^
Enter elements of second vector
2
3
ANSWER
     1     8
```

```
COMMAND WINDOW
VECTOR CALCULATOR
Enter length of vector
3
Enter elements of first vector
1
2
3
Enter choice of operator,
 +,-,*(element wise),/(element wise),&,|,~,e (ex-or),^(element wise)
+
Enter elements of second vector
3
2
1
ANSWER
     4     4     4
```

RESULT

The script written, successfully calculated basic operations in numbers according to the user's choice.

Familiarization of MATLAB: Part C

AIM:

Write a MATLAB/PYTHON 2.7 script to compute the conjugate of a given complex number (without using any inbuilt function) and hence compute its absolute value.
Verify the results using inbuilt functions.

SOFTWARE REQUIRED:

MATLAB 2017 / PYTHON 2.7 or higher.

ALGORITHM

STEP 1 Input the complex number from the user.

STEP 2 Find the conjugate of the complex number by using the np.conjugate(A.T)/ctranspose(A) operator.

STEP 3 Compute the absolute value by taking the square root of the product of the original complex number and its conjugate.

STEP 4 Print the results in the Command Window.

STEP 5 Verify the results using inbuilt functions.

SAMPLE INPUT/OUTPUT

COMMAND WINDOW

```
Enter a complex number
4+2i

c =

   4.0000 - 2.0000i

calculated conjugate and absolute
   4.0000 - 2.0000i

    4.4721

using function
   4.0000 - 2.0000i

    4.4721

Successfully verified conjugate
Succesfully verified absolute
>>
```

RESULT

The script written successfully calculated the conjugate and absolute values of the given complex numbers

Experiment II
Scientific Computing: Part A

AIM:

Write a MATLAB/PYTHON 2.7 function *sum_complex()* to compute the sum of N complex numbers and hence use the same to compute the absolute value, conjugate, real part and imaginary part of the sum output.

SOFTWARE REQUIRED:

MATLAB 2017 / PYTHON 2.7 or higher.

ALGORITHM

STEP 1 Input the value of *N* from the user.

STEP 2 Create a loop having *N* iterations. In each iteration, the user can input a complex number.

STEP 3 Store the *N* inputted complex numbers in an array.

STEP 4 Pass this array to *sum_complex()* which computes the sum and other values.

STEP 5 Display the computed values on the Command Window.

SAMPLE INPUT/OUTPUT

COMMAND WINDOW

```
Enter number of complex numbers
4

 Enter 4 complex numbers
3i
2+3i
4-5i
6
sum
  12.0000 + 1.0000i


 real part=12
 complex part=1
 absolute value of sum=   12.0416


 conjugate value of sum=  12.0000 - 1.0000i

>>
```

COMMAND WINDOW

```
Enter number of complex numbers
2

 Enter 2 complex numbers
2+4i
-2-4i
sum
     0


 real part=0
 complex part=0
 absolute value of sum=     0


 conjugate value of sum=     0

>>
```

RESULT

The script is written successfully calculated the sum of N complex numbers according to the users input, using a function and computed the absolute value, conjugate, real and imaginary part of the sum.

Scientific Computing: Part B

AIM:

Create a MATLAB/PYTHON 2.7 function to compute the factorial of a number and hence use the function to compute the factorial of a given set of numbers.

SOFTWARE REQUIRED:

MATLAB 2017 / PYTHON 2.7 or higher.

ALGORITHM

STEP 1 Create a MATLAB/PYTHON 2.7 function that can compute and display the factorial of a scalar number inputted by the user.

STEP 2 Input the value of *N,* the number of elements in the given set of numbers.

STEP 3 Create a loop having *N* iterations in the main iteration.

STEP 4 Call the function for computing and displaying the factorial in each iteration.

SAMPLE INPUT/OUTPUT

COMMAND WINDOW

```
Enter number of numbers,to get factorial
3
Enter number:
4
Factorial of 4 is 24
Enter number:
5
Factorial of 5 is 120
Enter number:
6
Factorial of 6 is 720
>>
```

RESULT

The function written successfully calculated the factorial of a number and the script successfully calculated the factorial of n numbers using the function.

Experiment III
Realization of Arrays and Matrices: Part A

AIM:

To write a MATLAB/PYTHON 2.7 Script to calculate Eigen values, Eigen vectors and rank of a 3x3 matrix without using any inbuilt functions and verify the results using inbuilt functions.

SOFTWARE REQUIRED:

MATLAB 2017 / PYTHON 2.7 or higher.

ALGORITHM

STEP 1 Input the square matrix from the user.

STEP 2 Using Symbolic Variable, define the characteristic equation of the matrix and solve the matrix to find the eigen values.

STEP 3 Compute the spanning vectors in the null space of $A - \lambda I$ where A is the matrix inputted by the user, λ is the eigen value and I is the identity matrix.

STEP 4 Verify the values of eigen values and eigen vectors using the built-in functions.

SAMPLE INPUT/OUTPUT

```
COMMAND WINDOW
Enter the matrix in square brackets
[2 3 4
5 6 5
4 4 3]
Eigen Values:
   -1.4743
    0.2781
   12.1963

Rank:
     3

Eigen Vectors:
    0.7973    0.5337   -0.4207
   -0.1407   -0.7707   -0.7506
   -0.5870    0.3483   -0.5095

Using Inbuilt functions:
Eigen Values:
   -1.4743
    0.2781
   12.1963

Rank:
     3

Eigen Vector:
   -0.4207   -0.7973    0.5337
   -0.7506    0.1407   -0.7707
   -0.5095    0.5870    0.3483

Eigen values are correct.
Rank is correct
>>
```

RESULT

The script written, successfully calculated the Eigen values, Rank and Eigen vectors of the 3x3 matrices entered by the user. The calculated values were successfully verified with the values provided by the inbuilt function.

Realization of Arrays and Matrices: Part B

AIM:

To write a MATLAB/PYTHON 2.7 script to create an N x 1 complex valued array and compute real part, imaginary part and conjugate of the complex numbers without using inbuilt functions.

SOFTWARE REQUIRED:

MATLAB 2017 / PYTHON 2.7 or higher.

ALGORITHM

STEP 1 Input the square matrix from the user.

STEP 2 Using Symbolic Variable, define the characteristic equation of the matrix and solve the matrix to find the eigen values.

STEP 3 Compute the spanning vectors in the null space of $A - \lambda I$ where A is the matrix inputted by the user, λ is the eigen value and I is the identity matrix.

STEP 4 Verify the values of eigen values and eigen vectors using the built-in functions.

SAMPLE INPUT/OUTPUT

```
COMMAND WINDOW
Enter the matrix in square brackets
[2 3 4
5 6 5
4 4 3]
Eigen Values:
   -1.4743
    0.2781
   12.1963

Rank:
     3

Eigen Vectors:
    0.7973    0.5337   -0.4207
   -0.1407   -0.7707   -0.7506
   -0.5870    0.3483   -0.5095

Using Inbuilt functions:
Eigen Values:
   -1.4743
    0.2781
   12.1963

Rank:
     3

Eigen Vector:
   -0.4207   -0.7973    0.5337
   -0.7506    0.1407   -0.7707
   -0.5095    0.5870    0.3483

Eigen values are correct.
Rank is correct
>>
```

RESULT

The script written, successfully calculated the Eigen values, Rank and Eigen vectors of the 3x3 matrices entered by the user. The calculated values were successfully verified with the values provided by the inbuilt function.

EXPERIMENT IV
Numerical Differentiation and Integration: Part A

AIM:

To create a MATLAB/PYTHON 2.7 script to obtain the derivatives of the function $f(t) = 3t^4 + 5$, up to an order equal to 3, and hence plot the function and its derivatives in the interval (-3,3).

SOFTWARE REQUIRED:

MATLAB 2017 / PYTHON 2.7 or higher.

ALGORITHM

STEP 1 Enter the given function using symbolic variables.

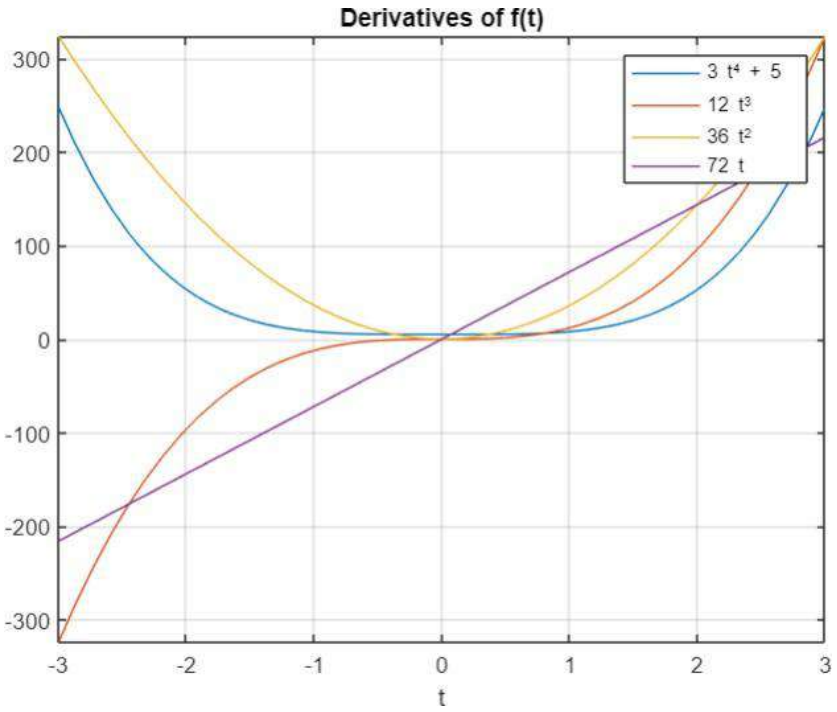
STEP 2 Find up to the third order derivative, one by one, using the inbuilt functions.

STEP 3 Display the derivatives in the Command Window using the inbuilt functions.

STEP 4 Plot the derivates using appropriate range for the x-axis.

SAMPLE INPUT/OUTPUT

```
COMMAND WINDOW
Enter the function:
3*(t^4)+5
First derivative:
12*t^3

Second derivative:
36*t^2

Third derivative:
72*t

Using Gradient function
First derivative:
12*t^3

Second derivative:
36*t^2

Third derivative:
72*t

>>
```



RESULT

The function written successfully calculated the derivative of the given function to an order of three and plotted the functions in a range (-3,3) successfully.

Numerical Differentiation and Integration: Part B

AIM:

To write a MATLAB/PYTHON 2.7 script that executes the integral $\int \mathrm{e}^{-|t|}) dt$ over the range (-3,3) and return the result.

SOFTWARE REQUIRED:

MATLAB 2017 / PYTHON 2.7 or higher.

ALGORITHM

STEP 1 Enter the given function using symbolic variables.

STEP 2 Find the integral of the function using the inbuilt functions.

STEP 3 Display the integral in the Command Window using the inbuilt functions.

SAMPLE INPUT/OUTPUT

```
COMMAND WINDOW
Enter the function:
exp(-abs(t))
Integral:2 - 2*exp(-3)
>>
```

RESULT

The script written, successfully accepted functions from the user and integrated the function with respect to t over the range (-3,3) successfully and displayed the result.

Experiment V
Solution of Ordinary Differential Equations: Part A

AIM:

To create a MATLAB/PYTHON 2.7 script to obtain the solution of the first order differential equation dx/dt + 2x = 0, with the initial condition x(0)=1.

SOFTWARE REQUIRED:

MATLAB 2017 / PYTHON 2.7 or higher.

ALGORITHM

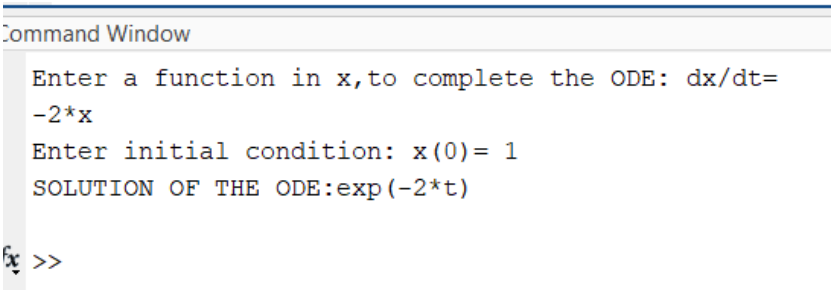
STEP 1 Define a symbolic variable

STEP 2 Input the given ODE using the symbolic variable.

STEP 3 Also, input the given initial conditions using the symbolic variable.

STEP 4 Solve the ODE using the inbuilt functions.

SAMPLE INPUT/OUTPUT



RESULT

The script successfully calculated the solution of the given code.

Solution of ordinary Differential Equations: Part B

AIM:

To compute the general response of a series RL circuit driven with 10u(t) and hence plot i(t) and VL(t) with R=10 ohm and L=1H and with i(0)=0.

SOFTWARE REQUIRED:

MATLAB 2017 / PYTHON 2.7 or higher.

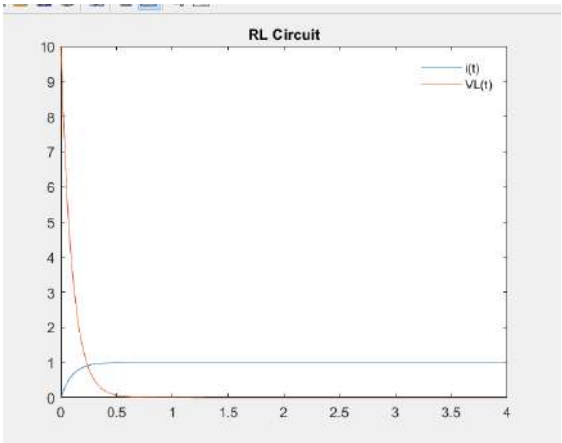
ALGORITHM

STEP 1 Enter the voltage, resistor and inductor values from the user.

STEP 2 Form the differential equation using a symbolic variable.

STEP 3 Solve the differential equation using the inbuilt functions.

SAMPLE INPUT/OUTPUT



RESULT

The script written, successfully accepted the R, L and V values and plotted the graph.

Solution of ordinary Differential Equations: Part C

AIM:

To create a MATLAB/PYTHON 2.7 script to obtain the solution of the second order differential equation $d^2x/dt^2 + 2\,dx/dt + 2x = e^{-t}$

SOFTWARE REQUIRED:

MATLAB 2017 / PYTHON 2.7 or higher.

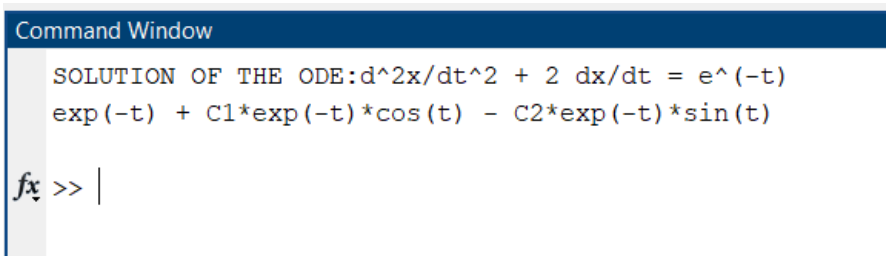
ALGORITHM

STEP 1 Enter the given differential equation using symbolic variables.

STEP 2 Solve the differential equation using the inbuilt functions.

STEP 3 Display the solution in the Command Window using the inbuilt functions.

SAMPLE INPUT/OUTPUT



RESULT

The script successfully calculated the solution of the given code.

Experiment VI
Simple data visualization

AIM

To generate and visualize different types of plots — stem plot, box plot, bar plot, and line plot — using random data arrays created with inbuilt functions in MATLAB/Python.

SOFTWARE REQUIRED:

MATLAB 2017 / PYTHON 2.7 or higher.

ALGORITHM

STEP 1 Input the number of stems for Stem Plot

STEP 2 Generate a random array of size equal to the number of stems using the inbuilt functions.

STEP 3 Create stem plot of the random array using the inbuilt functions.

STEP 4 Input the lower limit, upper limit and increment from the user.

STEP 5 Create a random array using the parameters in STEP 4

STEP 6 Plot the random array using the inbuilt functions.

STEP 7 Input the size of the random array for creating a box plot.

STEP 8 Generate a random array using the input size

STEP 9 Generate a box plot for the array using the inbuilt functions.

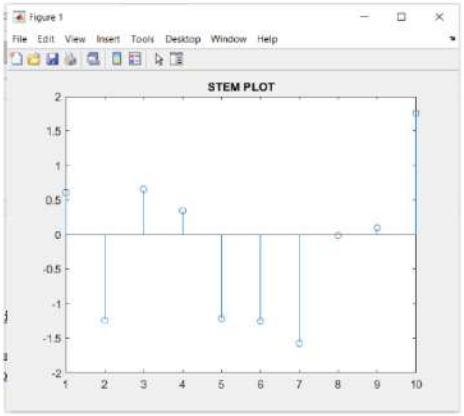
STEP 10 Repeat the above steps for bar plot and scatter plot

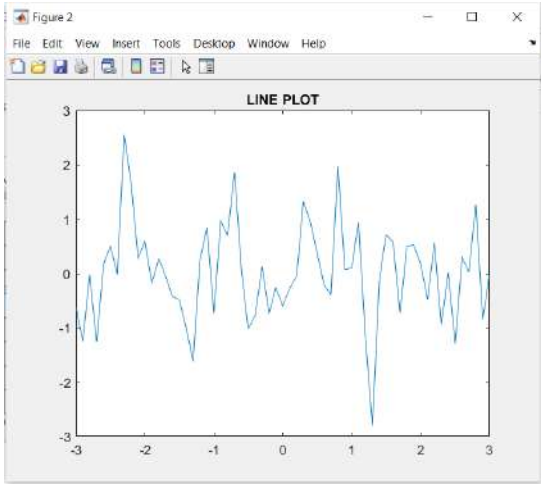
STEP 11 Generate the bar plot using the inbuilt functions.

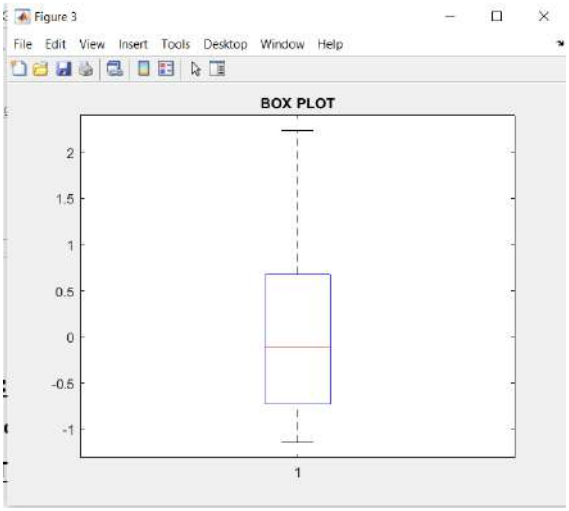
STEP 12 Generate the scatter plot using the inbuilt functions.

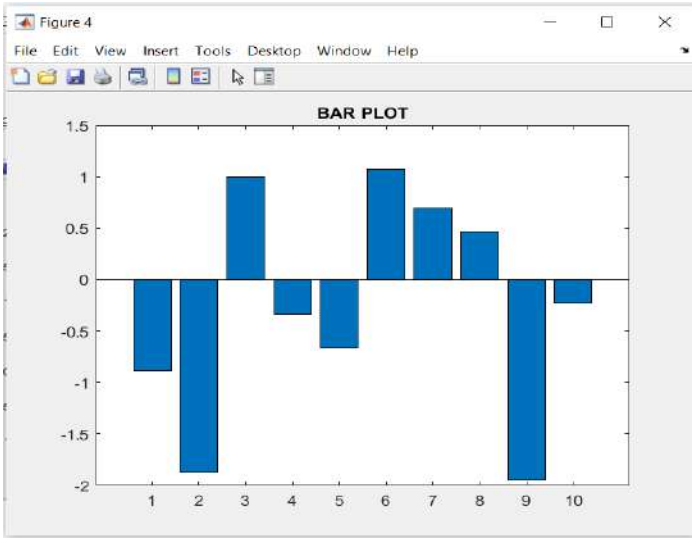
SAMPLE INPUT/OUTPUT

```
Command Window
  STEM PLOT
  Enter Number of stems: 10
  LINE PLOT
  Enter Lower Limit: -3
  Enter Upper Limit: 3
  Enter increment: 0.1
  BOX PLOT
  Enter number of values: 10
  BAR PLOT
  Enter number of bars: 10
  SCATTER PLOT
  Enter Lower Limit: -3
  Enter Upper Limit: 3
  Enter Increment: 0.1
fx >>
```









RESULT

The script successfully plotted and displayed the different plots.

Experiment VII
Simple Data Analysis with Spreadsheets: Part A

AIM:

To Read a .csv or .xls file as an array and plot it.

SOFTWARE REQUIRED:

MATLAB 2017 / PYTHON 2.7 or higher.

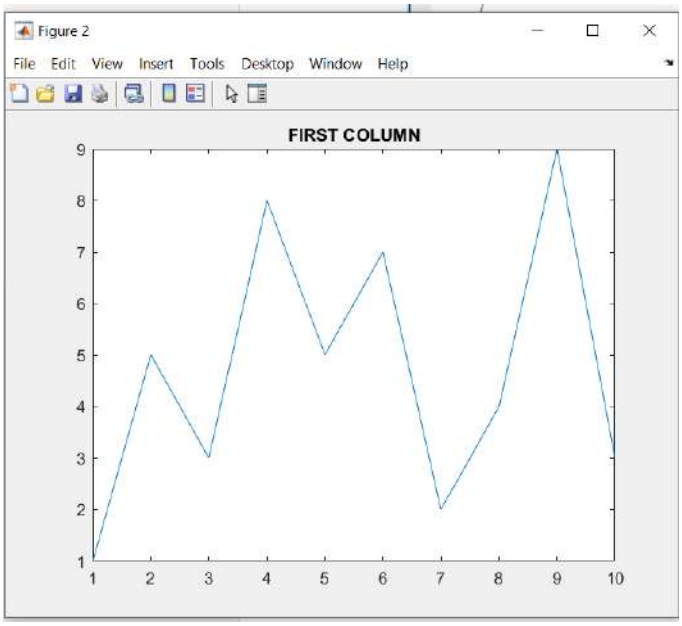
ALGORITHM

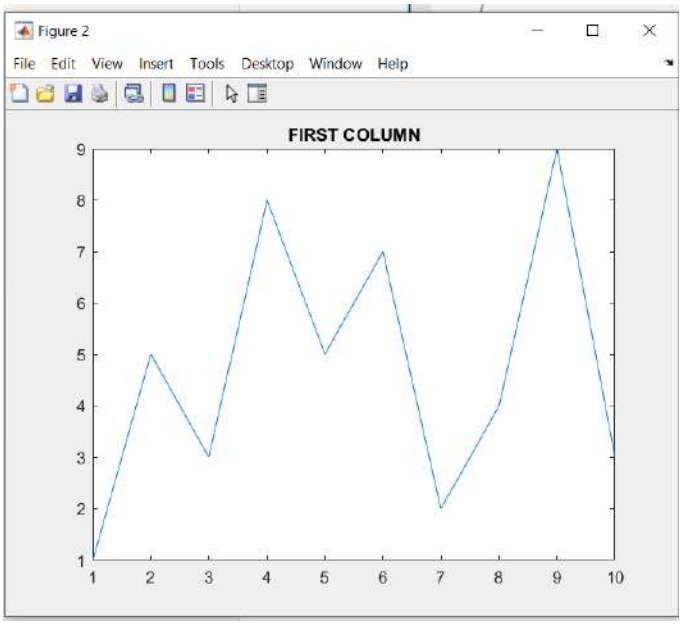
STEP 1 Make sure that the .xls file to be read is accessible to MATLAB/PYTHON 2.7

STEP 2 Read the .xls file using the inbuilt functions.

STEP 3 Plot the data using the inbuilt functions.

SAMPLE INPUT/OUTPUT





RESULT

The script successfully plotted the values from the given .xls file.

Simple Data Analysis with Spreadsheets: Part B

AIM:

To compute the mean and standard deviation of the signal and plot its histogram with an appropriate bin size.

SOFTWARE REQUIRED:

MATLAB 2017 / PYTHON 2.7 or higher.

ALGORITHM

STEP 1 Read the .xls file using the function the inbuilt functions.

STEP 2 Compute the mean of the data using the function the inbuilt functions.

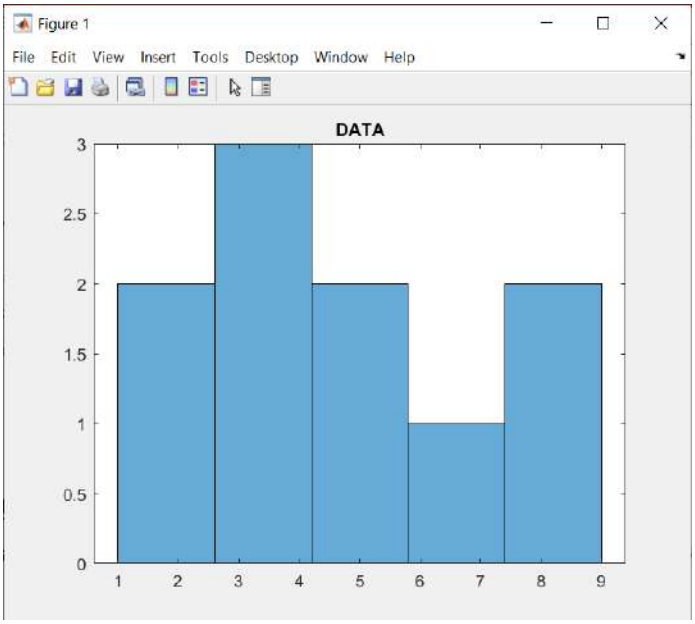
STEP 3 Compute the standard deviation of the function using the inbuilt functions.

STEP 4 Display the computed mean and standard deviation in the Command Window using the inbuilt functions.

STEP 5 Plot the histogram of the data using the function the inbuilt functions. Input the value of number of bins from the user.

SAMPLE INPUT/OUTPUT

```
Command Window

DATA =

     1
     5
     3
     8
     5
     7
     2
     4
     9
     3


Mean =

    4.7000


StandardDeviation =

    2.6268

Enter Bin value 5
>>
```



RESULT

The script written, successfully accepted the number of bins and plotted the histogram after reading the values from the given .xlsx file as an array.

Experiment VIII
Convergence of Fourier Series: Part A

AIM:

Realize the Fourier series f(t) =4/π [1 − 1/3 Cos 2π3t/T + 1/5 Cos 2π5t/T − 1/7 Cos 2π7t/T+ ⋯]. Realize the vector t = [0, 100] with an increment of 0.01 and keep T = 20. 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.

SOFTWARE REQUIRED:

MATLAB 2017 / PYTHON 2.7 or higher.

ALGORITHM

STEP 1 Input from the user the time window for the plot has to be created.

STEP 2 Input from the user the number of terms required.

STEP 3 Create a loop that runs from 1 to the value of the number of terms the user inputted.

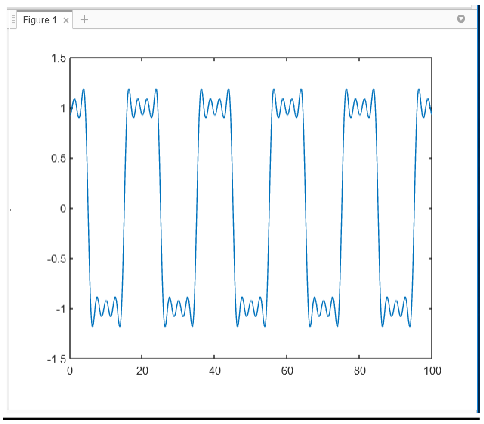
STEP 4 Compute the value of the Fourier series for each of these terms.

STEP 5 Sum all the terms using the inbuilt function

STEP 6 Plot the sum using the inbuilt functions.

SAMPLE INPUT/OUTPUT

```
COMMAND WINDOW
Enter Time Period:
20
Enter Lower Limit of T:
0
Enter Upper Limit of T:
100
Enter Increment:
0.01
Enter Number of Terms:
4
>>
```



RESULT

The script successfully realized the Fourier series and plotted it.

Convergence of Fourier Series: Part B

AIM:

To realize the Madhava series, $\pi = 4\ [1 - 1/3 + 1/5 - 1/7 + \cdots]$, with t made a zero vector, f (0) = 1 and then use this to compute π for the first 10, 20, 50 and 100 terms.

SOFTWARE REQUIRED:

MATLAB 2017 / PYTHON 2.7 or higher.

ALGORITHM

STEP 1 Input from the user the time period.

STEP 2 Input from the user the number of trials and the number of terms in each trial.

STEP 3 Generate a loop that runs through the number of trials.

STEP 4 Realize a Madhava series for each trial using the number of terms for that particular trial.

STEP 5 Display the sum of the series in the Command Window using the inbuilt functions for each trial.

SAMPLE INPUT/OUTPUT

```
COMMAND WINDOW
Enter Time Period:
20
Enter number of trials to print pi:
4
Enter number of terms for each trial
10
 π = 3.041840
20
 π = 3.091624
50
 π = 3.121595
100
 π = 3.131593
>>
```

RESULT

The script written, successfully accepted the values and displayed the value of pi using the Madhava series. It was observed by increasing the number of terms that the value of pi calculated approaches the accurate value of pi.

Appendix- Important Functions

Syntax	Description
`S = sum(A)`	`S = sum(A)` returns the sum of the elements of A along the first array dimension whose size does not equal 1.
`S = std(A)`	`S = std(A)` returns the standard deviation of the elements of `A` along the first array dimension whose size does not equal 1. By default, the standard deviation is normalized by `N-1`, where `N` is the number of observations.

`while expression`
`statements`
`end`
	`while expression, statements, end` evaluates an expression, and repeats the execution of a group of statements in a loop while the expression is true. An expression is true when its result is nonempty and contains only nonzero elements (logical or real numeric). Otherwise, the expression is false.

`for index = values`
`statements`
`end`
	`for index = values, statements, end` executes a group of statements in a loop for a specified number of times. `values` has one of the following forms:
	• `initVal:endVal` — Increment the `index` variable from `initVal` to `endVal` by `1`, and repeat execution of `statements` until `index` is greater than `endVal`.
	• `initVal:step:endVal` — Increment `index` by the value `step` on each iteration, or decrements `index` when `step` is negative.
	• `valArray` — Create a column vector, `index`, from subsequent columns of array `valArray` on each iteration. For example, on the first iteration, `index = valArray(:,1)`. The loop executes a maximum of *n* times, where *n* is the number of columns of `valArray`, given by `numel(valArray(1,:))`. The input `valArray` can be of any MATLAB® data type, including a character vector, cell array, or
	struct

`plot(X,Y)`	`plot(X,Y)` creates a 2-D line plot of the data in `Y` versus the corresponding values in `X`.

	• To plot a set of coordinates connected by line segments, specify `X` and `Y` as vectors of the same length. • To plot multiple sets of coordinates on the same set of axes, specify at least one of `X` or `Y` as a matrix.
`stem(Y)`	`stem(Y)` plots the data sequence, `Y`, as stems that extend from a baseline along the *x*-axis. The data values are indicated by circles terminating each stem. • If `Y` is a vector, then the *x*-axis scale ranges from 1 to `length(Y)`. • If `Y` is a matrix, then `stem` plots all elements in a row against the same *x* value, and the *x*-axis scale ranges from 1 to the number of rows in `Y`.
`histogram(X)`	`histogram(X)` creates a histogram plot of `X`. The `histogram` function uses an automatic binning algorithm that returns bins with a uniform width, chosen to cover the range of elements in `X` and reveal the underlying shape of the distribution. `histogram` displays the bins as rectangles such that the height of each rectangle indicates the number of elements in the bin.
`boxplot(x)`	`boxplot(x)` creates a box plot of the data in `x`. If `x` is a vector, `boxplot` plots one box. If `x` is a matrix, `boxplot` plots one box for each column of `x`.
`hold on`	`hold on` retains plots in the current axes so that new plots added to the axes do not delete existing plots

`if` *expression*
  statements
`elseif` *expression*
  statements
`else`
  statements
`end`
	`if` *expression*, *statements*, `end` evaluates an expression, and executes a group of statements when the expression is true. An expression is true when its result is nonempty and contains only nonzero elements (logical or real numeric). Otherwise, the expression is false.
	The `elseif` and `else` blocks are optional. The statements execute only if previous expressions in the `if...end` block are false. An `if` block can include multiple `elseif` blocks.