DBT-STAR COLLEGE SCHEME

Revised Practical Protocol Manual

Volume 3: MATHEMATICS

Since 1951

JAMAL MOHAMED COLLEGE (Autonomous)

College with Potential for Excellence

Accredited (3rd cycle) with "A" Grade by NAAC

DBT Star Scheme & DST-FIST Funded College

(Affiliated to Bharathidasan University)

TIRUCHIRAPPALLI-620 020

R PROGRAMMING

Introduction to R Programming

i History

R is an extension of the S-programming language, which was created by John Chambers at Bell Laboratories (formerly AT&T, now Lucent Technologies) in 1976. S was a premiere tool for statistical research, but it wasn't very feasible outside scholarly research.

In 1992, Ross Ihaka and Robert Gentleman created R at the University of Auckland, New Zealand, as a tool that their students could learn and use easily. Ihaka and Gentleman released the initial version in 1995, and a stable beta version was released in 2000. Since then, it is maintained by the R Development Core Team.

ii Background

R is a programming language and software environment for statistical computing and graphics supported by the R Foundation for Statistical Computing. R is an integrated suite of software facilities for data manipulation, calculation and graphical display. It includes

1. an effective data handling and storage facility,
2. a suite of operators for calculations on arrays, in particular matrices,
3. a large, coherent, integrated collection of intermediate tools for data analysis,
4. graphical facilities for data analysis and display either on-screen or on hard copy, and
5. a well-developed, simple and effective programming language which includes conditionals, loops, user-defined recursive functions and input and output facilities.

The term "environment" is intended to characterize it as a fully planned and coherent system, rather than an incremental accumulation of very specific and inflexible tools, as is frequently the case with other data analysis software. Many users think of R as a statistics system. It is preferable to think of it of an environment within which statistical techniques are implemented. R can be extended (easily) via packages. There are several packages supplied with the R distribution and many more are available through the CRAN family of Internet sites covering a very wide range of modern statistics.

The R language is widely used among statisticians and data miners for developing statistical software and data analysis. R is an implementation of the S programming language. R was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand, and is currently developed by the R Development Core Team, of which Chambers is a member. R is named partly after the first names of the first two R authors and partly as a play

on the name of S. R is a GNU project. The source code for the R software environment is written primarily in C, Fortran, and R. R is freely available under the GNU General Public License, and pre-compiled binary versions are provided for various operating systems.

R-Studio is an integrated development environment (IDE) for R. It includes a console, syntax-highlighting editor that supports direct code execution, as well as tools for plotting, history, debugging and workspace management. RStudio is available in open source and commercial editions and runs on the desktop (Windows, Mac, and Linux) or in a browser connected to RStudio Server or RStudio Server Pro (Debian/Ubuntu, RedHat/CentOS, and SUSE Linux). RStudio is a free and open-source integrated development environment (IDE) for R, a programming language for statistical computing and graphics. JJ Allaire, creator of the programming language ColdFusion, founded RStudio. Hadley Wickham is the Chief Scientist at RStudio. RStudio is available in two editions: RStudio Desktop, where the program is run locally as a regular desktop application; and RStudio Server, which allows accessing RStudio using a web browser while it is running on a remote Linux server. Pre-packaged distributions of RStudio Desktop are available for Windows, OS X, and Linux. RStudio is written in the C++ programming language and uses the Qt framework for its graphical user interface. Work on RStudio started at around December 2010, and the first public beta version (v0.92) was officially announced in February 2011.

iii Features of R Programming

- R has a massive community that works tirelessly to improve and add upon R's abilities. CRAN or Comprehensive R Archive Network has over 10,000 packages or extensions that can be used from producing high-definition graphics to creating interactive web-apps.
- R can perform complex mathematical and statistical operations on vectors, matrices, data frames, arrays, and other data objects of varying sizes.
- R is an interpreted language and does not need a compiler. It generates a machine-independent code that is easy to debug and is highly portable.
- R is a comprehensive programming language that supports object-oriented as well as procedural programming with generic and first-class functions.
- It supports matrix arithmetic.
- R can present data graphically. With static graphics, producing production quality visualizations and extended libraries providing interactive graphic capabilities, data visualization, and data representation becomes very easy. From concise charts to elaborate flow diagrams, all are well within R's repertoire.

- R can be used throughout the data analysis process. It helps to gather the data, to clean it, to investigate it, to model it, and finally, it helps you to compile the results in an eye-catching and easy to understand reports with R markdown. R can also help you build apps to show the results to the world.

- It can use distributed computing to process large datasets parallelly.

- R has packages that allow it to interact with multiple databases of different formats. It can also interact with various database management systems.

- R is compatible with many other programming languages like C, C++, Java, Python, etc.

iv Basic principles of R programming

R Data Types

There are fundamentally five data types in R.

1. Numeric data types : Numeric data consists of decimal values
2. Integer data type:

In R, there are two ways to create an integer variable.
- The first is to invoke the *as.integer()* function.
- The second is to creating an integer variable is to use 'the capital L'.

3. Complex data type:

The complex data type in R is for complex numbers or numbers with imaginary values.

4. Character data type:

The character data type is used to store strings in R. A character variable can be created in two ways in R.

- The first is to invoke the **as.character()** function.
- create a character variable is by using **inverted commas**.

5. Logical data type :

A logical variable can have two values either TRUE or FALSE. Logical are generally created when there is a comparison between variables.

v Simple calculations

Since the R environment can serve as an advanced calculator, it is worth noting this also allows for simple calculations. In the table below we show a few examples of such calculations where the first column gives a mathematical expression (calculation), the second gives the equivalent of this expression in R and finally in the third column we can find the result that is output from R.

Math.	R	Result
2+2	2+2	4
4/2	4/2	2
3·2−0.8	3*2^(-0.8)	1.723048
√22	sqrt(2)	1.414214
Ππ	Pi	3.141593
ln(2)	log(2)	0.6931472
log3(9)	log(9, base = 3)	2
e1.1	exp(1.1)	3.004166
cos(√0.9)	cos(sqrt(0.9))	0.5827536

vi Numerical Input

A first step in analysing numerical inputs is given by computing summary statistics of the data which, in this section, we can generally denote as x . For central tendency or spread statistics of a numerical input, we can use the following R built-in functions:

- mean calculates the mean of an input x;
- median calculates the median of an input x;
- var calculates the variance of an input x;
- sd calculates the standard deviation of an input x;
- IQR calculates the interquartile range of an input x;
- min calculates the minimum value of an input x;
- max calculates the maximum value of an input x;
- range returns a vector containing the minimum and maximum of all given arguments;
- summary returns a vector containing a mixture of the above functions (i.e. mean, median, first and third quartile, minimum, maximum).

vii Factor Input

If the data of interest is a factor with different categories or levels, then different summaries are more appropriate. For example, for a factor input we can extract counts and percentages to summarize the variable by using table. Using functions and data structures that will be described in the following chapters, below we create an example dataset with 90 observations of three different colors: 20 being Yellow, 10 being Green and 50 being Blue. We then apply the table function to it:

```
table(as.factor(c(rep("Yellow", 20), rep("Green", 10), rep("Blue", 50))))
##
##   Blue  Green Yellow
##     50     10     20
```

By doing so we obtain a frequency (count) table of the colors.

viii Lists

Lists can be extremely convenient to make text more readable or to take course notes during class. R Markdown allows to create different list structures as shown in the code below:

```
* You can create bullet points by using symbols such as *, +, or -.
+ simply add an indent or four preceding spaces to indent a list.
   + You can manipulate the number of spaces or indents to your liking.
      - Like this.
   * Here we go back to the first indent.
1. To make the list ordered, use numbers.
1. We can use one again to continue our ordered list.
2. Or we can add the next consecutive number.
```

which delivers the following list structure:

- create bullet points by using symbols such as *, +, or -.
- simply add an indent or four preceding spaces to indent a list.
- manipulate the number of spaces or indents to your liking.
- Like this.

Here we go back to the first indent.

1. To make the list ordered, use numbers.
2. We can use one again to continue our ordered list.

3. Or we can add the next consecutive number.

ix The most common “core” packages

- readr, for data import.
- ggplot2, for data visualization.
- dplyr, for data manipulation.
- tidyr, for data tidying.
- purrr, for functional programming.
- tibble, for tibbles, a modern re-imagining of dataframes.
- stringr, for string manipulation.
- forcats, for working with factors (categorical data).

To install packages in R we use the built-in install.packages() function.

Practical 1 : Develop a programme in R to create vectors

Aim: **Create a vectors as follows:**

1. **creating a vectors from 5 to 13.**
2. **creating a vectors from 6.6 to 12.6. Using R Programme**

Procedure:

Step 1: Write the R Programming code for the given vectors in R desktop.

Step 2: Run the Program.

Step 3: If any error occurs in your code then clear the errors and again run.

Step 4: Take the output as you like.

Results & Conclusion: Get the output and interpret the results.

Practical 2 : Develop a programme in R to create matrices

Aim: Create matrices.

1.
3 4 5
6 7 8
9 10 11
12 13 14

2.
9 18 12
11 10 4 Using R program

(You can create any order of matrices)

Procedure:

Step 1: Check the matrix of dimension.

Step 2: Write the R program codes to create given vectors in R desktop.

Step 3: Run the Program.

Step 4: If any error occurs in your code then clear the errors and again run.

Step 5: Take the output as you like.

Results & Conclusion: Get the output and interpret the results.

Practical 3 : Develop a programme in R using control statements

Aim: **Check value is less than or greater than 10 and Checks value is either positive, negative or zero**

1, 23, 45, -2, 0, -12, 11, -22, -10, 13, 7, 8, 3, 2, 9

(You can solve any large data set uploading format like excel sheet.)

Procedure:

Step 1: Verify the date is numerical values.

Step 2: Write the R programming codes to verify data is less than or greater than 10 and either positive, negative or zero in R desktop.

Step 3: Run the Program.

Step 4: If any error occurs in your code then clear the errors and again run.

Step 5: Take the output as you like.

Results & Conclusion: Get the output and interpret the results.

Practical 4 : Develop a programme in R to import spread sheet data

Aim: **R Programme to import spread sheet data into R.**

Procedure:

Step 1: Install the readxl package.

Step 2: Prepare your Excel File

Step 3: Import the Excel file into R.

Step 4: Run the Program.

Step 5: If any error occurs in your code then clear the errors and again run.

Step 6: Take the output as you like.

Results & Conclusion : Get the output and interpret the results.

Practical 5 : Develop a programme in R to calculate mean and median

Aim: **Calculate mean and median −21, −5, 2, 3, 4.2, 7, 8, 12, 18, 54 using R Programme.**

(You can solve any large data set uploading format like excel sheet.)

Procedure:

Step 1: create the given vector.

Step 2: Write the R program codes to find the value of mean and median

Step 3: Run the Program.

Step 4: If any error occurs in your code then clear the errors and again run.

Step 5: Take the output as you like.

Results & Conclusion: Get the output and interpret the results.

Practical 6 : Develop a programme in R to calculate standard deviation

Aim: **For the frequency distribution:**

x	2	3	4	5	6	7
f	**4**	**9**	**16**	**14**	**11**	**6**

Find the standard deviation using R program.

(You can solve any large data set uploading format like ex-sheet.)

Procedure:

Step 1: If possible you can simplify the vectors.

Step 2: Write the R programming codes to find the value of the standard deviation

Step 3: Run the Program.

Step 4: If any error occurs in your code then clear the errors and again run.

Step 5: Take the output as you like.

Results & Conclusion : Get the output and interpret the results.

Practical 7 : Develop a programme in R to present the data in tabulation and graphical representation

Aim: Develop a test of 50 marks is administered on a class of 40 students and the marks obtained by these students are as listed below

35, 40, 22, 32, 41, 18, 40, 36, 29, 24, 28,31, 11, 39,
12, 46, 28, 30, 31, 42, 15, 19, 24, 39, 49, 38, 23, 46,
11, 32, 33, 44, 22, 21, 34, 23, 16, 37, 32, 40,

Create a R Programme to present the data in tabulation and graphical representation.

(You can solve any large data set uploading format like excelsheet.)

Procedure:

Step 1: verify the given data is numerical data.

Step 2: Write the R progamming codes to formatting tabulation and graphical representation(like pie chart, Histogram, line graph, scatterplots).

Step 3: Run the Program.

Step 4: If any error occurs in your code then clear the errors and again run.

Step 5: Take the output as you like.

Results & Conclusion : Get the output and interpret the results.

Practical 8 : Develop a programme in R using chi-square test

Aim: Chi-square test.

Is gender independent of education level? A random sample of 395 people were surveyed and each person was asked to report the highest education level they obtained. The data that resulted from the survey is summarized in the following table:

	High School	Bachelors	Masters	Ph.d.	Total
Female	**60**	**54**	**46**	**41**	**201**
Male	**40**	**44**	**53**	**57**	**194**
Total	**100**	**98**	**99**	**98**	**395**

(You can solve any large data set uploading format like excel sheet.)

Procedure:

Step 1: verify the given data is suitable for chi-square test.

Step 2: Write the R program codes to solve the given Data.

Step 3: Run the Program.

Step 4: If any error occurs in your code then clear the errors and again run.

Step 5: Take the output as you like.

Results & Conclusion: Get the output and interpret the results.

Practical 9 : Develop a programme in R using student's t test

Aim: **Develop a programme in R.**

Calculate a paired t test by hand for the following data:

Subject	Score 1	Score 2
1	30	20
2	3	13
3	20	13
4	12	20
5	15	29
6	17	23

(You can solve any large data set uploading format like excel sheet.)

Procedure:

Step 1: verify the given data is student's t test.

Step 2: Write the R program codes to plot the given set of graphs in one set of axes.

Step 3: Run the Program.

Step 4: If any error occurs in your code then clear the errors and again run.

Step 5: Take the output as you like.

Results & Conclusion: Get the output and interpret the results.

Practical 10 : Develop a programme in R to calculate one way ANOVA

Aim:

An education researcher is comparing four different algebra curricula. Eighth grade students are randomly assigned to one of the four groups. Their state achievement test scores are compared at the end of the year. Use the appropriate statistical procedure to determine whether the curricula differ with respect to math achievement. An alpha criterion of .05 should be used for the test.

	N	Mean	SD
Curriculum 1	**50**	**170.5**	**14.5**
Curriculum 2	**50**	**168.3**	**12.8**
Curriculum 3	**50**	**167.6**	**17.7**
Curriculum 4	**50**	**172.8**	**16.8**

(You can solve any large data set uploading format like excel sheet.)

Procedure:

Step 1: Verify the average within-group variance; it is not sensitive to group mean differences.

Step 2: Verify estimating the variance is sensitive to group mean differences.

Step 3: Write the R programming code for the given equation in R desktop.

Step 4: Run the Program.

Step 5: If any error occurs in your code then clear the errors and again run.

Step 6: Take the output as you like.

Results & Conclusion : Get the output and interpret the results.

MATHEMATICA

Introduction to Mathematica

i. History

In 1979–1981, Stephen Wolfram constructed SMP (Symbolic Manipulation Program), the first modern computer algebra system (SMP was essentially Version Zero of Mathematica). In 1986–1988, Stephen Wolfram developed the first version of Mathematica. The concept of Mathematica was a single system that could handle many specific problems (e.g., symbolic, numerical, algebraic, graphical). In 1987, Wolfram founded a company, Wolfram Research, which continues to extend Mathematica. In 1991, the second version of Mathematica appears with more built-in functions, MathLink protocol for interprocess and network communication, sound support, notebook front end. In 1996, the third version introduced interactive mathematical typeset- ting system, exporting HTML, hyperlinks, and many other functions. In 1999, the fourth version of Mathematica appears with important enhancements in speed and efficiency in numerical calculation, publishing documents in a variety of formats, and enhancements to many built-in functions. In 2003, in the fifth version of Mathematica the core coding was improved and the horizons of Mathematica are more extended (e.g., in numerical linear algebra, in numerical solutions for differential

ii. Basic Features

- Symbolic, numerical, acoustic, graphical, parallel computations
- Static and Dynamic computations, Extensibility and elegance,
- Available for MS Windows, Linux, UNIX, Mac OS operating systems,
- Powerful and logical language,
- Extensive library of mathematical functions and specialized packages,
- An interactive front end with notebook interface,
- Interactive mathematical typesetting system,
- Free resources, The Mathematica Learning Center, Wolfram Demon- strations Project, Wolfram Information Center.

iii. Mathematica Language

Mathematica language is a very powerful programming language based on systems of transformation rules, functional, procedural, and object- oriented programming techniques. This distinguishes it from traditional programming languages. It supports a large collection of data structures or Mathematica objects (functions, sequences, sets, lists, arrays, tables, matrices, vectors, etc.) and operations on these objects (type-testing, selection, composition, etc.). The library can be enlarged with custom programs and packages.

iv. Basic Principles

Symbol in Mathematica, symb, refers to a symbol with the specified name, e.g., expressions, functions, objects, optional values, results, argument names.

A **name** of symbol, name, is a combination of letters, digits, or certain special characters, not beginning with a digit, e.g., a12new. Once defined, a symbol retains its value until it is changed, cleared, or removed.

Expression, expr, is a symbol that represents an ordinary Mathematica expression in readable form. The head of expr can be obtained with Head[expr]. The structure and various forms of expr can be analyzed with TreeForm, FullForm[expr], InputForm[expr], e.g., l1={5, 1/2, 9.1, 2+3*I, x, {A,B}, a+b, a*b} {Head /@ l1,FullForm[l1],InputForm[l1],TreeForm[l1], TraditionalForm[l1]}

Mathematica is case sensitive, there is a difference between lowercase and uppercase letters, e.g., Sin[Pi] and sin[Pi] are different. All Mathematica functions begin with a capital letter. Some functions (e.g., PlotPoints) use more than one capital. To avoid conflicts, it is best to begin with a lower-case letter for all user-defined symbols.

The **result** of each calculation is displayed, but it can be suppressed by using a semicolon (;), e.g., Plot[Sin[x],x,0,2*Pi]; a=9; b=3; c=a*b

v. Constants

Types of numbers: integer, rational, real, complex, root, e.g.,

{-5,5/6,-2.3^-4,ScientificForm [-2.3^-4],3-4*1,Root[#^2+#+1&,2]}

Mathematical constants: symbols for definitions of selected mathematical constants, e.g. , Catalan, Degree, E, EulerGamma, I, Pi, Infinity, GoldenRatio, e.g. , {60Degree//N, N [E, 30]}.

Scientific constants: valuable tools for scientists and engineers in Physics and Chemistry can be applied with the packages Units and Physical Constants.

vi. Special constants

Mathematica uses predefined symbols to represent built-in mathematical constants

- Pi or π is the ratio of the circumference of a circle to its diameter.

- E or e is the base of the natural logarithm.

Both pi and E are treated symbolically and do not have values, as such. However, they may be approximated to any degree of precision.

vii. Functions

Two classes of functions: pure functions and functions defined in terms of a variable (predefined and user-defined functions).

Pure functions are defined without a reference to any specific variable. The arguments are labeled #1, #2, and an ampersand & is used at the end of definition.

```
f:=Sin[#1]&;  g:Sin[#1^2+#2^2]&

{f[x], f[Pi], g[x,y], g[Pi,Pi]}
```

Predefined functions. Most of the mathematical functions are predefined.

Special functions: Mathematica includes all the common special functions of mathematical physics. We will discuss some of the more commonly used functions.

The names of mathematical functions are complete English words or the traditional abbreviations (for a few very common functions), e.g., Conjugate, Mod. Person's name mathematical functions have names of the form Person Symbol, for example, the Legendre polynomials $P_n(x)$, Legendre $P[n, x]$.

viii. Basic Arithmetic Operations

As we have seen, basic arithmetic operations such as addition are performed by inserting an operation symbol between two numbers. Thus the sum of 3 and 5 would be obtained by typing $3 + 5$. However, in more advanced applications it is sometimes useful to represent these operations as functions. Towards this end Mathematica includes the following:

- Plus $[a, b, \ldots]$ computes the sum of $a, b, ..$ Plus $[a, b]$ is equivalent to $a + b$.
- Times $[a, b, \ldots]$computes the product of $a, b, ...$Times $[a, b]$ is equivalent to $a * b$.
- Subtract $[a, b, \ldots]$computes the difference of a and b. Only two arguments are permitted. Subtract $[a, b]$ is equivalent to $a - b$.
- Divide $[a, b, \ldots]$ computes the quotient Of a and b. Only two arguments are permitted.
- Divide $[a, b]$ is equivalent to a/b.
- Minus $[a]$ produces the additive inverse (negative) of a. Minus $[a]$ is equivalent to $-a$.

- Power $[a, b]$ computes a^b'. Power $[a, b, c\ ]$ produces a^b, etc.

ix. Strings

A string is an (ordered) sequence of characters. Strings have no numerical value and are often used as labels for tables, graphs, and other displays.

In Mathematica, a string is enclosed within quotation marks. Thus • $abcde$ • is a string of five characters. Do not confuse " $abcde'$ with $abcde$, as the latter is not a string.

Mathematica comes equipped with a number of string manipulation commands.

- StringLength $[string]$ returns the number of characters in string.
- Stringioin , [string1, string2,…] or string1<> string2<>… . concatenates two or more strings to form a new string whose length is equal to the sum of the individual string lengths.
- StringReverse [string] reverses the characters in string.
- StringDrop eliminates characters from a string. There are five forms of this command.
- StringDrop [string, n] returns string with its first n characters dropped.
- StringDrop [string, -n] returns string with its last n characters dropped.
- StringDrop [string, {n}] returns string with its n^{th} character dropped.
- StringDrop [string,{-n}] returns string with the n^{th} character from the end dropped.
- StringDrop [string, $\{m, n\}$] returns string with characters m through n dropped.

x. Introduction to Graphing

The graph of a function offers tremendous insight into the function's behavior and can be of great value in the solution of problems. Mathematica offers some very powerful graphics commands which are remarkably easy to implement. Although there is a vast array of options available for customization of output, in this section we shall deal only with the most rudimentary forms using Mathematica's defaults.

The Plot command plots two-dimensional graphs.

- Plot$[f[x], \{x, xmin, xmax\}]$ plots a two-dimensional graph of the function $f(x)$ on the Interval $xmin \leq x \leq xmax$.

- Plot[{$f[x]$},{$g[x]$},{$x, xmin, xmax$}] plots two functions on one set of axes. This extends in a natural way to three or more functions.

Practical 1 : Solving higher degree equations

Aim:

To find the roots of the higher degree equation $'x^8 + 5x^6 - 3x^5 + 2 = 0'$

(You can solve any transcendental equations using this method)

Procedure:

Step 1: Verify it is a polynomial equation or not.

Step 2: Verify it has an integer power of rational power.

Step 3: Write the Mathematica code for the given equation in Mathematica desktop.

Step 4: Run the Program.

Step 5: If any error occurs in your code then clear the errors and again run.

Step 6: Take the output as you like.

Results & Conclusion: Get the output and interpret the results.

Practical 2 : Solving system of equations by matrix method and find the eigen values and eigen vectors of a matrix of order 4 by 4 or higher order

Aim:

To solve the system of equations

$a + b + c + d = 10, 2a + b + c + d = 11, a + 2b + c + d = 12, a + b + 2c + d = 13$.

by matrix method and find the eigen values and eigen vectors of a matrix of order 4X4.

(You can solve any system of equations by matrix methods of order more than 4 X 4)

Procedure:

Step 1: Verify the system of equations are linear or not.

Step 2: Check the matrix of order.

Step 3: Write the Mathematica codes to solve given system of equations and find the eigen values and eigen vectors in Mathematica desktop.

Step 4: Run the Program.

Step 5: If any error occurs in your code then clear the errors and again run.

Step 6: Take the output as you like.

Results & Conclusion: Get the output and interpret the results.

Practical 3 : Solving system of non-linear equations

Aim:

To solve the system of non linear equations

$$x^2 + y^2 = 5, \qquad 6\,x^2 - y^2 = 2.$$

(You can solve any system of non linear equations of any order.)

Procedure:

Step 1: Verify the system of equations are non linear.

Step 2: Write the Mathematica codes to solve given system of equations and find the eigen values and eigen vectors in Mathematica desktop.

Step 3: Run the Program.

Step 4: If any error occurs in your code then clear the errors and again run.

Step 5: Take the output as you like.

Results & Conclusion: Get the output and interpret the results.

Practical 4 : Finding the differentiation of different functions of second and third derivatives

Aim:

To find the second the third derivative of the function

$$\sin\left(\frac{\sqrt{x^2 - 5x + 3}}{\log(2x + 5)}\right)$$

(You can find any higher order derivative for any difficult functions.)

Procedure:

Step 1: If possible you can simplify the function.

Step 2: Write the Mathematica codes to find the second the third derivative of the function

Step 3: Run the Program.

Step 4: If any error occurs in your code then clear the errors and again run.

Step 5: Take the output as you like.

Results & Conclusion: Get the output and interpret the results.

Practical 5 : Finding the Integration of different functions with limits

Aim:

To find the

$$\int_0^{\infty} e^{-x^2}\, dx$$

(You can find any higher order derivative for any difficult functions.)

Procedure:

Step 1: If possible you can simplify the function.

Step 2: Write the Mathematica codes to find the value of the integral of the given function

Step 3: Run the Program.

Step 4: If any error occurs in your code then clear the errors and again run.

Step 5: Take the output as you like.

Results & Conclusion: Get the output and interpret the results.

Practical 6 : Evaluation of double integrals and triple integrals

Aim:

To find the

$$\int_0^1 \int_0^{1-x} \int_0^{1-x-y} 2\,(x^2 + y^2 + z^2)\, dxdydz$$

(You can find double integral with given limit for any functions.)

Procedure:

Step 1: If possible you can simplify the function.

Step 2: Write the Mathematica codes to find the value of the integral of the given function

Step 3: Run the Program.

Step 4: If any error occurs in your code then clear the errors and again run.

Step 5: Take the output as you like.

Results & Conclusion: Get the output and interpret the results.

Practical 7 : Solving ordinary differential equations with initial condition

Aim:

To solve the ordinary differential equations with initial condition.

$$\frac{d^2y}{dx^2} + 4y = 2x, \qquad y(0) = 1$$

(You can solve any ordinary differential equations of higher order.)

Procedure:

Step 1: verify the given function is ordinary differential equation with initial condition.

Step 2: Write the Mathematica codes to solve the given differential equation.

Step 3: Run the Program.

Step 4: If any error occurs in your code then clear the errors and again run.

Step 5: Take the output as you like.

Results & Conclusion: Get the output and interpret the results.

Practical 8 : Solving system of ordinary differential equations

Aim:

To solve the simultaneous ordinary differential equations with initial conditions.

$$\frac{d^2y}{dt^2} + 4y = 2t, \qquad \frac{d^2x}{dt^2} - x = \sin t, \qquad y(0) = x(0) = 1$$

(You can solve any* simultaneous *ordinary differential equations of higher order.)

Procedure:

Step 1: verify the given function is simultaneous ordinary differential equations with initial condition.

Step 2: Write the Mathematica codes to solve the given simultaneous differential equations.

Step 3: Run the Program.

Step 4: If any error occurs in your code then clear the errors and again run.

Step 5: Take the output as you like.

Results & Conclusion: Get the output and interpret the results.

Practical 9 : Creating and plotting 2-D and 3-D graphs

Aim:

To Plot the functions $y = x^2$ and $y = 2x + 10,\ -5 \leq x \leq 5,$ on the same set of axes.

(You can plot any number of graphs for any interval in one set of axes.)

Procedure:

Step 1: verify how many graphs are given to plot.

Step 2: Plot given set of graphs in one set of axes to compare these.

Step 3: Write the Mathematica codes to plot the given set of graphs in one set of axes.

Step 4: Run the Program.

Step 5: If any error occurs in your code then clear the errors and again run.

Step 6: Take the output as you like.

Results & Conclusion: Get the output and interpret the results.

Practical 10 : Solving Linear programming problems

Aim:

To solve the following set of linear programming problem:

$$Max\ z = 40x + 100y$$

subject to the conditions

$$12x + 6y \leq 3000$$
$$4x + 10y \leq 2000$$
$$2x + 3y \leq 900\ ,$$
$$x, y \geq 0$$

(You can solve minimization type of linear programming problems.)

Procedure:

Step 1: Verify given linear programming is linear or not.

Step 2: Verify it is maximize of minimize.

Step 3: Write the Mathematica code for the given equation in Mathematica desktop.

Step 4: Run the Program.

Step 5: If any error occurs in your code then clear the errors and again run.

Step 6: Take the output as you like.

Results & Conclusion: Get the output and interpret the results.

MAPLE

Introduction to Maple

i. History

We begin with a brief history of Maple, from a research project at a university to a leading position in education, research, and industry. The first concept of Maple and initial versions were developed by the Symbolic Computation Group at the University of Waterloo in the early 1980s. In 1988, the new Canadian company Waterloo Maple Inc., was created to commercialize the software. While the development of Maple was done mainly in research labs at Waterloo University and at the University of Western Ontario, with important contribution from worldwide research groups in other universities. In 1990, the first graphical user interface was introduced for Windows in version V. In 2003, a Java "standard" user interface was introduced in version 9. In 2005, Maple version 10 comes with a "document mode" as part of the user interface. In 2007, Maple version 11 is introduced, it comes with an improved smart document environment to facilitate the user-interface learning curve. This tool integrates in an optimal way, all different sources and types of related information to the problem that the user is solving in that moment. This version includes more mathematical tools for modelling and problem analysis.

Maple is a symbolic and numeric computing environment as well as a multi-paradigm programming language. It covers several areas of technical computing, such as symbolic mathematics, numerical analysis, data processing, visualization, and others. A toolbox, MapleSim, adds functionality for multidomain physical modeling and code generation.

Maple's capacity for symbolic computing include those of a general-purpose computer algebra system. Forinstance, it can manipulate mathematical expressions and find symbolic solutions to certain problems, such as those arising from ordinary and partial differential equations.

Maple is developed commercially by the Canadian software company Maplesoft. The name 'Maple' is a reference to the software's Canadian heritage.

ii Basic Features

- *Fast symbolic, numerical* computation, and interactive visualization,
- *Easy to use*, help can be found within the program or on the Internet,
- *Extensibility*, the system can easily incorporate new user defined capabilities to compute specific purpose applications,
- *Accessible* to large numbers of students and researchers,
- *Available* for almost all operating systems (MS Windows, Linux. Unix, Mac OS),
- *Powerful programming language*, intuitive syntax, easy debugging,
- *Extensive library* of mathematical functions and specialized packages,
- *Two forms* of interactive interfaces: a command-line and a graphic environment,
- *Free resources*, collaborative character of development, Maple Application Center, Teacher Resource Center, Student Help Center, Maple Community,
- *Understandable*, open-source software development path.

iii Basic Concepts

We type the Maple function to the right of the prompt symbol >, and at the end of the function we place a semicolon1, and then press Enter (or Shift+Enter to continue the function onto the next line). Maple evaluates the function, displays the result, and inserts a new prompt.

```
>?introduction
> evalf(gamma,40);
> solve(11*x^3-9*x+17=0, x);
> sort(expand((y+1)^(10)));
> plot({4*sin(2*x),cos(2*x)^2},x=0..2*Pi);
> plot3d(cos(x^2+y^2),x=0..Pi,y=0..Pi);
```

Above (typing line by line), the first line gives you introductory information about Maple, the second line returns a 40-digit approximation of Euler's constant γ, the third line solves the equation 11x3 −9x+17=0 for x, the fourth line expresses (y +1)10 in a polynomial form, the fifth line plots the functions 4 sin(2x) and cos2(2x) on the interval [0, 2π], and the last line plots the function cos(x2+y2) on the rectangle [0,π]×[0,π].

iv Basic Principles

Arithmetic operators: + - * / ^ mod.

Logic operators: and, or, xor, implies, not.

Relation operators: <, <=, >, >=, =, <>.

A variable name, var, is a combination of letters, digits, or the underline symbol ( ), beginning with a letter, e.g., a12 new.

Abbreviations for the longer Maple functions or any expressions: alias, e.g., alias(H=Heaviside); diff(H(t),t); to remove this abbreviation, alias(H=H);

Maple is case sensitive, there is a difference between lowercase and uppercase letters, e.g., evalf(Pi) and evalf(pi).

Various reserved keywords, symbols, names, and functions, these words cannot be used as variable names, e.g., operator keywords, additional

language keywords, global names that start with ( ) (see ?reserved, ?ininames, ?inifncs, ?names).

The assignment/unassignment operators: a variable can be “free” (with no assigned value) or can be assigned any value (symbolic, numeric) by the assignment operators a:=b or assign(a=b). To unassign (clear) an assigned variable (see ?:= and ?'): x:='x', evaln(x), or unassign('x').

The difference between the operators (:=) and (=). The operator var:=expr is used to assign expr to the variable var, and the operator A=B — to indicate equality (not assignment) between the left- and the right-hand sides (see ?rhs), e.g., Equation:=A=B; Equation; rhs(Equation); lhs(Equation);

The range operator (..), an expression of type range expr1..expr2, e.g., a[i] $ i=1..9; plot(sin(x),x=-Pi..Pi);

Statements, stats, are input instructions from the keyboard that are executed by Maple (e.g., break, by, do, end, for, function, if, proc, restart, return, save, while, not).

The new worksheet (or the new problem) it is best to begin with the statement restart for cleaning Maple' s memory. All examples and problems in the book assume that they begin with restart.

```
restart: with(LinearAlgebra): d:=10;
for i from 1 to 7 do
n:=2^i; A:=RandomMatrix(n,n,generator=-d..d):
t:=time(ReducedRowEchelonForm(A)); print(i,n,t); od:
```

The statement separators semicolon (;) and colon (:). The result of a statement followed with a semicolon (;) will be displayed, and it will not be displayed if it is followed by a colon (:), e.g., plot(sin(x),x=0..Pi); plot(sin(x),x=0..Pi): An expression, expr, is a valid statement, and is formed as a combination of constants, variables, operators and functions. Data types, every expression is represented as a tree structure in which each node (and leaf) has a particular data type. For the analysis of any node and branch, the functions type, what type, nops, op can be used. A boolean expression, bexpr, is formed with the logical operators and the relation operators.

An equation, eq, is represented using the binary operator (=), and has two operands, the left-hand side, lhs, and the right-hand side, rhs. Inequalities, ineq, are represented using the relation operators and have two operands, the left-hand side, lhs, and the right-hand side, rhs.
A string, str, is a sequence of characters having no value other than itself, cannot be assigned to, and will always evaluate to itself. For instance, x:="string"; and sqrt(x); is an invalid function. Names and strings can be used with the convert and printf functions. Incorrect response. If you get no response or an incorrect response you may have entered or executed the function incorrectly. Do correct the function or interrupt the computation (the stop button in the Tool Bar menu). Maple is sensitive to types of brackets and quotes.

v Types of brackets:

Parentheses (expr), for grouping expressions, (x+9)*3, for delimiting the arguments of functions, sin(x).

Square brackets [expr], for constructing lists, [a,b,c], vectors, matrices, arrays.

Curly brackets {expr}, for constructing sets, {a,b,c}.

vi Types of quotes:

Forward-quotes 'expr', to delay evaluation of expression, 'x+9+1', to clear variables, x:='x'; Back-quotes `expr`, to form a symbol or a name, `the name:=7`; k:=5; print(`the value of k is`,k);

Double quotes "expr", to create strings, and a single double quote ", to delimit strings.

Previous results (during a session) can be referred with symbols % (the last result), %% (the next-to-last result), %%...%, k times, (the k-th previous result), a+b; %^ 2; %%^ 2;

Comments can be included with the sharp sign # and all characters following it up to the end of a line. Also the text can be inserted with Insert →Text.

Maple source code can be viewed for most of the functions, general and specialized (package functions). interface(verboseproc=2); print(factor); print(`plots/arrow`);

vii Constants

Types of numbers: integer, rational, real, complex, root, e.g., -55, 5/6, 3.4, -2.3e4, Float(23,-45), 3-4*I, Complex(2/3,3); RootOf(Z^3-2,index=1);

Predefined constants: symbols for definitions of commonly used mathematical constants, true, false, gamma, Pi, I, infinity, Catalan, FAIL, exp(1) (see ?ininames, ?constants).

An angle symbolically has dimension 1. Maple knows many units of angle (see? Units / angle), e.g., convert(30*degrees,radians); convert(30,units,degrees,radians);

The packages ScientificConstants and Units (for ver ≥7) provide valuable tools for scientists and engineers in Physics and Chemistry (see ?ScientificConstants, ?Units).

viii Functions

Functions or function expressions have the form f(x) or expr(args) and represent a function call, or application of a function (or procedure) to arguments (args). Active functions (beginning with a lowercase letter) are used for computing, e.g., diff, int, limit. Inert functions (beginning with a capital letter) are used for showing steps in the problem-solving process; e.g., Diff, Int, Limit. Two classes of functions: the library functions (predefined functions) and user-defined functions. Predefined functions: most of the well known functions are predefined by Maple and they are known to some Maple functions (diff, evalc, evalf, expand, series, simplify). In addition, numerous special functions are defined (see ?FunctionAdvisor). We will

discuss some of the more commonly used functions. Elementary trascendental functions: the exponential function, the natural logarithm, the general logarithm, the common logarithm, the trigonometric and hyperbolic functions and their inverses.

exp ln log log[b] log10 ilog2

sin cos tan cot sec csc

sinh cosh tanh coth sech csch

arcsin arccos arctan arccot arcsec arccsc

arcsinh arccosh arctanh arccoth arcsech arccsch

exp(x^2); evalf(ln(exp(1)^3)); evalf(tan(3*Pi/4));

evalf(arccos(1/2)); evalf(tanh(1));

ix Procedures and Modules

In Maple language there are two forms of modularity: procedures and modules.

A procedure (see ?procedure) is a block of statements which one needs to use repeatedly. A procedure can be used to define a function (if the function is too complicated to write by using the arrow operator), to create a matrix, a graph, a logical value, etc.

proc(args) local v1; global v2; options ops; stats; end proc;
proc(args) local v1; global v2; options ops; stats; end;

Here args is a sequence of arguments, v1 and v2 are the names of local and global variables, ops are special options (see ?options), and stats are statements that are realized inside the procedure.

x Control Structures

In Maple language there are essentially two control structures: the selection structure if and the repetition structure for,

if cond1 then expr1 else expr2 end if;

if cond1 then expr1 elif cond2 then expr2 else expr3 end if;

for i from i1 by step to i2 do stats end do;

for i from i1 by step to i2 while cond1 do stats end do;

for i in expr1 do stats end do; for i in expr1 do stats od;

for i in expr1 while expr2 do stats end do;

where cond1 and cond2 are conditions, expr1, expr2 are expressions, stats are statements, i, i1, i2 are, respectively, the loop variable, the initial and the last values of i. These operators can be nested. The operators break, next, while inside the loops are used for breaking out of a loop, to proceed directly to the next iteration, or for imposing an additional condition. The operators end if and fi, end do and od are equivalent.

xi Complex Algebra

Maple and Mathematica perform complex arithmetic automatically, all operations are performed by assuming that the basic number system is the complex field C. In both systems, the imaginary unit i of the complex number x+y*I is denoted by I. Complex numbers and variables

abs(z); Re(z); Im(z); conjugate(z); argument(z); evalc(z); signum(z); csgn(z); polar(z); polar(r,theta); convert(z,polar); with(RandomTools): enerate(complex(integer(range=a..b))); abs, Re,Im, the absolute value and the real and imaginary parts, conjugate, argument, evalc, the complex conjugate, the complex argument, and complex evaluation function, signum, csgn, the sign of a real or complex number and the sign function for complex expressions, polar, convert,polar, polar representation of complex numbers and rewriting an expression in polar form, Generate,complex (of the RandomTools package), generating pseudorandom complex numbers.

```
z1:=5+I*7; abs(z1); Re(z1); Im(z1); conjugate(z1),
argument(z1); signum(z1); csgn(z1); polar(z1);
z2:=x+I*y; evalc(abs(z2)); evalc(Re(z2));
assume(x,real); assume(y,real); about(x,y); Re(z2); Im(z2);
unassign('x','y'); about(x,y); evalc(signum(z2));
evalc(polar(r,theta)); map(evalc,convert(z2, polar));
expand((1-I)^4); evalc(sqrt(-9)); expand((3+I)/(4-I));
with(RandomTools): Generate(list(complex(float(range=1..9)),9));
interface(showassumed=0): assume(k,complex);
f:=x->exp(-(Re(k)+I*Im(k))*(a+I*c*t)); f(x);
simplify(evalc(convert(f(x),polar)));
```

xii Complex Functions and Derivatives

Analytic and harmonic functions, multivalued “functions” :

```
f:=z->expr; diff(f(z),z); F:=evalc(f(x+I*y)); limit(f(z),z=z0);
u:=(x,y)->evalc(Re(F));v:=(x,y)->evalc(Im(F)); w:=(x,y)->expr;
w1:=unapply(u(x,y)+I*v(x,y),x,y); limit(w(x,y),{x=x0,y=y0});
```

xiii Numerical solutions of nonlinear equations and systems

```
Digits:=n;      evalf(solve(Eq,var));          evalf(root(x,n));
fsolve(Eq,var,ops);                            fsolve(Eq,var=a..b,ops);
fsolve(Eq,var,complex);                        fsolve(f,var=a..b,ops);
with(RootFinding);                             (f); NextZero(f(x),x0,ops);
Isolate([f1,f2],[x1,x2]);                      Homotopy([f1,f2],ops);
use RealDomain in solve(eq,var); end use;
```

fsolve, solving equations using iterative methods (the Newton methods) with options (see ?fsolve[details]),
fsolve(f,var=a..b,ops), for solving an equation defined as a procedure f,

NextZero, Isolate, Homotopy, functions of the RootFinding package, finding *next real* zero of a function f, isolating the real roots of a univariate polynomial or polynomial system, finding numerical approximations to roots of systems of polynomial equations, *root*, finding n-th root of an algebraic expression, the functions of the RealDomain package, performing computations under the assumption that the basic number system is the field of real numbers R.

```
Digits:=trunc(evalhf(Digits));          F:=proc(x) sin(x)^2-1 end;
fsolve(F(x)=0,x=0..Pi/2);       fsolve(sin(x)^2-1,x=0..Pi/2);
fsolve(x->F(x),0..Pi/2);
evalf(root(5,3));       evalf(root[3](5));       evalf(solve(sin(x)^2-1,x));
with(RootFinding);    NextZero(x->sin(x),    evalf(Pi/10));
Isolate(x^2-2*x+1);    Isolate([x^2-2*x+1,y^2+3*y-5],[x,y]);
Homotopy([x^2+x+1,y^2-y-1]);       use RealDomain in solve(x^4-1,x);
end use;        with(RealDomain):    fsolve(x^4-1,x,complex);
```

xiv Operations on Functions

Let us consider the most important results describing various operations on functions in both systems.

```
(f1@f2)(x); ((f1@f2)@f3)(x); (f1@f2@...@fn)(x); (f@@n)(x);
```

f1:=(f::function)->apply(op(0,f),x); f1(f2());

apply(f,expr); apply(f,x1,...,xn); apply(f,[x1,...,xn]);

map(f,expr); map(f,[x1,...,xn]); map2(f,g,[x1,...,xn]);

map(x->expr,expr); map(x->expr,[x1,...,xn]);

The Maple composition operator @,

Applying functions repeatedly, the repeated composition operator @@, Defining a function which takes a function name as an argument, ->, apply, Applying functions to other expressions and objects (e.g., lists), apply, Applying functions to each elements of other expressions and objects, map, map2, Functions without names (or anonymous functions), ->, map.

n:=5; (f@g)(x); ((f@g)@h)(x); (f@@n)(x);

f1:=(f::function)->apply(op(0,f),x+a)+apply(op(0,f),x-a);

f1(F()); apply(g,x^2+y^2); apply(g,x^2,y^2);

apply(g,[x^2,y^2]); map(g,x^2+y^2); map(g,[x^2,y^2]);

map2(f,g,[x^2,y^2]); map(x->x^2,x+y);

l1:=map(x->x^2,['x||i'$'i'=1..9]); convert(l1,`+`);

xv Inverse functions

The inverse function tables eval(invfunc), eval(Invfunc), the invfunc, Invfunc, unprotect functions:

eval(invfunc); eval(Invfunc); invfunc[sin]; (sin@@(-1))(1);

unprotect('invfunc'); invfunc[f]:=g; simplify((f@@(-1))@@n);

xvi Matrices

In matrix theory, a matrix is a rectangular table (rows and columns) of elements (numbers or symbols) and for which certain axioms of algebra hold. Matrices are used for describing equations and linear transformations, analyzing data that depend on multiple parameters and more. In our work we consider matrices whose elements are symbols and real and complex numbers. In symbolic mathematics, there exist differences in matrix representations, similar to the concept of a vector.

In Maple, a matrix is a mathematical object, but also a matrix can be defined as a data object represented as a two-dimensional array, and these objects are different. For example, one-column matrices and vectors, matrices and two-dimensional arrays are different objects in Maple.

xvii Matrix representations and components

with(LinearAlgebra): Matrix([[a11,...,a1m],...,[an1,...anm]]);
<<a11|...|a1m>,...,<an1|...|anm>>; map(x->f(x),M);
<<a11,...,an1>|...|<a1m,...,anm>>; Matrix(n,m,(i,j)->f(i,j));
Matrix(n,m,ops); Matrix(n,m,fill=c); Matrix(n,m,symbol=a);
with(Student[LinearAlgebra]): A:=MatrixBuilder(); M[i,j];
M[i,1..-1];M[1..-1,j];M[a..b,c..d]; SubMatrix(M,[a..b],[c..d]);
Dimension(M); ColumnDimension(M); RowDimension(M);

Matrix([[a11,..., a1m],...,[an1,...anm]]), constructing a matrix row-byrow (as a list of lists), <<a11|...|a1m>,...>>, <<a11,...,an1>|...>>, constructing a matrix, respectively row-by-row and column-by-column (as vectors), these forms are sufficiently slow for large matrices, *MatrixBuilder* (of the package Student[LinearAlgebra]), an interactive construction of a matrix (up to 5 × 5),

Matrix(n,m,(i,j)->f(i,j)), map(x->f(x),M), constructing a matrix n × m with elements that are defined, respectively, by a function f(i, j) and a function f(x) applied to each element of matrix M (for more details see Chapter 1),

Matrix(n,m,fill=c), Matrix(n,m,symbol=a), constructing a matrix n × m where, respectively, each element is c and the symbolic elements are aij. In general form, matrices can be constructed defining various options in the function Matrix (for more details see ? Matrix), *M[i,j]*, M[i,1..-1], M[1..-1,j], M[a..b,c..d], extracting elements, rows, columns, and submatrices of the matrix M,

Dimension, RowDimension, ColumnDimension, determining the dimension, row dimension, and column dimension of a matrix.

xviii Eigenvalues and Eigenvectors

Eigenvectors1 of a linear transformation T are vectors (_=0) that satisfy the equation Tx = λx for a scalar λ, called eigenvalue of T, that corresponds to the eigenvector x. An eigenspace of

a linear transformation for a particular eigenvalue is a space formed by all the eigenvectors associated with this eigenvalue.

A linear transformation can be considered as an operation on vectors that usually changes its magnitudes and directions. The direction of the eigenvector of T does not change (for positive eigenvalues) or changes in the opposite direction (for negative eigenvalues). The eigenvalue of an eigenvector is a scaling factor by which it has been multiplied. The spectrum of a linear transformation defined on finite-dimensional vector spaces is the set of all eigenvalues.

These concepts are of great importance in various areas of mathematics, especially in linear algebra, functional analysis, nonlinear mathematical equations, etc. We consider the most important concepts and methods, related to eigenvalues and eigenvectors, in both systems of symbolic algebra: characteristic matrices and polynomials, eigenvalues, eigenvectors, minimal polynomials, the diagonalization and diagonal factorization of a matrix, the trace of a matrix, the Cayley–Hamilton theorem.

```
with(LinearAlgebra):          Trace(M);       z:=Eigenvectors(M);   Eigenvalues(M);
CharacteristicMatrix(M,x);    CharacteristicPolynomial(M,x);
MinimalPolynomial(M,x);       p:=z[2];        d:=DiagonalMatrix(z[1]);
Equal(d,MatrixInverse(p).M.p);                Equal(M,p.d.MatrixInverse(p));
```

xiv Points in the Plane and Space

```
with(plots):                  pointplot(points,ops);
pointplot3d(points,ops);      matrixplot(matrix,ops);
listplot(list,ops);           listplot3d(list,ops);
with(Statistics):             ScatterPlot(seqX,seqY,ops);
```

xx Laplace Transforms

In Maple, the integral transforms (e.g., Fourier, Hilbert, Laplace, Mellin integral transforms) can be studied with the aid of the inttrans package.

Practical 1 : Simple programs using Mathematical constants

Aim: **To find the double factorial of a number 'n'.**

(You can find double factorial of any number)

Procedure:

Step 1: Verify given number is integer or not.

Step 2: Write the Maple program to find the double factorial of a given number.

Step 3: Run the Program.

Step 4: If any error occurs in your code then clear the errors and again run.

Step 5: Take the output as you like.

Results & Conclusion: Get the output and interpret the results.

Practical 2 : Programs using complex functions

Aim: **To find the derivatives of the functions** $f_1(z) = z^2 + i5z - 1$ *and* $f_2(z) = (f_1(z))5$.

(You can find the higher derivative of a given complex functions)

Procedure:

Step 1: Verify the given function is complex or not.

Step 2: Desire to find first, second or higher derivative.

Step 3: Write a maple program to find the desired derivates of a given complex functions.

Step 4: Run the Program.

Step 5: If any error occurs in your code then clear the errors and again run.

Step 6: Take the output as you like.

Results & Conclusion : Get the output and interpret the results.

Practical 3 : Numerical solutions of nonlinear equations and systems

Aim: **To find the root of the non linear equation $x^3 - 2x + 2 = 0$**

(You can find root of any transcendental equation of any degree.)

Procedure:

Step 1: Verify the given equation is non linear.

Step 2: Fix a lower limit, upper limits and tolerance.

Step 3: Write a maple program to find the root of the given non linear equation.

Step 4: Run the Program.

Step 5: If any error occurs in your code then clear the errors and again run.

Step 6: Take the output as you like.

Results & Conclusion : Get the output and interpret the results.

Practical 4 : Solving system of linear equations using Jacobi method

Aim: **To solve the system of equations**

$$3a + 0.1b + 0.2c = 7.85,$$
$$0.1a + 7b + 0.3c = -19.3,$$
$$0.2a + 0.3b + 10c = 71.4.$$ **by Jacobi method.**

(You can solve any system of equations by Jacobi methods of order more than 4 by 4)

Procedure:

Step 1: Verify the system of equations are linear or not.

Step 2: Check the matrix of order and diagonally dominance.

Step 3: Write the Maple program to solve given system of Equations by Jacobi method.

Step 4: Run the Program.

Step 5: If any error occurs in your code then clear the errors and again run.

Step 6: Take the output as you like.

Results & Conclusion : Get the output and interpret the results.

Practical 5 : Program using Trigonometric and Hyperbolic Expressions

Aim: **To find Maclaurin series expansion for tan(x) and also find the inverse series of the given series.**

(You can express any trigonometrical functions in Meclaurin series.)

Procedure:

Step 1: Verify the given function is trigonometrical or not.

Step 2: Fix a number of terms in the series.

Step 3: Write a maple program to express the given *trigonometrical functions in Meclaurin series.*

Step 4: Run the Program.

Step 5: If any error occurs in your code then clear the errors and again run.

Step 6: Take the output as you like.

Results & Conclusion : Get the output and interpret the results.

Practical 6 : Finding Eigen values and Eigen vectors of a matrix

Aim: **To find Eigen values, Eigen vectors characteristic equation and diagonal matrix of the given matrix.**

(You can* find Eigen values, Eigen vectors characteristic equation and diagonal matrix of the given matrix any order.*)

Procedure:

Step 1: Verify the given matrix is square or not.

Step 2: Write a maple program to Eigen values, Eigen vectors characteristic equation and diagonal matrix of the given matrix

Step 3: Run the Program.

Step 4: If any error occurs in your code then clear the errors and again run.

Step 5: Take the output as you like.

Results & Conclusion : Get the output and interpret the results.

Practical 7 : Plotting Points in the Plane and Space

Aim: **To draw a scatter plot in plane and the space.**

(You can draw any plane curve and the space curves.)

Procedure:

Step 1: Verify the given data is a data form or function form.

Step 2: Fix a scale in two or three dimensions.

Step 3: Write a maple program to draw a scatter plot in plane and the space

Step 4: Run the Program.

Step 5: If any error occurs in your code then clear the errors and again run.

Step 6: Take the output as you like.

Results & Conclusion : Get the output and interpret the results.

Practical 8 : Analyse data using Central Tendency and Measures of dispersion and distributions

Aim: **To find mean, standard deviation and polynomial fit for the data:**

X	1	2	3	4	5	6
Y	25	26	26	25	24	22

(You can find* mean, standard deviation and polynomial fit for any given data.*)

Procedure:

Step 1: Verify the given data is a data form or function form.

Step 2: Write a maple program to find mean, standard deviation and polynomial fit for the given data.

Step 3: Run the Program.

Step 4: If any error occurs in your code then clear the errors and again run.

Step 5: Take the output as you like.

Results & Conclusion : Get the output and interpret the results.

Practical 9 : Find the Laplace integral transforms for different functions

Aim: **To find the Laplace integral transforms of the function $t^2 \cos 2t$.**

(You can find Laplace transform of any functions like exponential, trigonometric and algebraic functions. Also combination of these.)

Procedure:

Step 1: Verify the given function is *exponential, trigonometric and algebraic functions. Also combination of these*.

Step 2: Write a maple program to find the desired derivates of a given complex functions.

Step 3: Run the Program.

Step 4: If any error occurs in your code then clear the errors and again run.

Step 5: Take the output as you like.

Results & Conclusion : Get the output and interpret the results.

Practical 10 : Solving the differential equations.

Aim: **To find the solution of the differential equation $\frac{dy}{dt} + t^2 = 0$.**

(You can find* the solution of the differential equation of any order and any degree.*)

Procedure:

Step 1: Verify the equation is differential equation or not.

Step 2: Check the initial conditions are given or not.

Step 3: Write a maple program to find the solution of the given differential equations.

Step 4: Run the Program.

Step 5: If any error occurs in your code then clear the errors and again run.

Step 6: Take the output as you like.

Results & Conclusion : Get the output and interpret the results.