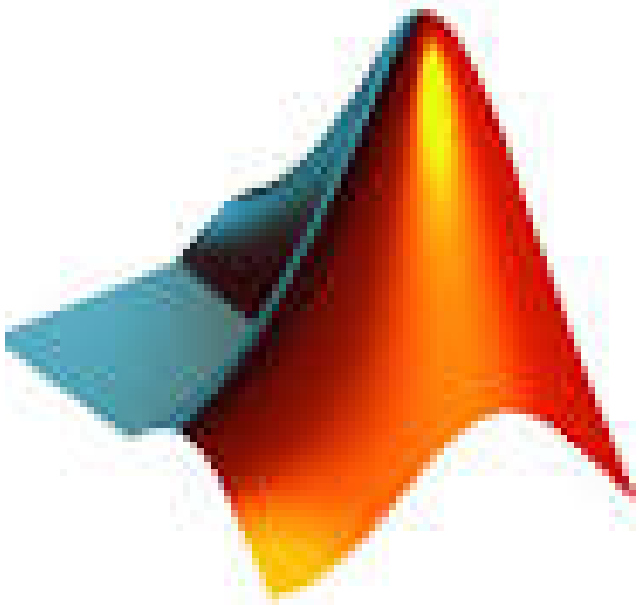
MATLAB

Laboratory Manual
for Mathematics Courses

Offered to

I Year B.Tech. Programmes

(Common to all Branches)



KALASALINGAM

ACADEMY OF RESEARCH AND EDUCATION

(DEEMED TO BE UNIVERSITY)

Under sec. 3 of UGC Act 1956. Accredited by NAAC with "A" Grade

MATLAB Laboratory Manual

for Mathematics Courses

Offered to

I Year B.Tech. Programmes

(211MAT1301 & 211MAT1303)

(Common to all Branches)

Name of the Student: _____

Reg. No.: _____

Branch: _____

Section: _____

KALASALINGAM
ACADEMY OF RESEARCH AND EDUCATION
(DEEMED TO BE UNIVERSITY)
Under sec. 3 of UGC Act 1956. Accredited by NAAC with "A" Grade

KALASALINGAM
ACADEMY OF RESEARCH AND EDUCATION
(DEEMED TO BE UNIVERSITY)
Under sec. 3 of UGC Act 1956. Accredited by NAAC with "A" Grade

OFFICE OF DEAN – FRESHMAN ENGINEERING

DEPARTMENT OF MATHEMATICS

UNIVERSITY VISION

To be a Center of Excellence of International Repute in Education and Research.

UNIVERSITY MISSION

To Produce Technically Competent, Socially Committed Technocrats and Administrators through Quality Education and Research.

VISION OF THE DEPARTMENT

To become a much sought-after centre for mathematics education and research through our sustained efforts and eminence.

MISSION OF THE DEPARTMENT

To sustain our enthusiasm for periodic updating of our UG and PG curriculum to meet the ever-growing needs of ACADEMIA, RESEARCH AND INDUSTRY and to identify and nurture the hidden talent in our valuable student community and transform them into responsible professionals of our country.

Program Outcomes (Pos):

PO 1: Engineering Knowledge: Apply knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.

PO 2: Problem Analysis: Identify, formulate, research literature and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences.

PO 3: Design/ Development of Solutions: Design solutions for complex engineering problems and design system components or processes that meet specified needs with appropriate consideration for public health and safety, cultural, societal and environmental considerations.

PO 4: Conduct investigations of complex problems using research-based knowledge and research methods including design of experiments, analysis and interpretation of data and synthesis of information to provide valid conclusions.

PO 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.

PO 6: The Engineer and Society: Apply reasoning informed by contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to professional engineering practice.

PO 7: Environment and Sustainability: Understand the impact of professional engineering solutions in societal and environmental contexts and demonstrate knowledge of and need for sustainable development.

PO 8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of engineering practice.

PO 9: Individual and Team Work: Function effectively as an individual, and as a member or leader in diverse teams and in multi-disciplinary settings.

PO 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.

PO 11: Project Management and Finance: Demonstrate knowledge and understanding of 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.

PO 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.

1. 211MAT1301 – CALCULUS AND LINEAR ALGEBRA

Prerequisite	NIL	**Syllabus Revision**	2021
Course Category	**Course Type**	**Course Level**	
B.Tech.	Integrated Course-Theory (ICT)	Foundation Core (ES)	

Course Outcomes (CO)

On successful completion of the course, the students would be able to;

CO1	Understand the concepts of Eigenvalues, Eigenvector and Diagonalization of matrices.
CO2	Use the conditions of vector spaces, linearly independent and linearly dependent to find the basis and dimension.
CO3	Apply the fundamental theorems on calculus and to find maxima and minima of functions of one variable**.**
CO4	Utilize the ideas of derivatives to find maxima and minima of functions of multi variables
CO5	Evaluate the problems on evolutes and simple applications of one-dimensional calculus.

CO – PO mapping:

CO / PO	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**
CO1	**2**	**2**										**1**
CO2	**2**	**2**										**1**
CO3	**1**	**2**										**1**
CO4	**2**	**2**										
CO5	**1**	**1**										**1**

*3 – Strong correlation; 2 – Medium correlation; 1 – Low correlation

2. 211MAT1303 – MULTIPLE INTEGRATION, ORDINARY DIFFERENTIAL EQUATIONS AND COMPLEX VARIABLES

Prerequisite	NIL	**Syllabus Revision**	2021
Course Category	**Course Type**	**Course Level**	
B.Tech.	Integrated Course-Theory (ICT)	Foundation Core (ES)	

Course Outcomes (CO)

On successful completion of the course, the students would be able to;

CO1	Distinguish the methods of solving differential equations of first and second orders
CO2	Recognize the concepts of double and triple integral and its applications
CO3	Realize about the applications of double and triple integral in vector calculus
CO4	Apply the concept and consequences of analyticity and the Cauchy-Riemann equations and of results on harmonic and entire functions including the fundamental theorem of algebra.
CO5	Evaluate complex contour integrals directly and apply the Cauchy integral theorem in its various versions, and the Cauchy integral formula.

CO – PO mapping:

CO / PO	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**
CO1	**2**	**2**										**1**
CO2	**1**	**2**										**1**
CO3	**2**	**2**										**1**
CO4	**2**	**1**										**1**
CO5	**2**	**2**										

*3 – Strong correlation; 2 – Medium correlation; 1 – Low correlation

INDEX

LIST OF EXPERIMENTS

TABLE OF CONTENTS - ODD

Ex. No.	Date	Name of the Experiment	Mark	Staff Signature	Remarks

TABLE OF CONTENTS - EVEN

Ex. No.	Date	Name of the Experiment	Mark	Staff Signature	Remarks

EXPERIMENT-I

1.1 OBJECTIVES

a. To Know the history and features of MATLAB
b. To Know the local environment of MATLAB

1.1.1 CONTENT

Introduction

MATLAB is a high-level language and interactive environment for numericalcomputation, visualization, and programming. Using MATLAB, you can analyse data, develop algorithms, and create models and applications. The language, tools, and built-inmath functions enable you to explore multiple approaches and reach a solution fasterthan with spread sheets or traditional programming languages, such as C/C++ or Java. You can use MATLAB for a range of applications, including signal processing andcommunications, image and video processing, control systems, test and measurement, computational finance, and computational biology. More than a million engineers andscientists in industry and academia use MATLAB, the language of technical computing.

History

- Developed primarily by Cleve Moler in the 1970'sDerived from FORTRAN subroutines LINPACK and EISPACK, linear and eigenvaluesystems.
- Developed primarily as an interactive system to access LINPACK and EISPACK.
- Gained its popularity through word of mouth, because it was not socially distributed.
- Rewritten in C in the 1980's with more functionality, which include plotting routines.
- The Math Works Inc. was created (1984) to market and continue development of MATLAB.

Strengths

- MATLAB may behave as a calculator or as a programming language
- MATLAB combine nicely calculation and graphic plotting.
- MATLAB is relatively easy to learn
- MATLAB is interpreted (not compiled), errors are easy to fix
- MATLAB is optimized to be relatively fast when performing matrix operations
- MATLAB does have some object-oriented elements

Weaknesses

- MATLAB is not a general purpose programming language such as C, C++, or FORTRAN
- MATLAB is designed for scientific computing, and is not well suitable for other applications
- MATLAB is an interpreted language, slower than a compiled language such as C++
- MATLAB commands are specific for MATLAB usage. Most of them do not have a direct equivalent with other programming language commands

Competition

One of MATLAB's competitors is Mathematica the symbolic computation program.MATLAB is more convenient for numerical analysis and linear algebra. It is frequentlyused in engineering community. Mathematica has superior symbolic manipulation, making it popular among physicists.
There are other competitors: Scilab, GNU Octave, and Rlab

Key Features

- 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.

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

Uses of MATLAB

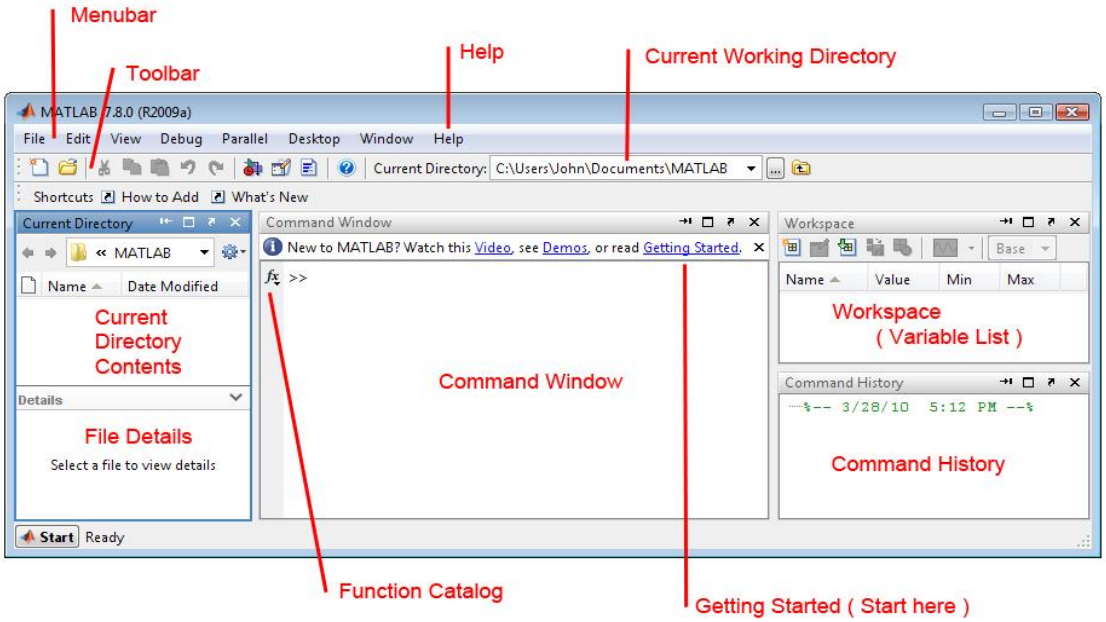
MATLAB is widely used as a computational tool in science and engineering encompassing the fields of physics, chemistry, math and all engineering streams. It is used in a range of applications including:

- Signal processing and Communications
- Image and video Processing
- Control systems
- Test and measurement
- Computational finance
- Computational biology

Understanding the MATLAB Environment

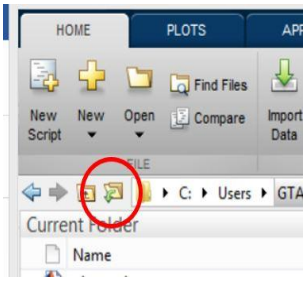
MATLAB development IDE can be launched from the icon created on the desktop. The main working window in MATLAB is called the desktop. When MATLAB is started, the desktop appears in its default layout:

The MATLAB Work Environment

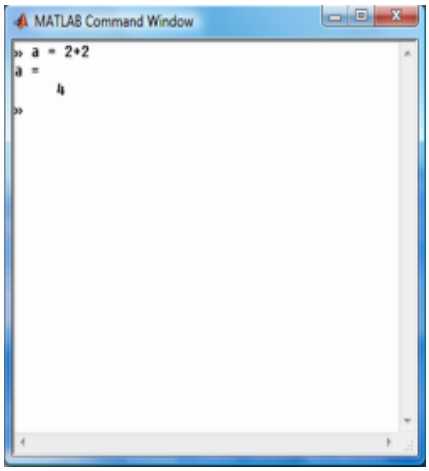


The desktop includes these panels:

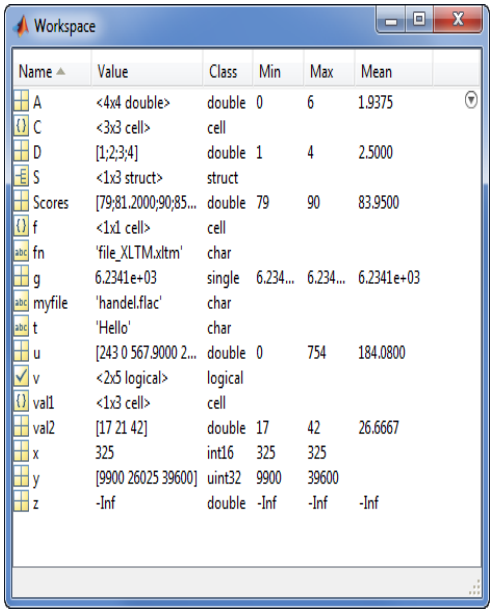
Current Folder - This panel allows you to access the project folders and files.



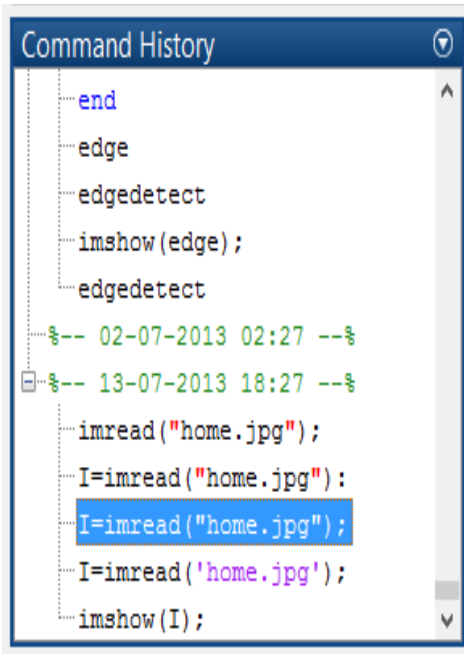
Command Window - This is the main area where commands can be entered at the command line. It is indicated by the command prompt (>>).

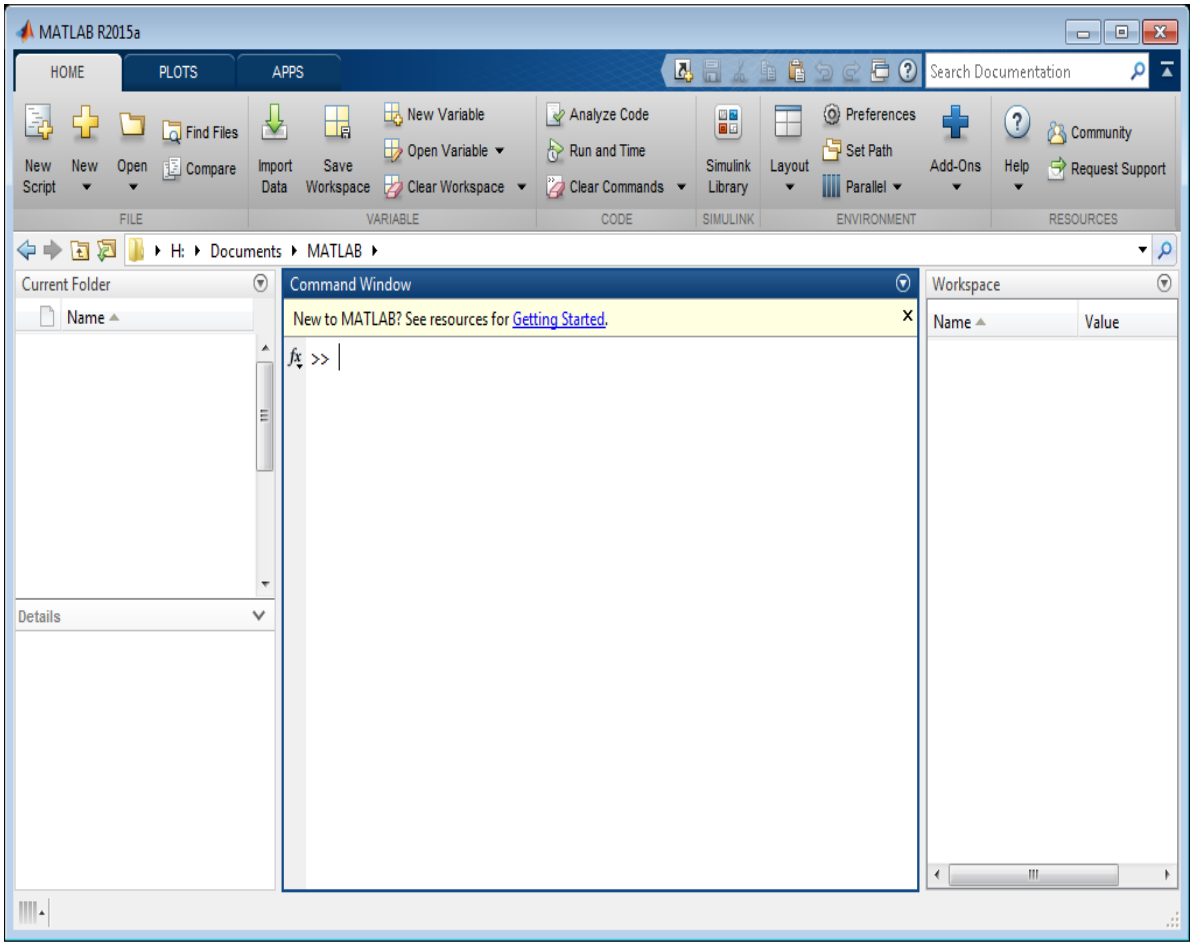


Workspace - The workspace shows all the variables created and/or imported from files.



Command History - This panel shows or rerun commands that are entered at the command line.





You are now faced with the MATLAB desktop on your computer, which contains the prompt (>>) in the Command Window. Usually, there are 2 types of prompt:

>>For full version
EDU> for educational version

Note:

1. To simplify the notation, we will use this prompt, >>, as a standard prompt sign, though our MATLAB version is for educational purpose.
2. MATLAB adds variable to the workspace and displays the result in the Command Window.

Managing workspace and file commands

Command	Description
cd	Change current directory
clc	Clear the Command Window
clear (all)	Removes all variables from the workspace
clear x	Remove x from the workspace
copy file	Copy file or directory
delete	Delete files

dir	Display directory listing
exist	Check if variables or functions are defined
help	Display help for MATLAB functions
look for	Search for specified word in all help entries
mkdir	Make new directory
move file	Move file or directory
pwd	Identify current directory
rmdir	Remove directory
type	Display contents of file
what	List MATLAB files in current directory
which	Locate functions and files
who	Display variables currently in the workspace
whos	Display information on variables in the workspace

Commonly used Operators and Special Characters

MATLAB supports the following commonly used operators and special characters:

Operator	Purpose
+	Plus; addition operator.
-	Minus; subtraction operator.
*	Scalar and matrix multiplication operator.
.*	Array multiplication operator.
^	Scalar and matrix exponentiation operator.
.^	Array exponentiation operator.
\	Left-division operator.
/	Right-division operator.
.\	Array left-division operator.
./	Array right-division operator.
:	Colon; generates regularly spaced elements and represents an entire row or column.
( )	Parentheses; encloses function arguments and array indices; overrides precedence.
[ ]	Brackets; enclosures array elements.
.	Decimal point.
...	Ellipsis; line-continuation operator
,	Comma; separates statements and elements in a row
;	Semicolon; separates columns and suppresses display.
%	Percent sign; designates a comment and specifies formatting.

_	Quote sign and transpose operator.
._	Non-conjugated transpose operator.
=	Assignment operator.

Note:

If you end a statement with a semicolon, MATLAB performs the computation, butsuppresses the display of output in the Command Window.

Special Variables and Constants

MATLAB supports the following special variables and constants:

Name	Meaning
ans	Most recent answer.
eps	Accuracy of floating-point precision.
i,j	The imaginary unit √-1.
Inf	Infinity.
NaN	Undefined numerical result (not a number).
pi	The number π

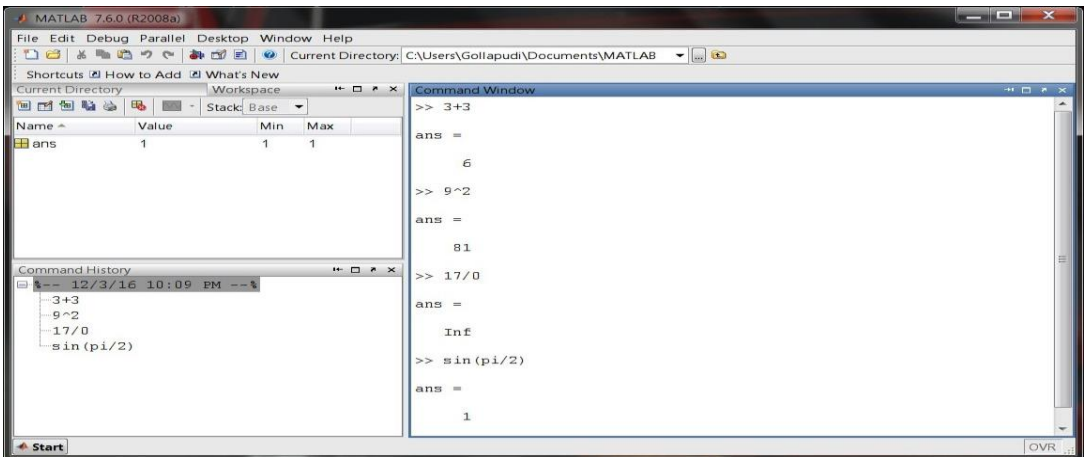
Naming Variables

Variable names consist of a letter followed by any number of letters, digits or underscore.
MATLAB is **case-sensitive**.
Variable names can be of any length; however, MATLAB uses only first N characters, where N is given by the function **namelengthmax**.

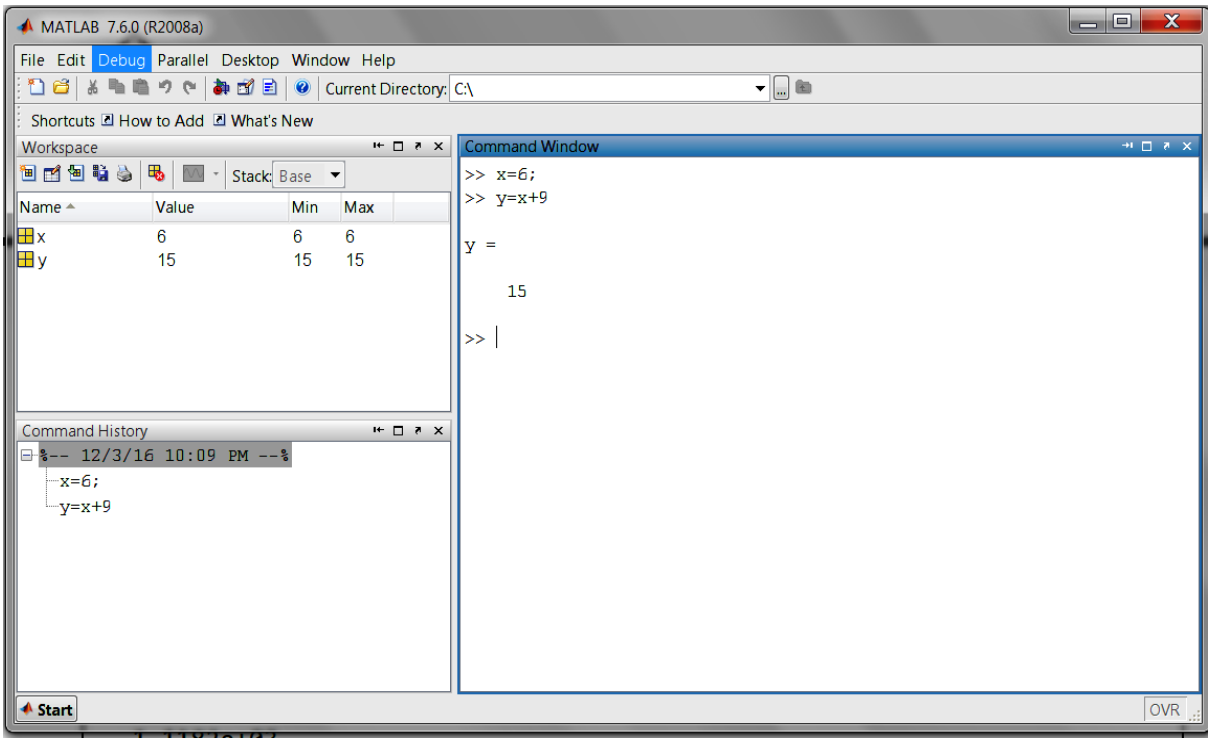
Saving Your Work

The **save** command is used for saving all the variables in the workspace, as a file with .mat extension, in the current directory.
For example, save myfile
You can reload the file anytime later using the **load** command.
load myfile

Example 1:

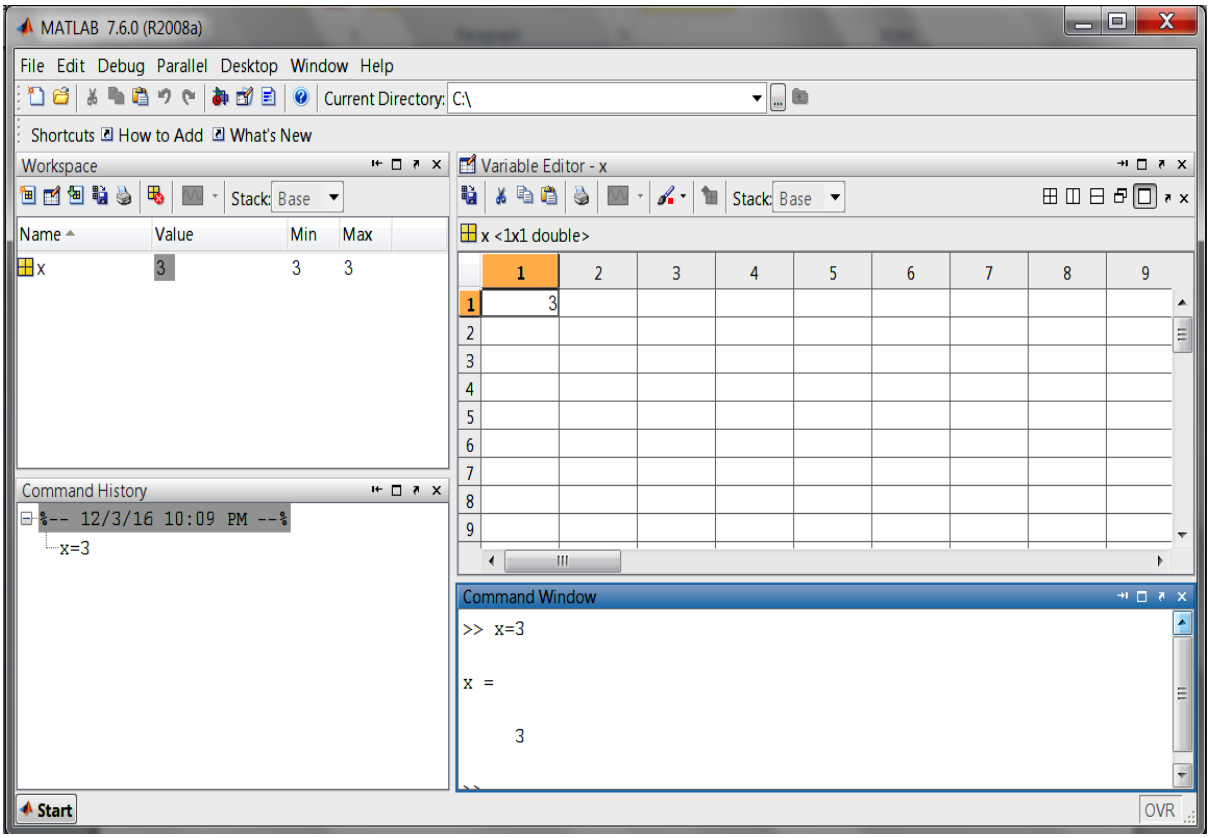


Example 2:



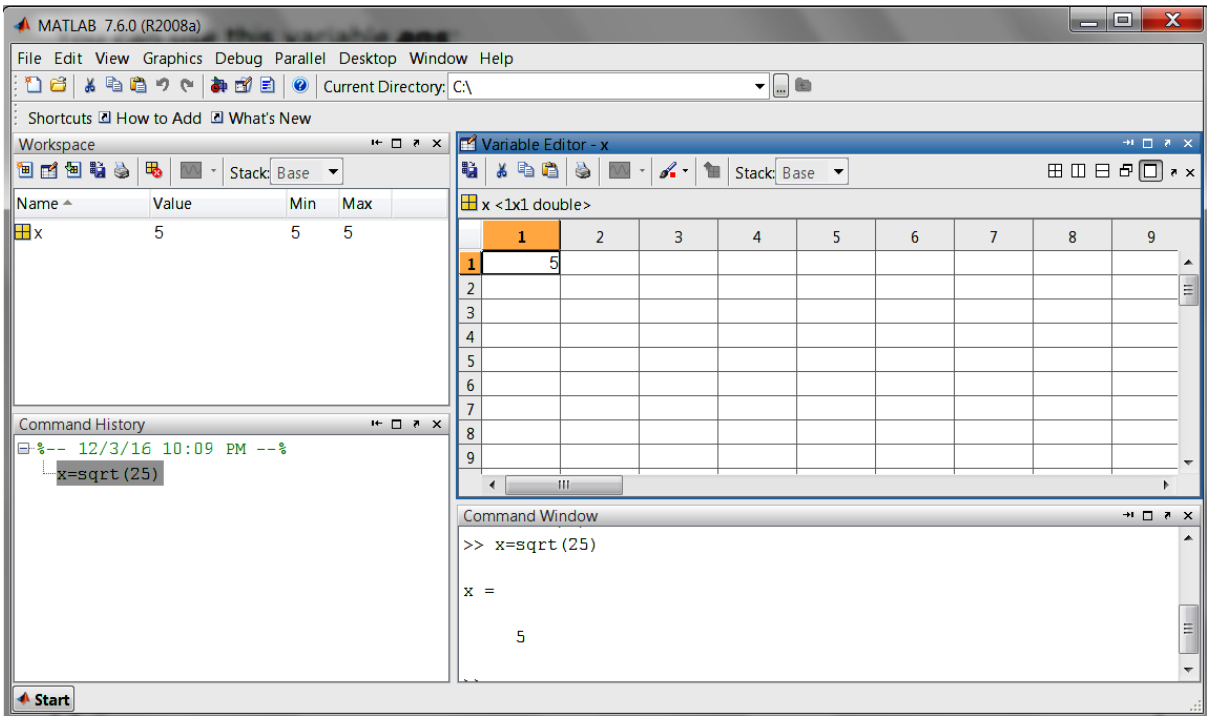
In MATLAB environment, every variable is an array or matrix.

Example 3:



In the above example it creates a 1-by-1 matrix named ‘x’and stores the value 3 in its element.

Example 4:



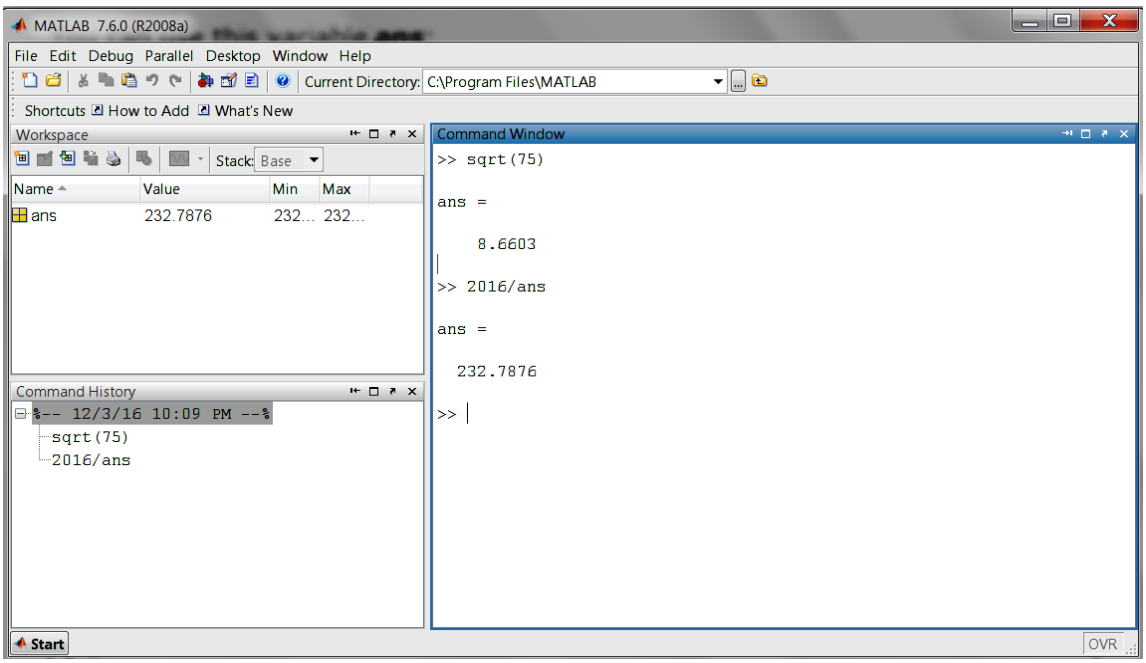
In this example x is to find the square root of 25 it creates a 1-by-1 matrix named ‘x’and stores the value 5 in its element

.

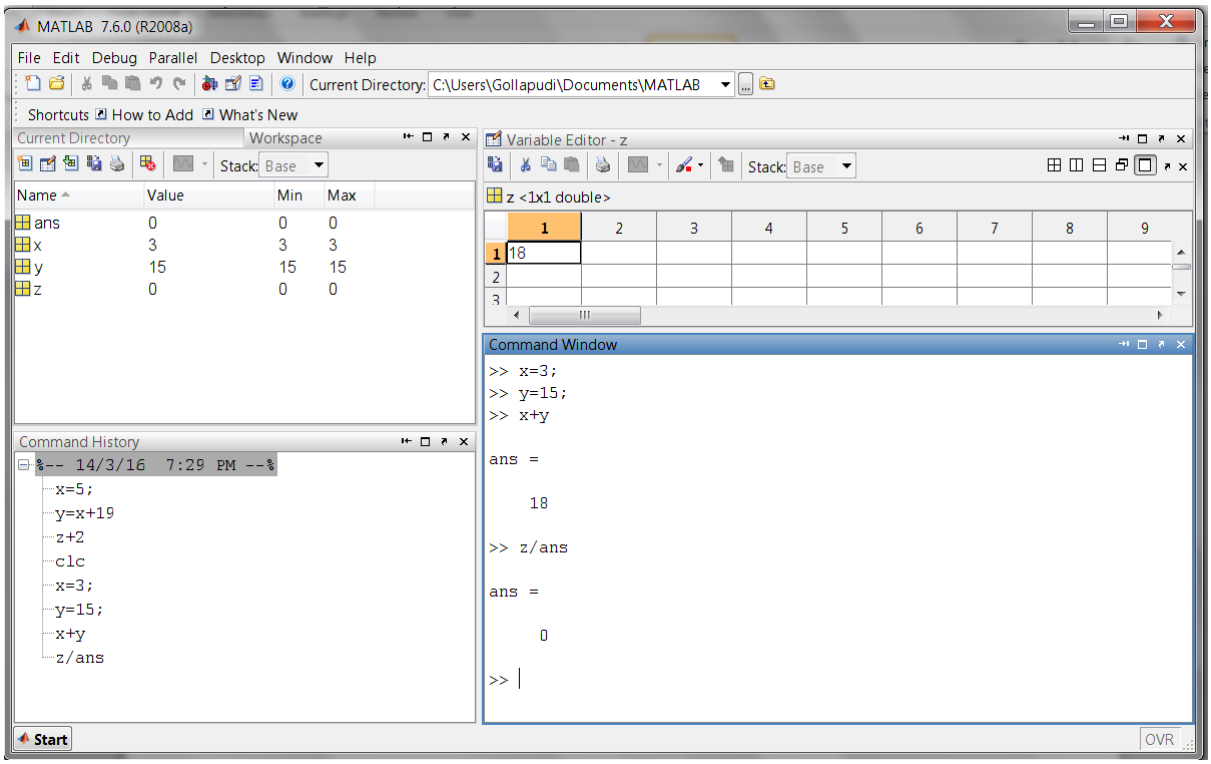
Note:

- Once a variable is entered into the system, you can refer to it later.
- Variables must have values before they are used.
- When you do not specify an output variable, MATLAB uses the variable ans, short for *answer*, to store the results of your calculation.

Example 6:



Example 7:



In the above example we have multiple assignments

Cayley Hamilton Theorem

Aim:

To write a MATLAB program to verify Cayley Hamilton theorem for the matrix $A = \begin{bmatrix} 7 & 2 & -2 \\ -6 & -1 & 2 \\ 6 & 2 & -1 \end{bmatrix}$. Also compute A^{-1} and A^4 using Cayley Hamilton Theorem.

Main Commands:

syms - to define symbolic variables and fynctions

charpoly - to create the characteristic polynomial

format rational - to create numbers in the rational form

Source Code:

```
syms A s
A=[7 2 -2; -6 -1 2; 6 2 -1];
I=[1 0 0; 0 1 0; 0 0 1];
charpoly(A,s)==0
```

```
ans = s^3 - 5 s^2 + 7 s - 3 = 0
ans =
      25             8            -8
     -24            -7             8
      24             8            -7
ans =
      79            26           -26
     -78           -25            26
      78            26           -25
```

```
verification=A^3-5*A^2+7*A-3*I
```

```
verification =
       0             0             0
       0             0             0
       0             0             0
```

```
format rational
inv_A=(1/3)*(A^2-5*A+7*I)
```

```
inv_A =
      -1          -2/3           2/3
       2           5/3          -2/3
      -2          -2/3           5/3
```

```
A_power4=5*A^3-7*A^2+3*A
```

```
A_power4 =
     241            80           -80
    -240           -79            80
     240            80           -79
```

Aim:

To write a MATLAB program to verify Cayley Hamilton theorem for the matrix $A = \begin{bmatrix} 2 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 2 \end{bmatrix}$. Also compute A^{-1} and A^4 using Cayley Hamilton Theorem.

Main Commands:

syms - to define symbolic variables and fynctions

charpoly - to create the characteristic polynomial

format rational - to create numbers in the rational form

Source Code:

```
syms A s
A=[2 1 1; 0 1 0; 1 1 2];
I=[1 0 0; 0 1 0; 0 0 1];
charpoly(A,s)==0
```

```
ans = s^3 - 5 s^2 + 7 s - 3 = 0
ans =
     5     4     4
     0     1     0
     4     4     5
ans =
    14    13    13
     0     1     0
    13    13    14
```

```
verification=A^3-5*A^2+7*A-3*I
```

```
verification =
     0     0     0
     0     0     0
     0     0     0
```

```
format rational
inv_A=(1/3)*(A^2-5*A+7*I)
```

```
inv_A =
     2/3    -1/3    -1/3
     0       1       0
    -1/3    -1/3     2/3
```

```
A_power4=5*A^3-7*A^2+3*A
```

```
A_power4 =
    41    40    40
     0     1     0
    40    40    41
```

Aim:

To write a MATLAB program to verify Cayley Hamilton theorem for the matrix $A = \begin{bmatrix} -2 & 2 & -3 \\ 2 & 1 & -6 \\ -1 & -2 & 0 \end{bmatrix}$. Also compute A^{-1} and A^4 using Cayley Hamilton Theorem.

Main Commands:

syms - to define symbolic variables and fynctions

charpoly - to create the characteristic polynomial

format rational - to create numbers in the rational form

Source Code:

```
syms A s
A=[-2 2 -3; 2 1 -6; -1 -2 0];
I=[1 0 0; 0 1 0; 0 0 1];
charpoly(A,s)==0
```

ans = $s^3 + s^2 - 21\,s - 45 = 0$

```
ans =
      11             4            -6
       4            17           -12
      -2            -4            15
ans =
      -8            38           -57
      38            49          -114
     -19           -38            30
```

```
verification=A^3+A^2-21*A-45*I
```

```
verification =
       0             0             0
       0             0             0
       0             0             0
```

```
format rational
inv_A=(1/45)*(A^2+A-21*I)
```

```
inv_A =
      -4/15          2/15          -1/5
       2/15         -1/15          -2/5
      -1/15         -2/15         -2/15
```

```
A_power4=-A^3+21*A^2+45*A
```

```
A_power4 =
     149           136          -204
     136           353          -408
     -68          -136           285
```

Aim:

To write a MATLAB program to verify Cayley Hamilton theorem for the matrix $A = \begin{bmatrix} 1 & 1 & 3 \\ 1 & 3 & -3 \\ 2 & -4 & -4 \end{bmatrix}$. Also compute A^{-1} and A^4 using Cayley Hamilton Theorem.

Main Commands:

syms - to define symbolic variables and fynctions

charpoly - to create the characteristic polynomial

format rational - to create numbers in the rational form

Source Code:

```
syms s
A=[1 1 3; 1 3 -3; 2 -4 -4];
I=[1 0 0; 0 1 0; 0 0 1];
charpoly(A,s)==0
```

ans = $s^3 - 32\,s + 56 = 0$

```
verification=A^3-32*A+56*I
```

```
verification =
       0             0             0
       0             0             0
       0             0             0
```

```
format rational
inv_A=(1/56)*(-A^2+32*I)
```

```
inv_A =
       3/7           1/7           3/14
       1/28          5/28         -3/28
       5/28         -3/28         -1/28
```

```
A_power4=32*A^2-56*A
```

```
A_power4 =
     200          -312          -552
    -120           536           360
    -432           416          1312
```

Diagonalization of Matrices

Aim:

To write a MATLAB program to diagonalize the matrix $A = \begin{bmatrix} 1 & 1 & 3 \\ 1 & 5 & 1 \\ 3 & 1 & 1 \end{bmatrix}$ using orthogonal transformation.

Main Commands:

syms - to define symbolic variables and fynctions

eig - to get the eigen values

charpoly - to create the characteristic polynomial

Source Code:

```
syms s
A =[1 1 3;1 5 1;3 1 1]
```

```
A =
      1            1            3
      1            5            1
      3            1            1
```

```
charpoly(A,s)==0
```

ans = $s^3 - 7\,s^2 + 36 = 0$

```
eig_A = eig(A)
```

```
ans =
     -2
      3
      6
```

```
[N D]= eig (A);
M(: ,1)=N(: ,1)* sqrt (2) ;
M(: ,2)=N(: ,2)* sqrt (3) ;
M(: ,3)=N(: ,3)* sqrt (6)
```

```
M =
     -1            1            1
      *           -1            2
      1            1            1
```

```
[N D]= eig (A)
```

```
N =
    -985/1393       780/1351       881/2158
       *           -780/1351       881/1079
     985/1393       780/1351       881/2158
D =
      -2            0            0
       0            3            0
       0            0            6
```

Aim:

To write a MATLAB program to diagonalize the matrix $A = \begin{bmatrix} 2 & 1 & -1 \\ 1 & 1 & -2 \\ -1 & -2 & 1 \end{bmatrix}$ using orthogonal transformation.

Main Commands:

syms - to define symbolic variables and fynctions

eig - to get the eigen values

charpoly - to create the characteristic polynomial

Source Code:

```
syms s
A =[2 1 -1; 1 1 -2; -1 -2 1]
```

```
A =
      2             1            -1
      1             1            -2
     -1            -2             1
```

```
charpoly(A,s)==0
```

ans = $s^3 - 4\,s^2 - s + 4 = 0$

```
eig_A = eig(A)
```

```
ans =
     -1
      1
      4
```

```
[N D]= eig (A);
M(: ,1)=N(: ,1)* sqrt (2) ;
M(: ,2)=N(: ,2)* sqrt (6) ;
M(: ,3)=N(: ,3)* sqrt (3)
```

```
M =
      *             2            -1
      1            -1            -1
      1             1             1
```

```
[N D]= eig (A)
```

```
N =
      *           881/1079      -780/1351
    985/1393     -881/2158      -780/1351
    985/1393      881/2158       780/1351
D =
     -1             0             0
      0             1             0
      0             0             4
```

Aim:

To write a MATLAB program to diagonalize the matrix $A = \begin{bmatrix} 3 & -1 & 1 \\ -1 & 5 & -1 \\ 1 & -1 & 3 \end{bmatrix}$ using orthogonal transformation.

Main Commands:

syms - to define symbolic variables and fynctions

eig - to get the eigen values

charpoly - to create the characteristic polynomial

Source Code:

```
syms s
A =[3 -1 1; -1 5 -1; 1 -1 3]
```

```
A =
      3            -1             1
     -1             5            -1
      1            -1             3
```

```
charpoly(A,s)==0
```

ans = $s^3 - 11\,s^2 + 36\,s - 36 = 0$

```
eig_A = eig(A)
```

```
ans =
      2
      3
      6
```

```
[N D]= eig (A);
M(: ,1)=N(: ,1)* sqrt (2) ;
M(: ,2)=N(: ,2)* sqrt (3) ;
M(: ,3)=N(: ,3)* sqrt (6)
```

```
M =
      1            -1             1
      *            -1            -2
     -1            -1             1
```

```
[N D]= eig (A)
```

```
N =
    985/1393      -780/1351       881/2158
      *           -780/1351      -881/1079
   -985/1393      -780/1351       881/2158
D =
      2             0             0
      0             3             0
      0             0             6
```

Aim:

To write a MATLAB program to diagonalize the matrix $A = \begin{bmatrix} 2 & 0 & 4 \\ 0 & 6 & 0 \\ 4 & 0 & 2 \end{bmatrix}$ using orthogonal transformation.

Main Commands:

syms - to define symbolic variables and fynctions

eig - to get the eigen values

charpoly - to create the characteristic polynomial

Source Code:

```
syms s
A =[2 0 4; 0 6 0; 4 0 2]
```

```
A =
      2             0             4
      0             6             0
      4             0             2
```

```
charpoly(A,s)==0
```

ans = $s^3 - 10\,s^2 + 12\,s + 72 = 0$

```
eig_A = eig(A)
```

```
ans =
     -2
      6
      6
```

```
[N D]= eig (A);
M(: ,1)=N(: ,1)* sqrt (2) ;
M(: ,2)=N(: ,2)* sqrt (2) ;
M(: ,3)=N(: ,3)
```

```
M =
      1             1             0
      0             0            -1
     -1             1             0
```

```
[N D]= eig (A)
```

```
N =
    985/1393      985/1393       0
      0             0            -1
   -985/1393      985/1393       0
D =
     -2             0             0
      0             6             0
      0             0             6
```

Linearly Independant / Dependent Vectors

Aim:

To write a MATLAB program to test the vectors $(1,2,1),(2,1,0)$ and $(1,1,2)$ in $V_3(R)$ are linearly independent or not.

Main Commands:

syms - to define symbolic variables and fynctions

det - to find the determinant of the matrix

disp - to display the statements

Source Code:

```
A=[1 2 1; 2 1 1; 1 0 2]
```

```
A =
     1     2     1
     2     1     1
     1     0     2
```

```
det_A = det(A)
```

```
det_A =
    -5
```

```
if det(A)==0
    disp('The given vectors are dependent')
else
    disp('The given vectors are independent')
end
```

```
The given vectors are independent
```

Aim:

To write a MATLAB program to test the vectors $(1,4,\ 2),(-2,1,3)$ and $(4,11,\ 5)$ in $V_3(R)$ are linearly independient or not.

Main Commands:

syms - to define symbolic variables and fynctions

det - to find the determinant of the matrix

disp - to display the statements

Source Code:

```
A=[1 -2 4; 4 1 11; 2 3 5]
```

```
A =
       1            -2             4
       4             1            11
       2             3             5
```

```
det_A = det(A)
```

```
det_A =
       8
```

```
if det(A)==0
    disp('The given vectors are dependent')
else
    disp('The given vectors are independent')
end
```

```
The given vectors are independent
```

Aim:

To write a MATLAB program to test the vectors $(1,-2,-1,0),(2,-1,1,0),(2,\ 1,-1,1)$ and $(-1,-1,2,-1)$ in $V_4(R)$ are linearly independient or not.

Main Commands:

syms - to define symbolic variables and fynctions

det - to find the determinant of the matrix

disp - to display the statements

Source Code:

```
A=[1 2 2 -1; -2 -1 1 -1; -1 1 -1 2; 0 0 1 -1]
```

```
A =
       1             2             2            -1
      -2            -1             1            -1
      -1             1            -1             2
       0             0             1            -1
```

```
det_A = det(A)
```

```
det_A =
       0
```

```
if det(A)==0
    disp('The given vectors are dependent')
else
    disp('The given vectors are independent')
end
```

```
The given vectors are dependent
```

Aim:

To write a MATLAB program to test the vectors $(1,1,\ 2,4),(2,-1,-5,2),(1,-1,-4,0)$ and $(2,1,1,6)$ in $V_4(R)$ are linearly independient or not.

Main Commands:

syms - to define symbolic variables and fynctions

det - to find the determinant of the matrix

disp - to display the statements

Source Code:

```
A=[1 2 1 2; 1 -1 -1 1; 2 -5 -4 1; 4 2 0 6]
```

```
A =
     1     2     1     2
     1    -1    -1     1
     2    -5    -4     1
     4     2     0     6
```

```
det_A = det(A)
```

```
det_A =
     0
```

```
if det(A)==0
    disp('The given vectors are dependent')
else
    disp('The given vectors are independent')
end
```

```
The given vectors are dependent
```

Rolle's Theorem

Aim:

To write a MATLAB program to determine $c \epsilon (a, b)$ for $f(x) = (x-1)^2(x-3) \text{ in } (1, 3)$ using Rolle's theorem

Main Commands:

syms - to define symbolic variables and fynctions

diff - to find the derivative of the function

solve - to solve the equation

format rational - to create numbers in the rational form

Source Code:

```
syms x
f(x)=(x-1)^2 *(x-3)
```

```
f(x) = (x - 1)^2 (x - 3)
```

```
I=[1,3];
a=I(1);
b=I(2);
df=diff(f,x,1);
f_a=f(a)
```

```
f_a = 0
```

```
f_b=f(b)
```

```
f_b = 0
```

```
c=solve(df==0);
c=c(a<c & c<b);
disp(c)
```

$$\frac{7}{3}$$

Aim:

To write a MATLAB program to determine $c \epsilon (a, b)$ for $f(x) = (x+2)^3(x-3)^4 \text{ in } (-2, 3)$ using Rolle's theorem

Main Commands:

syms - to define symbolic variables and fynctions

diff - to find the derivative of the function

solve - to solve the equation

format rational - to create numbers in the rational form

Source Code:

```
syms x
f(x)=(x+2)^3*(x-3)^4
```

f(x) = $(x+2)^3\ (x-3)^4$

```
I=[-2,3];
a=I(1);
b=I(2);
df=diff(f,x,1);
f_a=f(a)
```

f_a = 0

```
f_b=f(b)
```

f_b = 0

```
c=solve(df==0);
c=c(a<c & c<b);
disp(c)
```

$\frac{1}{7}$

Mean Value Theorem

Aim:

To write a MATLAB program to determine $c \in (a,b)$ for $f(x) = x + e^x \text{ in } (0,1)$ using Lagrange's mean value theorem

Main Commands:

syms - to define symbolic variables and fynctions

diff - to find the derivative of the function

solve - to solve the equation

format rational - to create numbers in the rational form

Source Code:

```
syms x
f(x)=x+exp(x)
```

f(x) = $x + \mathrm{e}^x$

```
I=[0,1];
a=I(1);
b=I(2);
df=diff(f,x,1);
m=(f(b)-f(a))/(b-a);
c=solve(df==m,x);
c=c(a<c & c<b);
disp(c)
```

$\log(\mathrm{e} - 1)$

Aim:

To write a MATLAB program to determine $c \in (a,b)$ for $f(x) = x(x-1)(x-2) \text{ in } \left(0, \frac{1}{2}\right)$ using Lagrange's mean value theorem

Main Commands:

syms - to define symbolic variables and fynctions

diff - to find the derivative of the function

solve - to solve the equation

format rational - to create numbers in the rational form

Source Code:

```
syms x
f(x)=x*(x-1)*(x-2)
```

f(x) = $x\ (x-1)\ (x-2)$

```
I=[0,0.5];
a=I(1);
b=I(2);
df=diff(f,x,1);
m=(f(b)-f(a))/(b-a);
c=solve(df==m,x);
c=c(a<c & c<b);
disp(c)
```

$$1-\frac{\sqrt{21}}{6}$$

Aim:

To write a MATLAB program to determine $c\,\epsilon\,(a,b)$ for $f(x)=(x+2)^3(x-3)^4$ in $(-2,3)$ using Lagrange's mean value theorem

Main Commands:

syms - to define symbolic variables and fynctions

diff - to find the derivative of the function

solve - to solve the equation

format rational - to create numbers in the rational form

Source Code:

```
syms x
f(x)=(x+2)^3*(x-3)^4
```

f(x) = $(x+2)^3\ (x-3)^4$

```
I=[-2,3];
a=I(1);
b=I(2);
df=diff(f,x,1);
m=(f(b)-f(a))/(b-a);
c=solve(df==m,x);
c=c(a<c & c<b);
disp(c)
```

$$\frac{1}{7}$$

Ordinary differential Equations

Aim

Solve $\frac{d^2y}{dx^2} + 4\frac{dy}{dx} - 5y = 0$

Main Commands

syms - Create symbolic variables and functions

diff - Differentiate symbolic expression or function

dsolve - Solve system of differential equations

Source Code

```
syms x y y(x)
y(x)=dsolve(diff(y,x,2)+4*diff(y,x,1)-5*y==0)
```

$y(x) = C_2\,e^{x} + C_1\,e^{-5\,x}$

Aim

Solve $\frac{d^2y}{dx^2} - 5\frac{dy}{dx} + 4y = 0$

```
syms x y y(x)
y(x)=dsolve(diff(y,x,2)-5*diff(y,x,1)+4*y==0)
```

$y(x) = C_1\,e^{x} + C_2\,e^{4\,x}$

Aim

Solve $\frac{d^2y}{dx^2} - 2\frac{dy}{dx} + 2y = 0$

```
syms x y y(x)
y(x)=dsolve(diff(y,x,2)-2*diff(y,x,1)+2*y==0)
```

$y(x) = C_1\,e^{x}\cos(x) - C_2\,e^{x}\sin(x)$

Aim

Solve $\frac{d^2y}{dx^2} - 4\frac{dy}{dx} + 4y = 0$

```
syms x y y(x)
y(x)=dsolve(diff(y,x,2)-4*diff(y,x,1)+4*y==0)
```

$y(x) = C_1\,e^{2\,x} + C_2\,x\,e^{2\,x}$

Aim

Solve $\frac{d^2y}{dx^2} - 5\frac{dy}{dx} + 4y = e^{5x}$

Main Commands

syms - Create symbolic variables and functions

diff - Differentiate symbolic expression or function

dsolve - Solve system of differential equations

Source Code

```
syms x y y(x)
y(x)=dsolve(diff(y,x,2)-5*diff(y,x,1)+4*y==exp(5*x))
```

```
y(x) =
```

$$\frac{e^{5x}}{4} + C_1 e^{x} + C_2 e^{4x}$$

Maxima and Minima

Aim:

To write a MATLAB program to find the maxima and minima of the function $f(x) = 3x^4 - 2x^3 - 6x^2 + 6x + 1.$

Main Commands:

syms - to define symbolic variables and fynctions

diff - to find the derivative of the function

solve - to solve the equation

Source Code:

```
syms x
f(x)=3*x^4-2*x^3-6*x^2+6*x+1
```

f(x) = $3\,x^4 - 2\,x^3 - 6\,x^2 + 6\,x + 1$

```
r=solve(diff(f));
r1=r(1,1)
```

r1 = -1

```
second_derivative=subs(diff(f,2),r(1,1))
```

second_derivative(x) = 36

```
if second_derivative>0
    disp('The above point is minimum point')
else if second_derivative<0
    disp('The above point is maximum point')
else
    disp('Further investigation required')
end
end
```

```
The above point is minimum point
```

```
value = subs(f,r(1,1))
```

value(x) = -6

```
r2=r(2,1)
```

r2 =

$\frac{1}{2}$

```
second_derivative=subs(diff(f,2),r(2,1))
```

second_derivative(x) = -9

```
if second_derivative>0
    disp('The above point is minimum point')
else if second_derivative<0
    disp('The above point is maximum point')
else
    disp('Further investigation required')
end
end
```

The above point is maximum point

```
value = subs(f,r(2,1))
```

value(x) =

$$\frac{39}{16}$$

```
r3=r(3,1)
```

r3 = 1

```
second_derivative=subs(diff(f,2),r(3,1))
```

second_derivative(x) = 12

```
if second_derivative>0
    disp('The above point is minimum point')
else if second_derivative<0
    disp('The above point is maximum point')
else
    disp('Further investigation required')
end
end
```

The above point is minimum point

```
value = subs(f,r(3,1))
```

value(x) = 2

Aim:

To write a MATLAB program to find the maxima and minima of the function $f(x) = 3x^4 - 2x^3 - 6x^2 + 6x + 1.$

Main Commands:

syms - to define symbolic variables and fynctions

diff - to find the derivative of the function

solve - to solve the equation

Source Code:

```
syms x
f(x)=4*x^3-16*x^2+12*x
```

f(x) = $4\,x^3-16\,x^2+12\,x$

```
r=solve(diff(f));
r1=r(1,1)
```

r1 =

$$\frac{4}{3}-\frac{\sqrt{7}}{3}$$

```
second_derivative=subs(diff(f,2),r(1,1))
```

second_derivative(x) = $-8\,\sqrt{7}$

```
if second_derivative > 0
    disp('The above point is minimum point')
else if second_derivative < 0
    disp('The above point is maximum point')
else
    disp('Further investigation required')
end
end
```

```
The above point is maximum point
```

```
value = subs(f,r(1,1))
```

value(x) =

$$16-16\left(\frac{\sqrt{7}}{3}-\frac{4}{3}\right)^2-4\left(\frac{\sqrt{7}}{3}-\frac{4}{3}\right)^3-4\,\sqrt{7}$$

```
r2=r(2,1)
```

r2 =

$$\frac{\sqrt{7}}{3}+\frac{4}{3}$$

```
second_derivative=subs(diff(f,2),r(2,1))
```

second_derivative(x) = $8\,\sqrt{7}$

```
if second_derivative > 0
    disp('The above point is minimum point')
else if second_derivative < 0
    disp('The above point is maximum point')
else
    disp('Further investigation required')
```

```
end
end
```

The above point is minimum point

```
value = subs(f,r(2,1))
```

value(x) =

$$4\sqrt{7} - 16\left(\frac{\sqrt{7}}{3}+\frac{4}{3}\right)^2 + 4\left(\frac{\sqrt{7}}{3}+\frac{4}{3}\right)^3 + 16$$

Limit of a function

Aim:

To write a MATLAB program to find $\lim_{\substack{x\longrightarrow 1\\ y\longrightarrow 2}}(x^2+y^2)$

Main Commands:

syms - to define symbolic variables and fynctions

limit - to find the limit of the function

Source Code:

```
syms x y f g
f=x^2+y^2
```

f = x^2+y^2

```
%limit_value=limit(limit(f,x,1),y,2)
```

Aim:

To write a MATLAB program to find $\lim_{\substack{x\longrightarrow 1\\ y\longrightarrow 2}}\left(\frac{2x^2y}{x^2+y^2+1}\right)$

Main Commands:

syms - to define symbolic variables and fynctions

limit - to find the limit of the function

Source Code:

```
syms x y
f=(2*x^2*y)/(x^2+y^2+1)
```

f =

$$\frac{2\,x^2\,y}{x^2+y^2+1}$$

```
%limit_value=limit(limit(f,x,1),y,2)
```

Derivatives of a function

Aim:

To write a MATLAB program to find the first and second order derivatives of the function $z = x^3 + y^3 - 3axy$.

Main Commands:

syms - to define symbolic variables and fynctions

diff - to find the derivative of the function

Source Code:

```
syms x y a
z=x^3+y^3-3*a*x*y
```

z = $x^3 - 3\,a\,x\,y + y^3$

```
disp('First derivative with respect to x')
```

```
First derivative with respect to x
```

```
diff(z,x,1)
```

ans = $3\,x^2 - 3\,a\,y$

```
disp('First derivative with respect to y')
```

```
First derivative with respect to y
```

```
diff(z,y,1)
```

ans = $3\,y^2 - 3\,a\,x$

```
disp('Second derivative with respect to x')
```

```
Second derivative with respect to x
```

```
diff(z,x,2)
```

ans = $6\,x$

```
disp('Second derivative with respect to y')
```

```
Second derivative with respect to y
```

```
diff(z,y,2)
```

ans = $6\,y$

```
disp('Second derivative with respect to x & y')
```

```
Second derivative with respect to x & y
```

```
diff(diff(z,x,1),y,1)
```

ans = $-3\,a$

Aim:

To write a MATLAB program to find the first and second order derivatives of the function $z = x^4 + y^4 - 2x^2 - 2y^2 + 4xy.$

Main Commands:

syms - to define symbolic variables and fynctions

diff - to find the derivative of the function

Source Code:

```
syms x y a
z=x^4+y^4-2*x^2-2*y^2+4*x*y
```

z = $x^4 - 2\,x^2 + 4\,x\,y + y^4 - 2\,y^2$

```
disp('First derivative with respect to x')
```

```
First derivative with respect to x
```

```
diff(z,x,1)
```

ans = $4\,x^3 - 4\,x + 4\,y$

```
disp('First derivative with respect to y')
```

```
First derivative with respect to y
```

```
diff(z,y,1)
```

ans = $4\,y^3 - 4\,y + 4\,x$

```
disp('Second derivative with respect to x')
```

```
Second derivative with respect to x
```

```
diff(z,x,2)
```

ans = $12\,x^2 - 4$

```
disp('Second derivative with respect to y')
```

```
Second derivative with respect to y
```

```
diff(z,y,2)
```

ans = $12\,y^2 - 4$

```
disp('Second derivative with respect to x & y')
```

Second derivative with respect to x & y

```
diff(diff(z,x,1),y,1)
```

ans = 4

Aim:

To write a MATLAB program to find the first and second order derivatives of the function $z = x^3 + 3xy^2 + 15x^2 - 15y^2 + 72x.$

Main Commands:

syms - to define symbolic variables and fynctions

diff - to find the derivative of the function

Source Code:

```
syms x y a
z=x^3+3*x*y^2+15*x^2-15*y^2+72*x
```

z = $x^3 + 15\,x^2 + 3\,x\,y^2 + 72\,x - 15\,y^2$

```
disp('First derivative with respect to x')
```

First derivative with respect to x

```
diff(z,x,1)
```

ans = $3\,x^2 + 30\,x + 3\,y^2 + 72$

```
disp('First derivative with respect to y')
```

First derivative with respect to y

```
diff(z,y,1)
```

ans = $6\,x\,y - 30\,y$

```
disp('Second derivative with respect to x')
```

Second derivative with respect to x

```
diff(z,x,2)
```

ans = $6\,x + 30$

```
disp('Second derivative with respect to y')
```

Second derivative with respect to y

```
diff(z,y,2)
```

ans = $6\,x - 30$

```
disp('Second derivative with respect to x & y')
```

```
Second derivative with respect to x & y
```

```
diff(diff(z,x,1),y,1)
```

ans = $6y$

Beta and Gamma functions

Aim:

To write a MATLAB program to find $\Gamma(1)$

Main Commands:

syms - to define symbolic variables and fynctions

gamma - to gamma function values

Source Code:

```
syms y
y = gamma(1)
```

```
y =
     1
```

Aim:

To write a MATLAB program to find $\Gamma(10)$

Main Commands:

syms - to define symbolic variables and fynctions

gamma - to gamma function values

Source Code:

```
syms y
y = gamma(10)
```

```
y =
  362880
```

Aim:

To write a MATLAB program to find $\Gamma(0.5)$

Main Commands:

syms - to define symbolic variables and fynctions

gamma - to gamma function values

Source Code:

```
syms y
y = gamma(0.5)
```

```
y =
   296/167
```

Aim:

To write a MATLAB program to find $\beta(1,5)$

Main Commands:

syms - to define symbolic variables and fynctions

gamma - to gamma function values

Source Code:

```
syms x
x=beta(1, 5)
```

```
x =
       1/5
```

Aim:

To write a MATLAB program to find $\beta(1,5)$

Main Commands:

syms - to define symbolic variables and fynctions

gamma - to gamma function values

Source Code:

```
syms x
x=beta(3, sqrt(2))
```

```
x =
     204/1189
```

Aim:

To write a MATLAB program to find $\beta(1,5)$

Main Commands:

syms - to define symbolic variables and fynctions

gamma - to gamma function values

Source Code:

```
syms x
x=beta(pi, exp(1))
```

```
x =
     125/3299
```

Aim:

To write a MATLAB program to find $\beta(1,5)$

Main Commands:

syms - to define symbolic variables and fynctions

gamma - to gamma function values

Source Code:

```
syms x
x=beta(0, 1)
```

```
x =
       1/0
```

COMPUTATION OF INDEFINITE INTEGRAL

Aim

To evaluate $\int \sin x \, dx$

Main Commands

syms - Create symbolic variables and functions

int - to integrate the function or variable

Source Code

```
syms x
f= sin(x);
I=int(f,x)
```

I = $-\cos(x)$

Aim

To evaluate $\int \cos x \, dx$

Main Commands

syms - Create symbolic variables and functions

int - to integrate the function or variable

Source Code

```
syms x
f= cos(x);
I=int(f,x)
```

I = $\sin(x)$

Aim

To evaluate $\int \tan x \, dx$

Main Commands

syms - Create symbolic variables and functions

int - to integrate the function or variable

Source Code

```
syms x
I=int(tan(x),x)
```

I = $-\log(\cos(x))$

Aim

To evaluate $\int e^{ax}\,dx$

Main Commands

syms - Create symbolic variables and functions

int - to integrate the function or variable

Source Code

```
syms a x
I=int(exp(a*x),x)
```

```
I =
```

$$\frac{e^{a\,x}}{a}$$

Aim

To evaluate $\int x^{3}\,dx$

Main Commands

syms - Create symbolic variables and functions

int - to integrate the function or variable

Source Code

```
syms x
I=int(x^3,x)
```

```
I =
```

$$\frac{x^4}{4}$$

COMPUTATION OF DEFINITE INTEGRAL

Aim

To evaluate $\int_0^{\frac{\pi}{2}} \sin x \ dx$

Main Commands

syms - Create symbolic variables and functions

int - to integrate the function or variable

Source Code

```
syms x
I=int(sin(x),x,0,pi/2)
```

```
I = 1
```

Aim

To evaluate $\int_0^{\frac{\pi}{2}} \cos^6 x \ dx$

Main Commands

syms - Create symbolic variables and functions

int - to integrate the function or variable

Source Code

```
syms x
I=int(cos(x)^6,x,0,pi/2)
```

```
I =
```

$$\frac{5\,\pi}{32}$$

Aim

To evaluate $\int_1^5 x^3 \ dx$

Main Commands

syms - Create symbolic variables and functions

int - to integrate the function or variable

Source Code

```
syms x
I=int(x^3,x,1,5)
```

```
I = 156
```

Aim

To evaluate $\int_0^a \sqrt{a^2 - x^2}\ \mathrm{dx}$

Main Commands

syms - Create symbolic variables and functions

int - to integrate the function or variable

Source Code

```
syms a x
I=int(sqrt(a^2-x^2),x,0,a)
```

```
I =
```

$$\frac{\pi\, a^2}{4}$$

Aim

To evaluate $\int_0^a \frac{\mathrm{dx}}{\sqrt{a^2 - x^2}}$

Main Commands

syms - Create symbolic variables and functions

int - to integrate the function or variable

Source Code

```
syms a x
I=int(1/sqrt(a^2-x^2),x,0,a)
```

```
I =
```

$$\frac{\pi}{2}$$

DOUBLE INTEGRAL (RECTAGULAR COORDINATES)

Aim

To evaluate $\int_0^1 \int_1^2 (x^2 + y^2)\ \mathrm{dydx}$

Main Commands

syms - Create symbolic variables and functions

int - to integrate the function or variable

Source Code

```
syms x y
I=int(int(x^2+y^2,y,1,2),x,0,1)
```

```
I =
```

$$\frac{8}{3}$$

Aim

To evaluate $\int_0^1 \int_{x^2}^{2-x} \mathrm{xy}\ \mathrm{dydx}$

Main Commands

syms - Create symbolic variables and functions

int - to integrate the function or variable

Source Code

```
syms x y
I=int(int(x*y,y,x^2,2-x),x,0,1)
```

```
I =
```

$$\frac{3}{8}$$

Aim

To evaluate $\int_0^1 \int_y^1 (x^2y^2)\ \mathrm{dxdy}$

Main Commands

syms - Create symbolic variables and functions

int - to integrate the function or variable

Source Code

```
syms x y
```

```
I=int(int(x^2*y^2,x,y,1),y,0,1)
```

```
I =
```

$$\frac{1}{18}$$

Aim

To find the area of the circle $x^2 + y^2 = a^2$

Main Commands

syms - Create symbolic variables and functions

int - to integrate the function or variable

Source Code

```
syms x y a
I=4*int(int(1,y,0,sqrt(a^2-x^2)),x,0,a)
```

```
I =
```

$\pi\, a^2$

Aim

To find the area between the parabolas $x^2 = 4\mathrm{ay}$ and $y^2 = 4\mathrm{ax}$

Main Commands

syms - Create symbolic variables and functions

int - to integrate the function or variable

Source Code

```
syms x y a
I=int(int(1,y,x^2/(4*a),2*sqrt(a*x)),x,0,4*a)
```

```
I =
```

$$\frac{16\, a^2}{3}$$

DOUBLE INTEGRAL (POLAR COORDINATES)

Aim

To evaluate $\int\int r^3 \, dr \, dt$ over the region between the circles $r = 2 \sin t$ and $r = 4 \sin t$.

Main Commands

syms - Create symbolic variables and functions

int - to integrate the function or variable

Source Code

```
syms r t
I=int(int(r^3,r,2*sin(t),4*sin(t)),t,0,pi)
```

```
I =
```

$$\frac{45\,\pi}{2}$$

Aim

To evaluate $\int\int r^3 \, dr \, dt$ over the region between the circles $r = 2 \cos t$ and $r = 4 \cos t$.

Main Commands

syms - Create symbolic variables and functions

int - to integrate the function or variable

Source Code

```
syms r t
I=2*int(int(r^3,r,2*cos(t),4*cos(t)),t,0,pi/2)
```

```
I =
```

$$\frac{45\,\pi}{2}$$

Aim

To evaluate $\int\int r \sin t \, dr \, dt$ over the upper half of the region bounded by the cardioid $r = a(1 - \cos t)$

Main Commands

syms - Create symbolic variables and functions

int - to integrate the function or variable

Source Code

```
syms a r t
I=int(int(r*sin(t),r,0,a*(1-cos(t))),t,0,pi)
```

```
I =
```

$$\frac{4\,a^2}{3}$$

Aim

To evaluate $\int\int r \sin t \, dr \, dt$ over the upper half of the region bounded by the cardioid $r = a(1 + \cos t)$

Main Commands

syms - Create symbolic variables and functions

int - to integrate the function or variable

Source Code

```
syms a r t
I=int(int(r*sin(t),r,0,a*(1+cos(t))),t,0,pi)
```

```
I =
```

$$\frac{4\,a^2}{3}$$

Aim

To find the area lying outside the corcle $r = a$ and inside the cardioid $r = a(1 + \cos t)$

Main Commands

syms - Create symbolic variables and functions

int - to integrate the function or variable

Source Code

```
syms a r t
I=int(int(r,r,a,a*(1+cos(t))),t,0,2*pi)
```

```
I =
```

$$\frac{\pi\,a^2}{2}$$

CHANGE OF ORDER OF INTEGRATION

Aim

Write a MATLAB program to change the order of integration and evaluate $\int_0^a \int_0^{\sqrt{a^2-x^2}} \mathrm{dydx}$.

Main Commands

syms - Create symbolic variables and functions

int - to integrate the function or variable

Source Code

```
syms x y a
I=int(int(1,x,0,sqrt(a^2-y^2)),y,0,a)
```

```
I =
```

$$\frac{\pi\, a^2}{4}$$

Aim

Write a MATLAB program to change the order of integration and evaluate $\int_0^\infty \int_x^\infty \frac{e^{-y}}{y} \mathrm{dydx}$.

Main Commands

syms - Create symbolic variables and functions

int - to integrate the function or variable

Source Code

```
syms x y
I=int(int((exp(-y))/y,x,0,y),y,0,inf)
```

```
I = 1
```

Aim

Write a MATLAB program to change the order of integration and evaluate $\int_0^a \int_y^a \frac{x}{x^2+y^2} \mathrm{dxdy}$.

Main Commands

syms - Create symbolic variables and functions

int - to integrate the function or variable

Source Code

```
syms x y a
I=int(int(x/(x^2+y^2),y,0,x),x,0,a)
```

```
I =
```

$$\frac{\pi a}{4}$$

Aim

Write a MATLAB program to change the order of integration and evaluate $\int_0^4 \int_{\frac{x^2}{4}}^{2\sqrt{x}} xy \, dydx$.

Main Commands

syms - Create symbolic variables and functions

int - to integrate the function or variable

Source Code

```
syms x y
I=int(int(x*y,x,y^2/4,2*sqrt(y)),y,0,4)
```

```
I =
```

$$\frac{64}{3}$$

Aim

Write a MATLAB program to change the order of integration and evaluate $\int_0^1 \int_{x^2}^{2-x} xy \, dydx$.

Main Commands

syms - Create symbolic variables and functions

int - to integrate the function or variable

Source Code

```
syms x y
I=int(int(x*y,x,0,sqrt(y)),y,0,1)+int(int(x*y,x,0,2-y),y,1,2)
```

```
I =
```

$$\frac{3}{8}$$

TRIPLE INTEGRAL

Aim

To evaluate $\int_0^a \int_0^b \int_0^c (x^2 + y^2 + z^2)\, dz\, dy\, dx$

Main Commands

syms - Create symbolic variables and functions

int - to integrate the function or variable

Source Code

```
syms x y z a b c
I=int(int(int(x^2+y^2+z^2,z,0,c),y,0,b),x,0,a)
```

```
I =
```

$$\frac{a\,b\,c\,(a^2 + b^2 + c^2)}{3}$$

Aim

To evaluate $\int_0^1 \int_0^2 \int_0^3 x^2 yz\, dz\, dy\, dx$

Main Commands

syms - Create symbolic variables and functions

int - to integrate the function or variable

Source Code

```
syms x y z
I=int(int(int(x^2*y*z,z,0,3),y,0,2),x,0,1)
```

```
I =
```

$$3$$

Aim

To find the volume of the tetrahedron bounded by $x = 0, y = 0, z = 0$ and $x + y + z = a$

Main Commands

syms - Create symbolic variables and functions

int - to integrate the function or variable

Source Code

```
syms x y z a
I=int(int(int(1,z,0,a-x-y),y,0,a-x),x,0,a)
```

```
I =
```

$\frac{a^3}{6}$

Aim

To find the volume of the cube bounded by $x = 0, y = 0, z = 0$ and $x = a, y = a, z = a$

Main Commands

syms - Create symbolic variables and functions

int - to integrate the function or variable

Source Code

```
syms x y z a
I=int(int(int(1,z,0,a),y,0,a),x,0,a)
```

I = a^3

Aim

To find the volume of the rectangular box bounded by $x = 0, y = 0, z = 0$ and $x = a, y = b, z = c$

Main Commands

syms - Create symbolic variables and functions

int - to integrate the function or variable

Source Code

```
syms x y z a
I=int(int(int(1,z,0,c),y,0,b),x,0,a)
```

I = $a\,b\,c$

Aim

To find the volume of the sphere $x^2 + y^2 + z^2 = a^2$

Main Commands

syms - Create symbolic variables and functions

int - to integrate the function or variable

Source Code

```
syms x y z a
I=8*int(int(int(1,z,0,sqrt(a^2-x^2-y^2)),y,0,sqrt(a^2-x^2)),x,0,a)
```

I =

$\frac{4\,\pi\,a^3}{3}$

VECTOR CALCULUS

GRADIENT

Aim

To find the gradient of $f = x^2 + y^2 - z - 1$

Main Commands

syms - Create symbolic variables and functions

gradient - to find the gradient of a scalar point function

disp - to display the things

Source Code

```
syms x y z
f= x^2+y^2-z-1;
grad_f=gradient(f);
display('grad(f)=');
```

```
grad(f)=
```

```
disp(grad_f')
```

$$(2\,\overline{x}\quad 2\,\overline{y}\quad -1)$$

Aim

To find the gradient of $f = x^2 + y^2 + z^3 - 1$ at (1,2,3)

Main Commands

syms - Create symbolic variables and functions

gradient - to find the gradient of a scalar point function

disp - to display the things

subs - to substitue the values

Source Code

```
syms x y z
f= x^2+y^2+z^3-1;
grad_f=subs(gradient(f),{x,y,z},{1,2,3});
display('grad(f)=');
```

```
grad(f)=
```

```
disp(grad_f')
```

$$(2\quad 4\quad 27)$$

Aim

To find the gradient of $f = x^2 - y + z^3 - 1$ at (1,0,1)

Main Commands

syms - Create symbolic variables and functions

gradient - to find the gradient of a scalar point function

disp - to display the things

subs - to substitue the values

Source Code

```
syms x y z
f= x^2-y+z^3-1;
grad_f=subs(gradient(f),{x,y,z},{1,0,1});
display('grad(f)=');
```

```
grad(f)=
```

```
disp(grad_f')
```

$$\begin{pmatrix} 2 & -1 & 3 \end{pmatrix}$$

DIVERGENCE

Aim

To find the divergence of $\vec{F} = x^2 y\,\vec{i} - 2xz\,\vec{j} + 2yz\,\vec{k}$

Main Commands

syms - Create symbolic variables and functions

divergence - to find the divergence of a vector

disp - to display the things

Source Code

```
syms x y z
F= [x^2*y,-2*x*z,2*y*z];
div_F=divergence(F);
display('div(F)=');
```

```
div(F)=
```

```
disp(div_F)
```

$$2y + 2xy$$

Aim

To find the divergence of $\vec{F} = x^2y\,\vec{i} - 2xz\,\vec{j} + 2yz\,\vec{k}$ at (1,2,1)

Main Commands

syms - Create symbolic variables and functions

divergence - to find the divergence of a vector

disp - to display the things

subs - to substitue the values

Source Code

```
syms x y z
F= [x^2*y,-2*x*z,2*y*z];
div_F=subs(divergence(F),{x,y,z},{1,2,1});
display('div(F)=');
```

```
div(F)=
```

```
disp(div_F)
```

$$8$$

CURL

Aim

To find the curl of $\vec{F} = x^2y\,\vec{i} - 2xz\,\vec{j} + 2yz\,\vec{k}$

Main Commands

syms - Create symbolic variables and functions

curl - to find the curl of a vector

disp - to display the things

Source Code

```
syms x y z
F= [x^2*y,-2*x*z,2*y*z];
curl_F=curl(F);
display('curl(F)=');
```

```
curl(F)=
```

```
disp(curl_F')
```

$$(2\,\overline{x} + 2\,\overline{z} \quad 0 \quad -\overline{x}^2 - 2\,\overline{z})$$

Aim

To find the curl of $\vec{F} = x^2y\,\vec{i} - 2xz\,\vec{j} + 2yz\,\vec{k}$ at (1,2,3)

Main Commands

syms - Create symbolic variables and functions

curl - to find the curl of a vector

disp - to display the things

subs - to substitue the values

Source Code

```
syms x y z
F= [x^2*y,-2*x*z,2*y*z];
curl_F=subs(curl(F),{x,y,z},{1,2,3});
display('curl(F)=');
```

```
curl(F)=
```

```
disp(curl_F')
```

```
(8   0   -7)
```

Aim

To find the curl of $\vec{F} = x^2y^2\,\vec{i} + 2xz\,\vec{j} + 2yz\,\vec{k}$ at (1,2,3)

Main Commands

syms - Create symbolic variables and functions

curl - to find the curl of a vector

disp - to display the things

subs - to substitue the values

Source Code

```
syms x y z
F= [x^2*y^2,2*x*z,2*y*z];
curl_F=subs(curl(F),{x,y,z},{1,2,3});
display('curl(F)=');
```

```
curl(F)=
```

```
disp(curl_F')
```

```
(4   0   2)
```

ANALYTIC FUNCTIONS

Aim

To test whether $f(z) = z$ is analytic or not

Main Commands

syms - Create symbolic variables and functions

diff - Differentiate symbolic expression or function

simplify - to simplify the values

Source Code

```
syms x y
u=x;
v=y;
condition1=simplify(diff(u,x,1)-diff(v,y,1));
condition2=simplify(diff(u,y,1)+diff(v,x,1));
if (condition1==0 && condition2==0)
disp('f(z) is analytic');
else
disp('f(z) is not analytic');
end
```

```
f(z) is analytic
```

Aim

To test whether $f(z) = |z|^2$ is analytic or not

Main Commands

syms - Create symbolic variables and functions

diff - Differentiate symbolic expression or function

simplify - to simplify the values

Source Code

```
syms x y
u=x^2+y^2;
v=0;
condition1=simplify(diff(u,x,1)-diff(v,y,1));
condition2=simplify(diff(u,y,1)+diff(v,x,1));
if (condition1==0 && condition2==0)
disp('f(z) is analytic');
else
disp('f(z) is not analytic');
end
```

```
f(z) is not analytic
```

Aim

To test whether $f(z) = (x^2 - y^2) + i\,2\,x\,y$ is analytic or not

Main Commands

syms - Create symbolic variables and functions

diff - Differentiate symbolic expression or function

simplify - to simplify the values

Source Code

```
syms x y
u=x^2-y^2;
v=2*x*y;
condition1=simplify(diff(u,x,1)-diff(v,y,1));
condition2=simplify(diff(u,y,1)+diff(v,x,1));
if (condition1==0 && condition2==0)
disp('f(z) is analytic');
else
disp('f(z) is not analytic');
end
```

```
f(z) is analytic
```

Aim

To test whether $f(z) = 2\,x\,y\, + i(x^2 - y^2)$ is analytic or not

Main Commands

syms - Create symbolic variables and functions

diff - Differentiate symbolic expression or function

simplify - to simplify the values

Source Code

```
syms x y
u=2*x*y;
v=x^2-y^2;
condition1=simplify(diff(u,x,1)-diff(v,y,1));
condition2=simplify(diff(u,y,1)+diff(v,x,1));
if (condition1==0 && condition2==0)
disp('f(z) is analytic');
else
disp('f(z) is not analytic');
end
```

```
f(z) is not analytic
```

Aim

To test whether $f(z) = \frac{x^2 - y^2}{(x^2 + y^2)^2} - i\frac{2xy}{(x^2 + y^2)^2}$ is analytic or not

Main Commands

syms - Create symbolic variables and functions

diff - Differentiate symbolic expression or function

simplify - to simplify the values

Source Code

```
syms x y
u=(x^2-y^2)/((x^2+y^2)^2);
v=(-2*x*y)/((x^2+y^2)^2);
condition1=simplify(diff(u,x,1)-diff(v,y,1));
condition2=simplify(diff(u,y,1)+diff(v,x,1));
if (condition1==0 && condition2==0)
disp('f(z) is analytic');
else
disp('f(z) is not analytic');
end
```

```
f(z) is analytic
```

Aim

To test whether $f(z) = e^z$ is analytic or not

Main Commands

syms - Create symbolic variables and functions

diff - Differentiate symbolic expression or function

simplify - to simplify the values

Source Code

```
syms x y
u=exp(x)*cos(y);
v=exp(x)*sin(y);
condition1=simplify(diff(u,x,1)-diff(v,y,1));
condition2=simplify(diff(u,y,1)+diff(v,x,1));
if (condition1==0 && condition2==0)
disp('f(z) is analytic');
else
disp('f(z) is not analytic');
end
```

```
f(z) is analytic
```

HARMONIC FUNCTIONS

Aim

To test whether $u = \frac{1}{2}\log(x^2 + y^2)$ is harmonic or not

Main Commands

syms - Create symbolic variables and functions

diff - Differentiate symbolic expression or function

Source Code

```
syms x y
u=(1/2)*log(x^2+y^2);
condition=simplify(diff(u,x,2)+diff(u,y,2));
if condition==0
disp('u is harmonic');
else
disp('u is not harmonic');
end
```

```
u is harmonic
```

Aim

To test whether $u = e^{-2x}\cos 2y$ is harmonic or not

Main Commands

syms - Create symbolic variables and functions

diff - Differentiate symbolic expression or function

Source Code

```
syms x y
u=exp(-2*x)*cos(2*y);
condition=simplify(diff(u,x,2)+diff(u,y,2));
if condition==0
disp('u is harmonic');
else
disp('u is not harmonic');
end
```

```
u is harmonic
```

Aim

To test whether $u = e^{x^2-y^2}\cos 2xy$ is harmonic or not

Main Commands

syms - Create symbolic variables and functions

diff - Differentiate symbolic expression or function

Source Code

```
syms x y
u=exp(x^2-y^2)*cos(2*x*y);
condition=simplify(diff(u,x,2)+diff(u,y,2));
if condition==0
disp('u is harmonic');
else
disp('u is not harmonic');
end
```

```
u is harmonic
```

Aim

To test whether $u = x^2 + y^2 + 16xy$ is harmonic or not

Main Commands

syms - Create symbolic variables and functions

diff - Differentiate symbolic expression or function

Source Code

```
syms x y
u=x^2+y^2+16*x*y;
condition=simplify(diff(u,x,2)+diff(u,y,2));
if condition==0
disp('u is harmonic');
else
disp('u is not harmonic');
end
```

```
u is not harmonic
```

BILINEAR TRANSFORM

Aim

To obtain the bilinear transformation which maps the points $z = 1,\ i,\ -1$ into the points $w = 0,\ 1, \infty$.

Main Commands

syms - Create symbolic variables and functions

det - to find the determinant

inv - to find the inverse

Source Code

```
syms z w
z(1)=1;
z(2)=i;
z(3)=-1;
w(1)=0;
w(2)=1;
w(3)=inf;
if w(1)==inf, W = [0 w(3)-w(2); -1 w(3)];
elseif w(2)==inf, W = [1 -w(1); 1 -w(3)];
elseif w(3)==inf, W = [-1 w(1); 0 w(1)-w(2)];
else W=[w(2)-w(3) (w(3)-w(2))*w(1); w(2)-w(1) (w(1)-w(2))*w(3)];
end
if z(1)==inf, Z=[0 z(3)-z(2); -1 z(3)];
elseif z(2)==inf, Z = [1 -z(1); 1 -z(3)];
elseif z(3)==inf, Z = [-1 z(1); 0 z(1)-z(2)];
else Z=[z(2)-z(3) (z(3)-z(2))*z(1); z(2)-z(1) (z(1)-z(2))*z(3)];
end
A=det(W)*inv(W)*Z;
A=A/A(1,1);
syms z;
w = ((A(1,1)*z + A(1,2)) / (A(2,1)*z +A(2,2)))
```

```
w =
```

$$\frac{z-1}{z\,\mathrm{i}+\mathrm{i}}$$

Aim

To obtain the bilinear transformation which maps the points $z = 0,\ -1, i$ into the points $w = i,\ 0, \infty$.

Main Commands

syms - Create symbolic variables and functions

det - to find the determinant

inv - to find the inverse

Source Code

```
syms z w
z(1)=0;
z(2)=-1;
z(3)=i;
w(1)=i;
w(2)=0;
w(3)=inf;
if w(1)==inf, W = [0 w(3)-w(2); -1 w(3)];
elseif w(2)==inf, W = [1 -w(1); 1 -w(3)];
elseif w(3)==inf, W = [-1 w(1); 0 w(1)-w(2)];
else W=[w(2)-w(3) (w(3)-w(2))*w(1); w(2)-w(1) (w(1)-w(2))*w(3)];
end
if z(1)==inf, Z=[0 z(3)-z(2); -1 z(3)];
elseif z(2)==inf, Z = [1 -z(1); 1 -z(3)];
elseif z(3)==inf, Z = [-1 z(1); 0 z(1)-z(2)];
else Z=[z(2)-z(3) (z(3)-z(2))*z(1); z(2)-z(1) (z(1)-z(2))*z(3)];
end
A=det(W)*inv(W)*Z;
A=A/A(1,1);
syms z;
w = ((A(1,1)*z + A(1,2)) / (A(2,1)*z +A(2,2)))
```

```
w =
```

$$\frac{z+1}{z-\mathrm{i}}$$

Aim

To obtain the bilinear transformation which maps the points $z = -1,\ 0, 1$ into the points $w = 0,\ i, 3i.$

Main Commands

syms - Create symbolic variables and functions

det - to find the determinant

inv - to find the inverse

Source Code

```
syms z w
z(1)=-1;
z(2)=0;
z(3)=1;
w(1)=0;
w(2)=i;
w(3)=3i;
if w(1)==inf, W = [0 w(3)-w(2); -1 w(3)];
elseif w(2)==inf, W = [1 -w(1); 1 -w(3)];
elseif w(3)==inf, W = [-1 w(1); 0 w(1)-w(2)];
else W=[w(2)-w(3) (w(3)-w(2))*w(1); w(2)-w(1) (w(1)-w(2))*w(3)];
end
if z(1)==inf, Z=[0 z(3)-z(2); -1 z(3)];
elseif z(2)==inf, Z = [1 -z(1); 1 -z(3)];
```

```
elseif z(3)==inf, Z = [-1 z(1); 0 z(1)-z(2)];
else Z=[z(2)-z(3) (z(3)-z(2))*z(1); z(2)-z(1) (z(1)-z(2))*z(3)];
end
A=det(W)*inv(W)*Z;
A=A/A(1,1);
syms z;
w = ((A(1,1)*z + A(1,2)) / (A(2,1)*z +A(2,2)))
```

```
w =
```

$$\frac{z+1}{\frac{z\,\mathrm{i}}{3}-\mathrm{i}}$$

Aim

To obtain the bilinear transformation which maps the points $z = 1,\ i, -1$ into the points $w = i,\ 0, -i.$

Main Commands

syms - Create symbolic variables and functions

det - to find the determinant

inv - to find the inverse

Source Code

```
syms z w
z(1)=1;
z(2)=i;
z(3)=-1;
w(1)=i;
w(2)=0;
w(3)=-i;
if w(1)==inf, W = [0 w(3)-w(2); -1 w(3)];
elseif w(2)==inf, W = [1 -w(1); 1 -w(3)];
elseif w(3)==inf, W = [-1 w(1); 0 w(1)-w(2)];
else W=[w(2)-w(3) (w(3)-w(2))*w(1); w(2)-w(1) (w(1)-w(2))*w(3)];
end
if z(1)==inf, Z=[0 z(3)-z(2); -1 z(3)];
elseif z(2)==inf, Z = [1 -z(1); 1 -z(3)];
elseif z(3)==inf, Z = [-1 z(1); 0 z(1)-z(2)];
else Z=[z(2)-z(3) (z(3)-z(2))*z(1); z(2)-z(1) (z(1)-z(2))*z(3)];
end
A=det(W)*inv(W)*Z;
A=A/A(1,1);
syms z;
w = ((A(1,1)*z + A(1,2)) / (A(2,1)*z +A(2,2)))
```

```
w =
```

$$-\frac{z-\mathrm{i}}{z+\mathrm{i}}$$

REFERENCES:

1. Won Y. Yang, Young K. Choi, Jaekwon Kim, Man cheol Kim, H. Jin Kim, Taeho Im, Engineering Mathematics with MATLAB, CRC Press, Taylor & Francis Group, 2018.

2. Dean G. Duffy, Advanced Engineering Mathematics with MATLAB, CRC Press, Taylor & Francis Group, 2017.

3. Paul Zimmermann, Alexandre Casamayou, Nathann Cohen, Guillaume Connan, Thierry Dumont, Laurent Fousse, François Maltey, Matthias Meulien, Marc Mezzarobba, Clement Pernet, Nicolas M. Thiéry, Erik Bray, John Cremona, Marcelo Forets, Alexandru Ghitza and Hugh Thomas, Computational Mathematics with SAGEMATH, 2018.

4. Sandeep Nagar, Introduction to Scilab for Engineers and Scientists, Apress, 2017.

5. https://in.mathworks.com/help/matlab/

6. http://dl.lateralis.org/public/sagebook/sagebook-ba6596d.pdf

7. https://www.scilab.org/

8. http://www.sagemath.org/

9. A MATLAB ® Companion to Complex Variables

REFERENCE LINKS:

https://help.scilab.org/doc/5.5.2/en_US/sum.html

http://www.srmuniv.ac.in/sites/default/files/2017/cse-lab-manual-scilab.pdf http://doc.sagemath.org/pdf/en/tutorial/SageTutorial.pdf

https://cloud.scilab.in/

https://in.mathworks.com/help/matlab/ref/polyint.html

https://www.scilab.org/tutorials/getting-started/plotting#sec2

http://www.ee.iitm.ac.in/~hsr/ec205/

http://portal.unimap.edu.my/portal/page/portal30/Lecture%20Notes/KEJURUTERAAN_MIKRO ELEKTRONIK/1/EKT%20230%20Signal%20and%20System/Lab/Lab5_Laplace%20Transform %2 0and%20Z-Transform_full%5B1%5D.pdf