INSTITUTE OF TECHNOLOGY
& MANAGEMENT
GWALIOR • MP • INDIA

Laboratory Manual

Matlab
(IT-406)

For

Second Year Students
Department of Information Technology

Department of Information Technology

Vision of IT Department

The department envisions to nurture students to become technologically proficient, research competent and socially accountable for the welfare of the society.

Mission of the IT Department

- To provide high quality education through effective teaching-learning process emphasizing active participation of students.
- To build scientifically strong engineers to cater to the needs of industry, higher studies, research and startups.
- To awaken young minds ingrained with ethical values and professional behaviors for the betterment of the society.

Programme Educational Objectives

Graduates will be able to

- Our engineers will demonstrate application of comprehensive technical knowledge for innovation and entrepreneurship.
- Our graduates will employ capabilities of solving complex engineering problems to succeed in research and/or higher studies.
- Our graduates will exhibit team-work and leadership qualities to meet stakeholder business objectives in their careers.
- Our graduates will evolve in ethical and professional practices and enhance socioeconomic contributions to the society.

Programme Outcomes (POs)

Engineering Graduates will be able to:

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

Course Outcomes

Course	Course Outcomes
IT406.1	To study MATLAB environment, data types, variables, operators and assignment statements in MATLAB and able to use MATLAB for interactive computations.
IT406.2	To implement the arithmetic operations and familiar with vector, matrices and array and their use..
IT406.3	Able to use basic flow controls (if-else, for, while) to implement the control structures in MATLAB. Able to program scripts and functions using the MATLAB development environment and generate plots and export this for use in reports and presentations.
IT406.4	Able to create web pages using HTML Tags and Cascading Style Sheets.

Course	CO Attainment	P01	P02	P03	P04	P05	P06	P07	P08	P09	P010	P011	P012	PSO1	PSO2	PSO3
IT404.1		3	3	2		-	-	-	-		-	-			-	-
IT404.2		2	3	2		-	-	-	-		-	-			-	-
IT404.3		3	2	2		-	-	-	-	1	1	-			-	1
IT404.4		2	2	-		-	-	-	-	1	1	-		2	-	1
IT404.5				-	2	1	-	-	-	-	-	-		2	-	-
Average		**2.5**	**2.5**	**2**	**2**	**1**				**1**	**1**	**#DIV/0!**	**#DIV/0!**			

List Of Experiment

1. BASICS OF MATLAB

MATLAB is a language of technical computing. The MATLAB development environment is a set of tools to help user use MATLAB functions and files. Many of these tools are graphical user interface.

STARTING AND QUITTING MATLAB

To start MATLAB on Microsoft Windows's platform, double-click MATLAB shortcut icon on your windows Desktop.

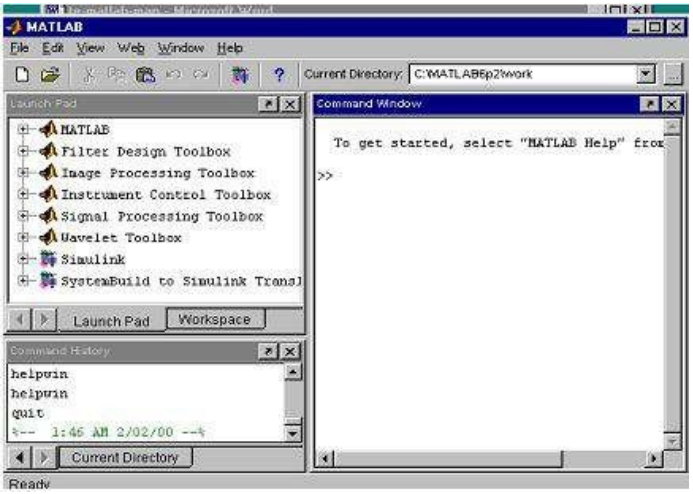
To quit MATLAB at any time, do one of the following:

- Select close box in MATLAB desktop
- Select exit MATLAB from desktop file menu
- Type quit at command window prompt.

DESKTOP TOOLS

Desktop is thought of as your instrument panel for MATLAB. When you start MATLAB, Desktop appears contains tools (GUI) for managing files, variables and applications associated with MATLAB.

MATLAB DESKTOP



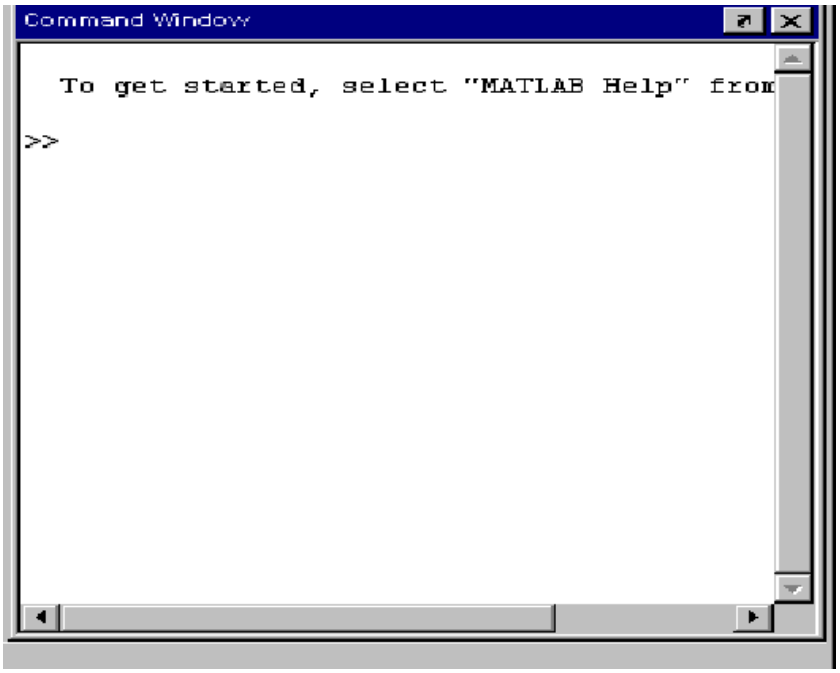
The following tools are managed by the MATLAB

DESKTOP:
- Command Window- Run MATLAB functions
- Command History- View a log of functions you entered in the command window, copy them, and execute them.
- Launch Pad- Run tools and access documentation for all of your MathWorks Products
- Current Directory Browser- View MATLAB files and related files and perform file operations such as open, and find content.
- Help Browser- View and search documentation for the full family of MATLAB products.
- Workspace Browser- View and make changes to the contents of workspace
- Array Editor- View array contents in a table format and edit values
- Editor/Debugger- Create, edit and debug M-files

THE COMMAND WINDOW

The command window is the main way you communicate with MATLAB. It appears in the desktop when you first start MATLAB.Use command window to run MATLAB functions and MATLAB operations.

THE COMMAND WINDOW



MATLAB WORKSPACE

The MATLAB workspace consists of the set of variables built up during a MATLAB session and stored in memory. You add variables to the workspace by using functions, running M-files and loading saved workspaces.

Use workspace browser to perform operations on the MATLAB workspace.

FILE OPERATIONS:

MATLAB file operations use the current directory as a reference point. Any file you want to run must either be in the current directory or on the search path.

The key tools for performing file operations are

Current Directory Field

Current Directory Browser

IMPORTING AND EXPORTING DATA

MATLAB provides many ways to load data from disk files or the clipboard into the workspace, a process called Importing data, and to save workspace variables to a disk file, a process called exporting data. You can import/export both text data and binary data. The easiest way to import the data is to use MATLAB

Import Wizard.

MATLAB provides following toolboxes:

1. Control System Toolbox
2. Filter Design Toolbox
3. Fuzzy Logic Toolbox
4. Image Processing Toolbox
5. Signal Processing Toolbox
6. Neural Networks Toolbox
7. Partial Differential Equation Toolbox
8. Statistics Toolbox
9. Mapping Toolbox
10. LMI Control Toolbox
11. Optimization Toolbox
12. Robust Control Toolbox
13. Wavelet Toolbox

14. Instrument Control Toolbox
15. System Identification Toolbox
16. Communication Toolbox

Demonstrations:

MATLAB provides very powerful Demonstrations for almost everything in the software. To start demo you have to use demo command.

It's syntax is:

Type "demo" at the command line to browse available demos.

With the optional action argument demo

('matlab'|'toolbox'|'simulink'|'blockset'|'stateflow'),

DEMO opens the demo screen to the specified

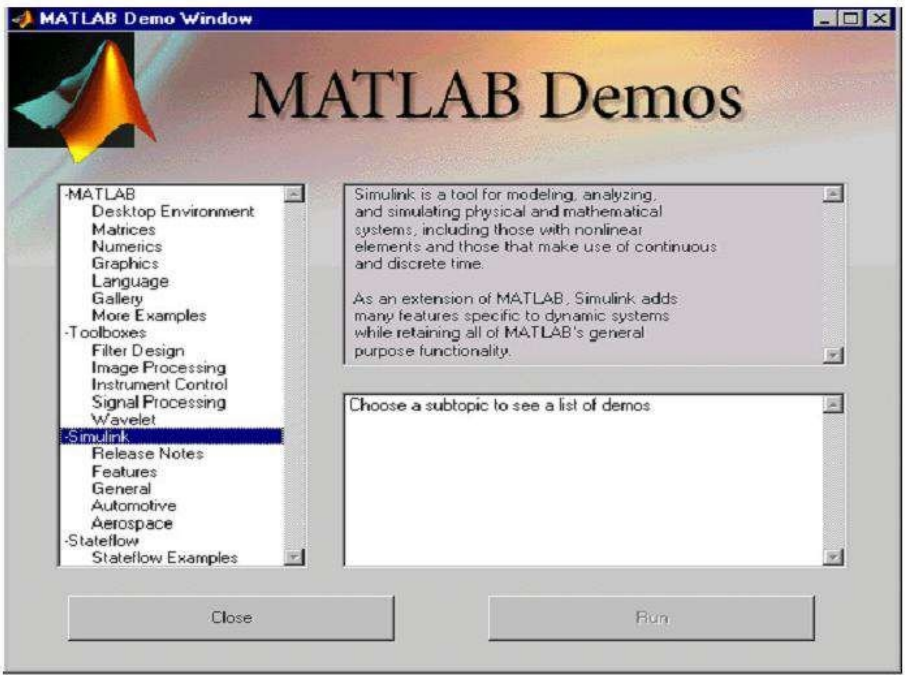
subtopic. With the optional categoryArg argument,

DEMO opens to the specified toolbox or category,

e.g. demo toolbox signal

demo matlab language

The starting Demo Screen looks as follows:



You have to select the topic whose demo is required from the left panel and click Run.

Some commonly used basic MATLAB commands are:

To learn more about each command type help

<command> in the command window.

- Ver - MATLAB, SIMULINK, and toolbox version information.
- help - M-file help, displayed at the command line.
- helpwin - M-file help, displayed in the help browser.
- demo - Run demonstrations.
- who - List current variables.
- Workspace - Display Workspace Browser, a GUI for managing workspace.
- clear - Clear variables and functions from memory.
- quit - Quit MATLAB session.
- type - List M-file.
- edit - Edit M-file.
- echo - Echo commands in M-files.
- format - Set output format.
- error - Display error message and abort function.
- warning - Display warning message.
- fprintf - Display formatted message.
- sprintf - Write formatted data to a string.
- input - Prompt for user input.
- pause - Wait for user response.

Programming language constructs.

Control flow

- if - Conditionally execute statements.
- else - IF statement condition.
- elseif - IF statement condition.
- End -Terminate scope of FOR, WHILE, SWITCH, TRY and IF statements.
- for - Repeat statements a specific number of times.
- while - Repeat statements an indefinite number of times.

b E or FOR loop.

r

e

a

k

-

T

e

r

m

i

n

a

t

e

e

x

e

c

u

t

i

o

n

o

f

W

H

I

L

- switch - Switch among several cases based on expression.
- case - SWITCH statement case.
- otherwise - Default SWITCH statement case.
- try - Begin TRY block.
- catch - Begin CATCH block.
- return - Return to invoking function.

2. BASIC OPERATIONS & EXPRESSIONS IN MATLAB

Different commands are available to carry out various mathematical operations such as Arithmetic, Trignometric., Exponential, Complex etc. on the numerical data.

ELEMENTARY MATH FUNCTIONS.

Trigonometric.
sin - Sine.
sec -
Secant.
cos -
Cosine.
csc -
Cosecant. tan
- Tangent. cot
- Cotangent.

Exponential.
exp - Exponential.
log - Natural logarithm.
log10 - Common (base 10) logarithm.
log2 - Base 2 logarithm and dissect floating point
number. pow2 - Base 2 power and scale floating point
number. sqrt - Square root.

nextpow2 - Next higher power of 2.

Complex.

abs - Absolute value.

angle - Phase angle.

complex - Construct complex data from real and imaginary

parts. conj - Complex conjugate.

imag - Complex imaginary

part. real - Complex real part.

unwrap - Unwrap phase

angle.

MATRIX FUNCTIONS - NUMERICAL LINEAR ALGEBRA.

Matrix analysis.

norm	-	Matrix or vector norm.
normest	-	Estimate the matrix 2-norm.
rank	-	Matrix rank.
det	-	Determinant.
trace	-	Sum of diagonal elements.
null	-	Null space.

Matrix functions.

expm	-	Matrix exponential.
logm	-	Matrix logarithm.
sqrtm	-	Matrix square root.
funm	-	Evaluate general matrix function.

3. BASIC OPERATIONS & EXPRESSIONS IN MATLAB

Exercise

1. Arithmetic operations: Compute the following quantities:

i. $\frac{2^5}{2^5-1}$ and compare with $(1-\frac{1}{2^5})^{-1}$.

ii. $3\frac{\sqrt{5}-1}{(\sqrt{5}+1)2}-1$

iii. Area = πr^2 with r = $\pi^{\frac{1}{3}}-1$

2. Exponential and logarithms:

The mathematical quantities e^x , ln x , and log x are calculated with exp (x) , log

(x) , and log10(x) , respectively. Calculate the following quantities:

i. e3 , ln(e3) , $\log_{10}$ (e3) , and $\log_{10}$ (105)

ii. $e^{\pi\sqrt{163}}$.

iii. Solve $3^x = 17$ for x and check the result.

3. Trigonometry:

The basic MATLAB trigonometric functions are sin , cos , tan , cot , sec , and csc. The inverses, e.g. , arcsin , arctan, etc. , are call: ulated with asin , atan , etc. The same is true for hyperbolic functions. The inverse function atan2 takes two arguments , y and x, and gives the four quadrant inverse tangent . The argument of these functions must be in radians.
Calculate the following quantities:

i. $\sin\frac{\pi}{6}$, cos π , and $\tan\frac{\pi}{2}$

ii. $\sin^2\frac{\pi}{6}+\cos^2\frac{\pi}{6}$

iii. y= $\cosh^2 x - \sinh^2 x$, with x=32π

4. Basic operations on a Vector

Aim: Write a program which illustrates the basic operations on a vector

SOFTWARE REQURIED:-

1.MATLAB R2010a.

2.Windows XP SP2, 7 etc.

THEORY:-

MATLAB, which stands for matrix laboratory, is a state-of-the-art mathematical software package, which is used extensively in both academia and industry. It is an interactive program for numerical computation and data visualization, which along with its programming capabilities provides a very useful tool for almost all areas of science and engineering. Unlike other mathematical packages, such as MAPLE or MATHEMATICA, MATLAB cannot perform symbolic manipulations without the use of additional Toolboxes. It remains however, one of the leading software packages for numerical computation. As you might guess from its name, MATLAB deals mainly with matrices. A scalar is a 1-by-1 matrix and a row vector of length say 5, is a 1-by-5 matrix.. One of the many advantages of MATLAB is the natural notation used. It looks a lot like the notation that you encounter in a linear algebra. This makes the use of the program especially easy and it is what makes MATLAB a natural choice for numerical computations. The purpose of this experiment is to familiarize MATLAB, by introducing the basic features and commands of the program.

Built in Functions:

Scalar Functions:

Certain MATLAB functions are essentially used on scalars, but operate element-wise when applied to a matrix (or vector).

They are summarized below.

1. sin - trigonometric sine
2. cos - trigonometric cosine
3. tan - trigonometric tangent
4. asin - trigonometric inverse sine (arcsine)
5. acos - trigonometric inverse cosine (arccosine)
6. atan - trigonometric inverse tangent (arctangent)
7. exp - exponential
8. log - natural logarithm
9. abs - absolute value
10. sqrt - square root
11. rem - remainder
12. round - round towards nearest integer
13. floor - round towards negative infinity
14. ceil - round towards positive infinity

Vector Functions:

Other MATLAB functions operate essentially on vectors returning a scalar value. Some of these functions are given below.

1. max largest component : get the row in which the maximum element lies
2. min smallest component
3. length length of a vector
4. sort sort in ascending order
5. sum sum of elements
6. prod product of elements
7. median median value
8. mean mean value std standard deviation

Algorithm:

1. Create vector- a of 9 elements
2. Add 2 to each element
3. Take square of each element

4. Create a new vector c from b by multiplying each element of b by 2

5. Plot graph of b with grid lines

6. Plot bar graph of b with x-axis label “Sample” and y-axis label [“kgs”]

7. Use symbols to mark the points

Program:

```
%create vector a of 9
elements a=[1 2 3 4 5 6
4 3 4 5]
%add 2 to each
element b=a+2
%take square of each
element b.^2
%create a new vector c from b by multiplying each element of
b by 2 c=b * 2
%plot graph of b with grid
lines plot(b)
grid on
%plot bar
graph of b
bar(b)
xlabel('Sampl
e')
ylabel('kgs')
```

```
%use symbols to mark the points
plot(b,'*')
axis([0 10 0 10])
```

Exercises

1. Command used to display the value of variable x.
2. What would be the output of the following code (in editor window)?

   A = [0 1]; B=2; C = A + B
3. What would be the output of the following code (in editor window)?

   A = [1 0 2]; b = [3 0 7]; c=a.*b;
4. What would be the output of the following code (in editor window)? a=1:5;

   c=a.^2
5. Create a 4 × 1 column vector that contains any values of your choosing.
6. Use one MATLAB command to evaluate the sine of 30o; 45o; 60o, and 120o.

   Subsequently, evaluate cosine, tangent and cotangent of the same angles.Find the sum of the integers from 1 to 100.
7. Find the sum of the integers from 1 to 100.
8. Create a 1 × 6 vector v containing the integer values from 20 to 25. Subsequently, create an 1 × 6 vector whose values are equal to 5 times the values in v.
9. Create a vector that goes at equal steps from −2 to +2 containing 50 components.
10. Create a vector spanning the range from 0 to 2, containing 100 equally spaced components, so that the first value is 0, and the last value is 2.

5. Matrix operations

AITo write a MATLAB program to perform some basic operation on matrices such as addition, subtraction, multiplication etc.

SOFTWARE REQURIED:-

1. MATLAB R2010a.
2. Windows XP SP2, Windows 7 etc.

THEORY:-

MATLAB, which stands for matrix laboratory, is a state-of-the-art mathematical software package, which is used extensively in both academia and industry. It is an interactive program for numerical computation and data visualization, which along with its programming capabilities provides a very useful tool for almost all areas of science and engineering. Unlike other mathematical packages, such as MAPLE or MATHEMATICA, MATLAB cannot perform symbolic manipulations without the use of additional Toolboxes. It remains however, one of the leading software packages for numerical computation. As you might guess from its name, MATLAB deals mainly with matrices. A scalar is a 1-by-1 matrix and a row vector of length say 5, is a 1-by-5 matrix.. One of the many advantages of MATLAB is the natural notation used. It looks a lot like the notation that you encounter in a linear algebra. This makes the use of the program especially easy and it is what makes MATLAB a natural choice for numerical computations. The purpose of this experiment is to familiarize MATLAB, by introducing the basic features and commands of the program.

Scalar Functions:

Certain MATLAB functions are essentially used on scalars, but operate element-wise when applied to a matrix (or vector).

Built in Functions:

They are summarized below.

1. sin - trigonometric sine
2. cos - trigonometric cosine
3. tan - trigonometric tangent
4. asin - trigonometric inverse sine (arcsine)
5. acos - trigonometric inverse cosine (arccosine)
6. atan - trigonometric inverse tangent (arctangent)
7. exp - exponential
8. log - natural logarithm
9. abs - absolute value
10. sqrt - square root
11. rem - remainder
12. round - round towards nearest integer
13. floor - round towards negative infinity
14. ceil - round towards positive infinity

Matrix Functions:

Much of MATLAB‟s power comes from its matrix functions.

These can be further separated into two sub-categories.
The first one consists of convenient matrix building functions, some of which are given below.

1. eye - identity matrix
2. zeros - matrix of zeros
3. ones - matrix of ones
4. diag - extract diagonal of a matrix or create diagonal matrices
5. triu - upper triangular part of a matrix
6. tril - lower triangular part of a matrix
7. rand - randomly generated matrix commands in the

second sub-category of matrix functions are

1. size size of a matrix

2. det determinant of a square matrix
3. inv inverse of a matrix
4. rank rank of a matrix
5. rref reduced row echelon form
6. eig eigenvalues and eigenvectors
7. poly characteristic polynomial

PROCEDURE:-

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

GIVEN NUMERICAL:

Subtraction: A – B

Multiplication: A * B

Division: A/B or A\B

Inverse of A: inv(A)

RESULTS:

```
>> A + B % Sum up matrix A
and B ans: 4 5 1

2 9 8
-2 1 1
>> A – B % Subtract matrix A and B
ans:
2 -1 1
-2 -3 0
0 1 -3
>> A * B % Multiply matrix A and B
ans:
6 21 10
2 18 20
2 3 2
```

>> A/B

>> % Divides matrix A and B(take inverse of B and multiply with A)

ans:

>> A\B

>> % Divides matrix A and B(take inverse of A and multiply with B)

ans:

>> inv(A) % inverse of A

Ans:

ADDITIONAL COMMANDS AND RESULTS:

>>a= magic(3) % gives 3 X 3 matrix whose sum from any angle is same

ans:

8 1 6

3 5 7

4 9 2

>>a= rand(3) % gives any 3 X 3 random matrix

ans:

>> a= ones(3) % gives 3X 3 matrix whose elements are one

ans:

1 1 1

1 1 1

1 1 1

>> b= 2*ones(3) % multiplication of 2 with ones(3)

ans:

2 2 2

2 2 2

2 2 2

>> a+2 % summation of 2 with matrix A

ans:

>> a(2,2) % gives second row and second column element of matrix A

ans:

1

>>a(2:3, :) %gives second and third row of matrix 'a'

ans:

1 1 1

1 1 1

>> a(:, 2:3)% gives second and third column of matrix 'a'

ans:

1 1

1 1

1 1

>> a(2:3,1:2)

>> %gives second and third row and first and second column of matrix

>> % 'a'

ans:

1 1

1 1

>>a(:,2) % gives second column of matrix 'a'

ans:

1

1

1

>>a(:,1:2)=[] %delete first and second column

ans:

1

1

1

>>eye(3) %gives 3 X 3 matrix whose diagonal are

```
One
Ans: 1  0  0
0  1  0
0  0  1
>> diag(a) % gives diagonal element of matrix 'a'
ans:
1  0  0
0  1  0
0  0  1
>> b = a' % gives inverse of matrix 'a'
ans:
1  1  1
```

Algorithm:

1.Create matrx of order 3,3

2.Take it's transpose

3.Multiply A and B and assign to C

4.Multiply corresponding elements of A with those of B

5.Add A and B and assign to D

6.Subtract B from A and assign result to D

7.Take inverse of A

8.Illustrate the fact that a matrix multiplied by it's inverse is identity matrix

9.Take square of matrix A

10.Find 4^{th} power of matrix B

11.create matrix of order 3,3 with each diagonal element=1

Program:

```
A=[1 2 0; 2 5 -1; 4 10 -1] %create matrix of order 3,3
B=A'      %take it's transpose
C=A * B  %multiply A and B and assign to C
C=A .* B %multiply corresponding elements of A with those of B
D=A + B  %add A and B and assign to D
D=A - B  %subtract B from A and assign result to D
X=inv(A) % Take inverse of A
%Illustrate the fact that a matrx multiplied by it's inverse is
%identity matrix
I=inv(A) * A
    A.^2    %Take square of matrix A
    B.^4%Find 4th power of matrix B
%create matrix of order 3,3 with each diagonal element=1

diag([ones(1,3)]
```

CONCLUSION: We have studied various commands to solve the basic matrix operations.

6. Multi-dimensional Array and Special Matrix operations

AIM: -

To write a MATLAB program to perform operation on array

SOFTWARE REQURIED:-

1. MATLAB R2010a.
2. Windows XP SP2, Windows 7 etc.

THEORY:-

MATLAB, which stands for MATrixLABoratory, is a state-of-the-art mathematical software package, which is used extensively in both academia and industry. It is an interactive program for numerical computation and data visualization, which along with its programming capabilities provides a very useful tool for almost all areas of science and engineering. Unlike other mathematical packages, such as MAPLE or MATHEMATICA, MATLAB cannot perform symbolic manipulations without the use of additional Toolboxes. It remains however, one of the leading software packages for numerical computation. As you might guess from its name, MATLAB deals mainly with matrices. A scalar is a 1-by-1 matrix and a row vector of length say 5, is a 1-by-5 matrix.. One of the many advantages of MATLAB is the natural notation used. It looks a lot like the notation that you encounter in a linear algebra. This makes the use of the program especially easy and it is what makes MATLAB a natural choice for numerical computations. The purpose of this experiment is to familiarize MATLAB, by introducing the basic features and commands of the program.

Built in Functions:

Special Matrix Functions:

Much of MATLAB"s power comes from its matrix functions.

These can be further separated into two sub-categories.
The first one consists of convenient matrix building functions, some of which are given below.

1. eye - identity matrix
2. zeros - matrix of zeros
3. ones - matrix of ones
4. diag - extract diagonal of a matrix or create diagonal matrices
5. triu - upper triangular part of a matrix
6. tril - lower triangular part of a matrix
7. rand - randomly generated matrix

PROCEDURE:-

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

COMMANDS:

```
>>a= magic(3) % gives 3 X 3 matrix whose sum from any angle is same
ans:
8 1 6
3 5 74 9 2
>>a= rand(3) % gives any 3 X 3 random matrix
ans:
>> a= ones(3) % gives 3X 3 matrix whose elements are one
ans:
1 1 1
1 1 1
1 1 1
>> b= 2*ones(3) % multiplication of 2 with ones(3)
```

ans:

2 2 2

2 2 2

2 2 2

>> a+2 % summation of 2 with matrix A

ans:

>> a(2,2) % gives second row and second column element of matrix A

ans:

1

>>a(2:3, :) %gives second and third row of matrix 'a'

ans:

1 1 1

1 1 1

>> a(:, 2:3) % gives second and third column of matrix 'a'

ans:

1 1

1 1

1 1

>> a(2:3,1:2)

>> %gives second and third row and first and second column of matrix

>> % 'a'

ans:

1 1

1 1

>>a(:,2) % gives second column of matrix 'a'

ans:

1

1

1

>>a(:,1:2)=[] %delete first and second column

1

Ans:

>>eye(3) %gives 3 X 3 matrix whose diagonal are

o

n

e

a

n

s

:

1 0 0

0 1 0

0 0 1

>> diag(a) % gives diagonal element of matrix 'a'

ans:

1 0 0

0 1 0

0 0 1

>> b = a' % gives inverse of matrix 'a'

ans:

1 1 1

7. Programming in MATLAB

AIM: -

To write a MATLAB program to using script and function during programming

SOFTWARE REQURIED:-

1. MATLAB R2010a.
2. Windows XP SP2, Windows 7 etc.

THEORY:-

MATLAB, which stands for matrix laboratory, is a state-of-the-art mathematical software package, which is used extensively in both academia and industry. It is an interactive program for numerical computation and data visualization, which along with its programming capabilities provides a very useful tool for almost all areas of science and engineering. Unlike other mathematical packages, such as MAPLE or MATHEMATICA, MATLAB cannot perform symbolic manipulations without the use of additional Toolboxes. It remains however, one of the leading software packages for numerical computation. As you might guess from its name, MATLAB deals mainly with matrices. A scalar is a 1-by-1 matrix and a row vector of length say 5, is a 1-by-5 matrix.. One of the many advantages of MATLAB is the natural notation used. It looks a lot like the notation that you encounter in a linear algebra. This makes the use of the program especially easy and it is what makes MATLAB a natural choice for numerical computations. The purpose of this experiment is to familiarize MATLAB, by introducing the basic features and commands of the program.

Theory:

So far in these lab sessions, all the commands were executed in the Command Window. The problem is that the commands entered in the Command Window cannot be saved and executed again for several times. Therefore, a different way of executing repeatedly commands with MATLAB is:

1. to create a file with a list of commands,
2. save the file, and
3. run the file.

If needed, corrections or changes can be made to the commands in the file. The files that are used for this purpose are called script files or scripts for short.

This section covers the following topics:

- M-File Scripts
- M-File Functions

M-File Scripts

A script file is an external file that contains a sequence of MATLAB statements. Script files have a filename extension .m and are often called M-files. M-files can be scripts that simply execute a series of MATLAB statements, or they can be functions that can accept arguments and can produce one or more outputs.

M-File functions

As mentioned earlier, functions are programs (or *routines*) that accept *input* arguments and return *output* arguments. Each M-¯le function (or *function* or *M-file* for short) has its *own* area of workspace, separated from the MATLAB base workspace.

Anatomy of a M-File function

This simple function shows the basic parts of an M-file.

```
function f = factorial(n) (1)
% FACTORIAL(N) returns the factorial of N. (2)
% Compute a factorial value.
(3) f = prod(1:n);
```

PROCEDURE:-

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

PROGRAM FOR PRACTICE:

Program 1:

Consider the system of equations:

$x + 2y + 3z = 1$ $3x + 3y + 4z = 1$ $2x + 3y + 3z = 2$

Find the solution x to the system of equations.

Use the MATLAB *editor* to create a file: File-> New ->M-file. Enter the following statements in the file:

```
A = [1 2 3; 3 3 4; 2 3 3];
b = [1; 1; 2];
x = A\b
```

Save the file, for example, example1.m.

Run the file, in the command line, by typing:

```
>>
exam
ple1
x =
-0.5000
1.5000
-0.5000
```

Program 2:

Plot the following cosine functions, $y_1 = 2\cos(x)$, $y_2 = \cos(x)$, and $y_3 = 0.5 * \cos(x)$, in the interval $0 \leq x \leq 2¼$.

Create a file, say example2.m, which contains the following commands:

```
x =
0:pi/100:2*
pi; y1 =
2*cos(x);
y2 = cos(x);
y3 = 0.5*cos(x);
plot(x,y1,'--',x,y2,'-',x,y3,':')
xlabel('0 \leq x \leq 2\pi')
ylabel('Cosine functions')
legend('2*cos(x)','cos(x)','0.5*cos
(x)') title('Typical example of
multiple plots') axis([0 2*pi -3
3])
```

8. if...end, if...else...end, if...elseif...else...end & nested if

AIM: -

To write a MATLAB program to perform decision making during programming

SOFTWARE REQURIED:-

1. MATLAB R2010a.
2. Windows XP SP2, Windows 7 etc.

THEORY:-

MATLAB, which stands for matrix laboratory, is a state-of-the-art mathematical software package, which is used extensively in both academia and industry. It is an interactive program for numerical computation and data visualization, which along with its programming capabilities provides a very useful tool for almost all areas of science and engineering. Unlike other mathematical packages, such as MAPLE or MATHEMATICA, MATLAB cannot perform symbolic manipulations without the use of additional Toolboxes. It remains however, one of the leading software packages for numerical computation. As you might guess from its name, MATLAB deals mainly with matrices. A scalar is a 1-by-1 matrix and a row vector of length say 5, is a 1-by-5 matrix.. One of the many advantages of MATLAB is the natural notation used. It looks a lot like the notation that you encounter in a linear algebra. This makes the use of the program especially easy and it is what makes MATLAB a natural choice for numerical computations. The purpose of this experiment is to familiarize MATLAB, by introducing the basic features and commands of the program.

Theory:

if…end

statements:

Syntex

```
if < expression>
<statements>
...
...
end
```

if…else…end statements:

Syntex

```
if <expression>
% statement(s) will execute if the boolean expression is true
<statement(s)>
```

if...elseif...elseif...else...end statements:

```
if <expression 1>
% Executes when the expression 1 is true
<statement(s)>
elseif <expression
2>
% Executes when the boolean expression 2 is true
<statement(s)>
Elseif <expression
3>
% Executes when the boolean expression 3 is true
<statement(s)>
else
% executes when the none of the above condition is true
<statement(s)>
end
```

```
e
l
s
e
```

End

```
                    statement(s)>

                    % statement(s) will execute if
<                   the boolean expression is fals
```

Nested if statements: Syntex

```
if <expression 1>
% Executes when the boolean expression 1 is true if
<expression 2>
% Executes when the boolean expression 2 is true
end
end
```

PROCEDURE:-

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

PROGRAM FOR PRACTICE:

Program 1: if...else...end Statement

```
a = 100;
% check the boolean
condition if a < 20
% if condition is true then print the following
fprintf('a is less than 20\n' );
el
se                % if condition is false then print the
                  following fprintf('a is not less than 20\n'
                  );
en
d
fprintf('value of a is : %d\n', a);
```

Results

```
a is not less than
20 value of a is :
100
```

Program 2: if...elseif...elseif...else...end Statements

```
a = 100;
%check the boolean
condition if a == 10
% if condition is true then print the following
fprintf('Value of a is 10\n' );
elseif( a == 20 )
% if else if condition is true
fprintf('Value of a is 20\n' );
elseif a == 30
% if else if condition is true
fprintf('Value of a is 30\n' );
else
% if none of the conditions is true '
fprintf('None of the values are matching\n');
fprintf('Exact value of a is: %d\n', a );
end
```

Results

```
None of the values are matching
Exact value of a is: 100
```

Program 3: if...elseif...elseif...else...end Statements

```
a = 100;
b = 200;
```

```
% check the boolean
condition if( a == 100 )
% if condition is true then check the following
```

```
if( b == 200 )
% if condition is true then print the
following fprintf('Value of a is 100 and b is
200\n' ); end
end
fprintf('Exact value of a is : %d\n', a );
fprintf('Exact value of b is : %d\n', b );
```

Results

```
Value of a is 100 and b is
200 Exact value of a is :
100 Exact value of b is :
200
```

if...end, if...else...end, if...elseif...else...end & nested if

EXPERIMENTAL QUESTION

Q.1 What would be the result of the following expressions?

```
'b' >= 'c' - 1
3 == 2 + 1
(3 == 2) + 1
xor(5 < 6, 8 > 4)
```

Q.2 Write a script that tests whether the user can follow instructions. It prompts the user to enter an x. If the user enters anything other than an x, it prints an error message; otherwise, the script does nothing.

Q.3 Write a function nexthour that will receive one integer argument, which is an hour of the day, and will return the next hour. This assumes a 12-hour clock, so for example the next hour after 12 would be 1. Here are two examples of calling this function..

```
>> fprintf('The next hour will be %d.\n',nexthour(3))
The next hour will be 4.

>> fprintf('The next hour will be %d.\n',nexthour(12))
The next hour will be 1.
```

Q.4 Write a function program "*ArithmeticOperation.m*" for performing arithmetic operation on the two operand by passing as arguments and desired arithmetic operation is passing as numeric value in the arguments like 1 for addition, 2 for subtraction, 3 for multiply and 4 for division implementing selection using if...elseif...else...end .

EXERCISE NO. 8

Switch-Case, Nested Switch Statements, Menu, if. Functions

AIM: -

To write a MATLAB program to perform decision making during programming

SOFTWARE REQURIED:-

1. MATLAB R2010a.
2. Windows XP SP2, Windows 7 etc.

THEORY:-

MATLAB, which stands for MATrixLABoratory, is a state-of-the-art mathematical software package, which is used extensively in both academia and industry. It is an interactive program for numerical computation and data visualization, which along with its programming capabilities provides a very useful tool for almost all areas of science and engineering. Unlike other mathematical packages, such as MAPLE or MATHEMATICA, MATLAB cannot perform symbolic manipulations without the use of additional Toolboxes. It remains however, one of the leading software packages for numerical computation. As you might guess from its name, MATLAB deals mainly with matrices. A scalar is a 1-by-1 matrix and a row vector of length say 5, is a 1-by-5 matrix.. One of the many advantages of MATLAB is the natural notation used. It looks a lot like the notation that you encounter in a linear algebra. This makes the use of the program especially easy and it is what makes MATLAB a natural choice for numerical computations. The purpose of this experiment is to familiarize MATLAB, by introducing the basic features and commands of the program.

Theory:

Switch

statements:

Syntex

```
switch <switch_expression> case
<case_expression>
     <statements> case
<case_expression>
<statements>
...
...
otherwise
<statements> end
```

Nested Switch statements:

Syntex

```
switch(ch1)
case 'A'
fprintf('This A is part of outer switch');
switch(ch2)
case 'A'
     fprintf('This A is part of inner switch' ); case
'B'
fprintf('This B is part of inner switch' );
end
case 'B'
fprintf('This B is part of outer switch' ); end
```

Menu function:

```
mypick = menu('Pick a pizza','Cheese','Shroom','Sausage');
```

PROCEDURE:-

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

PROGRAM FOR PRACTICE:

Program 1: switch-case

```
sprintf('Classical- 1; Laplace- 2; Fourier- 3') % assign nos to each mehod

i=input('Enter the choice: ')

switch i %select the case as per no entered case
1

    case 2                  disp('Classical

    case 3                  ')

                            disp('Laplace')

                            disp('Fourier')

                  otherwise
e                 disp('Not found')
n
d
```

Program 2: switch-case

```
grade = 'B';
switch(grade)
case 'A'
      fprintf('Excellent!\n'
); case 'B'
      fprintf('Well done\n'
); case 'C'
      fprintf('Good\n'
); case 'D'
      fprintf('You passed\n'
); case 'F'
      fprintf('Better try again\n' );
otherwise
fprintf('Invalid grade\n' );
end
```

Results

```
Well done
Your grade is B
```

Program 3: Nested switch-case

```
a = 100;
b = 200;
switch(a)
case 100
fprintf('This is part of outer switch %d\n', a ); switch(b)
case 200
fprintf('This is part of inner switch %d\n', a );
end
```

```
end
fprintf('Exact value of a is : %d\n', a );
fprintf('Exact value of b is : %d\n', b );
```

Results:

```
This is part of outer switch
100 This is part of inner
switch 100 Exact value of a
is : 100
Exact value of b is : 200
```

Program 4: menu function

```
%This script asks the user for a type of pizza
% and prints which type to order using a switch
mypick = menu('Pick a pizza','Cheese','Shroom','Sausage');

switch mypick
case 1
                        disp('Order a cheese pizza')
    case 2
                        disp('Order a
    case 3
                        mushroom

                        pizza')

                        disp('Order a

                        sausage

                        pizza')
                otherwise
                disp('No pizza for us today')
e
n
d
```

Program 5: is.. function

```
>> clear
>> evec = [];
>> isempty(evec)
ans =
1
>> evec = [evec 11];
>> isempty(evec)
ans =
0


>> isletter('h')
ans
= 1
>> isletter('4')
ans
= 0
```

EXPERIMENTAL QUESTION

Q.1 Rewrite the following switch statement as one nested if-else statement (elseif clauses may be used). Assume that there is a variable letter and that it has been initialized.

```
switch letter
case 'x'
      disp('Hello'
) case {'y', 'Y'}
      disp('Yes'
) case 'Q'
disp('Quit')
otherwise
disp('Error')
end
```

Q.2 Rewrite the following nested if-else statement as a switch statement that accomplishes exactly the same thing. Assume that num is an integer variable that has been initialized, and that there are functions f1, f2, f3, and f4. Do not use any if or if-else statements in the actions in the switch statement, only calls to the four functions.

```
if num < -2 || num > 4
      f1(num)
       e
       l                  if num < = 2
       s                  if num > = 0
       e                  f2(num)
                  else end
                                      f3(num)

          else

                                f4(num)
      e      end
      n
      d
```

Q.3 Write a script *area_menu* that will print a list consisting of cylinder, circle, and rectangle. It prompts the user to choose one, and then prompts the user for the appropriate quantities (e.g., the radius of the circle) and then prints its area. If the user enters an invalid choice, the script simply prints an error message. The script should use a nested **if-else** statement to accomplish this. Here are two examples of running it (units are assumed to be inches).

```
>> area_menu
```

```
   Menu
1. Cylinder
2. Circle
3. Rectangle
   Please choose one: 2
   Enter the radius of the circle:
   4.1 The area is 52.81

   >> area_menu
   Menu
1. Cylinder
2. Circle
3. Rectangle
   Please choose one:
   3 Enter the
   length: 4 Enter
   the width: 6 The
   area is 24.00
```

EXERCISE NO. 9

Loop – Structures: while..end, for…end, Nested for & while

AIM: -

To write a MATLAB program to perform loop control during programming

SOFTWARE REQURIED:-

1. MATLAB R2010a.
2. Windows XP SP2, Windows 7 etc.

THEORY:-

MATLAB, which stands for MATrixLABoratory, is a state-of-the-art mathematical software package, which is used extensively in both academia and industry. It is an interactive program for numerical computation and data visualization, which along with its programming capabilities provides a very useful tool for almost all areas of science and engineering. Unlike other mathematical packages, such as MAPLE or MATHEMATICA, MATLAB cannot perform symbolic manipulations without the use of additional Toolboxes. It remains however, one of the leading software packages for numerical computation. As you might guess from its name, MATLAB deals mainly with matrices. A scalar is a 1-by-1 matrix and a row vector of length say 5, is a 1-by-5 matrix.. One of the many advantages of MATLAB is the natural notation used. It looks a lot like the notation that you encounter in a linear algebra. This makes the use of the program especially easy and it is what makes MATLAB a natural choice for numerical computations. The purpose of this experiment is to familiarize MATLAB, by introducing the basic features and commands of the program.

Theory:

While….

end:

Syntex

```
while <expression>
<statements>
end
```

Nested While statements:

Syntex

```
while <expression1> while
      <expression2>
      <statements>
      end
      end
```

For…

end:

Syntex

```
for index = values
<program statements>
...
end
```

Nested For Statements:

Syntex

```
          for m = 1:j
            for n = 1:k
                      <statements>;
end
end
```

PROCEDURE:-

1. Open MATLAB
2. Open new M-file
3. Type the program
4. Save in current directory

5. Compile and Run the program
6. For the output see command window\ Figure window

PROGRAM FOR PRACTICE:

Program 1: while

```
a = 10;
% while loop execution
while( a < 20 )
fprintf('value of a: %d\n', a); a =
a + 1;
end
```

Result:

```
value of a:
10 value of
a: 11 value
of   a:   12
value of a:
13 value of
a: 14 value
of   a:   15
value of a:
16 value of
a: 17 value
of   a:   18
value of a:
19
```

Program 2: Error-Checking User Input in a While Loop

```
% Loop until the user enters a positive number
inputnum=input('Enter a positive number: ');
while inputnum < 0
inputnum = input('Invalid! Enter a positive number: ');
end
fprintf('Thanks, you entered a %.1f \n',inputnum)
```

Results

```
>> readonenum
Enter a positive number: -5
Invalid! Enter a positive number: -2.2
Invalid! Enter a positive number: c
??? Error using ==> input
Undefined function or variable 'c'.
Invalid! Enter a positive number: 44
Thanks, you entered a 44.0
```

Program 3: Multiple Conditions in a While Loop

```
function logresult = myanywhile(vec)
% Simulates the built-in function any
% Uses a while loop so that the action halts
% as soon as any true value is found
logresult = logical(0);
i = 1;
while i <= length(vec) && logresult ==
```

```
0 if vec(i) = 0
logresult = logical(1);
End
i = i + 1;
```

```
end
```

Program 4: for Loop

```
for a = 10:20
fprintf('value of a: %d\n', a);
end
```

Results:

```
value of a:
10 value of
a: 11 value
of   a:   12
value of a:
13 value of
a: 14 value
of   a:   15
value of a:
16 value of
a: 17 value
of   a:   18
value of a:
19 value of
a: 20
```

Program 5: for Loop

```
for a = 1.0: -0.1: 0.0
disp(a)
end
```

Results:

```
1
0.9000
0.8000
0.7000
0.6000
0.5000
0.4000
0.3000
0.2000
0.1000
```

Program 6: for Loop

```
for a = [24,18,17,23,28]
disp(a)
end
```

Results:

24
18
17
23
28

Program 7: For- Loop

```
function runsum = sum_1_to_n(n)
% This function returns the sum of
% integers from 1 to n
runsum = 0;
for i = 1:n
runsum = runsum + i;
end
```

Results:

```
15
```

Program 8: Nested

```
for i=2:100
for j=2:100
if(~mod(i,j))
break; % if factor found, not prime
end
end
if(j > (i/j))
fprintf('%d is prime\n', i);
end
end
```

Results:

```
2  is     prime
3  is     prime
5  is     prime
7  is     prime
.
.
61
67
71
73
79
83
89
97
```

EXPERIMENTAL QUESTION

Q.1 Write a script ***aveposnum*** that will repeat the process of prompting the user for positive numbers, until the user enters a negative number. Instead of echo-printing them, however, the script will print the average (of just the positive numbers). If no positive numbers are entered, the script will print an error message instead of the average.

>> *aveposnum*
Enter a positive number:
−5 No positive numbers to
average.
>> *aveposnum*
Enter a positive number: 8
Enter a positive number: 3
Enter a positive number: 4
Enter a positive number:
−6 The average was 5.00

Q.2 Write a function called geomser that will receive values of r and n, and will calculate and return the sum of the geometric series:

$1 + r + r^2 + r^3 + r^4 + ... + r^n$

The following examples of calls to this function illustrate what the result should be:

>>
geomser(1,5)
ans =
6
>>
disp(geomser(2,4))
31

EXERCISE NO. 10

Loop Control Statement: break- Continue etc

AIM: -

To write a MATLAB program to perform loop control during programming

SOFTWARE REQURIED:-

1. MATLAB R2010a.
2. Windows XP SP2, Windows 7 etc.

THEORY:-

MATLAB, which stands for MATrixLABoratory, is a state-of-the-art mathematical software package, which is used extensively in both academia and industry. It is an interactive program for numerical computation and data visualization, which along with its programming capabilities provides a very useful tool for almost all areas of science and engineering. Unlike other mathematical packages, such as MAPLE or MATHEMATICA, MATLAB cannot perform symbolic manipulations without the use of additional Toolboxes. It remains however, one of the leading software packages for numerical computation. As you might guess from its name, MATLAB deals mainly with matrices. A scalar is a 1-by-1 matrix and a row vector of length say 5, is a 1-by-5 matrix.. One of the many advantages of MATLAB is the natural notation used. It looks a lot like the notation that you encounter in a linear algebra. This makes the use of the program especially easy and it is what makes MATLAB a natural choice for numerical computations. The purpose of this experiment is to familiarize MATLAB, by introducing the basic features and commands of the program.

Theory:

While….

end:

Syntex

```
while <expression>
<statements>
end
```

Nested While statements:

Syntex

```
while <expression1> while
      <expression2>
      <statements>
      end
      end
```

For…

end:

Syntex

```
for index = values
<program statements>
...
end
```

Nested For Statements:

Syntex

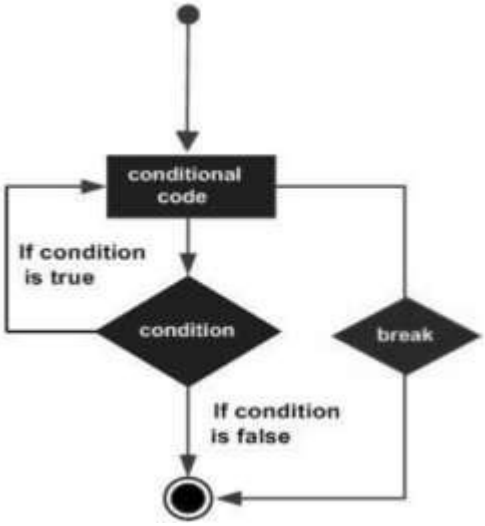
```
          for m = 1:j
            for n = 1:k
                      <statements>;
      end
      end
```

Control Statement	Description

	break statement	***Terminates the loop statement and transfers execution to the statement immediately following the loop.***
	continue statement	***Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.***

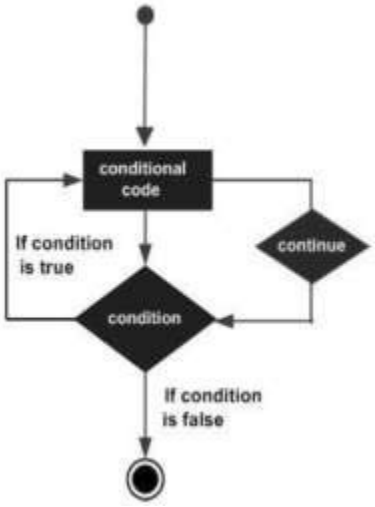
The break Statement

- The break statement terminates execution of for or while loop. Statements in the loop that appear after the break statement are not executed.
- In nested loops, break exits only from the loop in which it occurs. Control passes to the statement following the end of that loop.



The continue Statement:

- The continue statement is used for passing control to next iteration of for or while loop.
- The continue statement in MATLAB works somewhat like the break statement.
- Instead of forcing termination, however, 'continue' forces the next iteration of the loop to take place, skipping any code in between.



PROCEDURE:-

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

PROGRAM FOR PRACTICE:

Program 1: break

```
a = 10;
% while loop execution
while (a < 20 )
fprintf('value of a: %d\n', a); a =
a+1;
if( a > 15)
% terminate the loop using break statement
break;
end
end
```

Result:

```
value of a:
10 value of
a: 11 value
of   a:   12
value of a:
13 value of
a: 14 value
of a: 15
```

Program 2: continue

```
a = 10;
%while loop execution
while a < 20
if a == 15
% skip the iteration a =
a + 1;
continue; end
fprintf('value of a: %d\n', a); a =
a + 1;
end
```

Results

```
value of a:
10 value of
```

```
a: 11 value
of a: 12
```

```
value of a:
13 value of
a: 14 value
of   a:   16
value of a:
17 value of
a: 18 value
of a: 19
```

Program 3: Vectorizing

```
for i = 1:length(vec)
% do something with vec(i)
end

[r c] =
size(mat); for
row = 1:r
for col = 1:c
% do something with mat(row,col)
e
n
d
e
n
d

>> for i = 1:length(v)
v(i) = v(i) * 3;
end
>> v
v =
9   21    6     3
```

Program 4:

```
>> v = v*3
v =
9  21     6   3

>> v= [3 7 2 1];
>>
v/2
ans
=
1.5000 3.5000 1.0000 0.5000
```

EXPERIMENTAL QUESTION

Q.1 Write a function that will receive a matrix as an input argument, and will calculate and return the overall average of all numbers in the matrix. Use loops, not built-in functions, to calculate the average.

Q.2 Write a script that will print the following multiplication table:

1

2 4

3 6 9

4 8 12 16

5 10 15 20 25

Q.3 Create a 3 × 5 matrix. Perform each of the following two ways: using built-in functions, and also using loops (with if statements if necessary):

■■ Find the maximum value in each column.

■■ Find the maximum value in each row.

■■ Find the maximum value in the entire matrix.