LAB MANUAL

NEURAL NETWORK (MAT LAB)

INDEX

S No.	AIM OF EXPERIMENT
1.	To study about MATLAB.
2	Write a program to perform the basics matrix operations.
3	WAP to plot the Straight line.
4.	WAP to plot the Sine curve.
5.	How the weight & bias value effects the output of neurons.
6.	How the choice of activation function effect the output of neuron experiment with the following function purelin(n), bimary threshold(hardlim(n) haradlims(n)) ,Tansig(n) logsig(n)
7.	How the weight and biased value are able to represent a decision boundary in the feature space.
8.	How the Perceptron Learning rule works for Linearly Separable Problem.
9.	How the Perceptron Learning rule works for Non-Linearly Separable Problem.
10.	Write a program to draw a graph with multiple curve.

Program-1

To study about MATLAB.

History

Developed primarily by Cleve Moler in the 1970's. Derived from FORTRAN subroutines LINPACK and EISPACK, linear and eigenvalue systems. Developed primarily as an interactive system to access LINPACK and EISPACK. Gained its popularity through word of mouth, because it was not o±cially distributed. Rewritten in C in the 1980's with more functionality, which include plotting routines.

The MathWorks Inc. was created (1984) to market and continue development of MATLAB. According to Cleve Moler, three other men played important roles in the origins of MATLAB: J. H. Wilkinson, George Forsythe, and John Todd. It is also interesting to mention the authors of LINPACK: Jack Dongara, Pete Steward, Jim Bunch, and Cleve Moler. Since then another package emerged: LAPACK. LAPACK stands for Linear Algebra Package. It has been designed to supersede LINPACK and EISPACK.

Introduction

MATLAB is a high-level technical computing language and interactive environment for algorithm development, data visualization, data analysis, and numeric computation. Using the MATLAB product, you can solve technical computing problems faster than with traditional programming languages, such as C, C++, and Fortran.

The name MATLAB stands for MATrix LABoratory. MATLAB was written originally to provide easy access to matrix software developed by the LINPACK (linear system package) and EISPACK (Eigen system package) projects.

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming environment. Furthermore, MATLAB is a modern programming language environment: it has sophisticated data structures, contains built-in editing and debugging tools, and supports object-oriented programming. These factors make MATLAB an excellent tool for teaching and research.

MATLAB has many advantages compared to conventional computer languages (e.g., C, FORTRAN) for solving technical problems. MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. The software package has been commercially available since 1984 and is now considered as a standard tool at most universities and industries worldwide.

It has powerful built-in routines that enable a very wide variety of computations. It also has easy to use graphics commands that make the visualization of results immediately available. Specific applications are collected in packages referred to as toolbox. There are toolboxes for signal processing, symbolic computation, control theory, simulation, optimization, and several other fields of applied science and engineering.

Areas where MATLAB can be used

You can use MATLAB in a wide range of applications, including :

- signal and image processing
- communications
- control design
- test and measurement
- financial modeling and analysis
- and computational biology.

Add-on toolboxes (collections of special-purpose MATLAB functions, available separately) extend the MATLAB environment to solve particular classes of problems in these application areas.

MATLAB provides a number of features for documenting and sharing your work. You can integrate your MATLAB code with other languages and applications, and distribute your MATLAB algorithms and applications.

The MATLAB System

The MATLAB system consists of these main parts:

Desktop Tools and Development Environment

This part of MATLAB is the set of tools and facilities that help you use and become more productive with MATLAB functions and files. Many of these tools are graphical user interfaces. It includes: the MATLAB desktop and Command Window, an editor and debugger, a code analyzer, and browsers for viewing help, the workspace, and folders.

Mathematical Function Library

This library is a vast collection of computational algorithms ranging from elementary functions, like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix eigenvalues, Bessel functions, and fast Fourier transforms.

The Language

The MATLAB language is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features. It allows both "programming in the small" to rapidly create quick programs you do not intend to reuse. You can also do "programming in the large" to create complex application programs intended for reuse.

Graphics

MATLAB has extensive facilities for displaying vectors and matrices as graphs, as well as annotating and printing these graphs. It includes high-level functions for two-dimensional and three-dimensional data visualization, image processing, animation, and presentation graphics. It also includes low-level functions that allow you to fully

customize the appearance of graphics as well as to build complete graphical user interfaces on your MATLAB applications.

External Interfaces

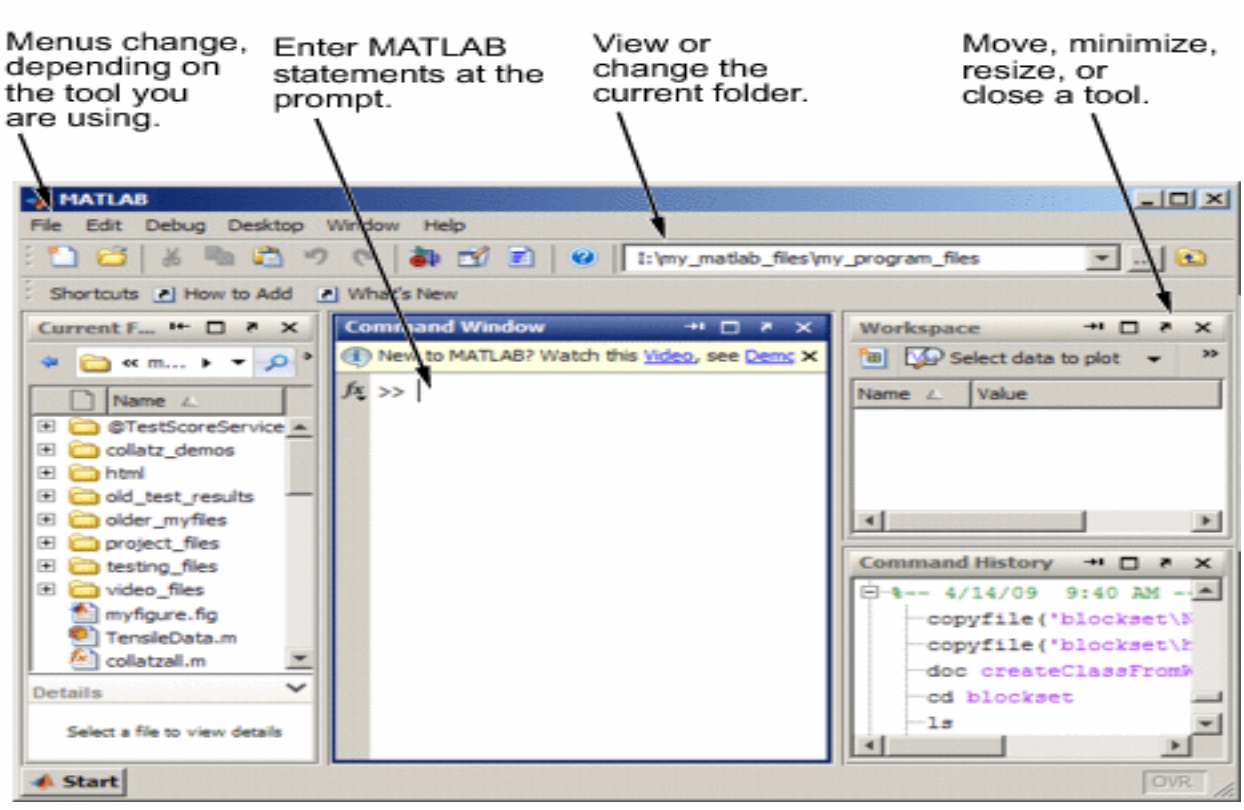
The external interfaces library allows you to write C/C++ and Fortran programs that interact with MATLAB. It includes facilities for calling routines from MATLAB (dynamic linking), for calling MATLAB as a computational engine, and for reading and writing MAT-files.

Starting MATLAB

After logging into your account, you can enter MATLAB by double-clicking on the MATLAB shortcut icon (MATLAB 7.0.4) on your Windows desktop. When you start MATLAB, a special window called the MATLAB desktop appears. The desktop is a window that contains other windows.

When you start MATLAB, the desktop appears, containing tools (graphical user interfaces) for managing files, variables, and applications associated with MATLAB.

The following illustration shows the default desktop. You can customize the arrangement of tools and documents to suit your needs.



The major tools within or accessible from the desktop are:

- The Command Window
- The Command History
- The Workspace
- The Current Directory
- The Help Browser
- The Start button

(>>) in the Command Window.

Usually, there are 2 types of prompt:

- >> for full version
- EDU> for educational version

Mathematical functions

MATLAB offers many predefined mathematical functions for technical computing which contains a large set of mathematical functions.
Typing help elfun and help specfun calls up full lists of elementary and special functions respectively.
There is a long list of mathematical functions that are built into MATLAB. These functions are called built-ins. Many standard mathematical functions, such as sin(x), cos(x), tan(x), ex, ln(x), are evaluated by the functions sin, cos, tan, exp, and log respectively in MATLAB.
Here is the lists of some commonly used functions, where variables x and y can be numbers, vectors, or matrices.

Key Features

The key features of MATLAB are:

- High-level language for technical computing
- Development environment for managing code, files, and data
- Interactive tools for iterative exploration, design, and problem solving
- Mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, and numerical integration
- 2-D and 3-D graphics functions for visualizing data
- Tools for building custom graphical user interfaces
- Functions for integrating MATLAB based algorithms with external applications and languages, such as C, C++, Fortran, Java, COM, and Microsoft Excel

Elementary functions

1. cos(x) Cosine
2. abs(x) Absolute value
3. sin(x) Sine
4. sign(x) Signum function
5. tan(x) Tangent
6. max(x) Maximum value
7. min(x) Minimum value
8. exp(x) Exponential
9. round(x) Round to nearest integer

10. sqrt(x) Square

Getting help

To view the online documentation, select MATLAB Help from Help menu or MATLAB Help directly in the Command Window. The preferred method is to use the Help Browser. The Help Browser can be started by selecting the ? icon from the desktop toolbar. On the other hand, information about any command is available by typing
>> help Command

Quitting the MATLAB Program

To end your MATLAB session, select **File** > **Exit MATLAB** in the desktop, or type quit in the Command Window. You can run a script file named finish.m each time MATLAB quits that, for example, executes functions to save the workspace.

Confirm Quitting

MATLAB can display a confirmation dialog box before quitting. To set this option, select **File** > **Preferences** > **General** > **Confirmation Dialogs**, and select the check box for **Confirm before exiting MATLAB**.

Advantages

- 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)
- MATLAB is optimized to be relatively fast when performing matrix operations
- MATLAB does have some object-oriented elements

Disadvantages

- 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

Program-2

a) Write a program to assign the following through a variable & display.

```
>> a=(3+4)/(5+6)
a =
   0.6364

>> b=2*3.14*3.14
b =
  19.7192

>> c=sqrt(2)
c =
   1.4142
```

b) Write a program to perform the basics matrix operations.

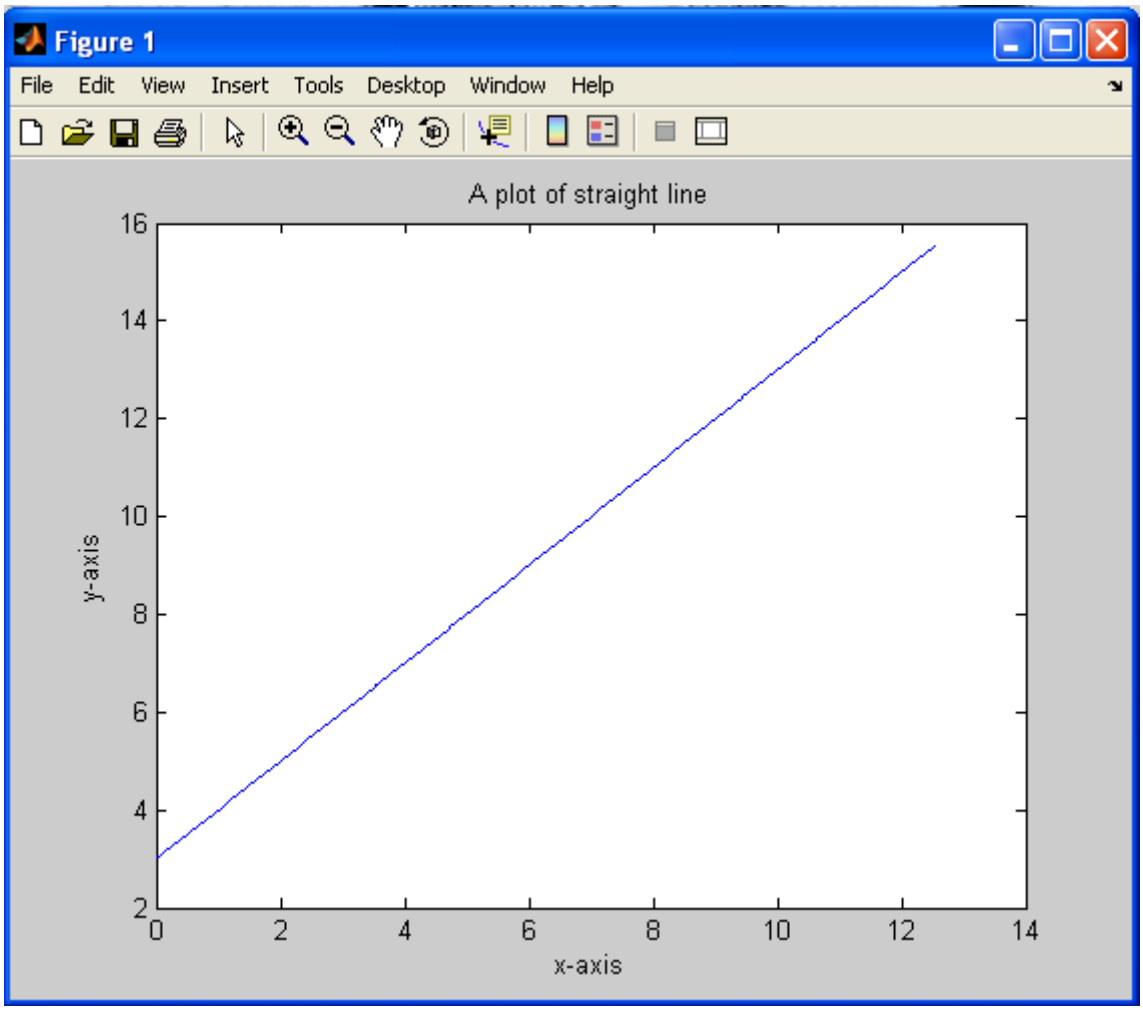
```
>> magic(3)
ans =
    8    1    6
    3    5    7
    4    9    2

>> a=ones(3)
a =
    1    1    1
    1    1    1
    1    1    1

>> a=a*2
a =
    2    2    2
    2    2    2
    2    2    2

>> b=2*a
b =
    4    4    4
    4    4    4
    4    4    4

>> operation=magic(4)
operation =
   16    2    3   13
    5   11   10    8
    9    7    6   12
    4   14   15    1
```

```
>> operation(2,3)
ans = 10

>> operation(2:3,:)
ans =
    5    11    10     8
    9     7     6    12

>> operation(:,2:3
ans =   2     3
       11    10
        7     6
       14    15

>> operation(2:3,2:2)
ans = 11
       7
```

Program-3

Write a program to Plot the Straight Line.

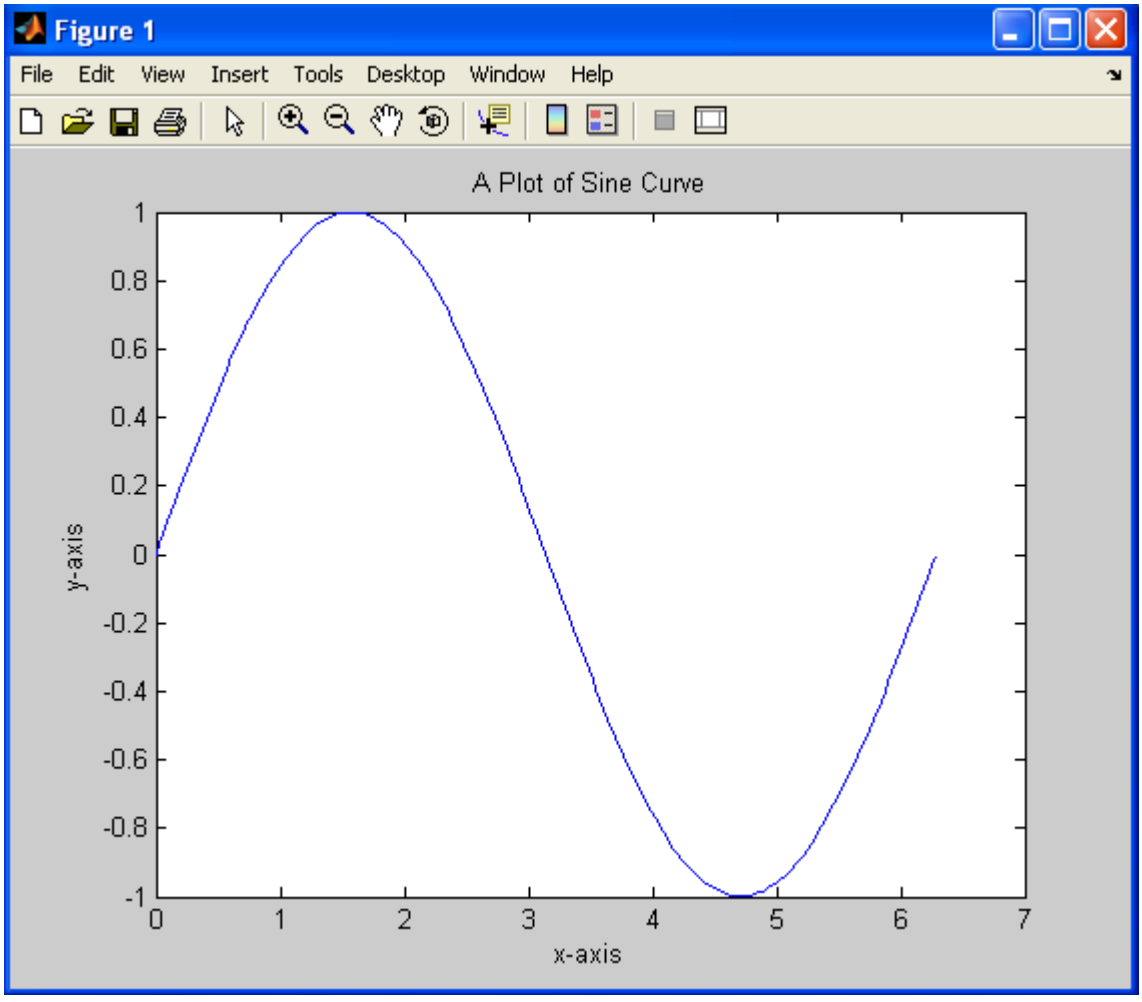
```
x=0:3.14/100:4*3.14;
y=x+3;
plot(x,y);
title('A Plot of Straight Line');
x label('x-axis');
y label('y-axis');
```



Program:-4

Aim:- Write a Program to draw a Sine curve.

```
x=0:3.14/100:2*3.14;
y=sin(x);
plot(x,y);
title('A Plot of Sine Curve');
xlabel ('x-axis');
ylabel ('y-axis');
```



Program-5

Aim:- Write a program how the Weight and Bios affect the value(s) of Neuron.

```
>> x=[1 2 3]

x =

    1     2     3

>> w=[3 4 5]

w =

    3     4     5

>> o=w' *x

o =

    3     6     9
    4     8    12
    5    10    15

>> b=1

b =

    1

>> e=b*w'

e =

    3
    4
    5

>>
```

Program-6

Aim:- How the choice of activation function affect the output of neuron. Experiment with the following
(i) Purelin
(ii) Binary Thresold (Hardlim, Hardlims)
(iii) Sigmoid (logsig,tansig)

```
>> x=[1,2,3]
x =
    1     2     3

>> w=[4,5,6]
w =
    4     5     6

>> o=x*w'
o =
   32

>> b=[1]
b =
    1

>>e =b*w'
e =
    4
    5
    6

>> n=o+e
n =
   36
   37
   38

>> hardlim(n)
ans =
    1
    1
    1

>> hardlims(n)
ans =
    1
    1
    1

>> purelin(n)
ans =
   36
```

```
    37
    38

>> logsig(n)
ans =
    1.0000
    1.0000
    1.0000

>> tansig(n)
ans =
     1
     1
     1
```

Practical No:7

Aim: How the weight and biased value are able to represent a decision boundary in the feature space.

Program: $w_{11} p_1 + w_{12} p_2 + b = 0$

```
> w=[1,2]

w =

    1    2

>> b=-1

b =

   -1

>> p2=-b/w(1,2)

p2 =

   0.5000

>> p1=-b/w(1,1)

p1 =

    1
```

Practical – 8

AIM : How the Perceptron Learning Rule works for linearly separable problems?

AND Case:

X1	X2	Y
-1	-1	-1
-1	1	-1
1	-1	-1
1	1	1

$b + w_1 x_1 + w_2 x_2 = 0$

```
>> w1 = 1

w1 =

     1

>> w2 = 1

w2 =

     1

>> b = -1

b =

    -1

>> x2 = -b/w2
```

x2 =

1

>> x1 = -b/w1

x1 =

1

OR Case:

X1	X2	Y
-1	-1	-1
-1	1	1
1	-1	1
1	1	1

$b + w_1 x_1 + w_2 x_2 = 0$

>> w1 = 1

w1 =

1

>> w2 = 1

w2 =

1

```
>> b = 1

b =

     1

>> x2 = -b/w2

x2 =

    -1

>> x1 = -b/w1

x1 =

    -1

>>
```

Practical – 9

AIM : How the Perceptron Learning Rule works for non-linearly separable problems?

XOR Case 1:

X1	X2	Y
-1	-1	-1
-1	1	1
1	-1	1
1	1	-1

```
>> w1 = 1

w1 =

     1

>> w2 = 1

w2 =

     1

>> b = 0

b =

     0

>> x2 = -b/w2

x2 =

     0

>> x1 = -b/w1

x1 =

     0

>>
```

XOR Case 2:

X1	X2	Y
-1	-1	-1
-1	1	1
1	-1	1
1	1	-1

```
>> w1 = -1

w1 =

-1

>> w2 = -1

w2 =

-1

>> b = 2

b =

2

>> x2 = -b/w2

x2 =

2

>> x1 = -b/w1

x1 =

2

>>
```

Practical – 10

AIM : Write a program to draw a graph with mul‘iple curves.

```
x=0:pi/100:2*pi;
y1=sin(2*x);
y2=cos(2*x);
plot(x,y1,x,y2);
title('a graph to plot multiple curves.');
```

OUTPUT