Jawaharlal Nehru Engineering College

Laboratory Manual

MATLAB

For

Second Year Students

INDEX

EXERCISE NO. 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 Mcrosoft Windows platform, double-click MATLAB shortcut icon on your windows desktop.

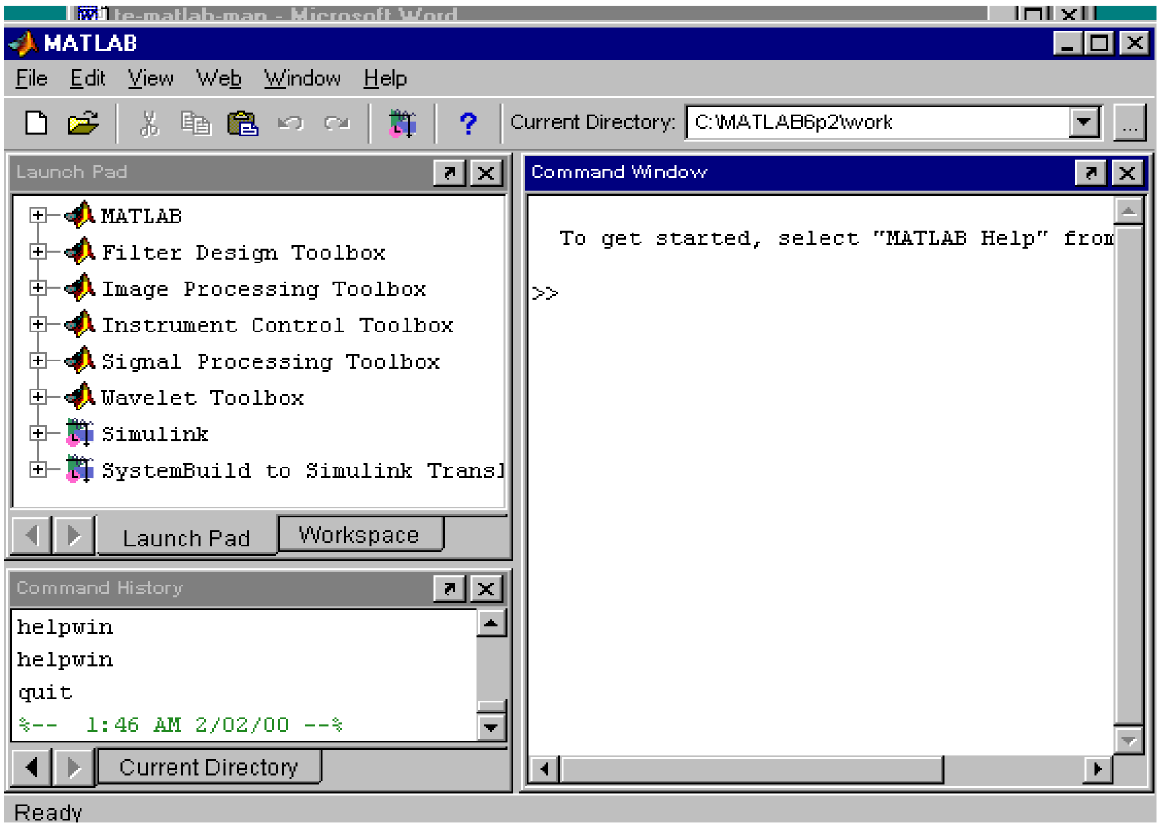
To quit MATLAB at any time, do one of the following:

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

DESKTOP TOOLS

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

MATLAB DESKTOP



```
 To get started, select "MATLAB Help" from
>>
```

```
helpwin
helpwin
quit
%-- 1:46 AM 2/02/00 --%
```

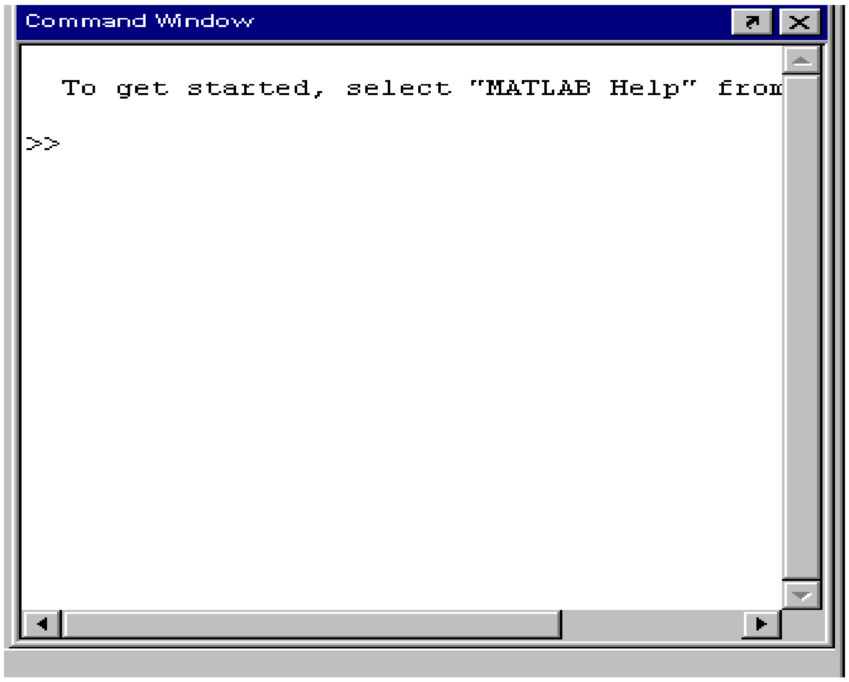
The following tools are managed by the MATLAB DESKTOP:

- 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

<u>THE COMMAND WINDOW</u>

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



```
  To get started, select "MATLAB Help" from

>>
```

<u>MATLAB WORKSPACE</u>

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.

<u>FILE OPERATIONS:</u>

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 Direcory 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 esiest way to import the data is to use MATLAB Import Wizard.

MATLAB provides folloing 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

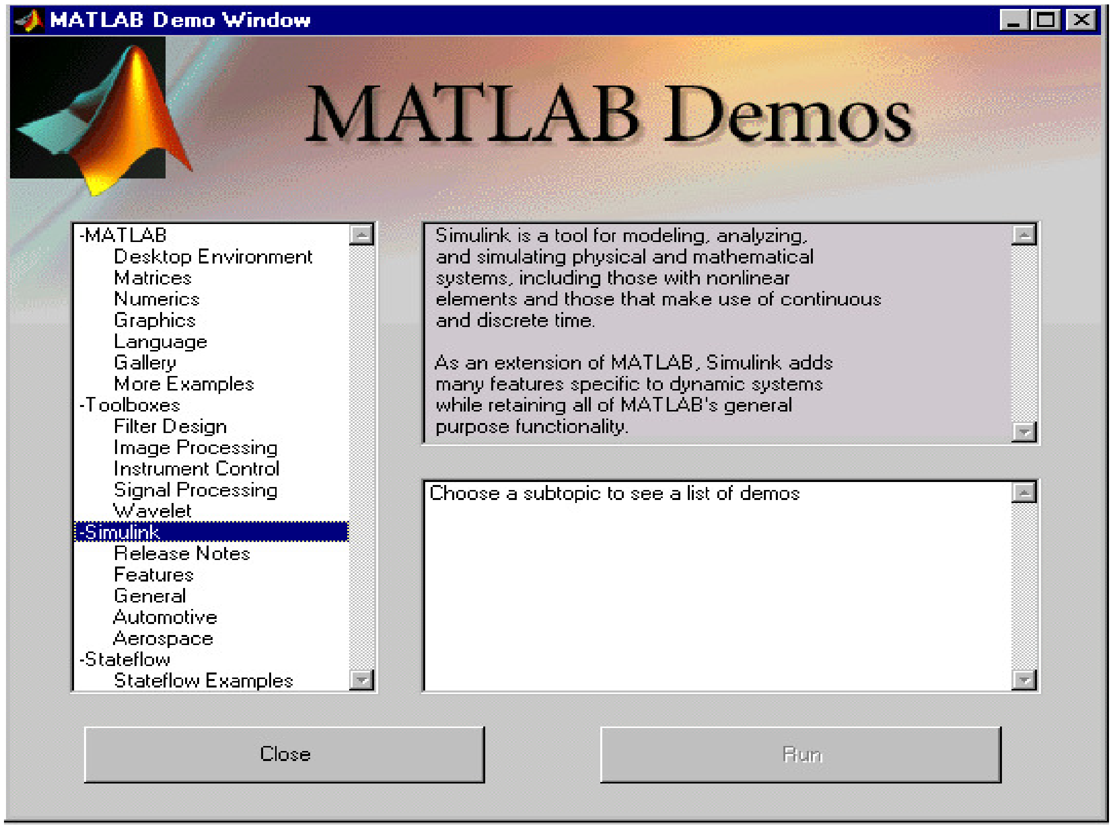
Demonstrations:
MATLAB provides very powerful Demonstrations for almost everything in the software.To start demo you have to use demo command.It's syntax is:

```
   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:



Simulink is a tool for modeling, analyzing, and simulating physical and mathematical systems, including those with nonlinear elements and those that make use of continuous and discrete time.

As an extension of MATLAB, Simulink adds many features specific to dynamic systems while retaining all of MATLAB's general purpose functionality.

Choose a subtopic to see a list of demos

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.
break	- Terminate execution of WHILE or FOR loop.
continue	- Pass control to the next iteration of FOR or WHILE loop.
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.

EXERCISE NO. 2

BASIC OPERATIONS 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.

Programs illustrating use of varius programming constructs

Program to find sum of numbers using For loop as well as While loop

Program1: For Loop

```
sum=0;   %  initialize  sum  to zero
for i=1:10;
n=input('Enter the number:') % ask for  the numbers
sum=sum+n  % add new no to previous sum
end
sprintf('Sum of numbers=') % display the sum
disp(sum)
```

Program 2: While Loop

```
sum=0;
i=0;
while i<10
  i=i+1
  n=input('Enter the number:')
 sum=sum+n
 end
 sprintf('Sum of numbers=')
 disp(sum)
```

Program to select required method out of given methods using switch-case

Program 3: Switch-case

```
sprinf('Classical- 1; Laplace- 2;  Fourier- 3') % assign nos to each mehod
i=inpu('Enter the choice')
swich i  %select the case as per no entered
  case 1
 disp('Classical')
 case 2
 disp('Laplace')
 case 3
 disp('Fourier')
otherwise
      disp('Not found')
   end
```

EXERCISE NO. 3

Basic operations on a Vector

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

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:

```
a=[1 2 3 4 5 6 4 3 4 5] %create vector a of 9 elements
b=a+2   %add 2 to each element
b.^2            %take square of each element
c=b * 2         %create a new vector c from b by multiplying each element of b by 2
plot(b)  %plot graph of b with grid lines
grid on
bar(b)          %plot bar graph of b
xlabel(‘Sample’)
ylabel(‘kgs’)
plot(b,’*’)     %use symbols to mark the points
axis([0 10 0 10])
```

<u>EXERCISE NO. 4</u>

Basic Matrix operations

Aim: Write a program which illustrates the basic matrix operations

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 matrx multiplied by it's inverse is identity matrix
9. Take square of matrix A
10. Find 4th 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 matrx 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
I=inv(A) * A                %Illustrate the fact that a matrx multiplied by it's inverse is identity matrix
A.^2     %Take square of matrix A
B.^4     %Find 4th power of matrix B

diag([ones(1,3)]           %create matrix of order 3,3 with each diagonal element=1
```

EXERCISE NO. 5

Operations on complex numbers

Aim: Write a program to illustrate operations on complex numbers

Algorithm:

```
1. Create a complex no
2. Show it's real part
3. Show it's imaginary part
4. Find conjugate of the complex no
5. Create vector of complex nos.
6. Take it's square
7. Create another vector of complex nos.
8. Convert complex nos in the matrix to equivalent polar form
```

Program:

```
a=4
b=6
c=complex(a, b)          %create a complex no
real(c)  %show it's real part
imag(c) %show it's imaginary part
conj(c)  %find conjugate of the complex no

y = [2+5*i 3+4*i 1+i]      %create vector of complex nos.
y.^2     % take it's square

A = [2+2*i 1+3*i ; 4+5*i 6-i]      %create another vector of complex nos.
abs(A)   %convert complex nos in the matrix to equivalent polar form
angle(A)
```

EXERCISE NO. 6

Polynomial evaluation

Aim: Write a program to evaluate polynomial at different values of variable and also to find out roots of a polynomial

%evaluate $3x^2+2x+1=0$ for x=5, x= 8, x=22

Algorithm:

1. Define vector p with coefficients of polynomiasl
2. Evaluate at x=5
3. Evaluate at x=5
4. Evaluate at x=22
5. Define vector a with values of x
6. Evaluate at vector a

%find roots of polynomial

7. Define vector q with coefficients of polynomiasl
8. Evaluate the roots

```
%POLYVAL Evaluate polynomial.
%   Y = POLYVAL(P,X), when P is a vector of length N+1 whose elements
%    are the coefficients of a polynomial, Y is the value of the
%   polynomial evaluated at X.
%       Y = P(1)*X^N + P(2)*X^(N-1) + ... + P(N)*X + P(N+1)
```

Program:

```
p = [3 2 1];        %define vector p with coefficients of polynomiasl
polyval (p,5)  %evaluate at x=5
polyval (p,8)  %evaluate at x=5
polyval (p,22) %evaluate at x=22

%This can also be done by defining vector a as [5 8 22] and using in polyval
a=[5 8 22] %define vector a with values of x
polyval(p,a) %evaluate at vector a


% ROOTS(C) computes the roots of the polynomial whose coefficients
% are the elements of the vector C. If C has N+1 components,
%  the polynomial is C(1)*X^N + ... + C(N)*X + C(N+1).

q=[1 -2 -24] %define vector q with coefficients of polynomiasl
roots(q)                    %evaluate the roots
q=[1 12 5 -150]
roots(q)
```

EXERCISE NO. 7

Structures

Aim: Write a program to illustrate use of structures

Program to store and retrieve patient's data in a hospital
Algorithm:
1. Create structure patient to store patient's data in a hospital
2. Display details of a patient
3. Enter details of another five patients using for loop
4. Display all details of all the patients

Program:
```
clear
patient.name='Ganesh Kale';          %create structure patient to store patient's data  in a hospital
patient.billing=600.00;
patient.bp=100;
patient.hm=13;
Patient %display details of a patient

for i=2:5
patient(i).name=input('Enter name of patient:','s') %enter details of other patients
patient(i).billing=input('Enter billing value:')
patient(i).bp=input('Enter bp test result:')
patient(i).hm=input('Enter hm test result:')
end

for i=1:5   % Display details of all patients
disp(patient(i).name)
disp(patient(i).billing)
disp(patient(i).bp)
disp(patient(i).hm)
end
```

EXERCISE NO. 8

User created functions

Aim: Write a program illusraing use of user created functions

Program to find out mean and standard deviation of given numbers
Algorithm:

```
    Main Program mstat.m
    1.  Ask the user for the numbers
    2.  Call function to find statistics
    Function stat.m
    1.  Define function
    2.  Find length of array passed
    3.  Find mean of numbers by calling subfunction avg
    4.  find standard deviation
    5.  Define subfunction avg
```

```
mstat.m
sprintf('Enter the numbers whose statistics has t be studied
for n=1:10,
  x(n)=input('no:')
end
stat(x)

stat.m
funcion[mean,stdev]=stat(x)
n=length
mean=avg(x,n)
stdev=sqrt(sum((x-avg(x,n)).^2)/n)
function[mean]=avg(x,n);
%avg subfunction
mean=sum(x)/n
```

EXERCISE NO. 9

Implicit and Explicit solutions

Aim: Write a program to find out implicit and explicit solutions of linear differential equations

```
% Finding out solutions t ordinary differential equations when no initial
% conditions are stated
%Here Symblic Toolbox is used.
%dsolve is a function available with Symblic Toolbax

 sprintf('Implicit solution of dx/dt=-ax is ')
 x = dsolve('Dx = -a*x')
 sprintf('Implicit solution of dx/dt=-ax is ')
 x = dsolve('Dx = -a*x')
 sprintf('Implicit solution of (dv/dt)^2 + v^2=4 is ')
 v = dsolve('(Dv)^2 + v^2 = 4')
 sprintf('Implicit solution of dx/dt=x+a is ')
 x = dsolve('Dx = x + a')
 sprintf('Implicit solution of dv/dt=(1-v)v^2 is ')
 v = dsolve('Dv = v^2*(1-v)')

% Finding out solutions t ordinary differential equations when initial
% conditions are stated
%Here Symblic Toolbox is used.
%dsolve is a function available with Symblic Toolbax

sprintf('Explicit solution of dx/dt=-ax, for x(0)=2 is ')
 x = dsolve('Dx = -a*x','x(0) = 2')
 sprintf('Explicit solution of (dv/dt)^2 + v^2=4, for v(0)=0 is ')
 v = dsolve('(Dv)^2 + v^2 = 4','v(0) = 0')
 sprintf('Explicit solution of dx/dt=x+a, for x(0)=2 is ')
 x = dsolve('Dx = x + a','x(0) = 2')
 sprintf('Explicit solution of dy/dt=-a+y, for y(0)=2 is ')
 y = dsolve('Dy = -a + y','y(0) = 2')
```

EXERCISE NO. 10

Toolboxes in MATLAB

Aim: Write in short description of commonly used toolboxes in MATLAB

1. **The Control System Toolbox** is a collection of functions for the modeling, analysis, and design of automatic control systems. It provides classical tools such as Bode plots, Nichols plots, or root locus, as well as modern techniques such as state-space LQG or pole placement design. Custom data structures allow for compact representation and easy manipulation of LTI models. The Toolbox contains functions to analyze model dynamics (poles, zeros, gain), plot time and frequency responses, discretize models, connect systems together, and design static or dynamic feedback compensators.

2. The Communications Toolbox is a collection of computation functions for research, development, system design, analysis, and simulation in the communications area. The Toolbox is designed to be suitable for both experts and beginners in the communications field.

3. The Data Acquisition Toolbox provides MATLAB users with tools to allow them to integrate live measured data with their data analysis and visualization applications. The Toolbox currently provides analog input, analog output, and digital I/O functionality for data acquired from, or targeted to, data acquisition hardware on a PC platform.

4. The Filter Design Toolbox is a collection of tools that provides advanced techniques for designing, simulating, and analyzing discrete-time filters. It extends the capabilities of the Signal Processing Toolbox by adding advanced filter design methods, and by adding the ability to simulate and analyze arbitrary-precision fixed-point and floating-point discrete-time filters.

5. The Fuzzy Logic Toolbox contains comprehensive tools to help you do automatic control, signal processing, system identification, pattern recognition, time series prediction, data mining, and financial applications.

6. The Instrument Control Toolbox provides MATLAB users with tools to allow them to communicate with instruments that support the GPIB (IEEE-488, HP-IB), VISA, serial port, TCPIP and UDP interfaces.

7. The Neural Network Toolbox includes many kinds of powerful networks for solving problems including: function approximation, modeling, signal processing and prediction classification, and clustering. These tools are an essential part of many applications, including engineering, finance, medicine, and artificial intelligence.

8. The Partial Differential Equation (PDE) Toolbox provides a powerful and flexible environment for the study and solution of partial differential equations in two space dimensions and time.

9, The Signal Processing Toolbox is a collection of tools built on the MATLAB technical computing environment. The toolbox provides two categories of tools: Command line functions for: Analog and digital filter analysis, Digital filter implementation FIR and IIR, digital filter design, Analog filter design Filter discretization ,Statistical signal processing ect.and Interactive graphical user interfaces for: Filter design and analysis, Window design and analysis

10. The Statistics Toolbox is a collection of MATLAB functions for descriptive, inferential, and graphical statistics, probability modeling, and random number generation. The toolbox includes several interactive graphic environments for dynamic visualization of data, functions, and probability distributions.

11. The Symbolic Math Toolbox integrates powerful symbolic and variable precision computing into the MATLAB environment.

12. The System Identification Toolbox is a collection of MATLAB functions for building models of systems and signals from observed data.

13. The Robust Control Toolbox is an extension of the Control Toolbox which enables the design of uncertainty-tolerant multiloop feedback systems. Capabilities include: Robustness Analysis sigma plots (singular value Bode plots), Mu-analysis (real, complex and mixed), Model-Order Reduction balanced truncation (for small absolute error) ,balanced-stochastic-truncation (for small percentage error) ,Multiloop Controller Synthesis

14. The Mapping Toolbox is a collection of MATLAB functions for display, manipulation and creation of geographic data. The Mapping Toolbox allows you to plot geographic data as easily as X-Y data. Both vector and matrix map data can be layered using one of 60 different projections. The toolbox provides functions for a wide variety of geographic and navigational calculations. It allows statistical analysis to account for the curvature of the Earth. Almanac functions provide spherical and elipsoidal representations of the Earth and other planets.

15. The Financial Toolbox provides a powerful set of financial functions essential to financial and quantitative analysis. Applications include pricing securities, calculating interest and yields, charting financial data, computing cash flows, analyzing derivatives, and optimizing portfolios.

16. The Database Toolbox allows you to use the power of MATLAB for sophisticated analysis of data stored in databases. From within the MATLAB environment, you can use Structured Query Language (SQL) commands to: Read data to and write data from a database. Apply simple and advanced conditions to your database queries. The Visual Query Builder allows users unfamiliar with SQL to query their database from MATLAB.