

Introduction to MATLAB

(Engineering Computing)

Contents:

1. Introduction : Background to Matlab

- 1.1 Objectives of the course
- 1.2 Course Literature
- 1.3 What is MatLab
- 1.4 Construction
- 1.5 Logging in

2. Basic Operations : Interactive Calculations

- 2.1 save / load function
- 2.2 Precedence of Operators

3. Vectors and Matrices

4. Graphics and Visulisation

5. The programming environment

- 5.1 Methodology for Solving Engineering Problems
- 5.2 Strings
- 5.3 Display
- 5.4 Keyboard User Interactions
- 5.5 Developing Programs
- 5.6 m-file scripts and function

6. Flow Control statements

- 6.1 Control Statements:
- 6.2 Loop Commands
- 6.3 If Statement

7. Polynomial Analysis / Curve fitting

- 7.1 Polynomials:
- 7.2 Multiplication of Polynomials:
- 7.3 Division of Polynomials
- 7.4 Curve fitting and Interpolation

8. Solution to systems of linear Equations

1. Introduction : Background to Matlab

1.1 Objectives of the course

The syntax of Matlab commands, Graphics and visualisation Feature, Matlab programming language.
To learn how to use Matlab as a mathematical tool in your research.

To give insight into a state of the art scientific tool in technical computation and visualisation.

1.2 Course Literature

The Student Edition of MATLAB, Prentice Hall.

MATLAB 6, FOR ENGINEERS, A. Biran & M Breiner, Addison Wesley.

The MATLAB 6, HANDBOOK, E. Part-Enander & A. Sjoberg, Addison Wesley.

Engineering Programming C, MATLAB, JAVA, M. Austin, D. Chancogne, John Wiley & Sons, Inc.

1.3 What is MatLab MATLAB = MATrix LABoratory

Matlab is a powerful, interactive numeric-computation and scientific visualisation software.

It has a simple and structured programming environment.

It can be used as a data analysis and visualisation Tool.

It can be extended via the available Toolboxes

It can be used for modelling, simulation and prototyping

Originally a user interface to provide access to matrix software developed by the LINPACK and EISPACK projects.

1.4 Construction

MATLAB works on PC (Windows/Linux), Mac and Unix systems.

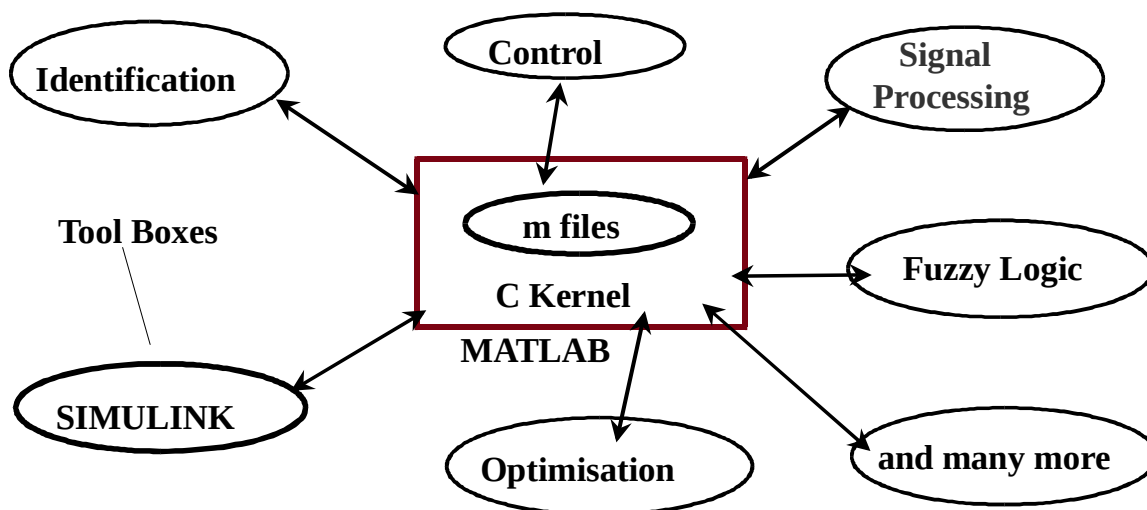
MATLAB is an open system, it contains source codes called mfiles that can be modified.

MATLAB's m-files are generally platform independent

The latest version is MATLAB 6.2.

The student version has a limited functionality

The core functionality in MATLAB contains compiled C-routines

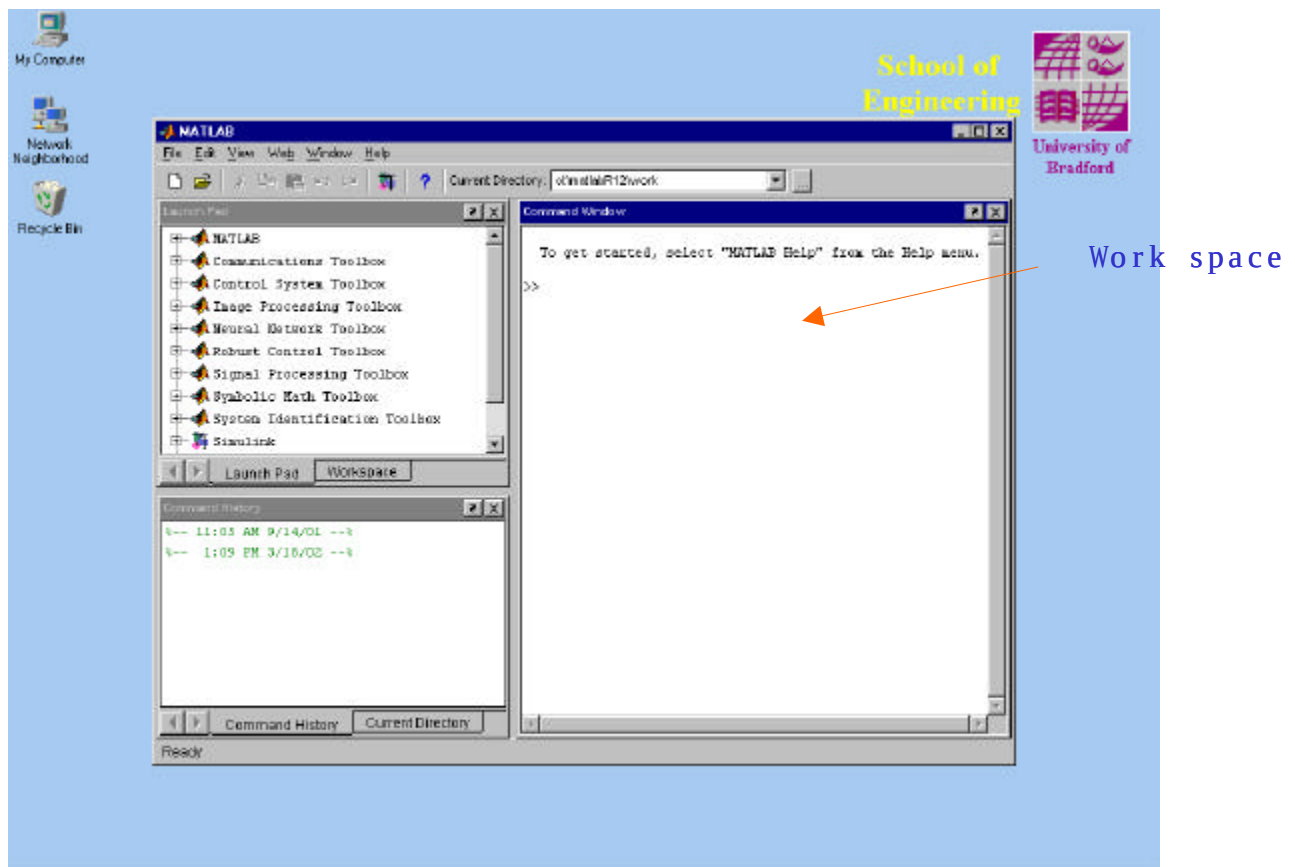


Further function can be added to MatLab by installing toolboxes. For example dynamical simulation can be preformed in Simulink toolbox.

Other toolboxes include:

System Identification, Model Predictive Control, Signal Processing, Wavelet, Optimisation, Control System, mu analysis and synthesis, Fuzzy Logic Neural Networks, Statistics, Symbolic maths. partial diff. Equations, Image Processing, Financial, ... etc.

1.5 Logging in



2. Basic Operations

Matlab is interactive

```
>> 5+3-2*(4/6)
ans = 6.6667
```

```
>> cos(pi)
ans = -1
```

variables can be declared.

```
>> x=2; y=5; z=x+y
z = 7
```

scalar variables

```

>> x = 10;           % define a value for variable x
>> y = 25;           % define a value for variable y
>> z = x*y;          % display the result in z
>> v = 2 + 4.5*i      % v is a complex number
>> u = x+j*y          % complex operators i=j
>> A = 214;           % case sensitive
>> B=sin(y)           % predefined functions
>> help               % help with syntax and function definition

```

who , whos get a list of all the variables defined in Workspace.

```

>> who
Your variables are:  ans      x      y      z

```

```

>> whos
      Name      Size      Bytes Class
      ans       1x1         8 double array
      x         1x1         8 double array
      y         1x1         8 double array
      z         1x1         8 double array
Grand total is 4 elements using 32 bytes

```

2.1 save / load function

The working directory is controlled by

```

>> cd → D:\matlab\bin
>> cd a:\           % change directory to a:
>> save test1
>> clear all
>> whos
>> dir → test1.mat   *.mat is extension of MatLab files
>> load test1
>> who → Your variables are:  ans      x      y      z
Matlab uses double precision which gives approximately 16 significant figures

```

```

>> format long
>> format compact

```

Simple calculation / diary / import data / read data / Input output. ascii file

2.2 Precedence of Operators

- 1 Parentheses
- 2 Exponentiation, left to right
- 3 Multiplication and division, left to right
- 4 Addition and subtraction, left to right Operators

+, -, *, / , \ are all matrix operation symbols

3. Vectors and Matrices

An order collection of numbers $a_1, a_2, a_3, \dots, a_n$ can be defined in MATLAB as an array (Vector).

```
>>A=[a1, a2, a3, ... , an]
```

The elements of a vector can be separated by "space" (blanks) or a ",".

```
>>A=[a1, a2, a3, ... , an] or >>A=[a1 a2 a3 ... an]
```

A row vector can be converted to a column vector by Transposition. The operation is performed by the apostrophe (prime) '

```
>>A=[a1, a2, a3, ... , an]
```

```
>>B=A'
```

```
B=
```

```
a1
```

```
a2
```

```
a3
```

```
.
```

```
an
```

Matrices are two Dimensional arrays. They are defined similarly:

```
>> A = [6,2,8; 4,-5,6 ; 5,-6,7]
```

Elements are separated by "," or space

```
>> A = [6 2 8 ; 4 -5 6 ; 5 -6 7]
```

$$A = \begin{bmatrix} 6 & 2 & 8 \\ 4 & -5 & 6 \\ 5 & -6 & 7 \end{bmatrix}$$

Transpose is given by

```
>> B = A' (try this example)
```

Common operators are available

```
>> A*B
```

```
>> A+B
```

(MATLAB is case-sensitive A and a are two different variables)

Indexing using parentheses:

Any element of a matrix is identified by two indices; the first indicates the row , and the second, the column.

```
>>A(2,1)
ans
    4
```

```
>>A(3,2)
ans
   -6
```

Indexing of a complete row / column is performed by the colon-operator

```
>> A(1:2,1:2)
```

In the above example the colon is used to show rows from 1 to 2 and columns from 1 to 2.

Elementary functions also work with Vectors or Matrices:

```
>> sin(A)
```

Specialized matrix functions and operators

```
>> sqrtm(A)
>> A.^2
>> A^2
>> A.*B
```

```
>> help elmat; help elfun
```

Numerical linear algebra

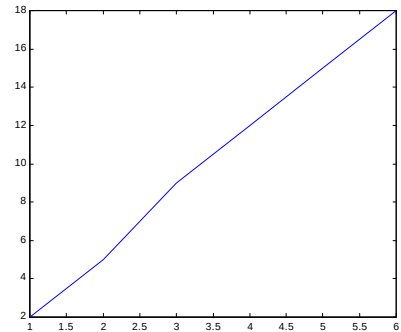
```
>> inv(A)
>> BA
>> det(A)
>> rank(A)
>> [V,D]=eig(A)
```

4. Graphics and Visualization

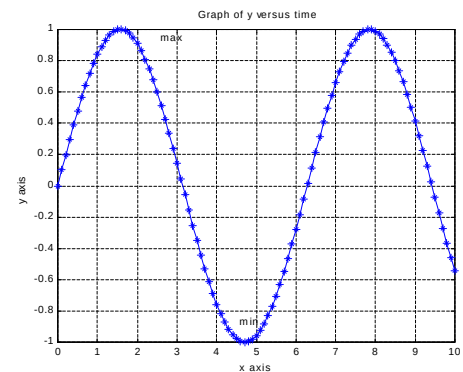
2D Plotting Functions

Two-dimensional graphics

```
>> A = [2,5,9,12,15,18]
>> plot(A)
```



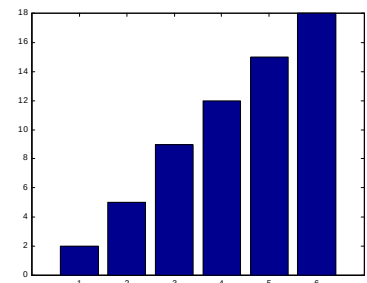
```
» t=0:0.1:10;
» y=sin(t);
» plot(t,y)
» plot(t,y,'*-')
» grid % grid : adds grid-lines
» xlabel('x axis')
» ylabel('y axis') % xlabel: adds x and y axis labels
» title(' Graph of y versus time') %Title : adds title
» gtext('max') %gtext : to add text on a graph.
```



Bar plot

(Two-dimensional graphics)

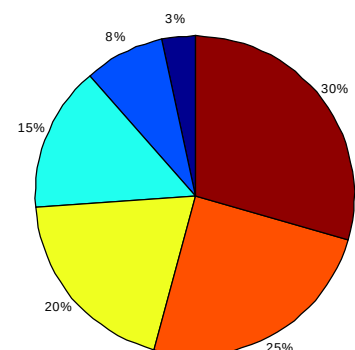
```
>> A = [2,5,9,12,15,18]
>> bar(A)
```



Pie plot

(Two-dimensional graphics)

```
>> A = [2,5,9,12,15,18]
>> pie(abs(A))
```



Three-dimensional graphics

```
>> t=-10*pi:pi/16:4*pi
```

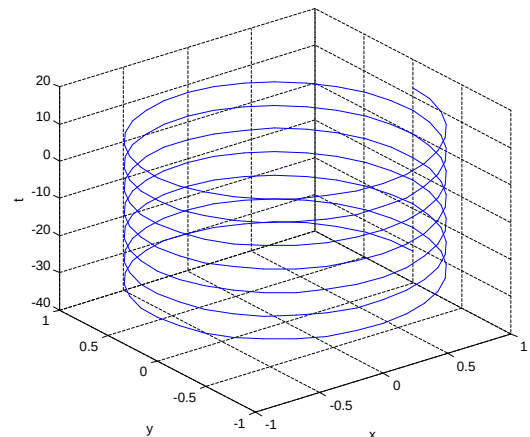
```
>> x=cos(t)
```

```
>> y=sin(t);
```

```
>> plot3(x,y,t)
```

Subplots

```
>> Subplot(221),plot(t,x1)
```



5. The programming environment

5.1. Engineering Problems “Solving Methodology”

- State the problem clearly
- Describe the input and output information
- work the problem by hand (using calculator).
- Develop a MatLab Solution (program)
- Test the program

5.2. Strings:

Strings are one dimensional arrays (vectors) containing characters.

The statement

```
» x='HELLO'
```

creates a vector whose components are the ASCII codes for the characters.

```
» x(3)
```

```
ans =
```

```
L
```

```
» x(2)
```

```
ans =
```

```
E
```

```
» y='WORLD'
```

```
» y(5)
```

```
ans =
```

```
D
```

```
» z=[ x y]
z =
HELLOWORLD
```

5.3 disp command (displays the array)

```
» A=[1 5;8 -13]
A =
     1     5
     8    -13
```

```
» disp(A)
     1     5
     8    -13
```

disp (displays the string)

If variable x is a string, the text is displayed.

```
» disp(x)
HELLO
```

```
» disp('x')
x
```

5.4 Keyboard User Interactions

Input Command

Prompting for input and reading it can be achieved by the use of the input function:

Example:

```
n = input ( 'Enter a number:' );
will prompt for a number and read it into the variable named n.
```

Other useful commands to control or monitor execution: Echo, Pause

```
» w1=input('give a number to w1: ')
give a number to w1: 100
w1 =
    100
```

```
» name=input('what is your name ? ')
what is your name ? 'john'
name =
John
```

Engineering Problem Solving Methodology

5.5. Developing Programs

- To write programs successfully:

Understand the problem thoroughly.

Break it down into its most fundamental logical stages.

Generate a systematic logical procedure or Algorithm for solving the problem using pseudo-code.

- Pseudo - Code

In “pseudo-code” statements for solving the problem are presented in plain English.

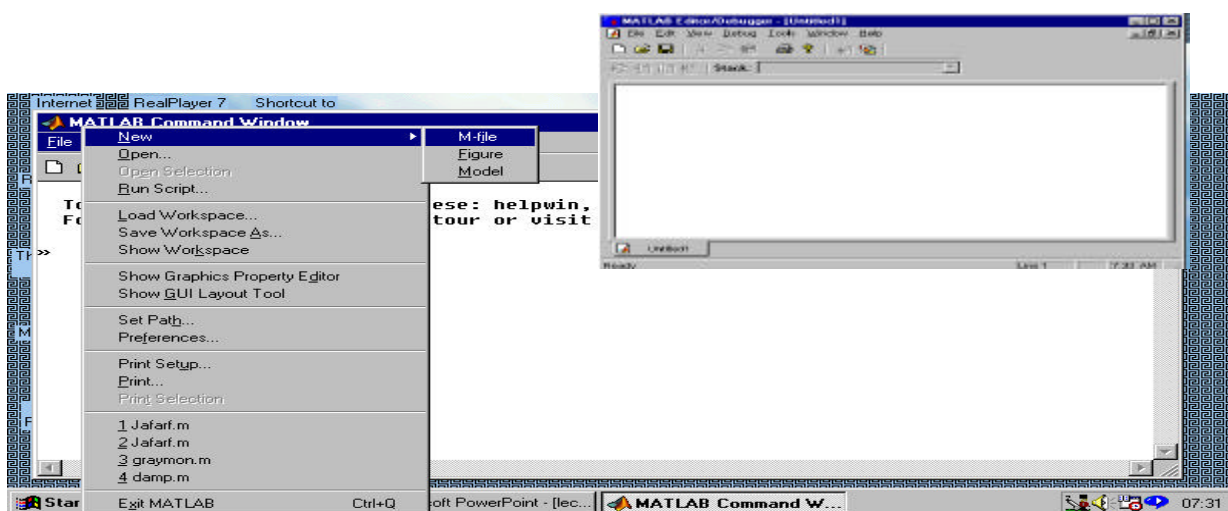
Therefore, a “structure plan” is a representation of an algorithm in *pseudo-code*.

In “Top Down” approach the overall structure of a program is clearly thought out before worrying about the details of syntax (coding), and this reduces the number of errors.

5.6 Executable Programs in MATLAB (MatLab Scripts or m-files)

A sequence of MatLab commands can be put into a file which can be executed by invoking its name.

How Matlab Scripts or m-files works.



Executable m-files should be located in the current working directory or somewhere on the search path.

```
Example 1 (A simple script file)
% Start up program (mystart1.m)
% this program greets you
%*****
% created by yourname on date
%*****
disp('Hello')
% disp displays the string array
name=input('Enter your name enclosed between inverted commas = ');
% input .....
de=date;
% date .....
% clc .....
clc
any_parameter=[' Hello ' name ' . ' ' Today is Tuesday ' de];
disp(any_parameter)
```

There are two types of .m files:

1. Script files
2. Function files

Functions (function file):

An m file that contains the word function at the beginning of the first line is a function file.

The first line of the function file must contain the word function.
The second line should contain help text and must begin with %.

Function files are saved as mfiles.

An example of an existing function file in MATLAB for calculating mean value of a series of numbers.

```
function y = mean(x)
% MEAN      Average or mean value.
%      For vectors, MEAN(X) is the mean value of the elements in X.
%      For matrices, MEAN(X) is a row vector containing the meanvalue
%      of each column.
%      See also MEDIAN, STD, MIN, MAX.
%      Copyright (c) 1984-94 by The MathWorks, Inc.
[m,n] = size(x);
if m == 1
    m = n;
end
y = sum(x) / m;
```

Exercise: Find the mean.m , help.m and plot.m files in the MATLAB directory.

Create the following function file:

```
function h=hyp(a,b)

% hypotenuse of right angled triangle
% Pythagoras theorem
% input the sides of the triangle (a and b).
h=sqrt(a^2+b^2);
```

Exercise : run the program , check help hyp

Differences between scripts and functions:

Arguments can be passed to function files and also results are returned as the value of the function.

All variables defined before a call to script remain available inside the script and those created inside the scripts remains available when returned from it.

Where as only variables passed to a function files are available within the function and any created within the function are not available when exited from the function.

The function/variable duality causes MatLab to search in following order:

1. variable in current workspace
2. built-in variable
3. built-in m-file
4. m-file in current directory
5. m-file on search path

Built-in Functions in MatLab

Check the following Examples:

general (abs, fix,exp,log,sqrt etc.)

mathematical special (bessel, erf, gamma etc.)

trigonometric (sin, cos, ... etc.)

data analysis (min, max, std , sort , diff cov etc.)

chol(A) Cholesky factorisation

cond(A) matrix condition number

6. Flow control - selection

So far we have looked at sequential steps.

However there are many problems which need to execute different sets of commands if a specified condition is true or false.

selection

Sequential steps:

One operation is executed after another until the end of the program.

Control Statements:

Loop Commands

Selection Commands

If Statement

Is a conditional selection statement, it is used to test a condition or ask a question:

```
If logical expression
    statements
end
```

To perform switching actions use If statement in combination with else or elseif

If - else - end

```
if    <condition1>
    < Program 1 >
elseif <condition2>
    < Program 2 >
elseif <condition3>
    < Program 3 >
.....
elseif <conditionN>
    < Program N >
end;
```

Relational Operators:

<	less than
<=	less than or equal
==	equal
~=	not equal
>	greater than
>=	greater than or equal

EXAMPLES:

```
% test1.m
a=input('input a:');
b=input('input b:');
t=(0:1:100)';
y=sin(t); x=cos(t); z=y.*x

% switching action
if a < b
plot(t,y);title('condition 1')
elseif a==b
plot(t,x);title('condition 2')
elseif a > b
plot(t,z);title('condition 3')
end;

% test2.m
t=(0:1:100)';y=sin(t); x=cos(t);
m=input('type 1 for sin or 2 for cos:');
if m==1
plot(t,y)
elseif m==2
plot(t,x)
elseif m>=3
'wrong choice try again'
end;
```

7 . Polynomial Analysis/ Curve fitting

7.1 Polynomials:

A polynomial (P) is a function of a single variable that can be expressed in the following general form:

$$P = a_0 + a_1 x + a_2 x^2 + \dots + a_n x^n$$

x is the variable and $a_0, a_1, a_2, \dots, a_n$ are the coefficients. Degree of the polynomial is the largest exponent.

Examples:

$$W(s) = b_n s^n + \dots + b_2 s^2 + b_1 s + b_0$$

$$Q(y) = y^5 + 17y^4 + 9y^2 - 34y + 5$$

$$X(s) = 9s^2 - 34s + 5 \quad \text{Quadratic equation}$$

In MATLAB Polynomials are represented by their coefficient terms, in order of descending powers of the variables. For example:

```
P1 = 1 x2 + 9 x + 20
» p1=[1, 9 , 20]
or
P2 = x3 -15 x - 4
» p2=[1, 0 , -15 , - 4]

» p1=[1, 9 , 20]
» r1=roots(p1)
r1 =
    -5
    -4
» p2=[1, 0 , -15 , -4]
» r2=roots(p2)
r2 =
    4.0000
   -3.7321
   -0.2679
```

Multiplication of Polynomials:

Division of Polynomials

Use convolution function (conv)

```
» p1=[1, 2]                % P1 = x + 2
» p2=[2, 0, 4]             % P2 = 2 x2 + 4
» p3=conv(p1,p2)
```

```
p3 = 2    4    4    8
```

which is

$$P_3 = 2x^3 + 4x^2 + 4x + 8 \text{ or } (x+2)(2x^2+4)$$

Retrieving Polynomial

Polynomial coefficients can be retrieved from polynomial roots:

```
» r1=[-5,-4]
» p1=poly(r1)
p1 =    1    9   20
```

ROOTS and POLY are inverse functions of each other.

In $q=(x-2)^4$

there are four roots equal to 2. Therefore

```
» R=[2 2 2 2]
» q = poly(R)
q = 1   -8   24  -32   16
(use conv to find the same results)
```

Division of Polynomials

use de-convolution function

```
» R1=deconv(p3,p1)
```

```
R1= 2 0 4
```

Polynomial Derivatives

polyder(p) returns the derivative of the whose coefficients are the elements of vector p.

$$P_3 = 2x^3 + 4x^2 + 4x + 8$$

```
» P3=[2 4 4 8]
```

```
» de =polyder(P3)
```

```
de =
```

```
6 8 4      de = 6x2 + 8x + 4
```

Polynomial Evaluation

```
» x=(0:0.01:1)';
```

```
» v=polyval(p,x);
```

```
» plot(x,v)
```

Curve fitting and Interpolation

To fit an analytic function to a measured data.

Curve fitting -

Finds a smooth curve that best fits the data (not necessarily passes through any data point).

Interpolation -

Any point between two data points can be found. (assumes data is correct).

1. Curve fitting or regression

Least squares curve fitting to polynomials

(minimising the sum of the squared error)

POLYFIT Fit polynomial to data polyfit(x,y,n)

finds the coefficients of a polynomial P(X) of degree n that fits a set of data y (measured at intervals given by x), using least-squares method.

```
» x=0:10
```

```
» y=[2,3.5,5,9.8,10.3,11.8,12.2,13.13,13.5,13.9,10]
```

```
» plot(x,y)
```

```
» f1=polyfit(x,y,1)
```

f1 = 1.0515 4.3000

» L1=polyval(f1,x,1);

» plot(x,L1)

» f2=polyfit(x,y,2)

f2 = -0.2240 3.2913 0.9402

»L2=polyval(f2,x,2)

» plot(x,[L1,L2])

» f5=polyfit(x,y,5)

f5 =-0.0042 0.1025 -0.8987 3.1293 -1.4733 2.1112

» L5=polyval(f5,x,5)

» plot(x,[y,L1,L2,L5])

2. Interpolation

MatLab interpolates the data point.

interp1

2.1. linear interpolation

» y=[2.0, 3.5, 5.0, 9.8, 10.3, 11.8, 12.2, 13.13, 13.5, 13.9, 10.0]
 1 2 3 4 5 6 7 8 9 10 11

» t1=interp1(y,8.5)

t1 =

13.3150

2.2. third order polynomial interpolation (cubic splines)

» t2=interp1(y,8.5,'spline')

t2 = 13.3268

8. Solution of Linear equations

Example:

Solve the following set of linear equations;

$$2x + 3y + z = 17$$

$$x + 4y + 2z = 22$$

$$x + y + 5z = 25$$

$$A = \begin{bmatrix} 2 & 3 & 1 \\ 1 & 4 & 2 \\ 1 & 1 & 5 \end{bmatrix};$$

$$B = \begin{bmatrix} 17 \\ 22 \\ 25 \end{bmatrix};$$

$$R = A \setminus B$$

$$C = \text{inv}(A)$$

If matrix A is ill conditioned {check the determinant $\det(A)$ }

Important Matrix and Vector Functions

L/U factorisation;

$$[L \ U] = \text{lu}(A)$$

Q/R factorisation

$$[Q \ R] = \text{qr}(A)$$

Eigenvalue Decomposition for an n-by-n matrix A, n numbers that satisfy the eqn. $Ax = \lambda x$ can be found by $\text{eig}(A)$

To obtain the eigen vectors as well use

$$[X, D] = \text{eig}(A);$$

in which case eigen vectors will be stored in the columns of X and D will be a diagonal matrix with the eigen values making up the diagonal terms such that $A \cdot X = X \cdot D$ is satisfied.