|  |  |
| --- | --- |
| **Faculty Member:\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
| **Semester: \_\_\_\_\_\_\_\_\_\_\_\_\_** | **Section: \_\_\_\_\_\_\_\_\_\_\_\_\_\_** |

**EE-211: Electrical Network Analysis**

**Lab 01: Introduction to MATLAB**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PLO4/CLO4** | | **PLO5/CLO5** | **PLO8/CLO6** | **PLO9/CLO7** |
| **Name** | **Reg. No** | **Viva /Quiz / Lab Performance**  **5 marks** | **Analysis of data in Lab Report**  **5 marks** | **Modern Tool Usage**  **5 marks** | **Ethics and Safety**  **5 marks** | **Individual and Team Work**  **5 marks** |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

**Lab # 01: Introduction to MATLAB**

#### **Learning Objectives:**

In this lab, you will learn:

* The interface of MATLAB
* Basic arithmetic operators
* Matlab variables
* Matrix manipulation

**Software Used:** MATLAB

**Description**:

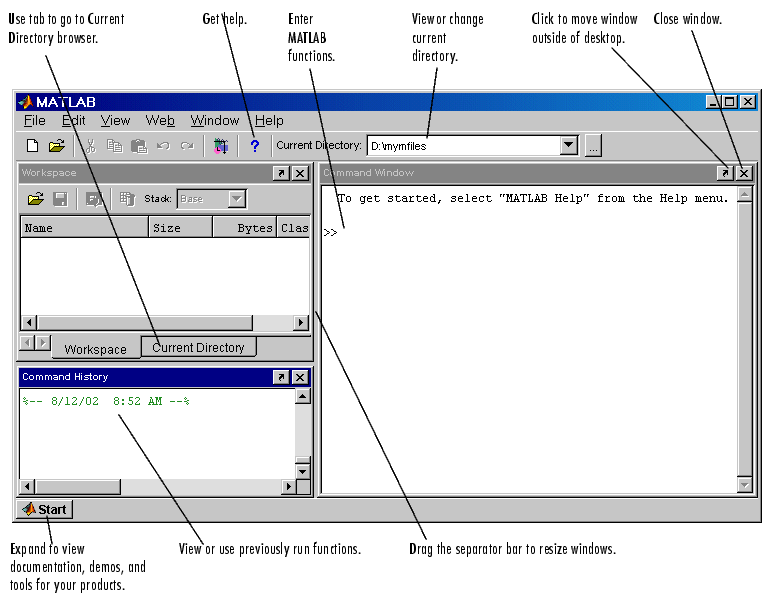
Matlab (stands for Matrix Laboratory), it integrates computation, visualization, and programming in an easy-to-use environment, where problems and solutions are expressed in familiar mathematical notation.

**Examples:**

* Matrix computations and linear algebra
* Solving nonlinear equations
* Numerical solution of differential equations
* Mathematical optimization
* Statistics and data analysis
* Signal processing
* Modelling of dynamical systems
* Solving partial differential equations
* Simulation of engineering systems

**Strengths of MATLAB**

* MATLAB is relatively easy to learn.
* MATLAB code is optimized to be relatively quick when performing matrix operations.
* MATLAB may behave like a calculator or as a programming language.
* MATLAB is interpreted, errors are easier to fix.

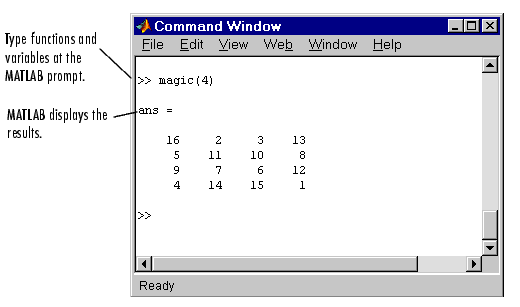


“Fig 1.1”

**Desktop Tools**

1. Command Window

Use the Command Window to enter variables and run functions and M-files.



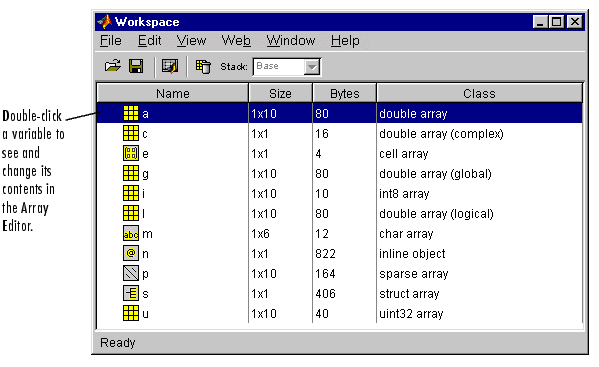
“Fig 1.2”

1. Command History:

Statements you enter in the Command Window are logged in the Command History. In the Command History, you can view previously run statements, and copy and execute selected statements.

1. Workspace Browser:

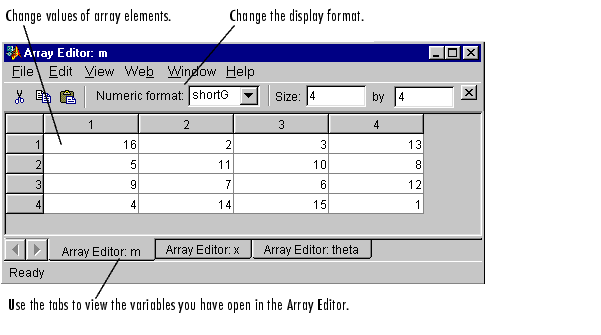
The MATLAB workspace consists of the set of variables (named arrays) built up during a MATLAB session and stored in memory.



“Fig 1.3”

1. Array Editor:

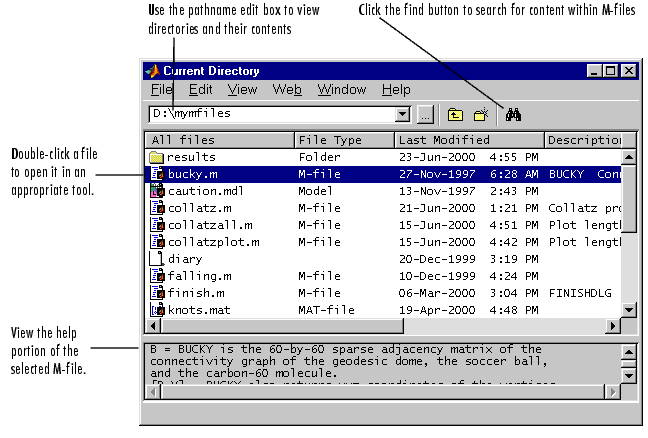
* Double-click a variable in the Workspace browser to see it in the Array Editor
* Use the Array Editor to view and edit a visual representation of one- or two-dimensional numeric arrays, strings, and cell arrays of strings that are in the workspace.



“Fig 1.4

1. Current Directory Browser:

It lists all the files that are available in current directory.



“Fig 1.5”

**Reserved Words:**

* Reserved Words List:

|  |  |  |  |
| --- | --- | --- | --- |
| for | end | if | while |
| function | else if | case | otherwise |

|  |  |  |  |
| --- | --- | --- | --- |
| switch | continue | else | try |
| catch | global | persistent | break |

* This list is returned as an output of the ‘iskeyword’ function. For example,

>> iskeyword(‘case’)

ans =

1 (else 0 will be displayed)

* The function isvarname(‘MCS’) work in the same manner.
* Matlab will report an error if you try to use a reserved word as variable.

**Basic Arithmetic Operations:**

|  |  |  |
| --- | --- | --- |
| Operations | Symbol | Example |
| Addition  Subtraction  Multiplication  Division | +  -  \*  / or \ | 3 + 22  54.4 – 16.5  3.14 \* 6  10/2 or 2\10 |

* Expressions are evaluated from left to right, with precedence;

Exponentiation > Multiplication & Division > Addition & Subtraction.

**Variables Naming Rule:**

* The first character MUST be alphabetic followed by any number of letters, digits, or underscore.
* The variable names are case sensitive.
* Blanks are NOT allowed in a variable name.
* Variable names can contain up to 63 characters.
* Punctuation characters are not allowed, because many of them have special meanings in Matlab.

**Matlab Special Variables:**

ans Default variable name for results

pi Value of p

inf infinity

NaN Not a number e.g. 0/0

i and j i = j = -1

realmin The smallest usable positive real number

realmax The largest usable positive real number

Examples:

>>a = 2

a =

2

>> 2\*pi

ans =

6.2832

**Types of Variables:**

|  |  |
| --- | --- |
| Type | Examples |
| Integer | 1362,-5656 |
| Real | 12.33,-56.3 |
| Complex | X=12.2 – 3.2i (i = sqrt(-1)) |

Complex numbers in MATLAB are represented in rectangular form. To separate real & imaginary part

* H = real(X)
* K = imag(X)
* Conversion between polar & rectangular
* C1 = 1-2i
* Magnitude: mag\_c1 = abs(C1)
* Angle: angle\_c1 = angle(C1)
* Note that angle is in radians

**Useful Matlab Commands:**

* Who List known variables
* Whos List known variables plus their size
* clear all Clear all variables from work space
* clear x y Clear variables x and y from work space
* Clc Clear the command window only and not any variable
* close all closes all open figures
* Extras:

Using up-arrow key allow user to recall most recently used commands

Another trick is to use a ‘letter’ prior using up-arrow

**Other MATLAB symbols:**

|  |  |
| --- | --- |
| **,**  **%**  **;**  **:** | separate statements and data    start comment which ends at end of line  (1) suppress output  (2) used as a row separator in a matrix  specify range |

**MATLAB Matrices:**

* MATLAB treats all variables as matrices. For our purposes a matrix can be thought of as an array, in fact, that is how it is stored.
* Vectors are special forms of matrices and contain only one row OR one column.
* Scalars are matrices with only one row AND one column.
* A matrix can be created in MATLAB as follows (note the commas AND semicolons):

» matrix = [1 , 2 , 3 ; 4 , 5 ,6 ; 7 , 8 , 9]

matrix =

1 2 3

4 5 6

7 8 9

Row Vector:

* A matrix with only one row is called a row vector. A row vector can be created in MATLAB as follows (note the commas):

» rowvec = [12, 14 , 63]

rowvec =

12 14 63

* Row vector can also defined in a following way:

rowvec = 2 : 2 : 10; % start : step size : stop

* rowvec =

2 4 6 8 10

Column Vector:

* A matrix with only one column is called a column vector. A column vector can be created in MATLAB as follows (note the semicolons):

» colvec = [13 ; 45 ; -2]

colvec =

13

45

-2

Extracting a Sub-Matrix:

* A portion of a matrix can be extracted and stored in a smaller matrix by specifying the names of both matrices and the rows and columns to extract. The syntax is:

sub\_matrix = matrix ( r1 : r2 , c1 : c2 ) ;

where r1 and r2 specify the beginning and ending rows and c1 and c2 specify the

beginning and ending columns to be extracted to make the new matrix.

* A column vector can be extracted from a matrix. As an example we create a matrix below:

» matrix=[1,2,3;4,5,6;7,8,9] Here we extract the column 2 of matrix and

make a column vector:

matrix = » col\_two =matrix( 1:3 ,2: 2)

1 2 3 col\_two = 2

4 5 6 5

7 8 9 8

* A row vector can be extracted from a matrix. As an example we create a matrix below:

» matrix=[1,2,3;4,5,6;7,8,9] Here we extract row 2 of matrix and make a row

vector. Note that 2:2 specifies the 2nd row and

matrix = 1:3 specifies columns of 2nd row.

1 2 3 » rowvec =matrix(2 : 2 , 1 : 3)

4 5 6 rowvec =

7 8 9 4 5 6

Concatenation:

* New matrices may be formed out of old ones
* Suppose we have:

a = [1 2; 3 4]

a = 1 2

3 4

Input output

[a , a, a] ans = 1 2 1 2 1 2

3 4 3 4 3 4

[a ; a; a] ans =

1 2

3 4

1 2

3 4

1 2

3 4

[a, zeros(2); zeros(2), a'] ans = 1 2 0 0

3 4 0 0

0 0 1 3

0 0 2 4

Scalar Matrix Addition & Subtraction:

» a=3;

» b=[1, 2, 3;4, 5, 6]

b =

1 2 3

4 5 6

» c= b+a % Add a to each element of b

c =

4 5 6

7 8 9

Scalar - Matrix Multiplication:

» a=3;

» b=[1, 2, 3; 4, 5, 6]

b =

1 2 3

4 5 6

» c = a \* b % Multiply each element of b by a

c =

3 6 9

12 15 18

Other matrices Operations:

Let a=[1 4 3;4 2 6 ;7 8 9]

* det(a) : Find the determinent of a matrix.

ans = 48

* inv(a) : Find the inverse of matrix.

ans =

-0.6250 -0.2500 0.3750

0.1250 -0.2500 0.1250

0.3750 0.4167 -0.2917

* a' : Find the transpose of a matrix.

ans =

1 4 7

4 2 8

3 6 9

* min(a) :Return a row vector containing the minimum element from each column.

ans = 1 2 3

* min(min(a)): Return the smallest element in matrix:

ans = 1

* max(a) : Return a row vector containing the maximum element from each column.

ans = 7 8 9

* max(max(a)): Return the max element from matrix:

ans = 9

* a.^2 :Bitwise calculate the square of each element of matrix:

ans =

1 16 9

16 4 36

49 64 81

* sum (a) : treats the columns of ‘a’ as vectors, returning a row vector of the sums of each column.

ans = 12 14 18

* sum(sum(a)): Calculate the sum of all the elements in the matrix.

ans = 44

* size (a) : gives the number or rows and the number of columns:

>> [r c] = size(a)

r = 3 c = 3

* Let a =[ 4 5 6] , length(a) finds the number of elements in row vector.

Bitwise Multiplication of Two Vectors:

Let a=[1 2 3] ; b=[4 5 6];

* a.\*b :Bitwise multiply the each element of vector ‘a’ and ‘b’:

ans =

4 10 18

Matrix Division:

* MATLAB has several options for matrix division. You can “right divide” and “left divide”.

Right Division: use the slash character

» A / B

This is equivalent to the MATLAB expression

» A\*inv (B)

Left Division: use the backslash character

» A \ B

This is equivalent to the MATLAB expression

» inv (A)\*B

Matrix of Zeros:

* Syntax : zeros array
* Format : zeros(N), zeros(M,N)
* Description:

This function is used to produce an array of zeros, defined by the arguments.

(N) is an N-by-N matrix of array.

(M,N) is an M-by-N matrix of array.

* Example;

>> zeros(2) >> zeros(1,2)

ans = ans =

0 0 0 0

0 0

Matrix of Ones:

* Syntax : ones array
* Format : ones(N), ones(M,N)
* Description:

This function is used to produce an array of ones, defined by the arguments.

(N) is an N-by-N matrix of array.

(M,N) is an M-by-N matrix of array.

* Example;

>> ones(2) >> ones(1,2)

ans = ans =

1 1 1 1

1 1

Identity Matrix:

* Syntax : identity matrix
* Format : eye (N), eye (M,N)
* Description:

Create an NxN or MxN identity matrix (i.e., 1’s on the diagonal elements with all others equal to zero). (Usually the identity matrix is represented by the letter “I”. Type

* Example;

>> I=eye(3)

I =

1 0 0

0 1 0

0 0 1

**Lab # 02: Introduction to MATLAB II**

**M-FILES, FLOW CONTROL AND PLOTTING:**

**Objective:** In this Matlab lab, you will learn:

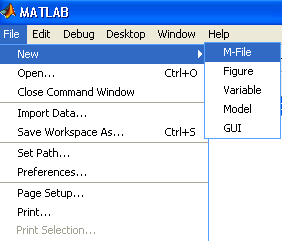
* How to create functions and script files.
* Input and output function
* Control flow statements
* Two dimensional plot
* Plot formatting
* Displaying multiple plots

**Tools Used:** **MATLAB 7.9.0**

**Description:**

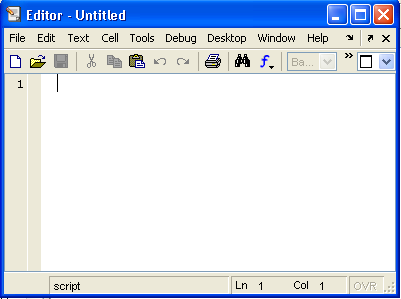
M-files:

* MATLAB can execute a sequence of MATLAB statements stored on disk. Such files are called "M-files"
* They must have the file type of ".m"
* To make the m-file click on **File** next select **New** and click on **M-File** from the pull-down menu as shown in **“Fig 2.1”**



**“Fig 2.1”**

* You will be presented with the **MATLAB Editor/Debugger** screen as shown in **“Fig 2.2”**



**“Fig 2.2”**

* Here you will type your code, can make changes, etc.
* Once you are done with typing, click on **File**, in the **MATLAB Editor/Debugger** screen and select **Save As…**
* Chose a name for your file, e.g., **firstgraph.m**
* Click on **Save**.
* Make sure that your file is saved in the directory that is in MATLAB's search path.
* There are two types of M-files:
  + Script files
  + Function files. (you don’t need to make it)

**Script Files:**

* Script files do not take the input arguments or return the output arguments
* Example (plotxy.m)

x = [1 2 3; 4 5 6]

y = [2 3 4; 6 7 8]

plot(x,y)

**Input Function:**

* Syntax : Prompt for user input
* Format : input(‘String to display’)
* Description:

Input function is used to get data from user and prompts until data has been entered.

* Example;

>> marks = input (‘Enter total marks = ’)

Enter total marks = 5

marks =

5

**Disp Function:**

* Syntax : Display array
* Format : disp(variable\_name), disp(‘String to display’)
* Description:

Displays the array, without printing the array name. In all other ways it's the same as leaving the semicolon off an expression except that empty arrays don't display.

* Example;

>> disp(‘ hello ’)

hello

>> a=[1 3 5];

>> disp(a)

1 3 5

**Control Flow Statements:**

**Examples:**

if (x < 10)

disp(x); % only displays when x < 10

end

if ((0 < x) & (x < 100))

    disp('OK');

end

**If block**

if (<condition>)

**<body>**

end

**Example:**

if ((0 < x) & (x < 100))

    disp('OK');

else

    disp('x contains invalid number');

end

**If block**

if (<condition>)

<**body**>

else

<**body**>

end

**Example:**

if (n <= 0)

disp('n is negative or zero');

elseif (rem(n,2)==0)

    disp('n is even');

else

disp('n is odd');

end

**If block**

if (<condition>)

<**body**>

else if

<**body**>

else

<**body**>

end

**for block**

for i = 1:1:10

<**body**>

end

**Example:**

a = zeros(k,k) % Preallocate matrix

for m = 1:k

for n = 1:k

a(m,n) = 1/(m+n -1);

end

end

**while block**

while (<condition>)

<**body**>

end

**Example**

i=1;

while (i<=10)

    disp(i);

    i = i + 1;

end

**switch statement**

switch<expression>

case <condition>,

<statement>

otherwise <condition>, <statement>

end

**Example:**

method = 'bilinear';

switch lower(method)

case {'linear','bilinear'}

disp('Method is linear')

case 'cubic'

disp('Method is cubic')

otherwise

disp('Unknown method.')

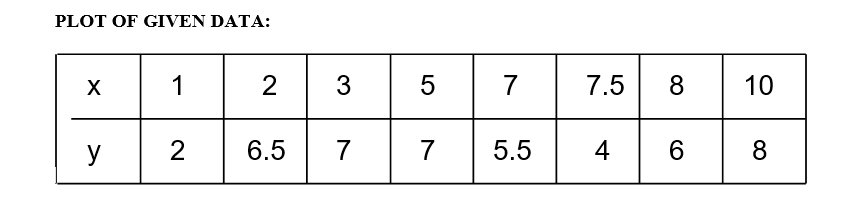
end

Method is linear

**PLOTTING**

**TWO-DIMENSIONAL plot() COMMAND**

* where **x** is a vector (one dimensional array), and **y** is a vector. Both vectors **must** have the same number of elements.
* The plot command creates a single curve with the **x** values on the abscissa (horizontal axis) and the **y** values on the ordinate (vertical axis).
* The curve is made from segments of lines that connect the points that are defined by the **x** and **y** coordinates of the elements in the two vectors.



**plot (x,y)**

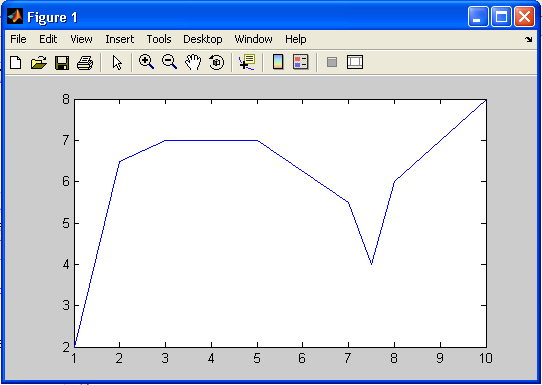
* A plot can be created by the commands shown below. This can be done in the Command Window, or by writing and then running a script file.

>> x=[1 2 3 5 7 7.5 8 10];

>> y=[2 6.5 7 7 5.5 4 6 8];

>> plot(x,y)

* Once the plot command is executed, the Figure Window opens with the following plot as shown in **“Fig 2.3”**



**“Fig 2.3”**

**LINE SPECIFIERS IN THE plot() COMMAND:**

Line specifiers can be added in the **plot** command to:

* Specify the style of the line.
* Specify the color of the line.
* Specify the type of the markers (if markers are desired).

**plot(x,y,’line specifiers’)**

The following tables lists the line specifiers:

**Line Style Specifiers:**

|  |  |
| --- | --- |
| **Specifier** | **Line Style** |
| **-** | Solid line (default) |
| **--** | Dashed line |
| **:** | Dotted line |
| **-.** | Dash dot line |

**Marker Specifier:**

|  |  |
| --- | --- |
| **Specifier** | **Marker Type** |
| **+** | plus sign |
| **O** | Circle |
| **\*** | Asterisk |
| **.** | dot |
| **s** | square |
| **d** | diamond |

**Color Specifiers:**

|  |  |
| --- | --- |
| **Specifier** | **Line Color** |
| **r** | red |
| **g** | green |
| **b** | blue |
| **c** | cyan |
| **m** | magenta |
| **y** | yellow |
| **k** | black |

* The specifiers are typed inside the **plot()** command as strings.
* Within the string the specifiers can be typed in any order.
* The specifiers are optional. This means that none, one, two, or all the three can be included in a command.

**Example:**

* plot(x,y) A solid blue line connects the points with no markers.
* plot(x,y,’r’) A solid red line connects the points with no markers.
* plot(x,y,’--y’) A yellow dashed line connects the points.
* plot(x,y,’\*’) The points are marked with \* (no line between the points.)
* plot(x,y,’g:d’) A green dotted line connects the points which are marked with

diamond markers.

**Plot of given data using the line specifiers in the plot ( ) command**

Year

Sales (M)

1988

1989

1990

1991

1992

1993

1994

127

130

136

145

158

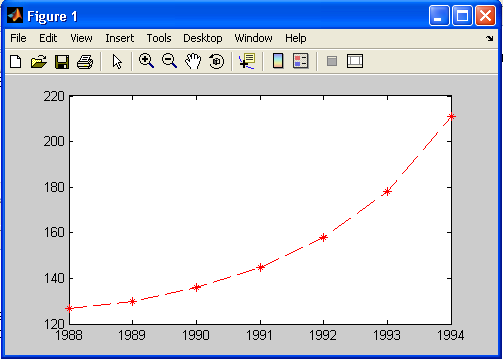
178

211

**>>**year = [1988:1:1994];

>> sales = [127, 130, 136, 145, 158, 178, 211];

>> plot(year,sales,'--r\*')



**“Fig 2.4”**

Formatting the Plots:

A plot can be formatted to have a required appearance.

With formatting you can:

* Add title to the plot.
* Add labels to axes.
* Change range of the axes.
* Add legend.
* Add text blocks.
* Add grid.

**Formatting Commands:**

**title(‘string’)**

Adds the string as a title at the top of the plot.

**xlabel(‘string’)**

Adds the string as a label to the *x*-axis.

**ylabel(‘string’)**

Adds the string as a label to the *y*-axis.

**axis([xmin xmax ymin ymax])**

Sets the minimum and maximum limits of the *x*- and y-axes.

**legend(‘string1’,’string2’,’string3’)**

Creates a legend using the strings to label various curves (when several curves are

in one plot). The location of the legend is specified by the mouse.

**text(x,y,’string’)**

Places the string (text) on the plot at coordinate x,y relative to the plot axes.

**Example of Formatted Plot:**

x=[0:0.1:2\*pi];

y=sin(x);

z=cos(x);

plot(x,y,x,z)

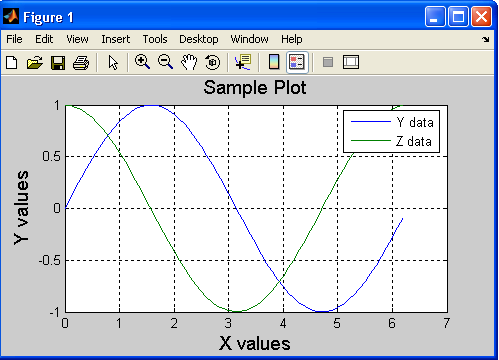
title('Sample Plot','fontsize',14);

xlabel('X values','fontsize',14);

ylabel('Y values','fontsize',14);

legend('Y data','Z data')

grid on



“Fig 2.5”

Displaying the Multiple Plots:

* Three typical ways to display multiple curves in MATLAB (other combinations are possible…)
  + One figure contains one plot that contains multiple curves
    - Requires the use of the command “hold” (see MATLAB help)
  + One figure contains multiple plots, each plot containing one curve
    - Requires the use of the command “subplot”
  + Multiple figures, each containing one or more plots, each containing one or more curves
    - Requires the use of the command “figure” and possibly “subplot”

**One Plot Contains Multiple Curves:**

**Example:**

x=[0:0.1:2\*pi];

y=sin(x);

z=cos(x);

plot(x,y,x,z)

grid on

**Or**

x=[0:0.1:2\*pi];

y=sin(x);

z=cos(x);

plot(x,y,’b’)

hold on

Plot(x,z,’g’)

hold off

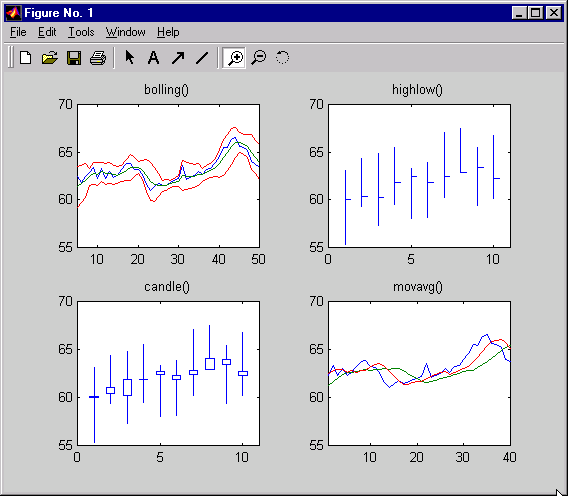
grid on

* The plot figure produced by above code is shown in **“Fig 2.5”.**

**Subplots:**

* Subplot divides the current figure into rectangular panes that are numbered row wise.
* **Syntax:**

**subplot(rows,cols,index)**



* **subplot(2,2,1);**
* **…**
* **subplot(2,2,2)**
* **...**
* **subplot(2,2,3)**
* **...**
* **subplot(2,2,4)**
* **...**

**“Fig 2.6”**

**Example of Subplot:**

x=[0:0.1:2\*pi];

y=sin(x);

z=cos(x);

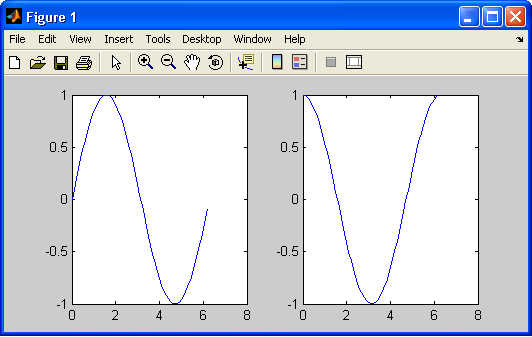
subplot(1,2,1);

plot(x,y)

subplot(1,2,2)

plot(x,z)

grid on

****

**“Fig 2.7”**

**Multiple Figures:**

**Example:**

x=[0:0.1:2\*pi];

y=sin(x);

z=cos(x);

figure;

plot(x,y)

figure;

plot(x,z)

grid on