Computer since

First Stage

Lec8

Plotting In Matlab

MS.c Mortada Sabri

MS.c Lubna Ali

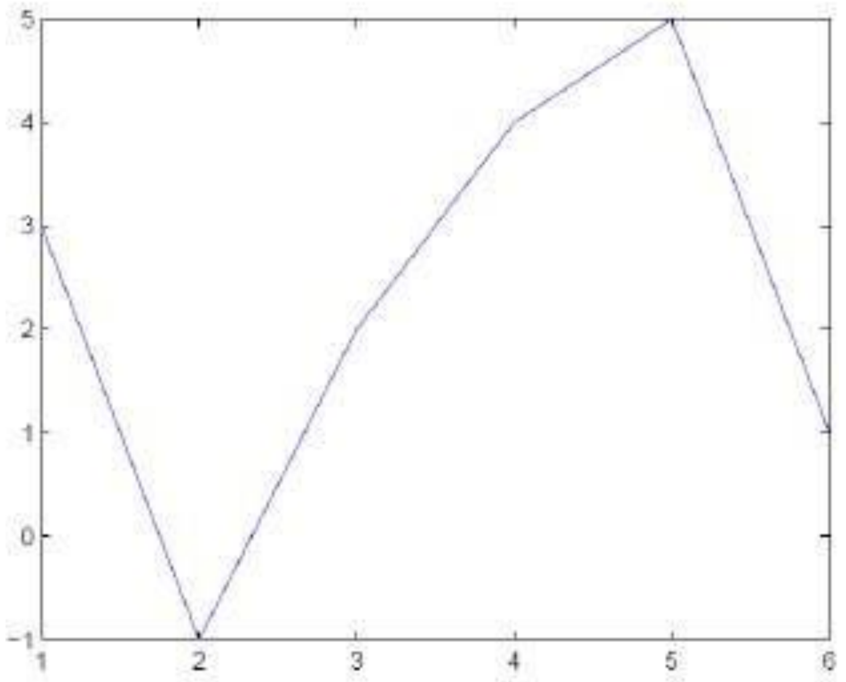
Introduction

- MATLAB has an excellent set of graphic tools. Plotting a given data set or the results of computation is possible with very few commands. You are highly encouraged to plot mathematical functions and results of analysis as often as possible. Trying to understand mathematical equations with graphics is an enjoyable and very efficient way of learning mathematics. Being able to plot mathematical functions and data freely is the most important step, and this section is written to assist you to do just that. Engineers use graphing techniques to make the information easier to understand. With a graph, it is easy to identify trends, pick out highs and lows, and isolate data points that may be measurement or calculation errors. Graphs can also be used as a quick check to determine whether a computer solution is yielding expected results.

➢ Basic Plotting

The basic MATLAB graphing procedure is to take a vector of *x*-coordinates, x = ($x_1; : : : ; x_N$), and a vector of *y*-coordinates, y = ($y_1; : : : ; y_N$), locate the points ($xi; yi$), with $i = 1; 2; : : : ; n$ and then join them by straight lines. You need to prepare *x* and *y* in an identical array form; namely, *x* and *y* are both row arrays or column arrays of the *same* length.
The MATLAB command to plot a graph is plot(x,y). The vectors x = (1; 2; 3; 4; 5; 6) and y = (3;¡1; 2; 4; 5; 1) produce the picture shown in below.

```
>> x = [1 2 3 4 5 6];
>> y = [3 -1 2 4 5 1];
>> plot (x,y)
```



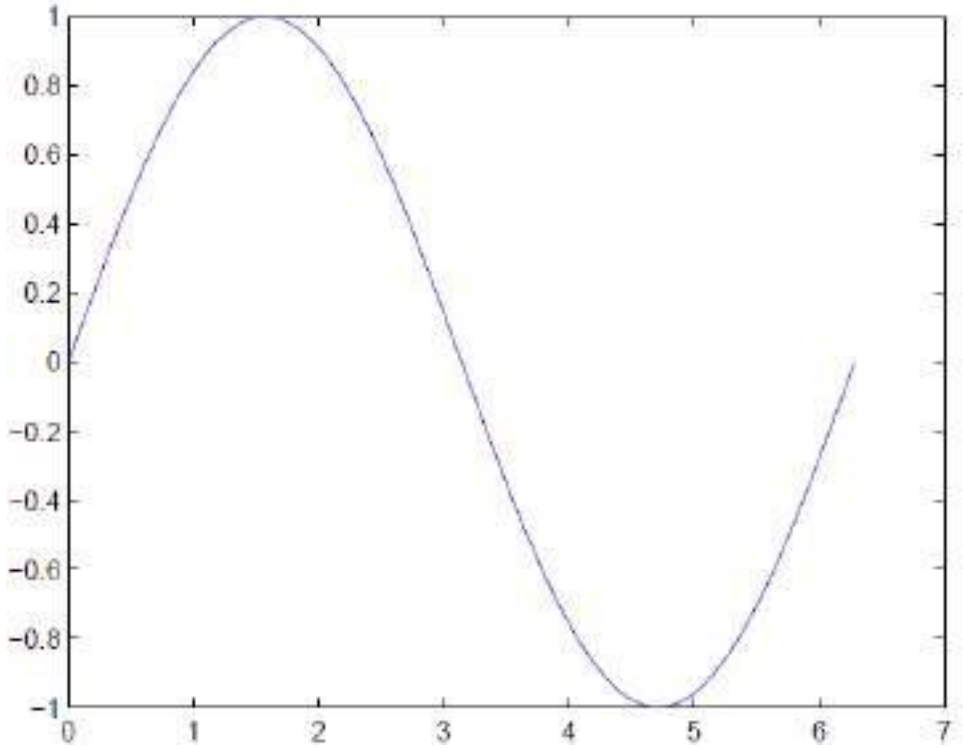
Note: The plot functions has different forms depending on the input arguments. If y is a vector plot (y) produces a piecewise linear graph of the elements of y versus the index of the elements of y. If we specify two vectors, as mentioned above, plot (x,y) produces a graph of y versus x.

For example, to plot the function sin (*x*) on the interval [0, 2π], we first create a vector of *x* values ranging from 0 to 2π, then compute the *sine* of these values, and finally plot the result:

```
>> x = 0:pi/100:2*pi;
>> y = sin(x);
>> plot(x,y)
```

Notes:

- 0:pi/100:2*pi yields a vector that
- starts at 0,
- takes steps (or increments) of *π*/100,
- stops when 2*π* is reached.
- If you omit the increment, MATLAB automatically increments by 1.



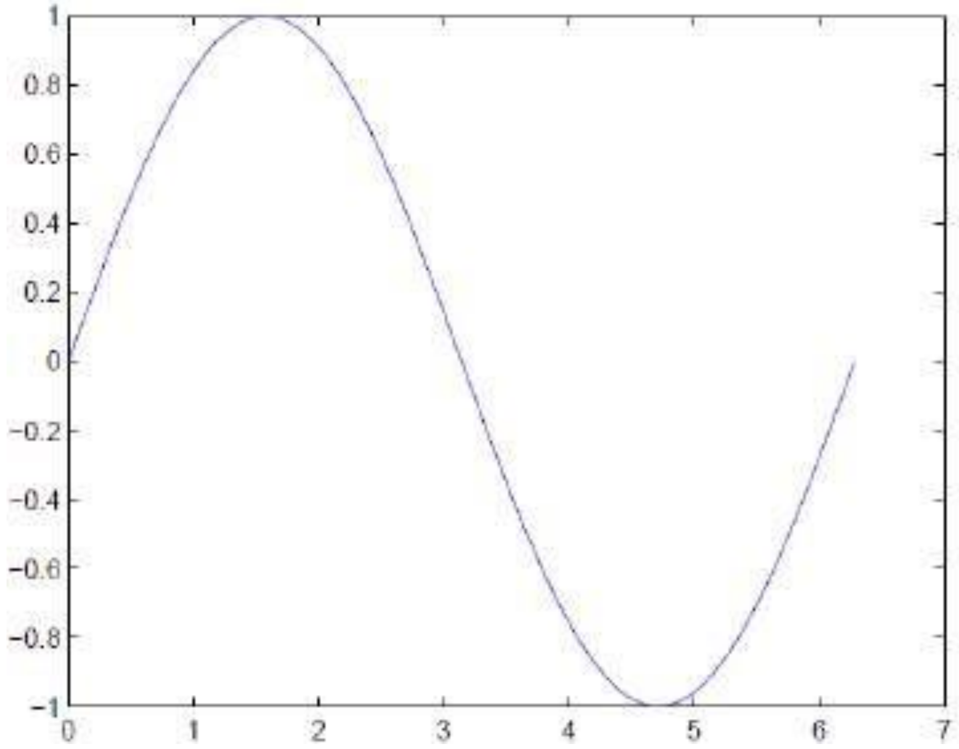
Note: The plot functions has different forms depending on the input arguments. If y is a vector plot (y) produces a piecewise linear graph of the elements of y versus the index of the elements of y. If we specify two vectors, as mentioned above, plot (x,y) produces a graph of y versus x.

For example, to plot the function sin (*x*) on the interval [0, 2π], we first create a vector of *x* values ranging from 0 to 2π, then compute the *sine* of these values, and finally plot the result:

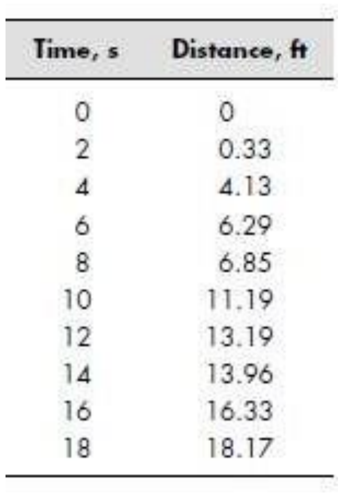
```
>> x = 0:pi/100:2*pi;
>> y = sin(x);
>> plot(x,y)
```

Notes:

- 0:pi/100:2*pi yields a vector that
- starts at 0,
- takes steps (or increments) of $\pi/100$,
- stops when 2π is reached.
- If you omit the increment, MATLAB automatically increments by 1.

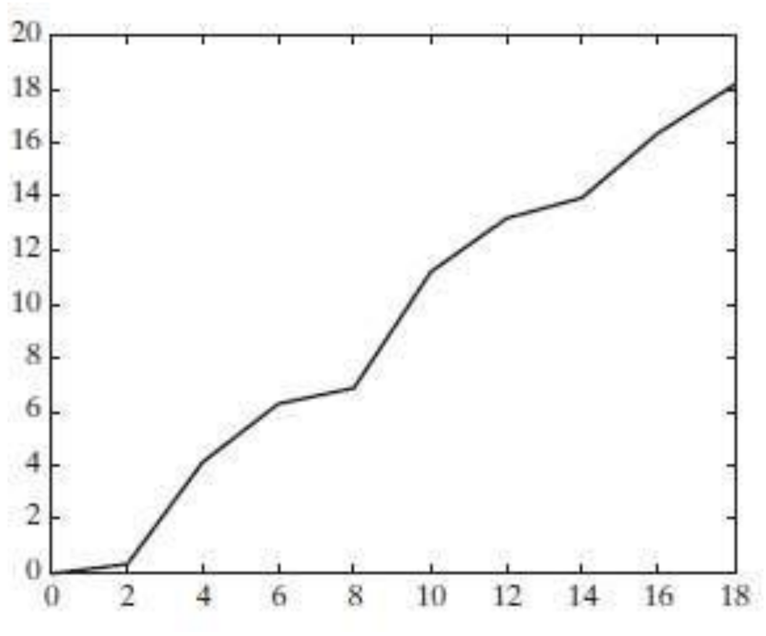


Suppose a set of time versus distance data were obtained through measurement. We can store the time values in a vector called **x** (the user can define any convenient name) and the distance values in a vector called **y** :



Time, s	Distance, ft
0	0
2	0.33
4	4.13
6	6.29
8	6.85
10	11.19
12	13.19
14	13.96
16	16.33
18	18.17

```
x = [0:2:18];
y = [0, 0.33, 4.13, 6.29, 6.85, 11.19, 13.19, 13.96, 16.33,18.17];
plot(x,y)
```



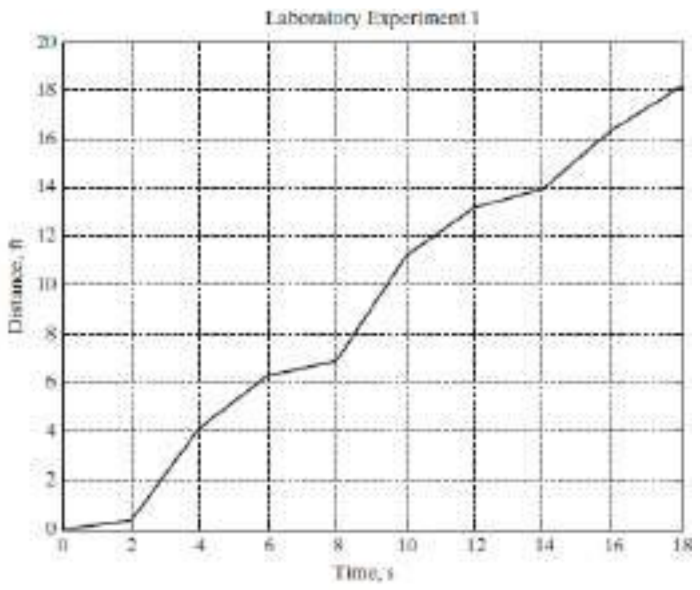
➢ Adding titles, axis labels, and annotations

Good engineering practice requires that we include axis labels and a title in our plot. The following commands add a title, *x* - and *y* -axis labels, and a background grid:

```
plot(
x,y)
xlab
el('Ti
me,
sec')
ylab
el('D
istan
ce,
ft')
grid
on
```

These commands generate the plot in Figure below. As with any MATLAB commands, they could also be combined onto one or two lines, separated by commas:

```
plot(x,y) , title('Laboratory
Experiment 1') xlabel('Time,
sec' ), ylabel('Distance, ft'),
grid
```



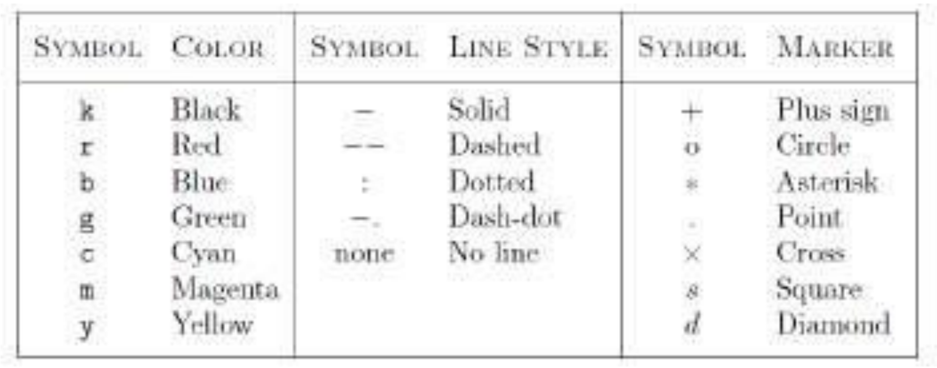
➢ -Specifying line styles and colors

By default, MATLAB uses *line style* and *color* to distinguish the data sets plotted in the graph. However, you can change the appearance of these graphic components or add annotations to the graph to help explain your data for presentation.

It is possible to specify *line styles*, *colors*, and *markers* (e.g., circles, plus signs, . . .) using the

plot command: Plot (x,y,'style_color_marker')

where style_color_marker is a *triplet* of values from Table below:



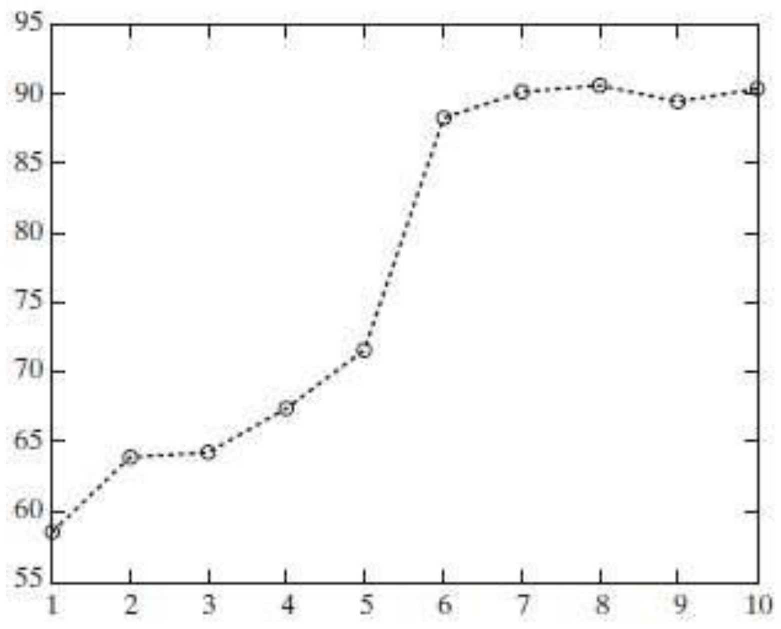
SYMBOL	COLOR	SYMBOL	LINE STYLE	SYMBOL	MARKER
k	Black	–	Solid	+	Plus sign
r	Red	– –	Dashed	o	Circle
b	Blue	:	Dotted	*	Asterisk
g	Green	–.	Dash-dot	.	Point
c	Cyan	none	No line	×	Cross
m	Magenta			s	Square
y	Yellow			d	Diamond

The following commands illustrate the use of line, color,

and mark styles: x = [1:10];

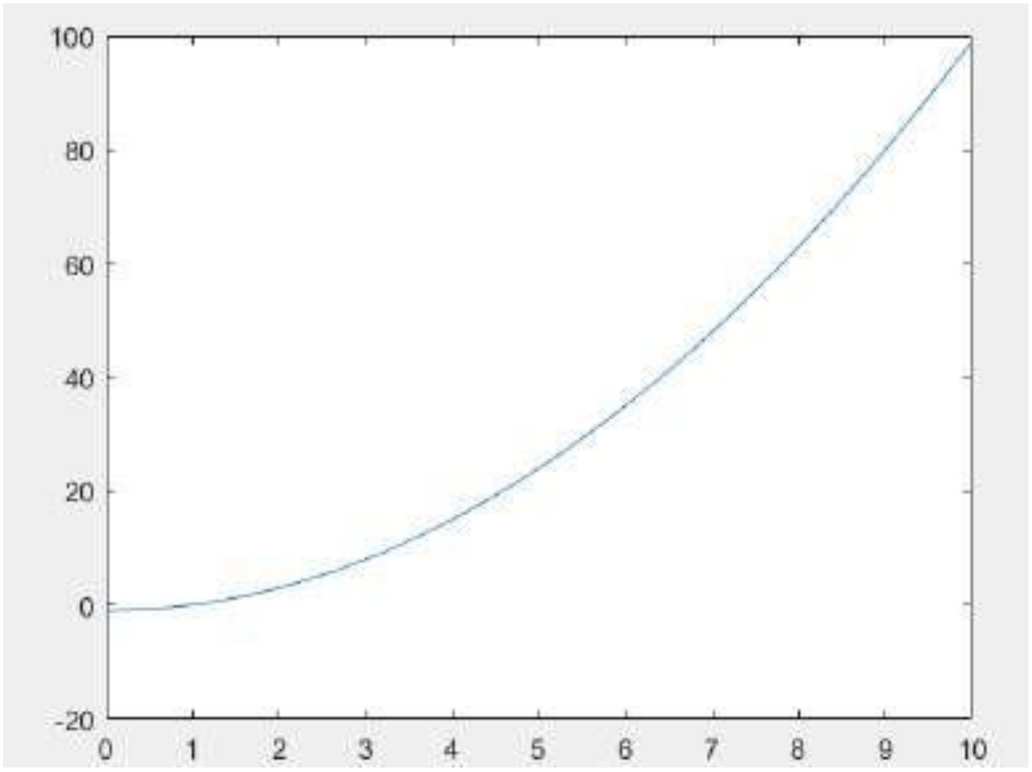
y = [58.5, 63.8, 64.2, 67.3, 71.5, 88.3, 90.1, 90.6, 89.5,90.4];

plot(x,y,':ok')



Plot the polynomial $y = x^2 - 1$ between x=0 and x=10 (using twenty points).

```
>>x=linspace(0,10,20); // or x=0:10/20:10;
>>y= x.^2-1;
>>plot(x,y)
```


100
80
60
40
20
0
-20
0
1
2
3
4
5
6
7
8
9
10