Plotting & Sub-plotting

Two-Dimensional Plots

❖Plots are a very useful tool for presenting information in science and engineering, where MATLAB is mostly used.

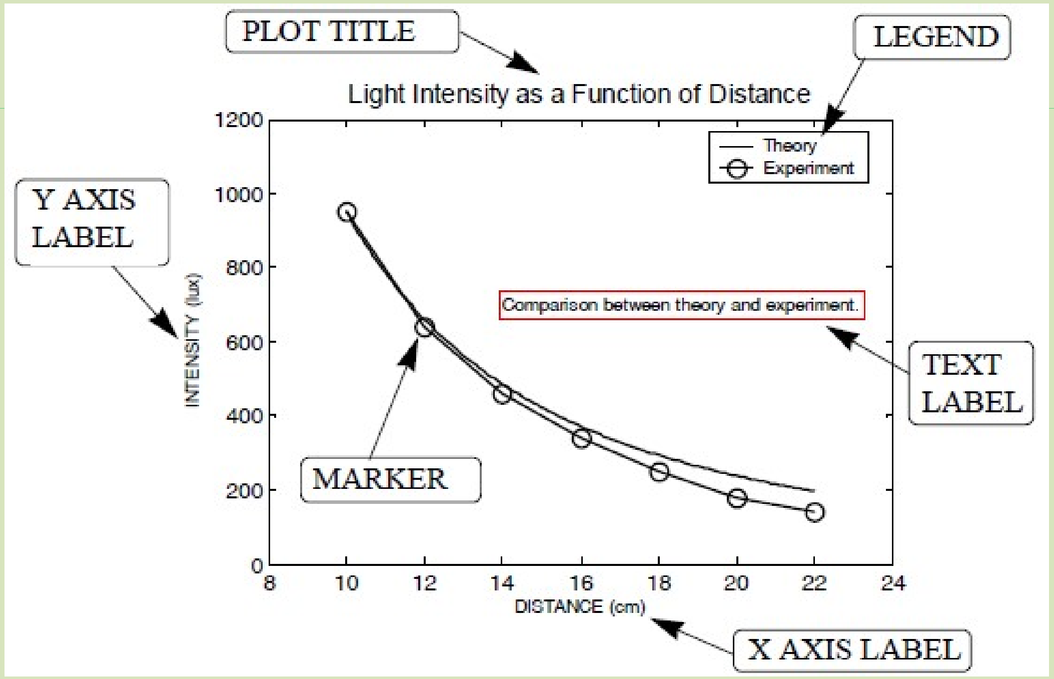
❑ different types of plots include :

- ❖standard plots with linear axes,
- ❖plots with logarithmic and semi-logarithmic axes,
- ❖ bar and stairs plots,
- ❖polar plots,
- ❖ three-dimensional contour surface and mesh plots,
- ❖and many more. The plots can be formatted to have a desired appearance.

Two-Dimensional Plots

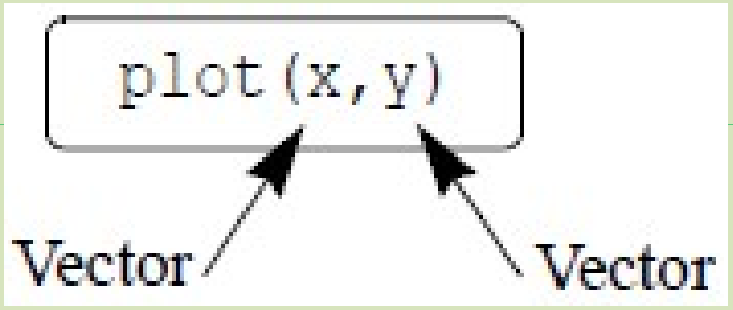
- ❑The line type (solid, dashed, etc.), color, and thickness can be prescribed,
- ❑ line markers, grid lines , titles and text comments can be added.
- ❑Several graphs can be created in the same plot, and several plots can be placed on the same page.
- ❑ When a plot contains several graphs and/or data points, a legend can be added to the plot as well.

- **This chapter describes how MATLAB can be used to create and format many types of two-dimensional plots.**
- a simple two-dimensional plot that was created with MATLAB is shown in Figure below.



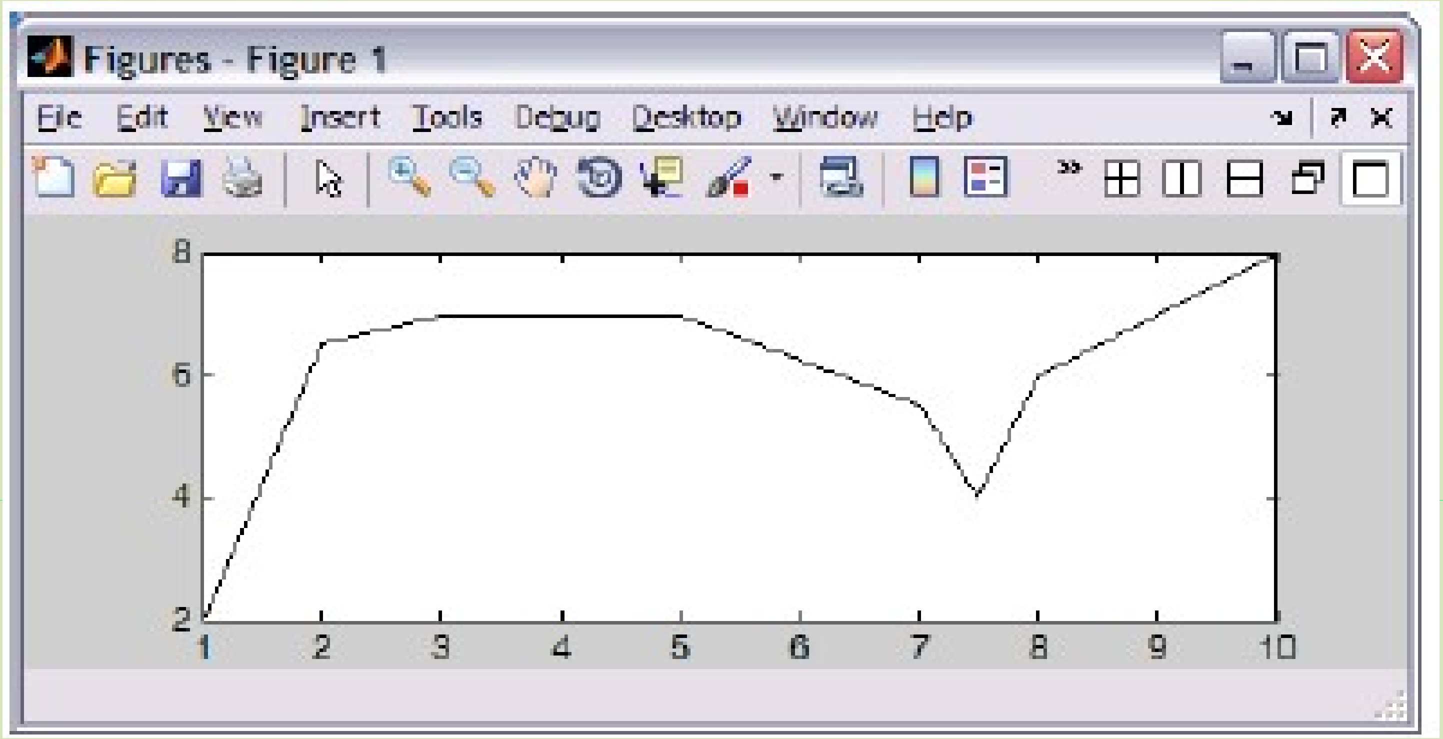
The plot Command

The plot command is used to create two-dimensional plots. The simplest form of the command is:



Example:

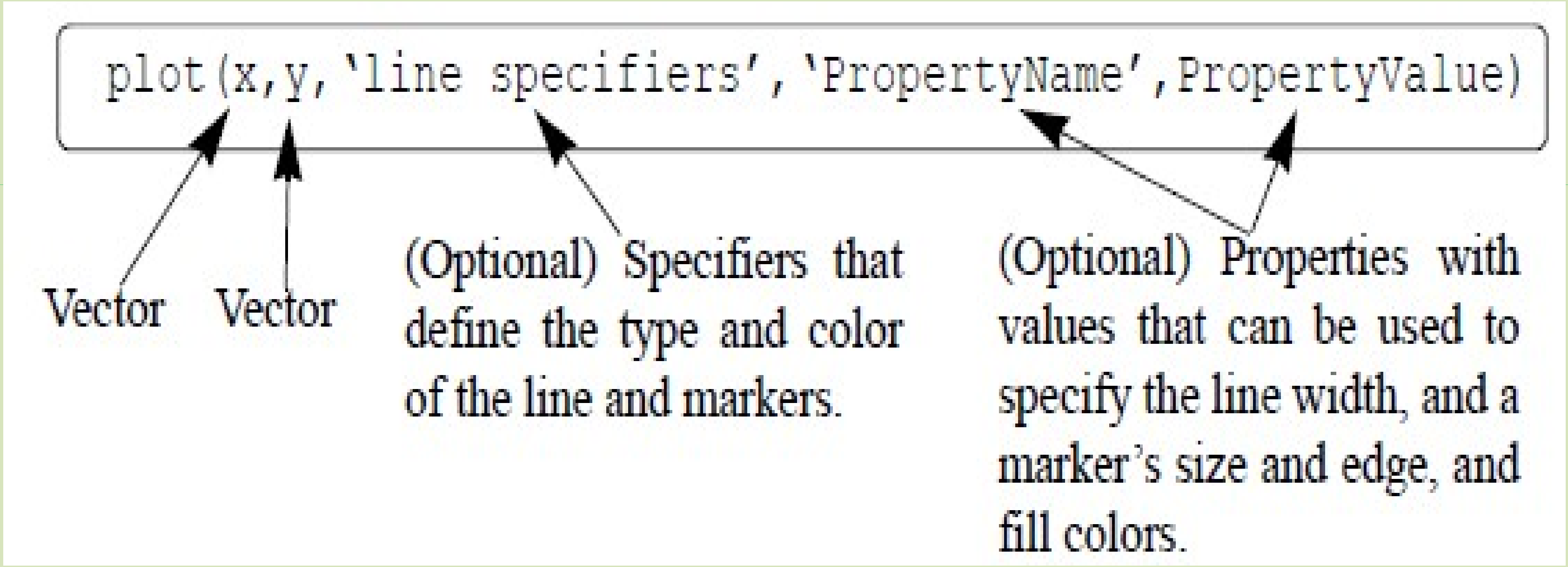
```
>> x=[1  2  3  5  7  7.5  8  10];
>> y=[2  6.5  7  7  5.5  4  6  8];
>> plot(x,y)
```



The plot command has additional, optional arguments that can be used to Specify **the color and style of the line** & **the color and type of markers**, if any are desired. With these options the command has the form:

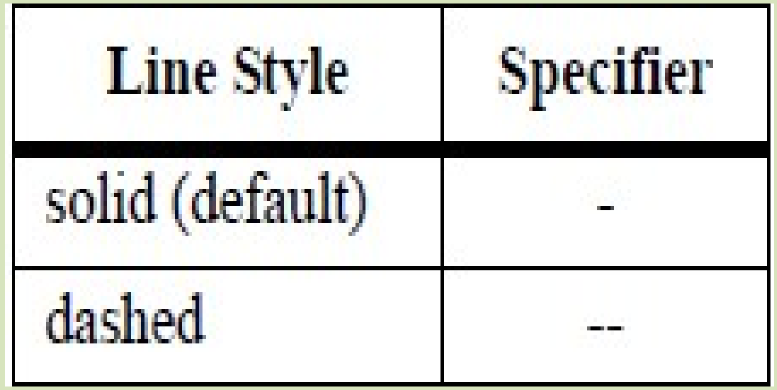
The General Form Of Plot Command

With these options the command has the form:

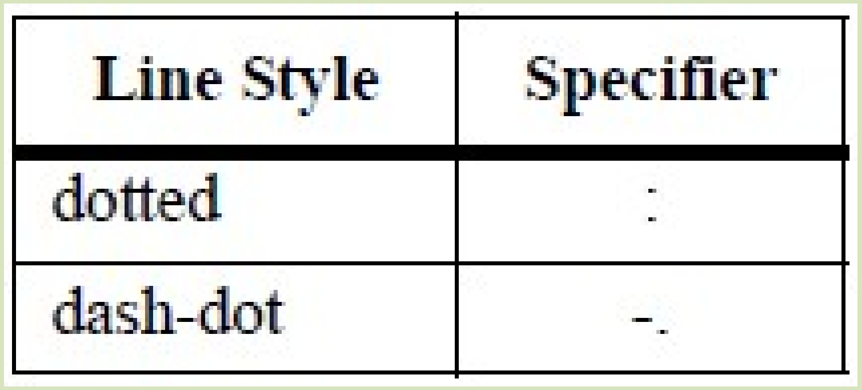

plot(x,y,'line specifiers','PropertyName',PropertyValue)
Vector
Vector
(Optional) Specifiers that define the type and color of the line and markers.
(Optional) Properties with values that can be used to specify the line width, and a marker's size and edge, and fill colors.

Line Specifiers:

- **Line specifiers** are optional and can be used to define the style and color of the line and the type of markers (if markers are desired).
- **The line style specifiers** are:

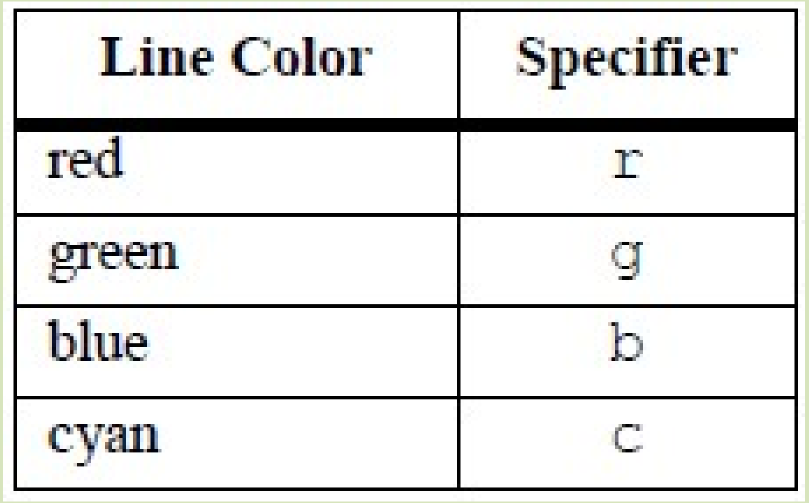


Line Style	Specifier
solid (default)	-
dashed	--

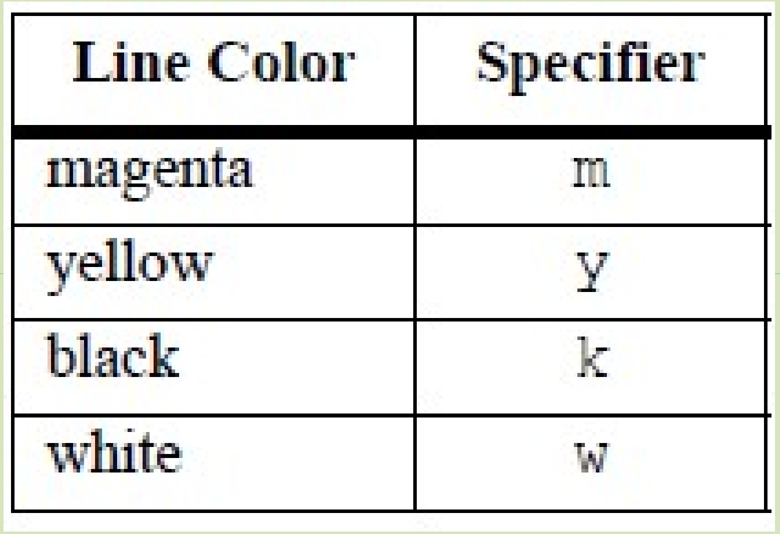


Line Style	Specifier
dotted	:
dash-dot	-.

The line color specifies are:

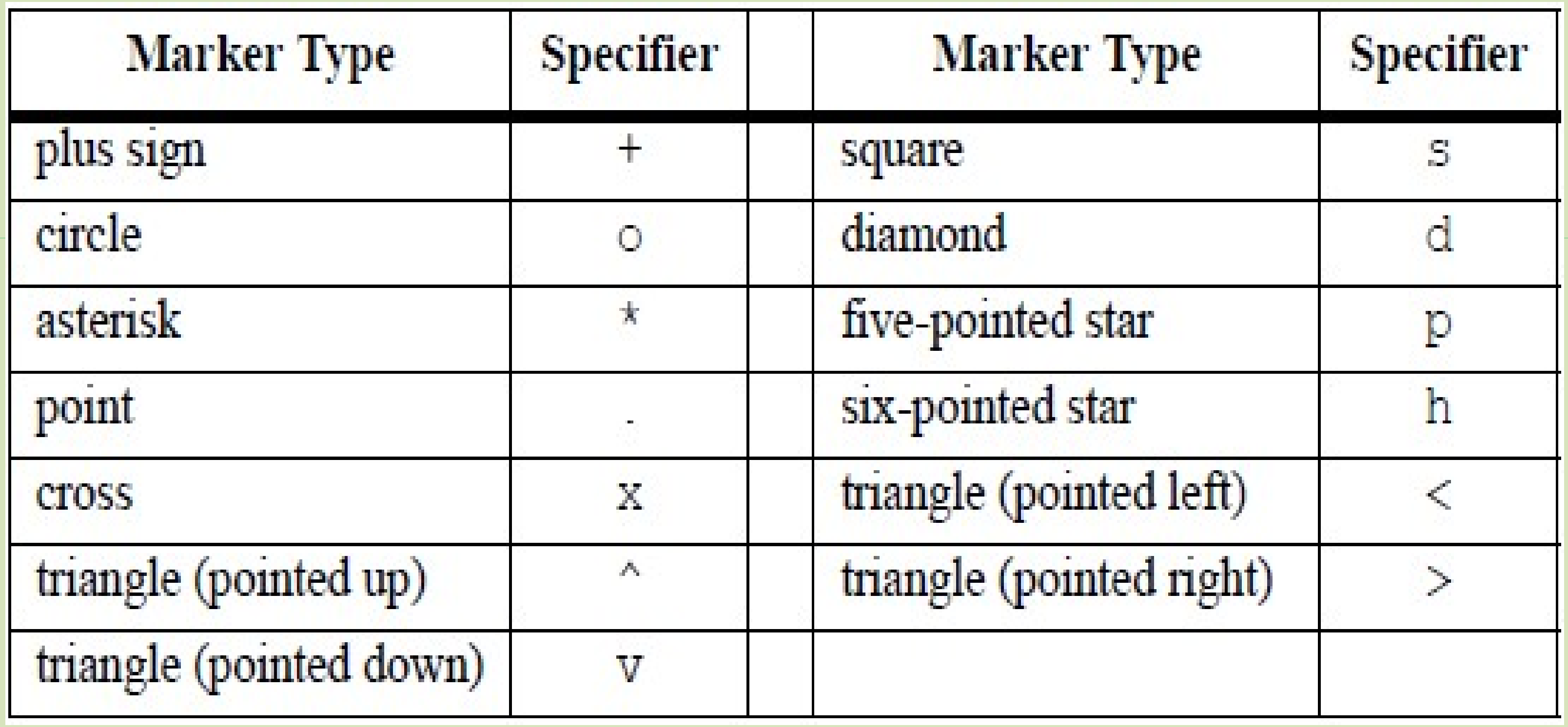


Line Color	Specifier
red	r
green	g
blue	b
cyan	c



Line Color	Specifier
magenta	m
yellow	y
black	k
white	w

The marker type specifies are:



Marker Type	Specifier		Marker Type	Specifier
plus sign	+		square	s
circle	o		diamond	d
asterisk	*		five-pointed star	p
point	.		six-pointed star	h
cross	x		triangle (pointed left)	<
triangle (pointed up)	^		triangle (pointed right)	>
triangle (pointed down)	v			

Notes On the specifiers:

- 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 three types can be included in a command.

Some examples:

plot(x , y)	A blue solid line connects the points with no markers (default).
plot(x, y, ‘r ’)	A red solid line connects the points.
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 that are marked with diamond markers.

Property Name and Property Value:

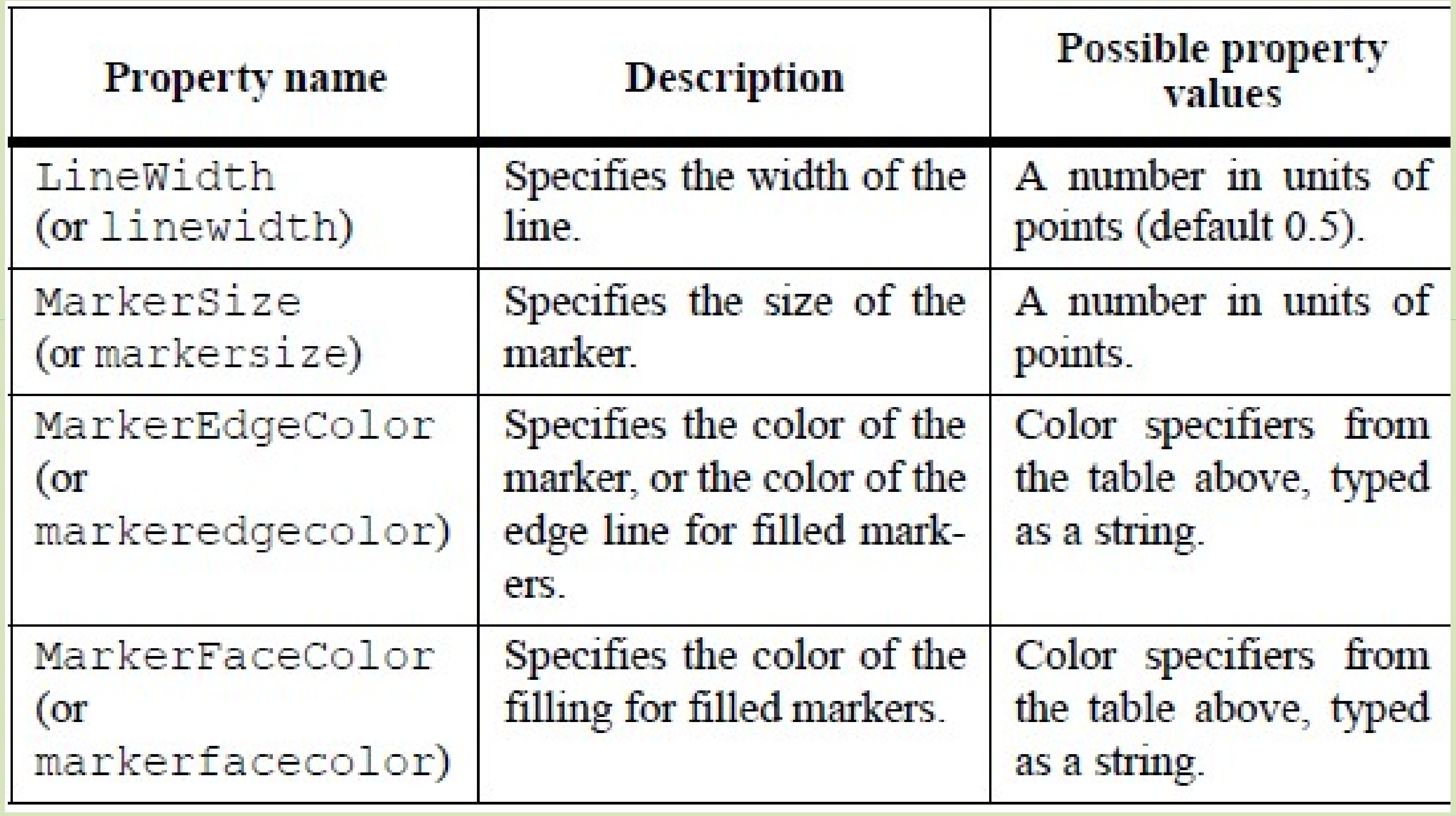
❖Properties are optional and can be used to specify

- the thickness of the line,
- the size of the marker, and
- the colors of the marker's edge line and fill.

❖ **The Property Name**

is typed as a string, followed by **a comma** and a value for the property, all inside the plot command.

❖Four properties and their possible values are:



Property name	Description	Possible property values
`LineWidth` (or `linewidth`)	Specifies the width of the line.	A number in units of points (default 0.5).
`MarkerSize` (or `markersize`)	Specifies the size of the marker.	A number in units of points.
`MarkerEdgeColor` (or `markeredgecolor`)	Specifies the color of the marker, or the color of the edge line for filled markers.	Color specifiers from the table above, typed as a string.
`MarkerFaceColor` (or `markerfacecolor`)	Specifies the color of the filling for filled markers.	Color specifiers from the table above, typed as a string.

For example , the command

```
plot(x , y ,'-mo ','LineWidth ' , 2,'markersize' ,12,
        'MarkerEdgeColor' , 'g',' markerfacecolor', 'y')
```

<u>creates a plot that</u>

- connects the points with a magenta solid line and circles as markers at the points.
- The line width is 2 points and
- size of the circle markers is 12 points.
- The markers have a green edge line and
- The markers have a yellow filling.

A note about line specifiers and properties:

The three line specifiers, which indicate

the style ,

color of the line, and

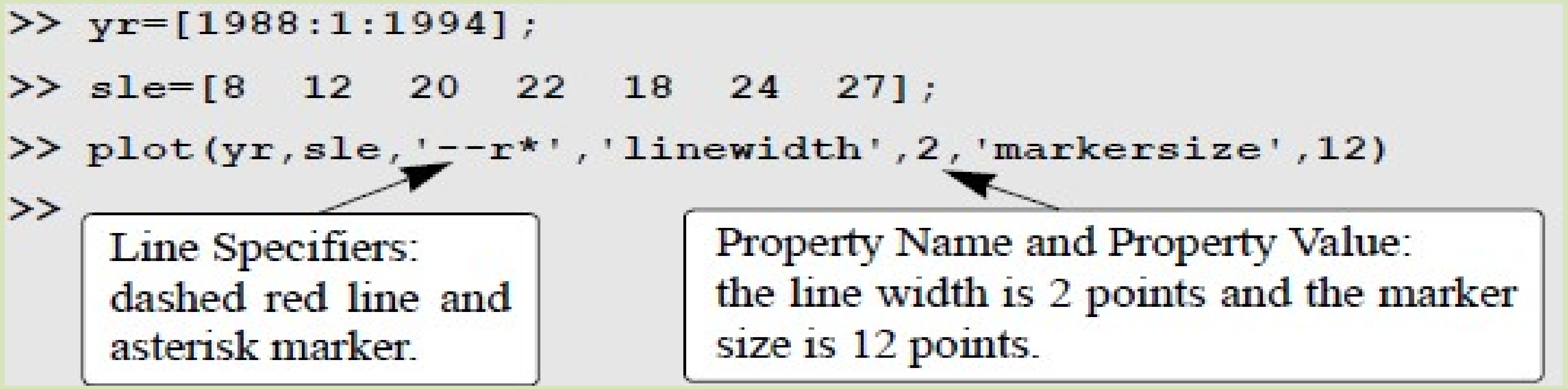
the type of the marker can also be assigned with a PropertyName argument followed by a PropertyValue argument. The Property Names for the line specifiers are:

Plot of Given Data

In this case given data is used to create vectors that are then used in the plot command. The following table contains sales data of a company from 1988 to 1994.

Year	1988	1989	1990	1991	1992	1993	1994
Sales(millions)	8	12	20	22	18	24	27

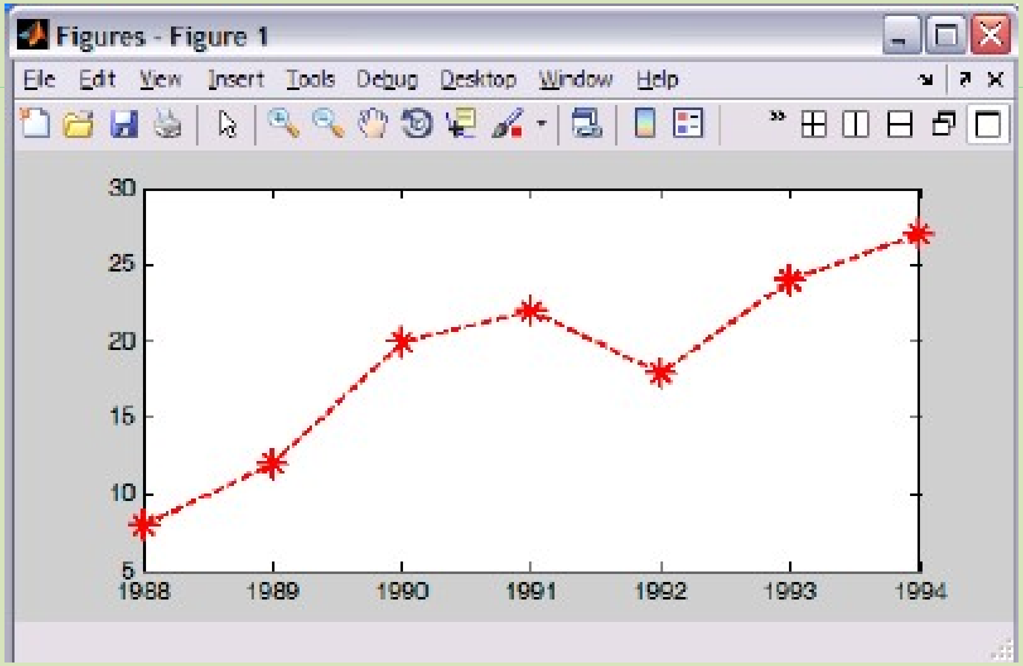
To plot this data, the list of years is assigned to one vector (named yr), and the corresponding sales data is assigned to a second vector (named sle). The Command Window where the vectors are created and the plot command is used is shown below:



```
>> yr=[1988:1:1994];
>> sle=[8  12  20  22  18  24  27];
>> plot(yr,sle,'--r*','linewidth',2,'markersize',12)
>>
```

Line Specifiers: dashed red line and asterisk marker.

Property Name and Property Value: the line width is 2 points and the marker size is 12 points.



Plot of a Function

to plot a function y=f(x) with the plot command, the user needs

- first create a vector of values of *x for the domain over which the function will* be plotted.

-Then a vector *y is created with the corresponding values of f(x) by* using element-by-element calculations .

- **Once the two vectors are defined, they can be used in the plot command**.

As an example, use the plot command is to plot the function :

*Y= 3.5^ (–0.5*x) * cos(6*x)   for –2 ≤ x ≤ 4*

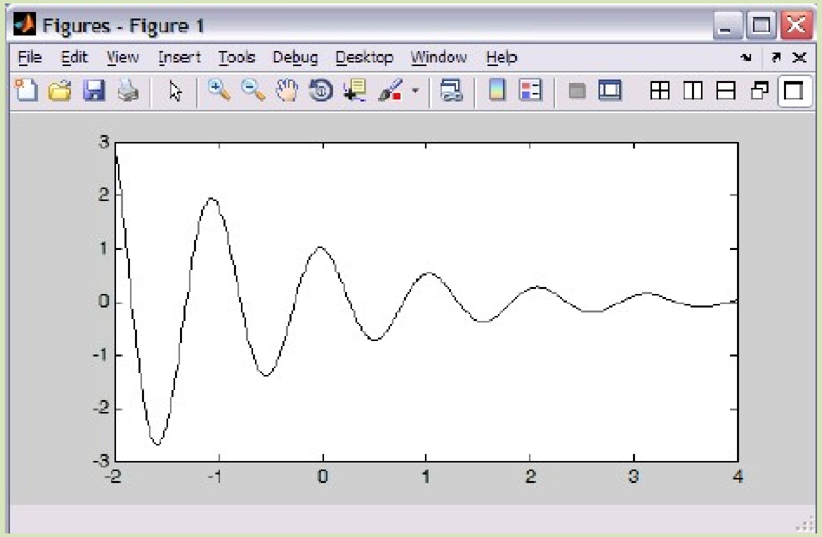
```
% the function: 3.5.^(-0.5*x).*cos(6x)
x=[-2:0.01:4];
y=3.5.^(-0.5*x).*cos(6*x);

plot(x,y)
```

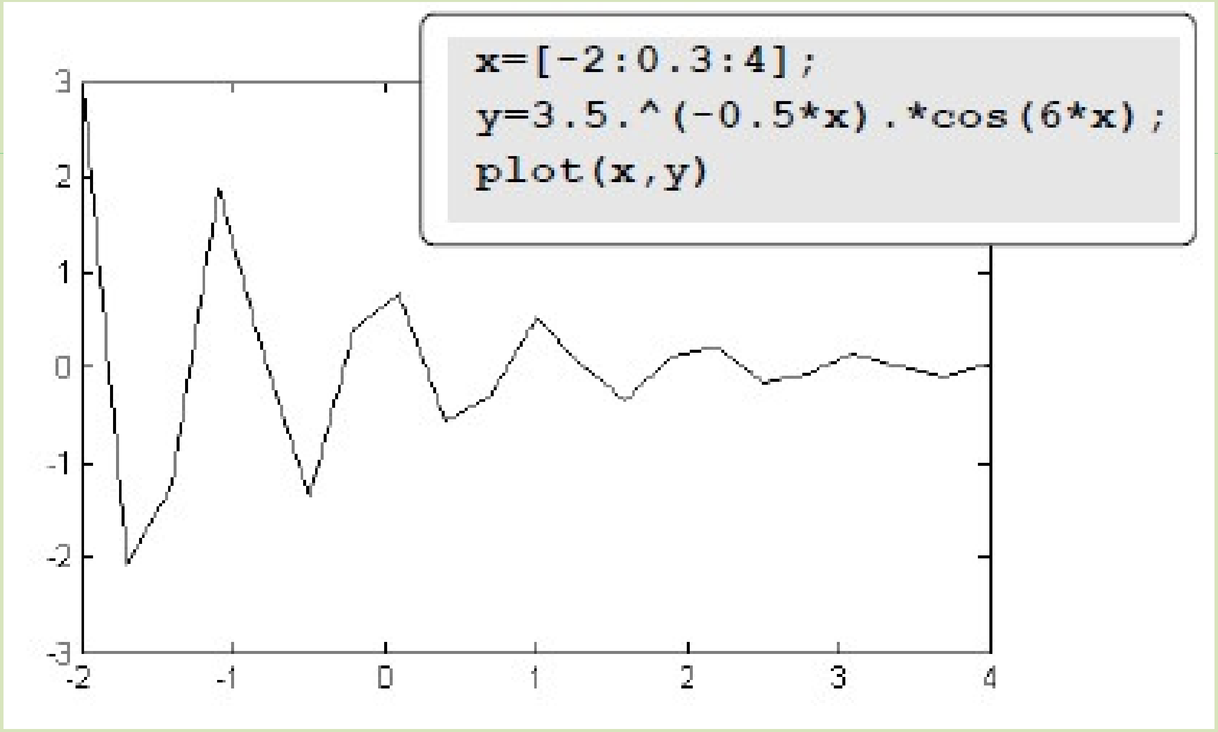
Create vector x with the domain of the function.

Create vector y with the function value at each x.

Plot y as a function of x.



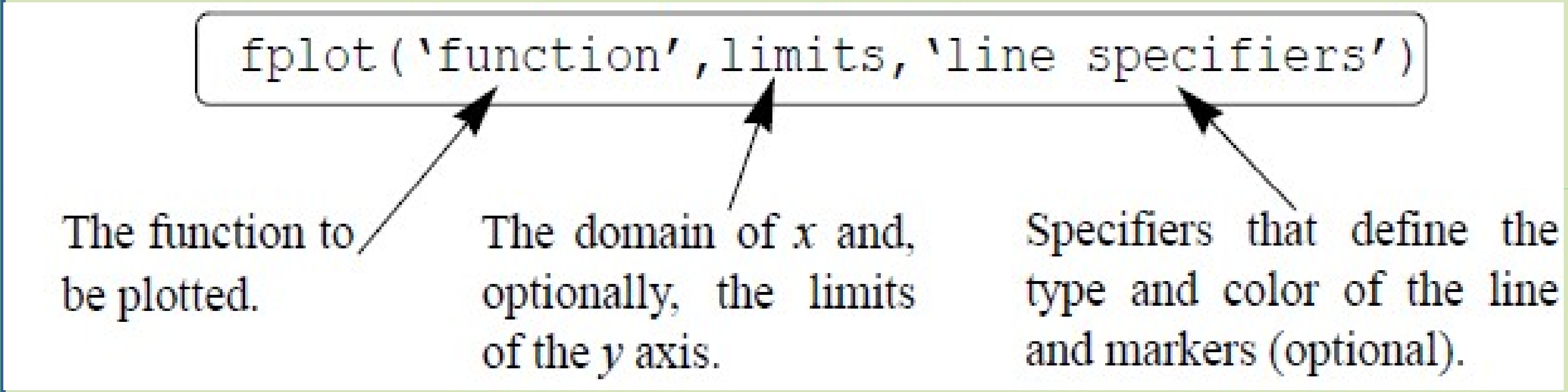
- a small spacing of 0.01 produced the plot that is shown in the above Figure However,
- if the same function in the same domain is plotted with much larger spacing—for example, 0.3—the plot that is obtained, shown in the following Figure , it gives a distorted picture of the function.



```
x=[-2:0.3:4];
y=3.5.^(-0.5*x).*cos(6*x);
plot(x,y)
```

The fplot Command

The ***fplot*** command plots a function with the form y=f(x) between specified limits. The command has the form:



```
fplot('function',limits,'line specifiers')
```

The function to be plotted.

The domain of x and, optionally, the limits of the y axis.

Specifiers that define the type and color of the line and markers (optional).

'function':

The function can be typed directly as a string inside the command.

-The function to be plotted can be typed as a function of any letter.
-The function cannot include previously defined variables.

limits:

-The limits argument is a vector with two elements that specify the domain of *x* ***[xmin ,xmax],*** *or a vector with four elements tha specifies the* domain of *x and the limits of the y-axis* ***[xmin,xmax,ymin,ymax].***

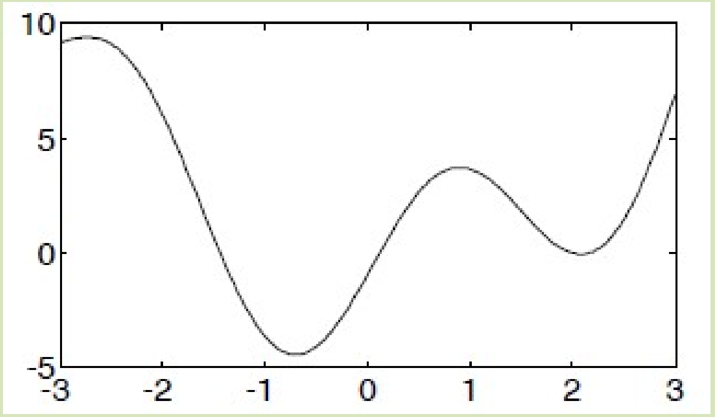
line specifiers:

The line specifiers are the same as in the plot command.

As an example:

*y = x^2 + 4*sin(2*x)-1 for – 1 –3 ≤ x ≤ 3*

```
>> fplot ('x^2+4*sin(2*x)-1',[-3 3])
```



PLOTTING MULTIPLE GRAPHS IN THE SAME PLOT

In many situations there is a need to make several graphs in the same plot. This is shown, for example, in Figure on slide 26 two graphs are plotted in the same figure.

There are three methods to plot multiple graphs in one figure.

1- One is by using the plot command,

2- the second is by using the hold on and hold off commands,

3-and the third is by using the line command.

1-Using the plot Command

Two or more graphs can be created in the same plot by typing pairs of vectors inside the plot command. The command

plot(x,y,u,v,t,h)

creates three graphs—**y vs. x**, **v vs. u**, and **h vs. t** —all in the same plot. The vectors of each pair must be of the same length

✓MATLAB automatically plots the graphs in different colors.

✓It is also possible to add line specifiers following each pair.

For example the command

```
plot(x,y, '-b' ,u,v, '--r',t,h, 'g:')
```

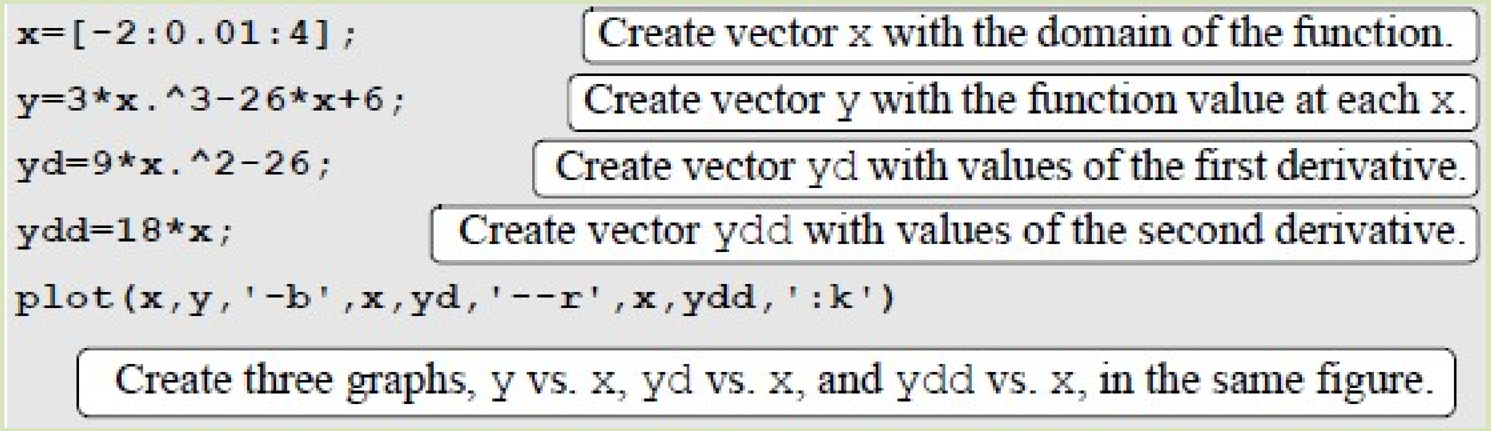
plots ***y vs. x*** with a solid blue line, ***v vs.u*** with a dashed red line, and ***h vs. t*** with a dotted green line.

Example:

Plot the function $y = 3x3 - 26x + 10$, and its first and second derivatives, for $-2 \leq x \leq 4$, all in the same plot.

The first derivative of the function is: $y' = 9x2 - 26.$

The second derivative of the function is: $y'' = 18x$



```
x=[-2:0.01:4];
y=3*x.^3-26*x+6;
yd=9*x.^2-26;
ydd=18*x;
plot(x,y,'-b',x,yd,'--r',x,ydd,':k')
```

Create vector x with the domain of the function.

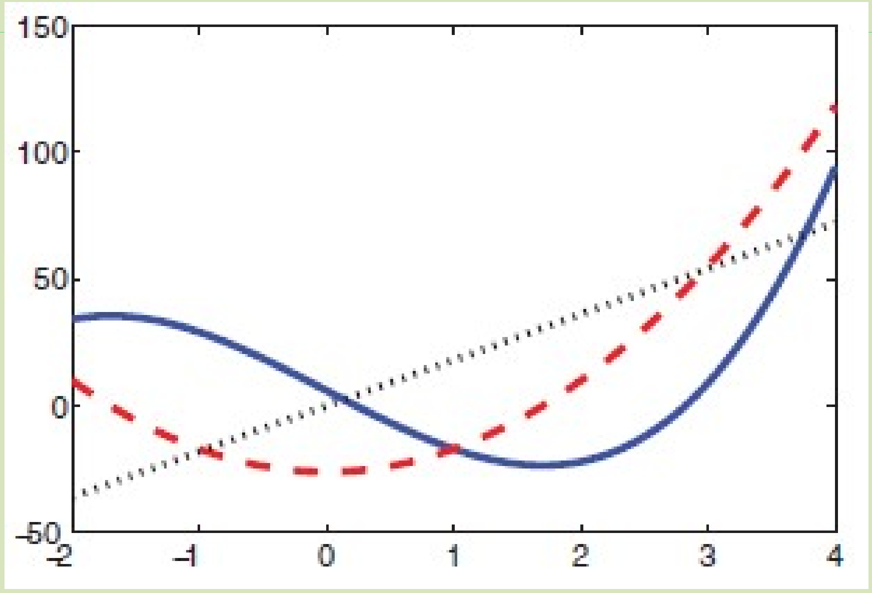
Create vector y with the function value at each x.

Create vector yd with values of the first derivative.

Create vector ydd with values of the second derivative.

Create three graphs, y vs. x, yd vs. x, and ydd vs. x, in the same figure.

The plot that is created is shown below:



2-Using the hold on and hold off Commands

To plot several graphs using the hold on and hold off commands,

1- one graph is plotted first with the plot command.

2-Then the hold on command is typed.

This keeps the Figure Window with the first plot open,

3- Additional graphs can be added with plot commands.

4-Each plot command creates a graph that is added to that figure.

5-The hold off command stops this process.

6- It returns MATLAB to the default mode, in which the plot

7-command erases the previous plot and resets the axis properties.

Example:

Plot the function *y = 3x3 – 26x + 10* , and its first and second derivatives, for –2 ≤ *x* ≤ *4* , all in the same plot ***using hold on hold off command*** .

The first derivative of the function is: *y' = 9x2 – 26.*

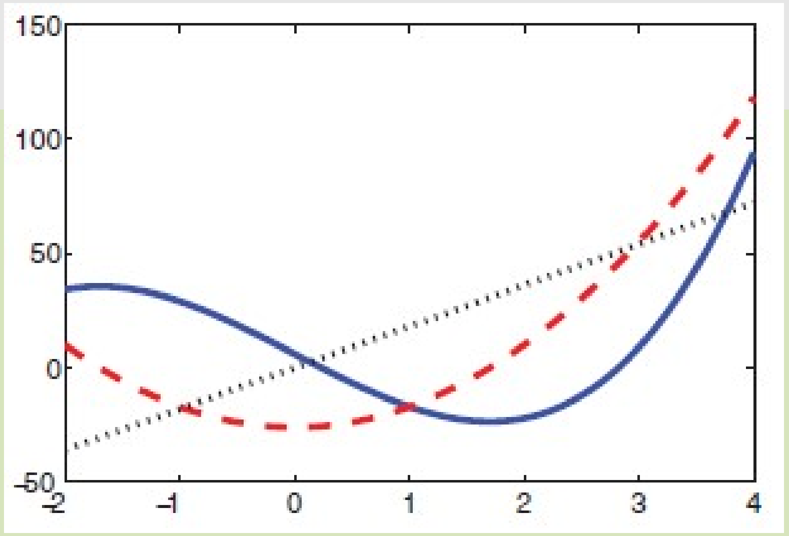
The second derivative of the function is: *y'' = 18x*

Computer Programming EEN170

```
x=[-2:0.01:4];
y=3*x.^3-26*x+6;
yd=9*x.^2-26;
ydd=18*x;
plot(x,y,'-b')
hold on
plot(x,yd,'--r')
plot(x,ydd,':k')
hold off
```

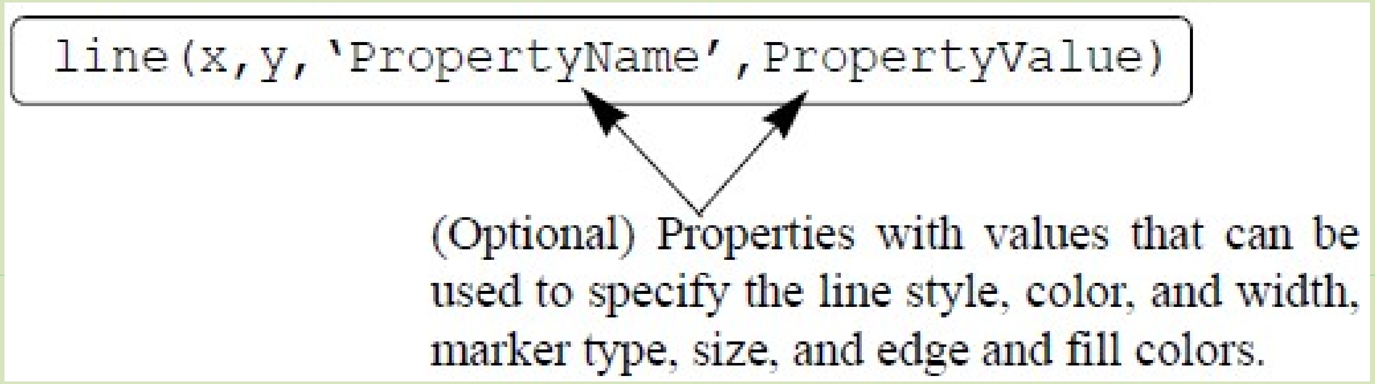
The first graph is created.

Two more graphs are added to the figure.



3-Using the line Command

With the line command additional graphs (lines) can be added to a plot that already exists. The form of the line command is:


```
line(x,y,'PropertyName',PropertyValue)
```
(Optional) Properties with values that can be used to specify the line style, color, and width, marker type, size, and edge and fill colors.

❖The format of the line command is almost the same as the plot command

❖ The line command does not have the line specifiers, but the line style, color, and marker can be specified with the Property Name and property value features.

❖The properties are optional & if none are entered MATLAB uses default properties and values.

❖For example, the command:

line(x,y, 'linestyle','--','color', 'r', 'marker ', 'o')

will add a dashed red line with circular markers to a plot that already exists.

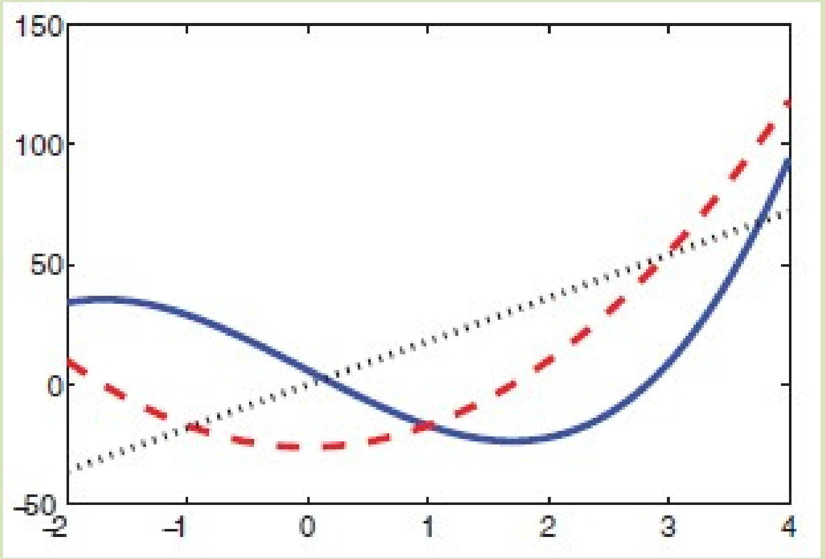
- The major difference between the plot and line commands is that the plot command starts a new plot every time it is executed, while the line command adds lines to a plot that already exists.
- To make a plot that has several graphs, a plot command is typed first and then line commands are typed for additional graphs.

- *(If a line command is entered before a plot command an error message is isplayed.)*

Example:
Plot the function *y = 3x3 – 26x + 10* , and its first and second derivatives, for –2 ≤ *x ≤ 4* , all in the same plot *using line command*.

The first derivative of the function is: *y' = 9x2 – 26.*
The second derivative of the function is: *y'' = 18x*

```
x=[-2:0.01:4];
y=3*x.^3-26*x+6;
yd=9*x.^2-26;
ydd=18*x;
plot(x,y,'LineStyle','-','color','b')
line(x,yd,'LineStyle','--','color','r')
line(x,ydd,'linestyle',':','color','k')
```



FORMATTING A PLOT

-The **plot** and **fplot** commands create bare plots.

Usually,

- figure that contains a plot needs to be formatted to have a specific look
- to display information in addition to the graph itself.

This can include specifying axis labels, plot title, legend, grid, range of custom axis, and text labels.

Plots can be formatted by using MATLAB commands that follow the plot or fplot command, or by using the plot editor in the Figure Window.

FORMATTING A PLOT(cont.)

-The first method is useful when a plot command is a part of a computer program (script file).

-When the formatting commands are included in the program, a formatted plot is created every time the program is executed.

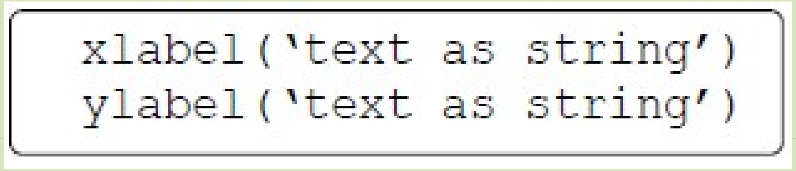
-On the other hand, formatting that is done in the Figure Window with the plot editor after a plot has been created holds only for that specific plot, and will have to be repeated the next time the plot is created.

Formatting a Plot Using Commands

The formatting commands are entered after the **plot** or the **fplot** command. The various formatting commands are:

The xlabel & ylabel commands:

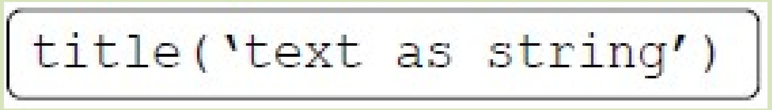
Labels can be placed next to the axes with the xlabel and ylabel command which have the form:



```
xlabel('text as string')
ylabel('text as string')
```

The title command:

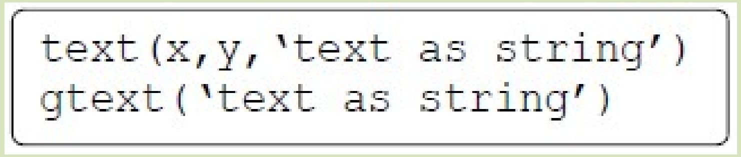
A title can be added to the plot with the command:



```
title('text as string')
```

The text command:

A text label can be placed in the plot with the **text** or **gtext** commands:



```
text(x,y,'text as string')
gtext('text as string')
```

❖The **text** command places the text in the figure such that the first character is positioned at the point with the coordinates x, y

❖The **gtext** command places the text at a position specified by the user.

The legend command:

The legend command places a legend on the plot. The legend shows a sample of the line type of each graph that is plotted, and places a label, specified by the user, beside the line sample.

The form of the command is:

legend('string1','string2', ... ,pos)

- The strings are the labels that are placed next to the line sample.
- Their order corresponds to the order in which the graphs were created.
- The pos is an optional number that specifies where in the figure the legend is to be placed.

The options are:

pos = -1 Places the legend outside the axes boundaries on the right side.

pos = 0 Places the legend inside the axes boundaries in a location that interferes the least with the graphs.

pos = 1 Places the legend at the upper-right corner of the plot (default).

pos = 2 Places the legend at the upper-left corner of the plot.

pos = 3 Places the legend at the lower-left corner of the plot.

pos = 4 Places the legend at the lower-right corner of the plot.

Formatting the text within the xlabel, ylabel, title, text and legend commands:

- The text in the string that is included in the command can be formatted.
- The formatting can be used to define the font, size, position superscript, subscript), style (italic, bold, etc.), and color of the characters, the color of the background, and many other details of the display.
- Some of the more common formatting possibilities are described below.

The formatting can be done either by adding modifiers inside the string, or by adding to the command optional PropertyName & Property Value arguments following the string.

Subscript and superscript:

❖A single character can be displayed as a subscript or a superscript by typing _ (the underscore character) or ^ in front of the character, respectively.

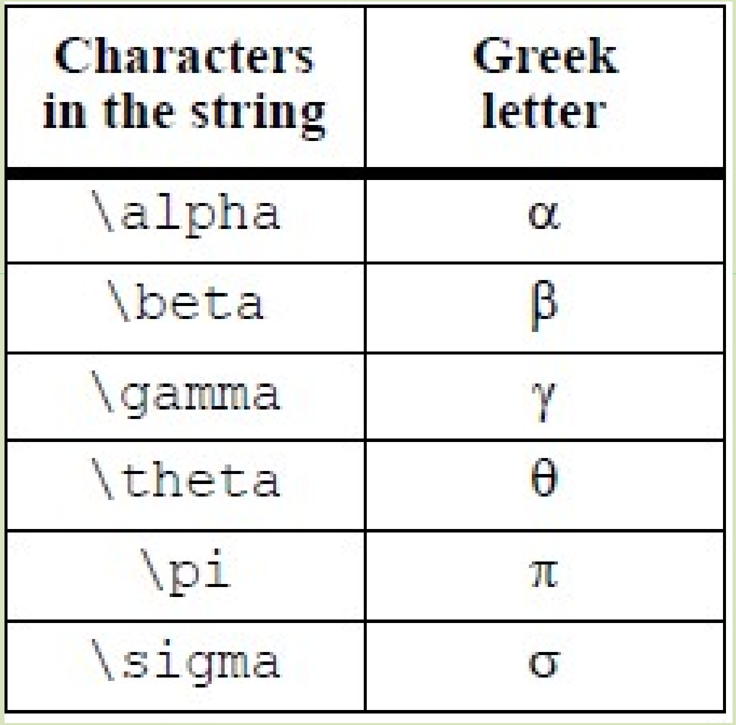
❖Several consecutive characters can be displayed as a subscript or a superscript by typing the characters inside braces { } following the _ or the ^.

Greek characters:

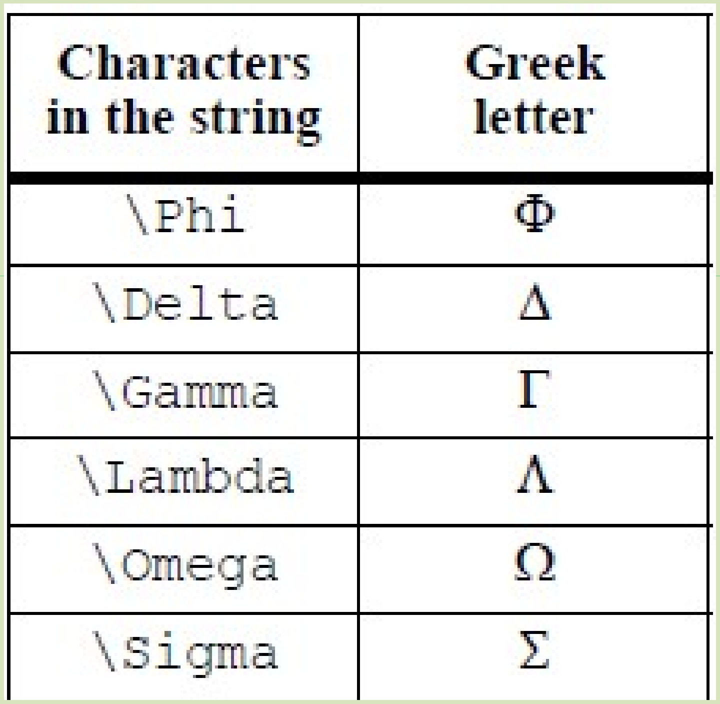
❖Greek characters can be included in the text by typing \name of the letter within the string.

❖ To display a lowercase Greek letter the name of the letter should be typed in all lowercase English characters,

❖ To display a capital Greek letter the name of the letter should start with a capital letter



Characters in the string	Greek letter
\alpha	α
\beta	β
\gamma	γ
\theta	θ
\pi	π
\sigma	σ



Characters in the string	Greek letter
\Phi	Φ
\Delta	Δ
\Gamma	Γ
\Lambda	Λ
\Omega	Ω
\Sigma	Σ

The axis command:

When the plot(x,y) command is executed, MATLAB creates axes with limits that are based on the minimum and maximum values of the elements of x and y.

The axis command can be used to change the range and the appearance of the axes. In many situations a graph looks better if the range of the axes extend beyond the range of the data.

The following forms of the axis commands:

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

Sets the limits of both the *x and y* axes (xmin, xmax, ymin, and ymax are umbers).

axis equal

Sets the same scale for both axes.

axis square

Sets the axes region to be square.

axis tight

Sets the axis limits to the range of the data.

The grid command:

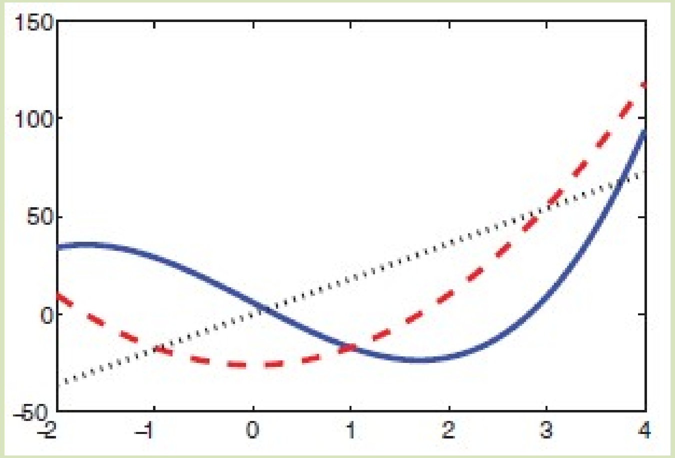
grid on	Adds grid lines to the plot.
grid off	Removes grid lines from the plot.

Example:

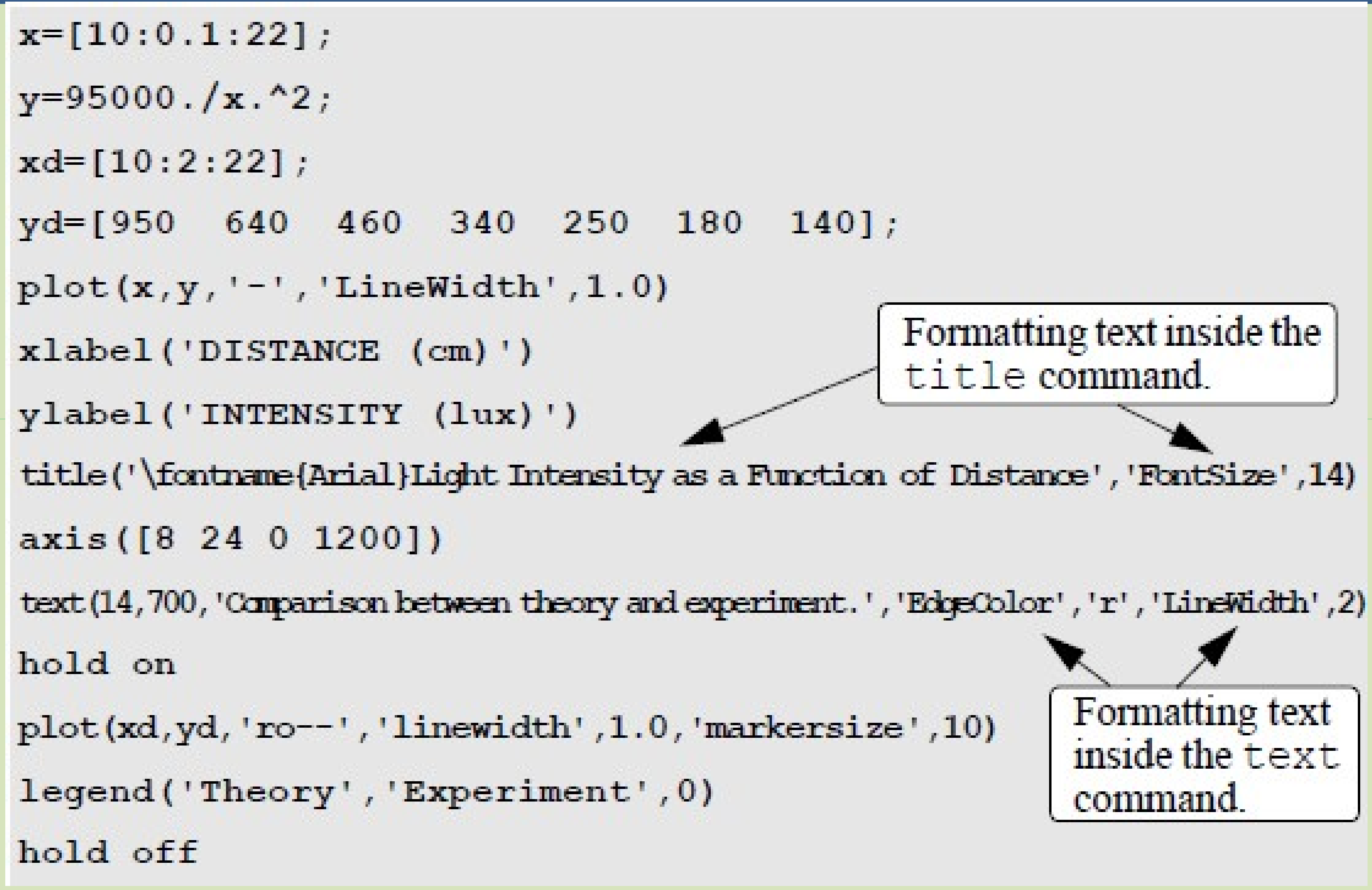
Plot the function $y = 3x3 – 26x + 10$, and its first and second derivatives, for $–2 \le x \le 4$, all in the same plot.

The first derivative of the function is: $y' = 9x2 – 26.$

The second derivative of the function is: $y'' = 18x$



Computer Programming EEN170



```
x=[10:0.1:22];
y=95000./x.^2;
xd=[10:2:22];
yd=[950   640   460   340   250   180   140];
plot(x,y,'-','LineWidth',1.0)
xlabel('DISTANCE (cm)')
ylabel('INTENSITY (lux)')
title('\fontname{Arial}Light Intensity as a Function of Distance','FontSize',14)
axis([8 24 0 1200])
text(14,700,'Comparison between theory and experiment.','EdgeColor','r','LineWidth',2)
hold on
plot(xd,yd,'ro--','linewidth',1.0,'markersize',10)
legend('Theory','Experiment',0)
hold off
```

Formatting text inside the `title` command.

Formatting text inside the `text` command.

Formatting a Plot Using the Plot Editor

Click the arrow button to start the plot edit mode. Then click on an item. A window with formatting tool for the item opens.

Use the **Edit** and **Insert** menus to add formatting objects, or to edit existing objects.

Change position of a label, legend or other object by clicking on the object and dragging.

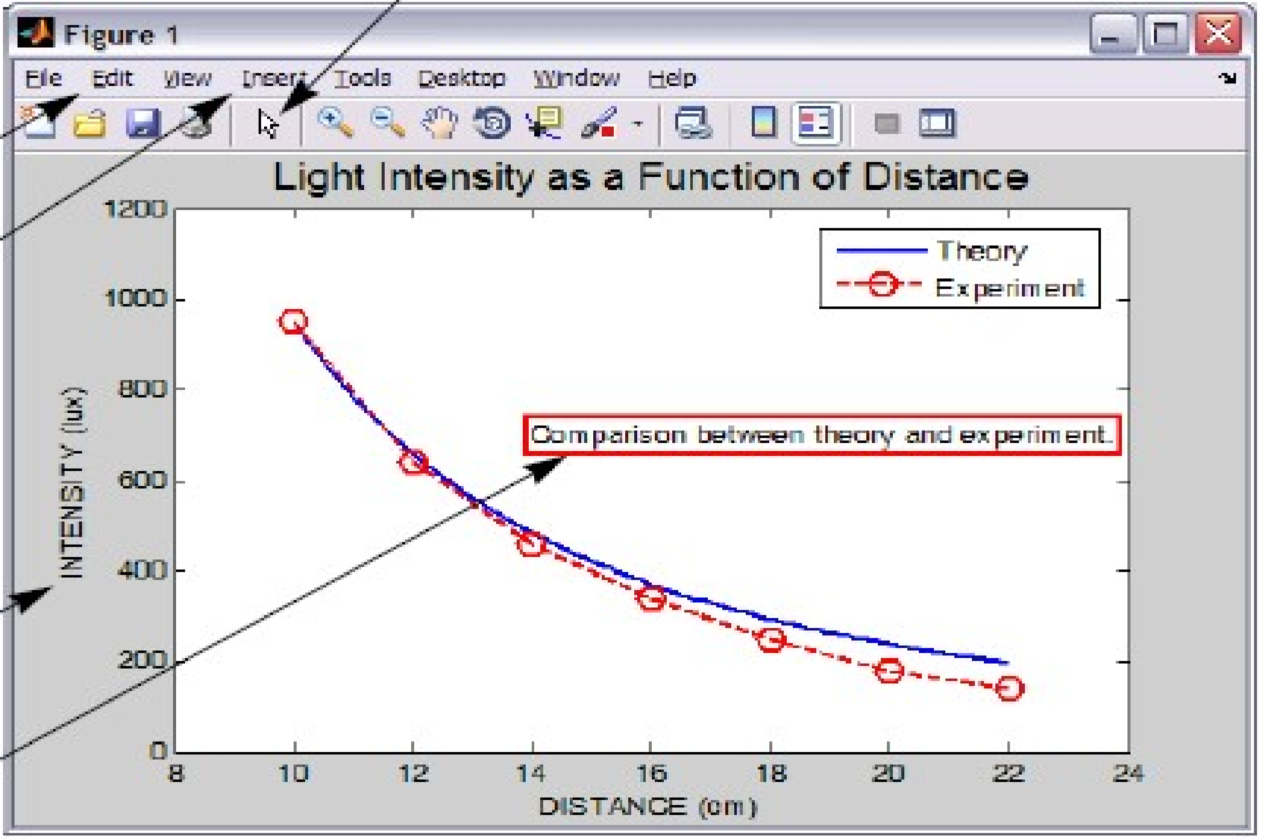


Figure 1
File Edit View Insert Tools Desktop Window Help
Light Intensity as a Function of Distance
Theory
Experiment
Comparison between theory and experiment.
INTENSITY (lux)
DISTANCE (cm)
0
200
400
600
800
1000
1200
8
10
12
14
16
18
20
22
24

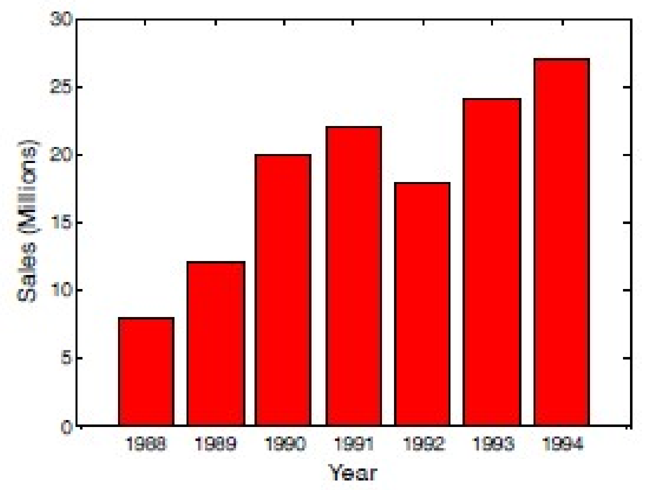
PLOTS WITH SPECIAL GRAPHICS

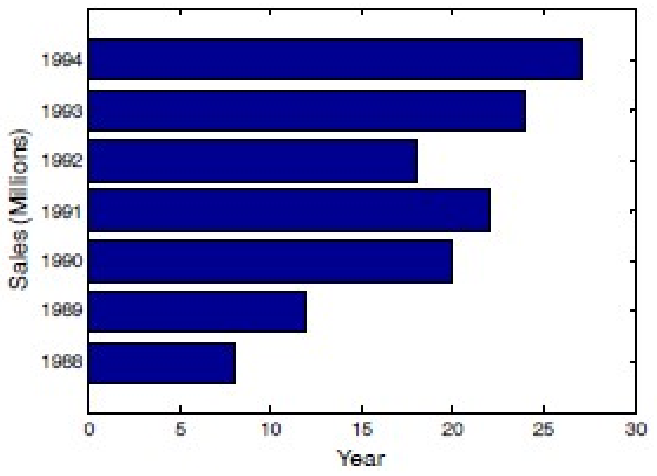
-with line plots in which the data points are connected by lines, different graphics or geometry can present data more effectively.

-MATLAB has many options for creating a wide variety of plots.

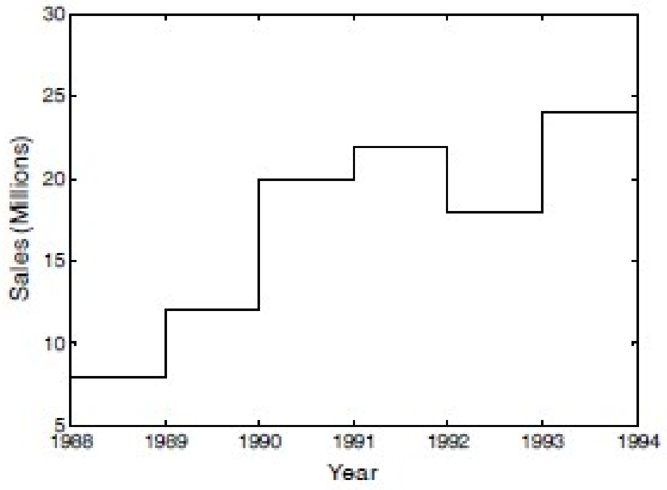
- These include ***bar***, ***stairs***, ***stem***, and ***pie*** plots and many more.

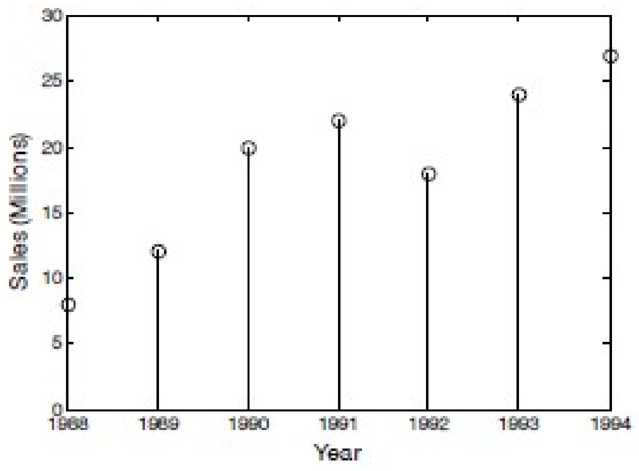
-Following are some of the special graphics plots that can be created with MATLAB..

<u>Vertical Bar Plot</u>
 	
Function format:
 	
`bar(x,y)`
		`yr=[1988:1994];`
		`sle=[8 12 20 22 18 24 27];`
		`bar(yr,sle,'r')` ← The bars are in red.
		`xlabel('Year')`
		`ylabel('Sales (Mil-`
		`lions)')`

<u>Horizontal Bar Plot</u>
 	
Function format:
 	
`barh(x,y)`
		`yr=[1988:1994];`
		`sle=[8 12 20 22 18 24 27];`
		`barh(yr,sle)`
		`xlabel('Sales (Millions)')`
		`ylabel('Year')`

Computer Programming EEN170

<u>Stairs Plot</u>
 	
Function format:
 	
`stairs(x,y)`
		`yr=[1988:1994];`
		`sle=[8 12 20 22 18 24 27];`
		`stairs(yr,sle)`

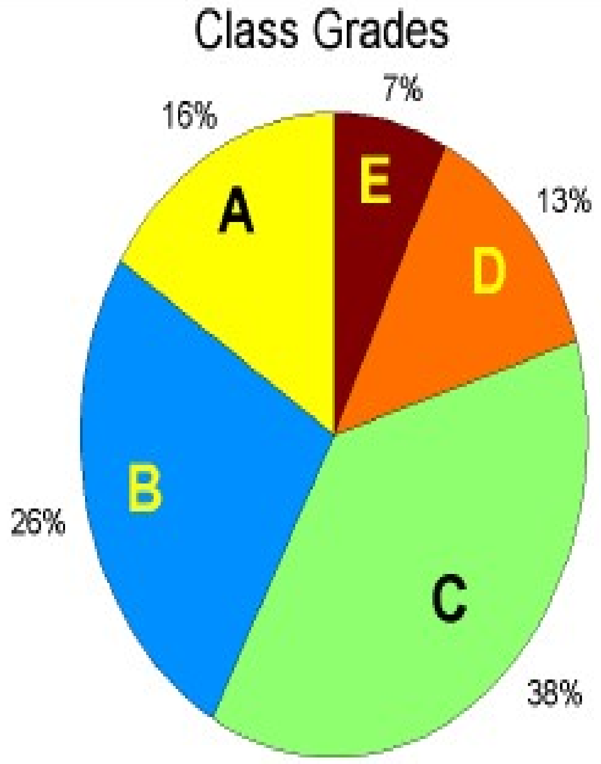
<u>Stem Plot</u>
 	
Function Format
 	
`stem(x,y)`
		`yr=[1988:1994];`
		`sle=[8 12 20 22 18 24 27];`
		`stem(yr,sle)`

Grade	A	B	C	D	E
Number of students	11	18	26	9	5

Pie Plot

Function format:

```
pie(x)
```



```
grd=[11 18 26 9 5];
pie(grd)
title('Class Grades')
```

MATLAB draws the sections in different colors. The letters (grades) were added using the Plot Editor.

Computer Programming EEN170

HISTOGRAMS

-Histograms are plots that show the distribution of data.

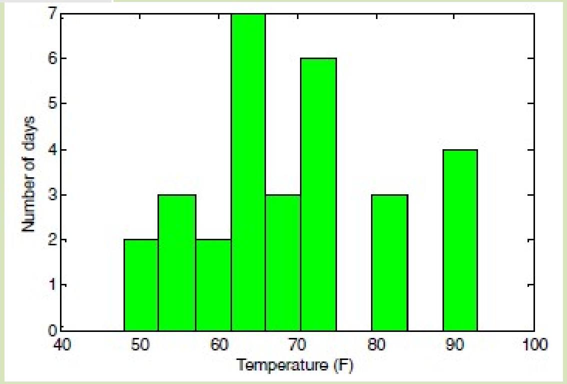
- The overall range of a given set of data points is divided into sub ranges (bins), and the histogram shows how many data points are in each bin.

- The histogram is a vertical bar plot in which the width of each bar is equal to the range of the corresponding bin and the height

```
>> y=[58 73 73 53 50 48 56 73 73 66 69 63 74 82 84 91 93 89
91 80 59 69 56 64 63 66 64 74 63 69];

>> hist(y)
```

```
hist(y)
```

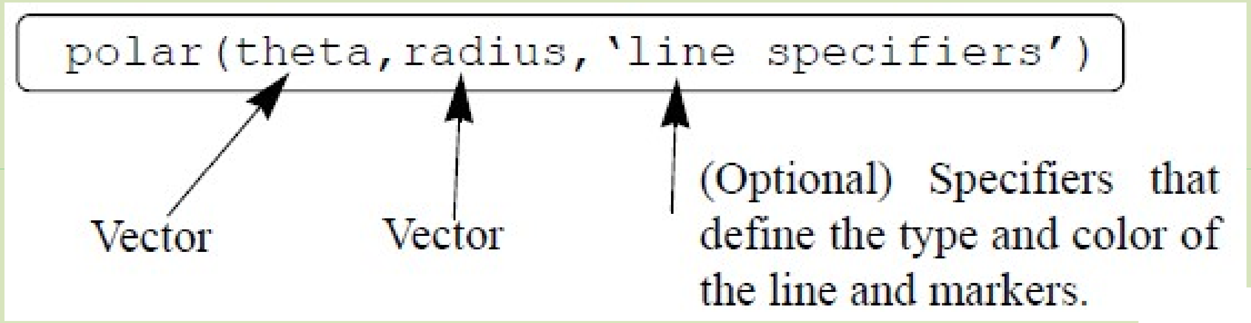


POLAR PLOTS

-Polar coordinates, in which the position of a point in a plane is defined by the angle θ and the radius (distance) to the point, are frequently used in

-The polar command is used to plot functions in polar coordinates.

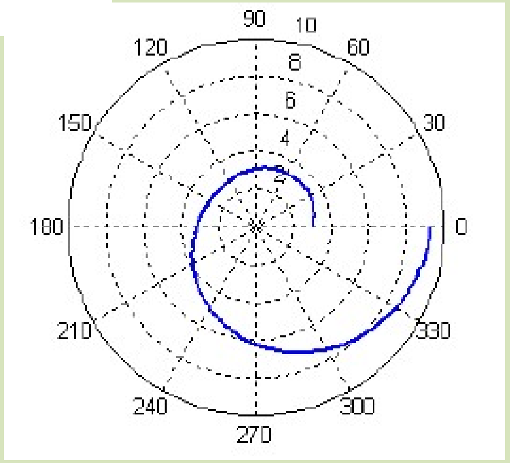
- The command has the form:



```
polar(theta,radius,'line specifiers')
```

Vector
Vector
(Optional) Specifiers that define the type and color of the line and markers.

```
t=linspace(0,2*pi,200);
r=3*cos(0.5*t).^2+t;
polar(t,r)
```

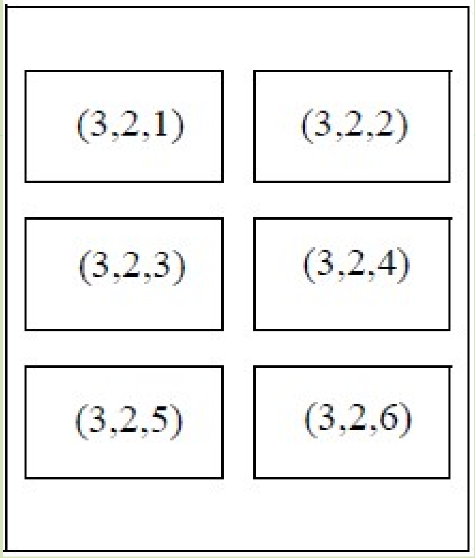


PUTTING MULTIPLE PLOTS ON THE SAME PAGE

Multiple plots can be created on the same page with the subplot command , which has the form:

```
subplot(m,n,p)
```

❖The command divides the Figure Window into $m \times n$ rectangular subplots.
❖The subplots are arranged like elements in an $m \times n$ matrix where each element is a subplot.
❖The subplots are numbered from 1 through $m \cdot n$.
❖The upper left subplot is numbered 1 and the lower right subplot is numbered $m \cdot n$.

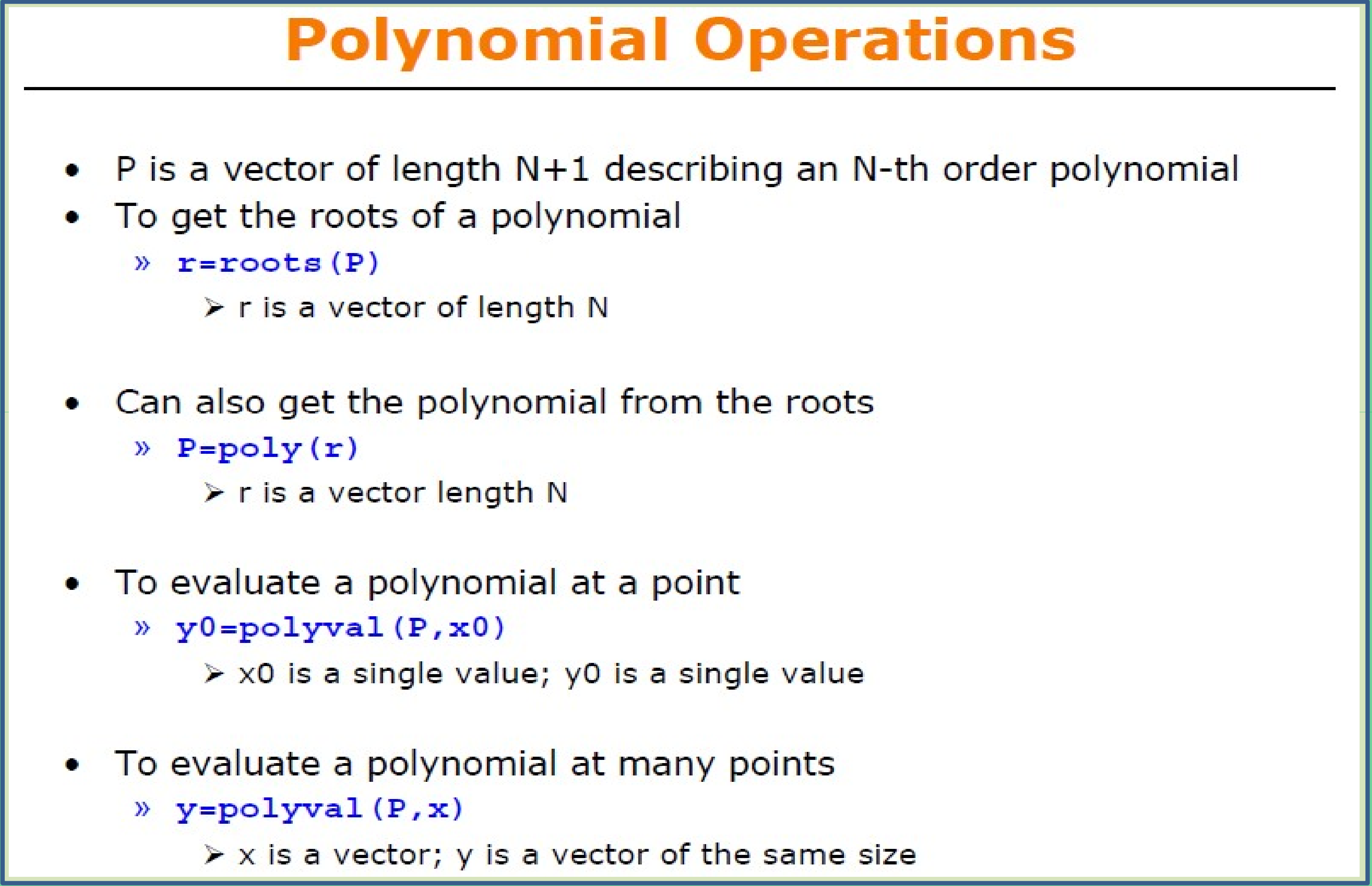


Polynomials

- Many functions can be well described by a high-order polynomial
- MATLAB represents a polynomials by a vector of coefficients
    - ➢ if vector P describes a polynomial

$$\underset{\nearrow P(1)}{a}x^3+\underset{\nearrow P(2)}{b}x^2+\underset{\nearrow P(3)}{c}x+\underset{\nwarrow P(4)}{d}$$

- P=[1 0 -2] represents the polynomial x^2-2
- P=[2 0 0 0] represents the polynomial $2x^3$



Polynomial Operations

- P is a vector of length N+1 describing an N-th order polynomial
- To get the roots of a polynomial

```
» r=roots(P)
```

  - ➢ r is a vector of length N

- Can also get the polynomial from the roots

```
» P=poly(r)
```

  - ➢ r is a vector length N

- To evaluate a polynomial at a point

```
» y0=polyval(P,x0)
```

  - ➢ x0 is a single value; y0 is a single value

- To evaluate a polynomial at many points

```
» y=polyval(P,x)
```

  - ➢ x is a vector; y is a vector of the same size

Polynomial Fitting

- MATLAB makes it very easy to fit polynomials to data
- Given data vectors X=[-1 0 2] and Y=[0 -1 3]

```
» p2=polyfit(X,Y,2);
```

  - ➢ finds the best second order polynomial that fits the points (-1,0),(0,-1), and (2,3)
  - ➢ see **help polyfit** for more information

```
» plot(X,Y,'o', 'MarkerSize', 10);
» hold on;
» x = -3:.01:3;
» plot(x,polyval(p2,x), 'r--');
```

Exercise: Polynomial Fitting

- Evaluate $y = x^2$ for x=-4:0.1:4.
- Add random noise to these samples. Use **randn**. Plot the noisy signal with **.** markers
- Fit a 2nd degree polynomial to the noisy data
- Plot the fitted polynomial on the same plot, using the same x values and a red line

Exercise: Polynomial Fitting

- Evaluate $y = x^2$ for x=-4:0.1:4.

```
» x=-4:0.1:4;
» y=x.^2;
```

- Add random noise to these samples. Use **randn**. Plot the noisy signal with **.** markers

```
» y=y+randn(size(y));
» plot(x,y,'.');
```

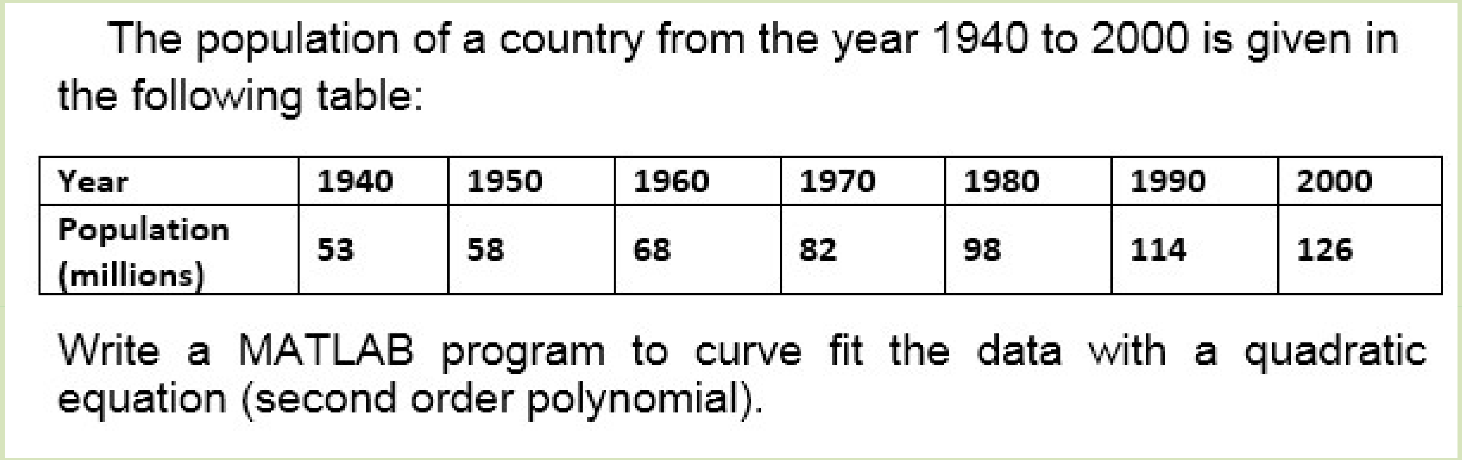
- Fit a 2nd degree polynomial to the noisy data

```
» p=polyfit(x,y,2);
```

- Plot the fitted polynomial on the same plot, using the same x values and a red line

```
» hold on;
» plot(x,polyval(p,x),'r')
```

Example 1

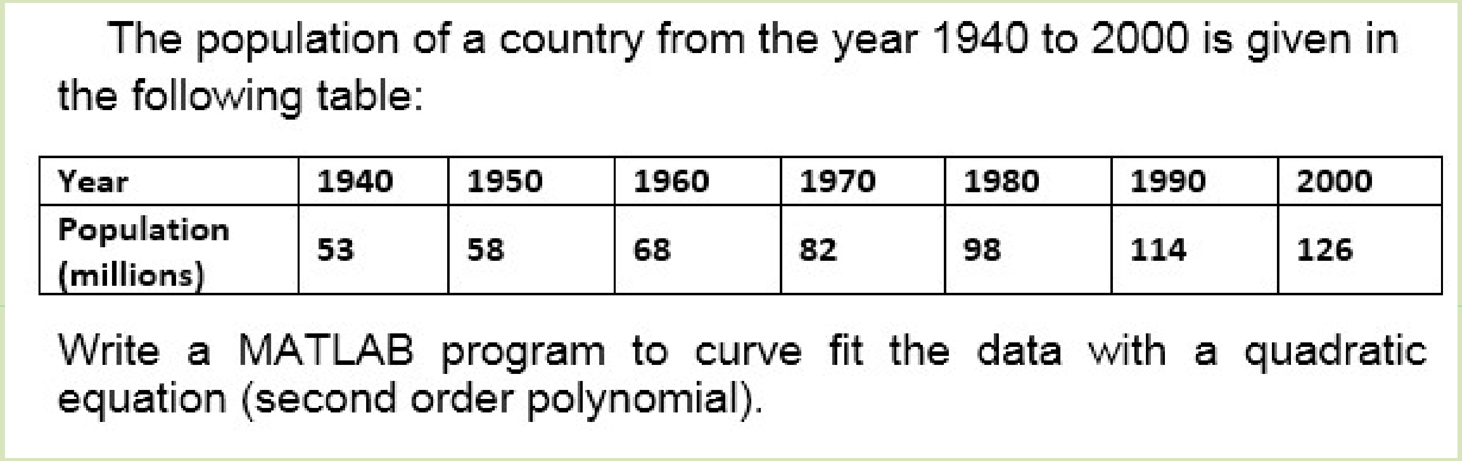


The population of a country from the year 1940 to 2000 is given in the following table:

Year	1940	1950	1960	1970	1980	1990	2000
Population (millions)	53	58	68	82	98	114	126

Write a MATLAB program to curve fit the data with a quadratic equation (second order polynomial).

Example 1



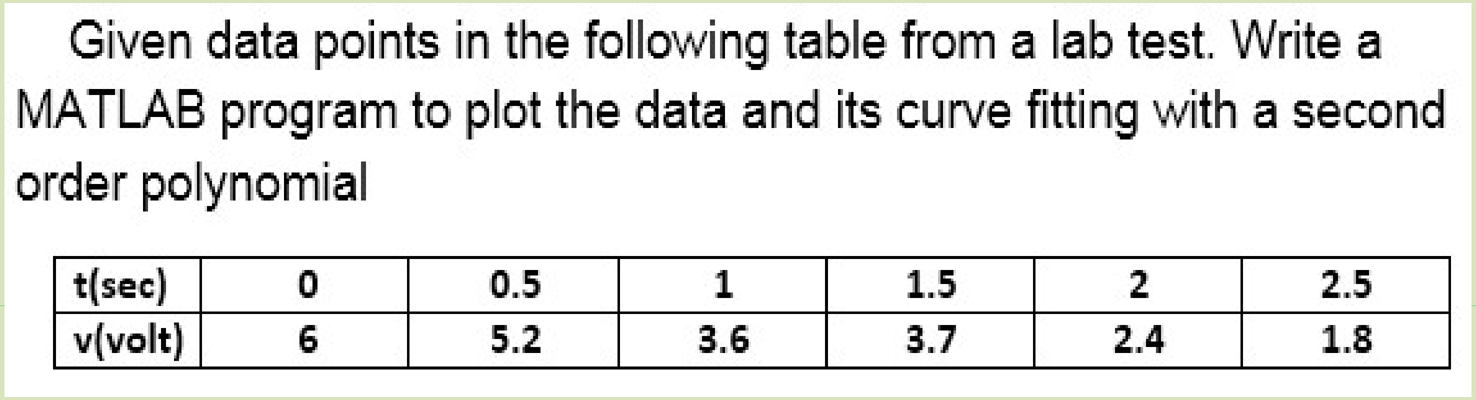
The population of a country from the year 1940 to 2000 is given in the following table:

Year	1940	1950	1960	1970	1980	1990	2000
Population (millions)	53	58	68	82	98	114	126

Write a MATLAB program to curve fit the data with a quadratic equation (second order polynomial).

```
>>x=[1940:10:2000];
>>y=[53  58  68  82  98  114  126];
>>polyfit(x,y,2)
```

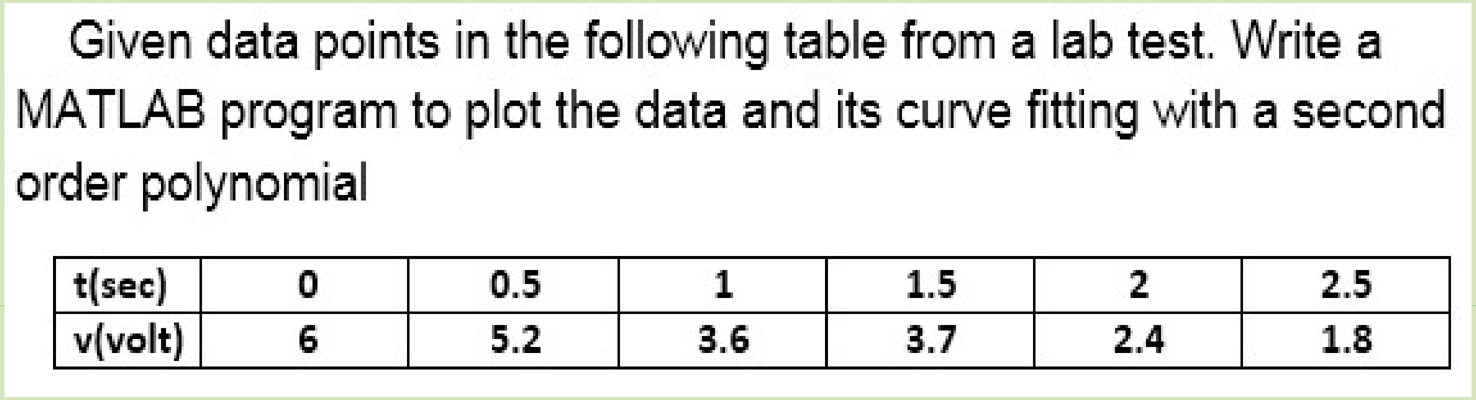
Example 2



Given data points in the following table from a lab test. Write a MATLAB program to plot the data and its curve fitting with a second order polynomial

t(sec)	0	0.5	1	1.5	2	2.5
v(volt)	6	5.2	3.6	3.7	2.4	1.8

Example 2



Given data points in the following table from a lab test. Write a MATLAB program to plot the data and its curve fitting with a second order polynomial

t(sec)	0	0.5	1	1.5	2	2.5
v(volt)	6	5.2	3.6	3.7	2.4	1.8

```
>>t=[0:0.5:2.5];
>>v=[6   5.2   3.6   3.7   2.4   1.8];
>>p= polyfit ( t , v , 2);
>>m= polyval ( p , t);
>>plot( t, v , ‘r - *', t , m)
```