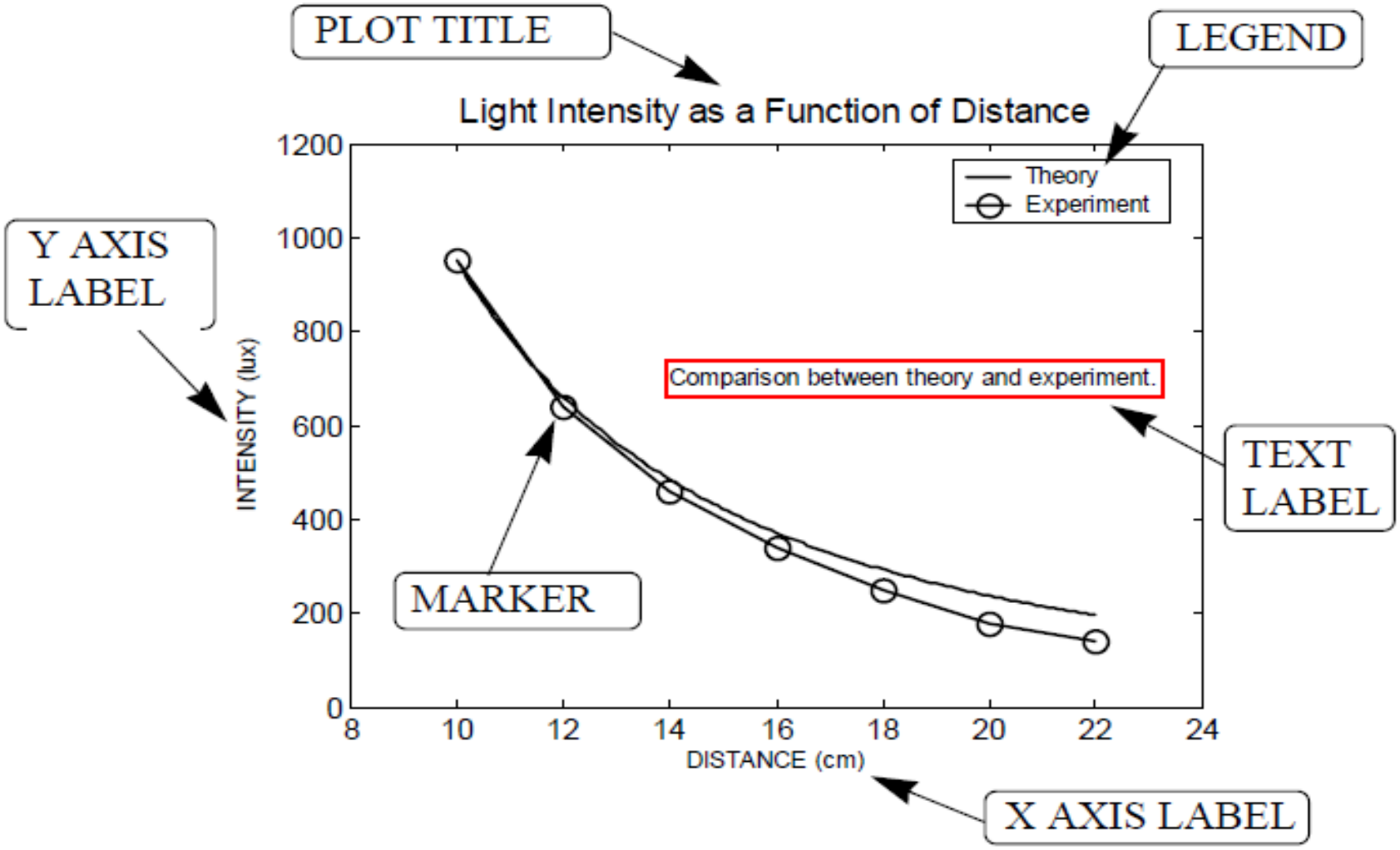
Computer Programming: 2D Plots

Asst. Prof. Dr. Yalçın İşler
Izmir Katip Celebi University

Outline

- Plot
- Fplot
- Multiple Plots
- Formatting Plot
- Logarithmic Plots
- Errorbar Plots
- Special plots: Bar, Stairs, Stem, Pie
- Histogram Plots
- Polar Plots
- Multiple Plots
- Multiple Figures

2D Formatted Plot Example



Plot command

```
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 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	-.

Plot command (cont'd)

The line color specifiers are:

Line Color	Specifier
red	r
green	g
blue	b
cyan	c

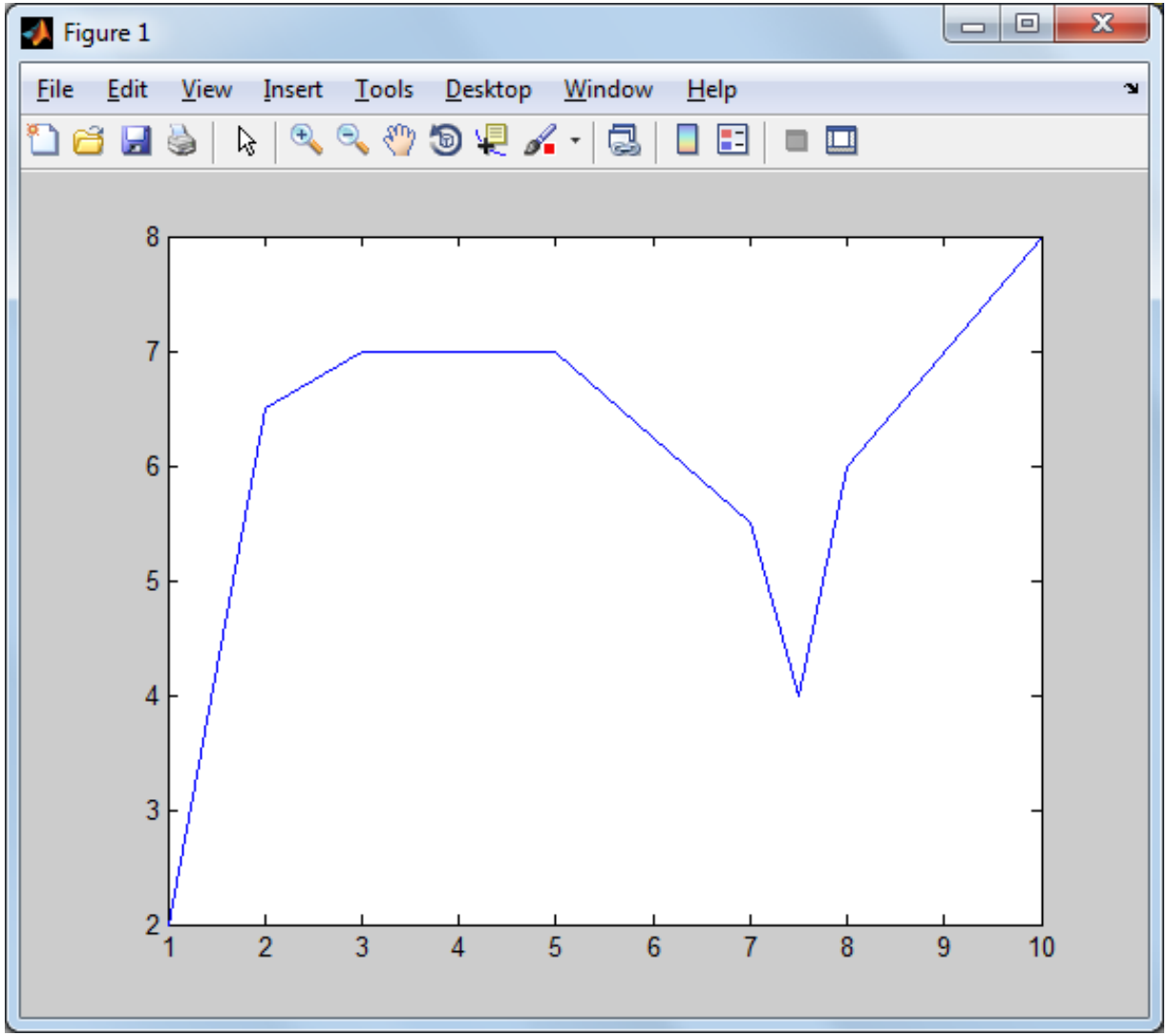
Line Color	Specifier
magenta	m
yellow	y
black	k
white	w

The marker type specifiers 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			

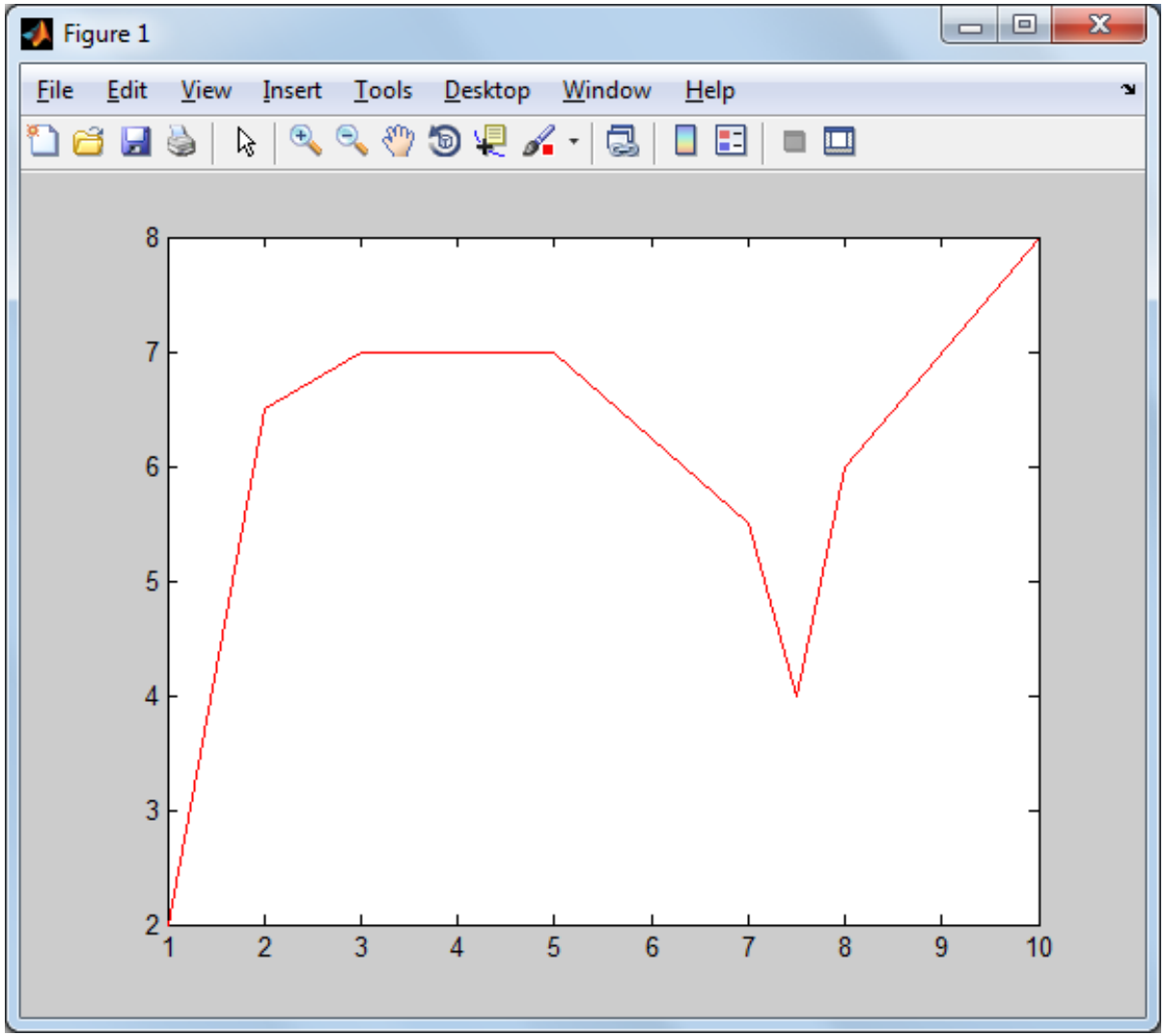
Plot examples (1)

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



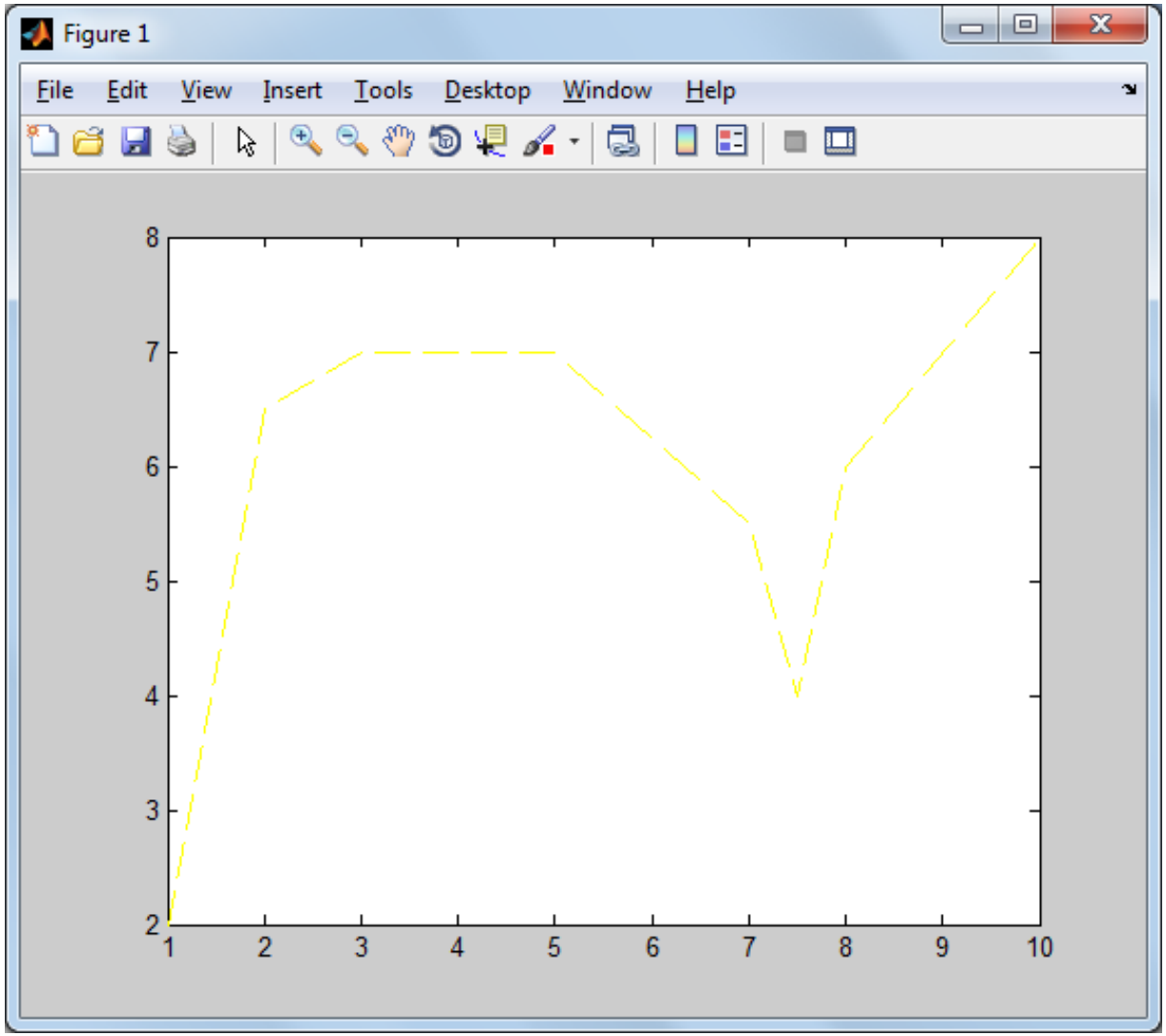
Plot examples (2)

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



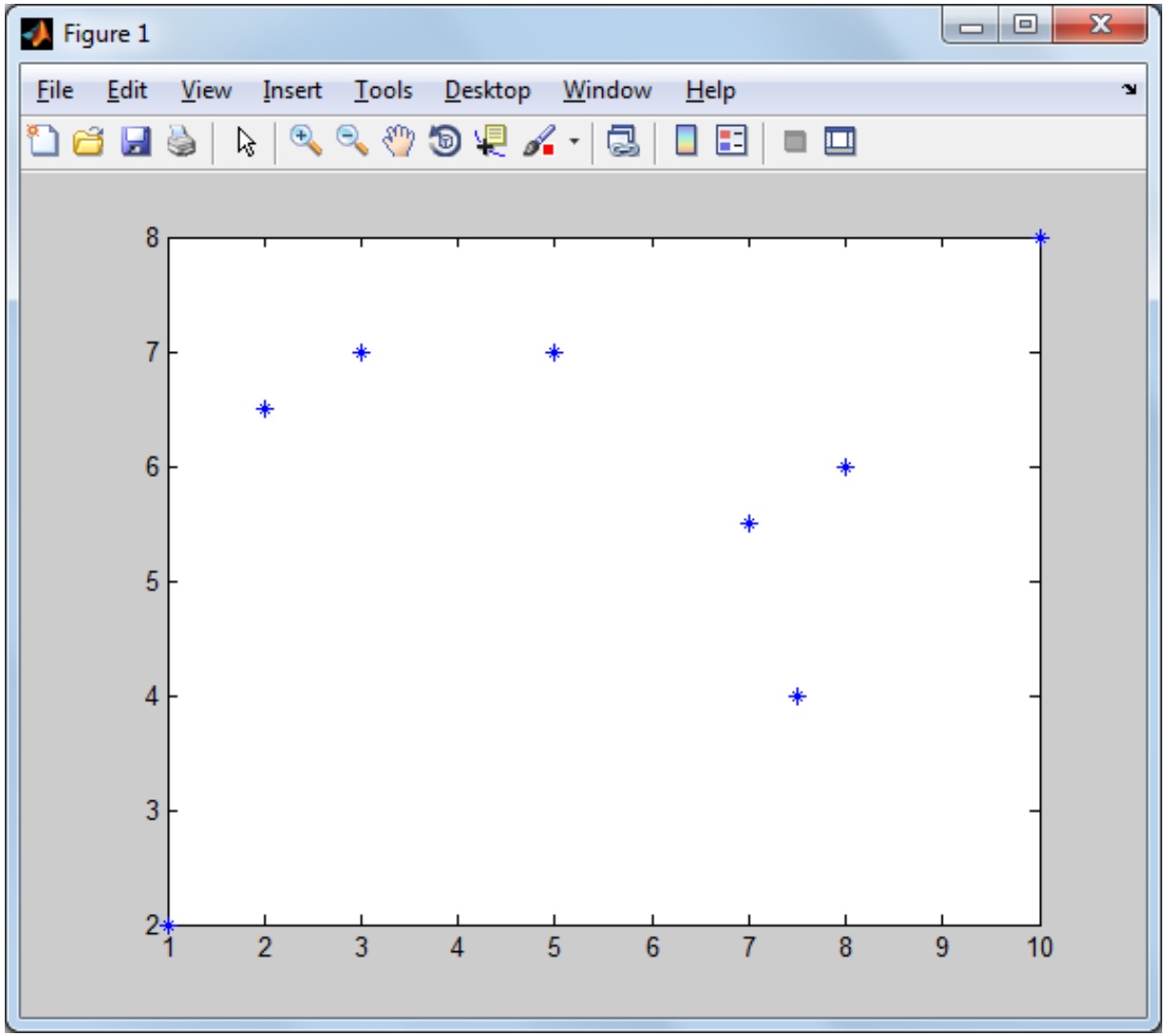
Plot examples (3)

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



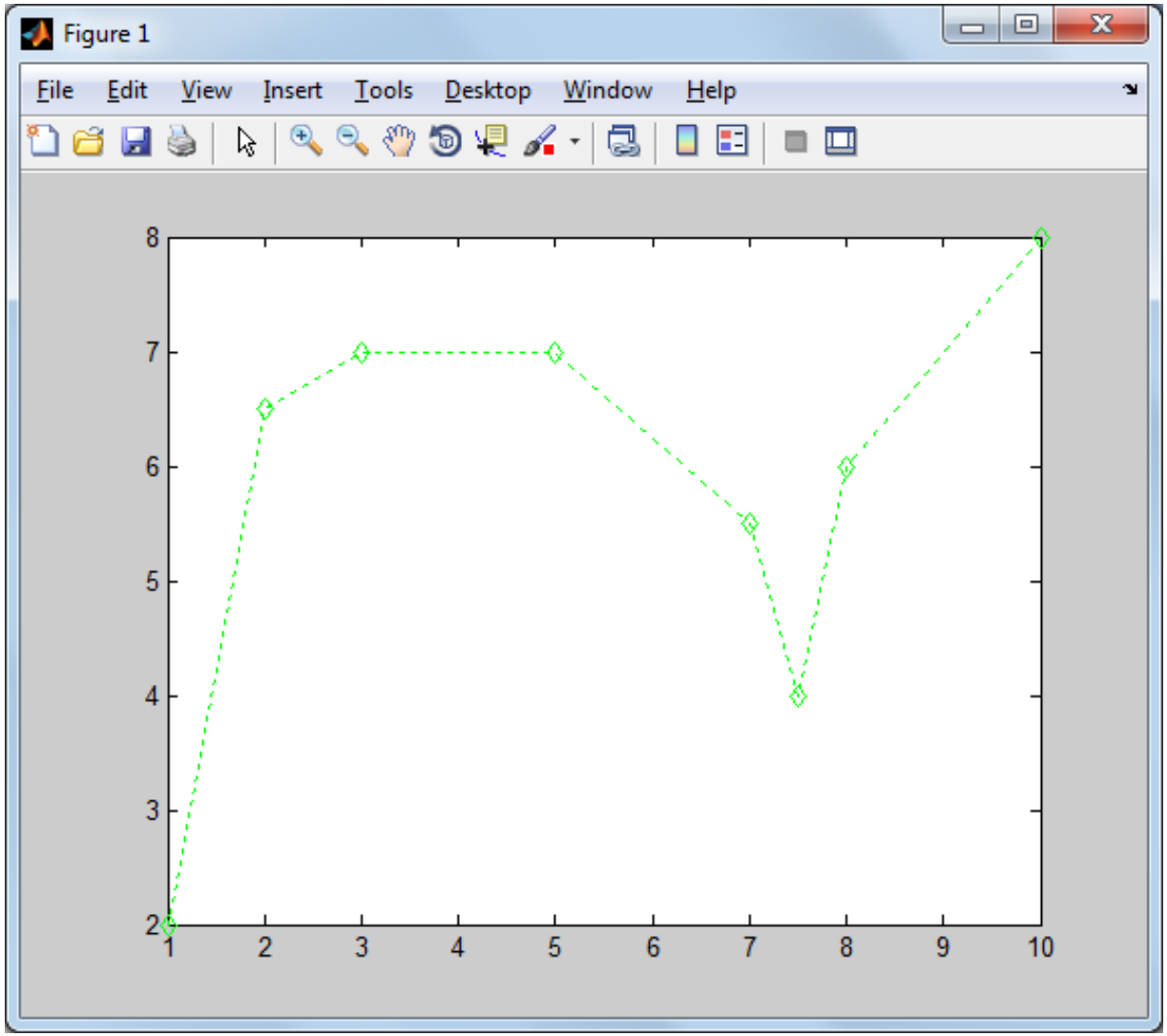
Plot examples (4)

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



Plot examples (5)

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



Plot properties

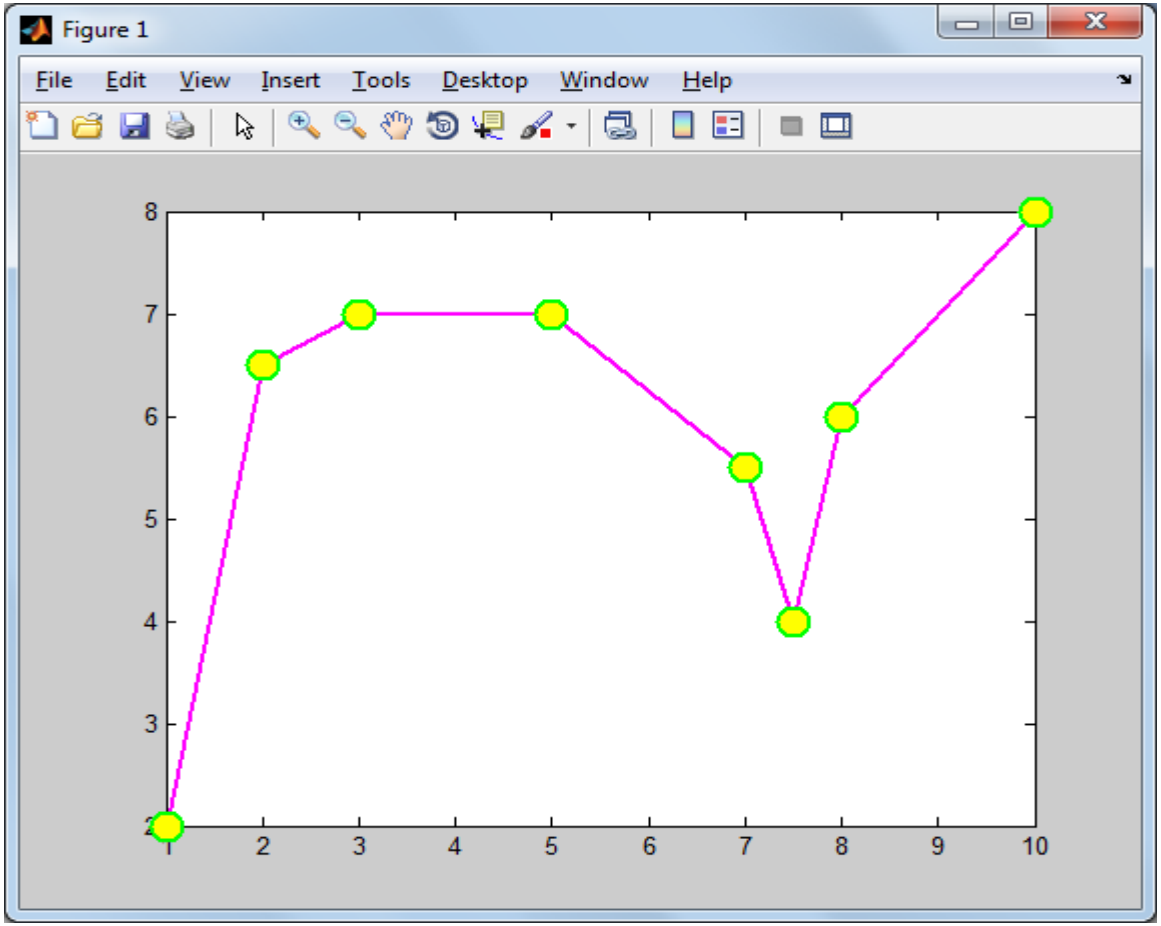
Four properties and possible values are:

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.

Plot examples (6)

```
>> x=[1 2 3 5 7 7.5 8 10];
>> y=[2 6.5 7 7 5.5 4 6 8];
>> plot(x,y,'-mo','LineWidth',2,'markersize',12,'MarkerEdgeColor','g','markerfacecolor','y')
```

This creates a plot that connects the points with a magenta solid line and circles as markers at the points. The line width is two points and the size of the circle markers is 12 points. The markers have a green edge line and yellow filling.



Line specifiers

The Property Names for the line specifiers are:

Specifier	Property Name	Possible Property Values
Line Style	`linestyle` (or `LineStyle`)	Line style specifier from the table above, typed as a string.
Line Color	`color` (or `Color`)	Color specifiers from the table above, typed as a string.
Marker	`marker` (or `Marker`)	Marker specifier from the table above, typed as a string.

Plot examples (7)

As an example, the `plot` command is used to plot the function $y = 3.5^{-0.5x}\cos(6x)$ for $-2 \le x \le 4$. A program that plots this function is shown in the following script file.

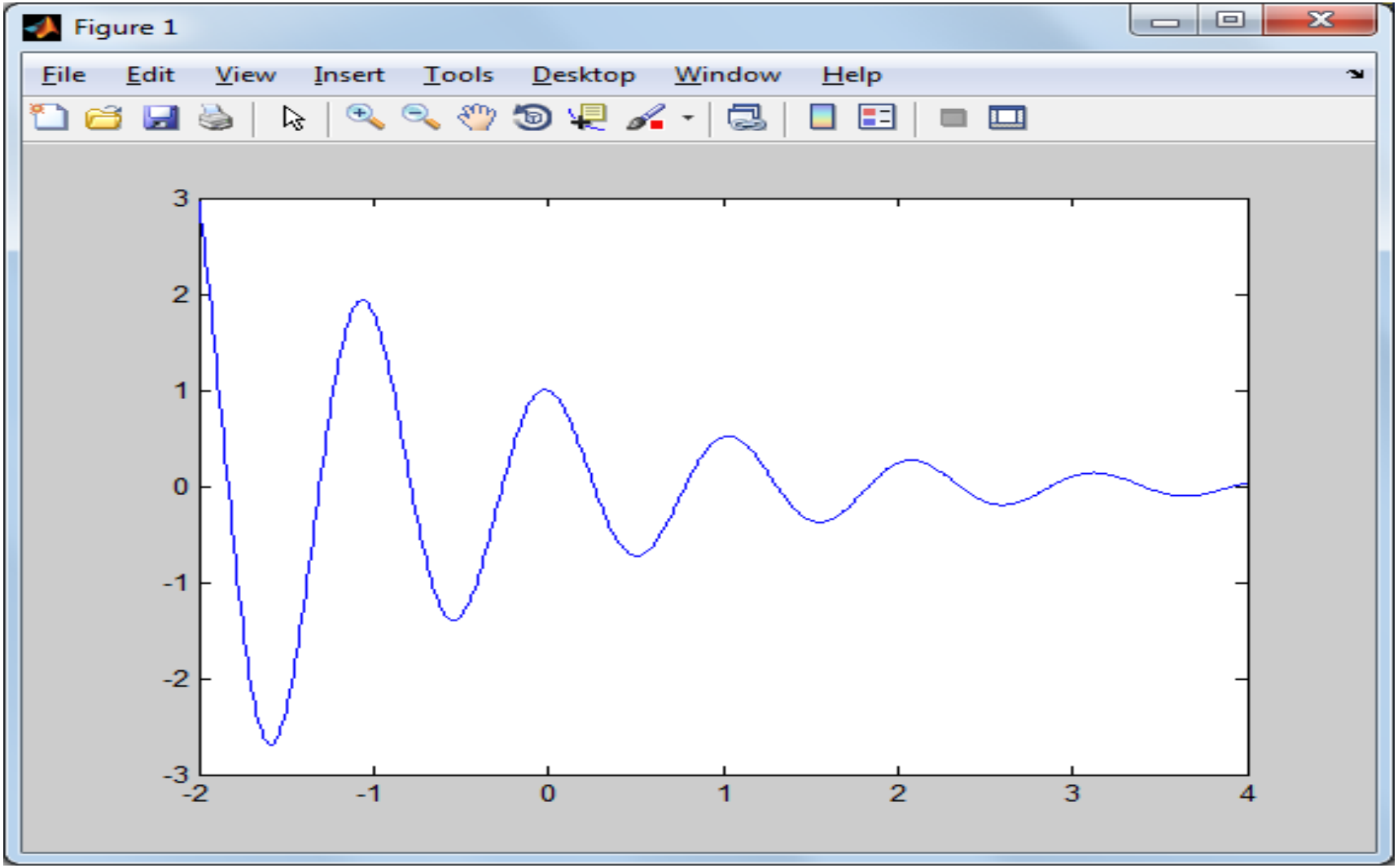
```
%  A script file that creates a plot of
%  the function: 3.5.^(-0.5*x).*cos(6x)
x=[-2:0.01:4];
y=3.5.^(-0.5*x).*cos(6*x);

plot(x,y)
```

Create vector x with the domain of the function.

Create vector y with the function value at each x.

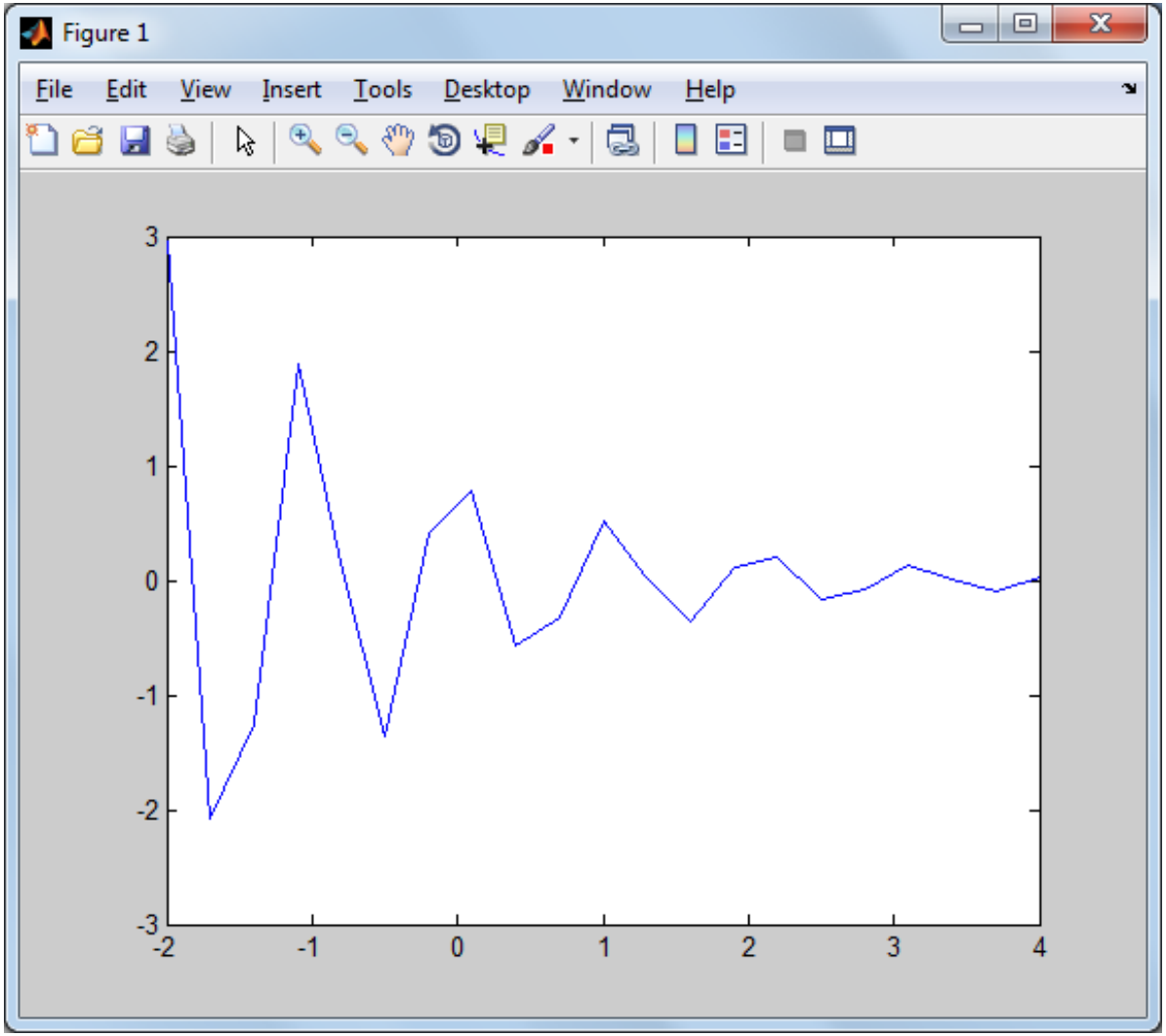
Plot y as a function of x.

Plot examples (7)



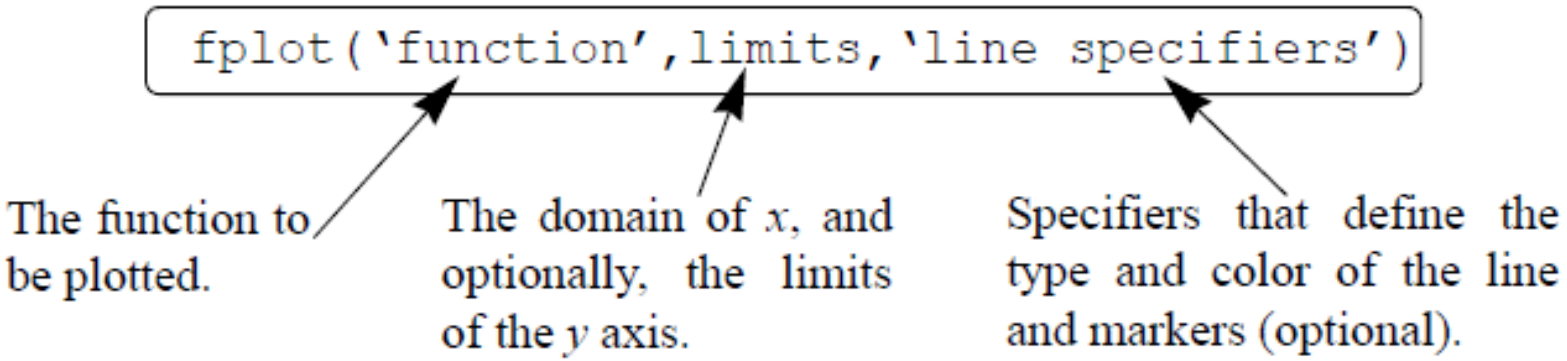
Plot examples (8)

However, if the same function in the same domain is plotted with much larger spacing, for example 0.3, the plot that is obtained gives a distorted picture of the function.



fplot

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



- The function can include MATLAB built-in functions and functions that are created by the user.
- The limits is a vector with two elements that specify the domain of x [xmin,xmax], or a vector with four elements that specifies the domain of x and the limits of the y-axis [xmin,xmax,ymin,ymax].
- The line specifiers are the same as in the plot command.

Fplot example

For example, a plot of the function $y = x^2 + 4\sin(2x) - 1$ for $-3 \le x \le 3$ can be created with the `fplot` command by typing:

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

in the Command Window. The figure that is obtained in the Figure Window is shown in Figure 5-6.

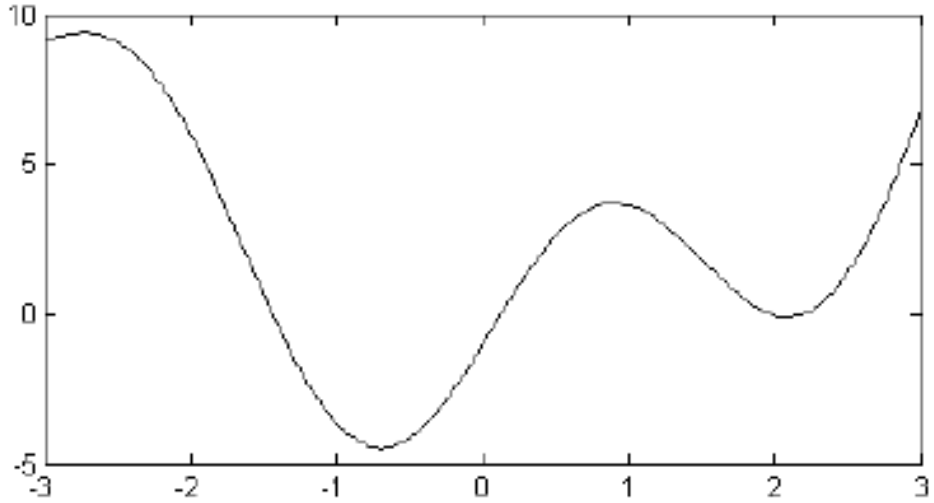


Figure 5-6: A plot of the function $y = x^2 + 4\sin(2x) - 1$.

Multiple plots (1)

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

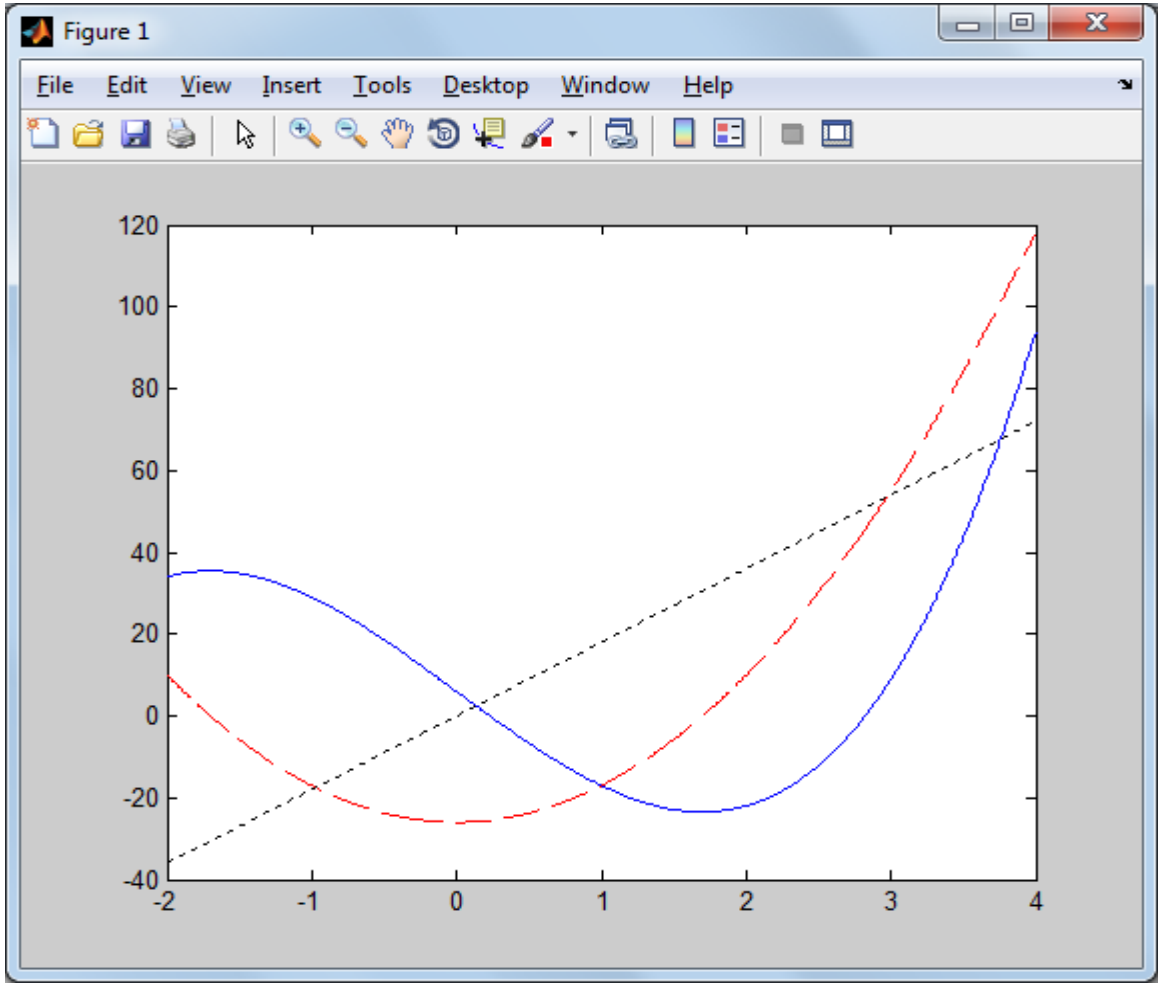
  plot(x,y,u,v,t,h)

  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 so that they can be identified. 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.

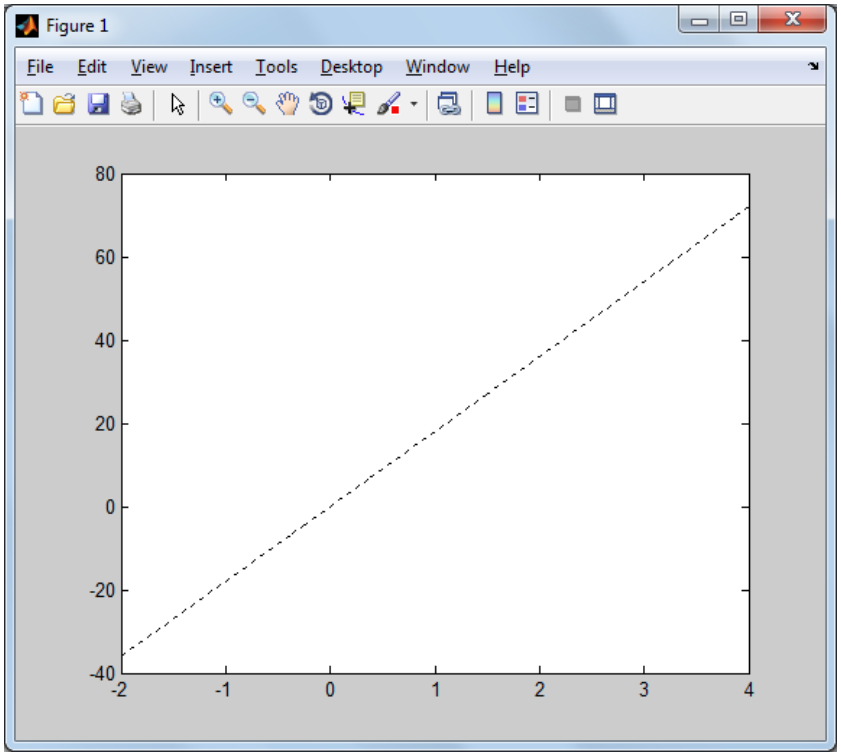
```
x=[-2:0.01:4];
y=3*x.^3-26*x+6;
yd=9*x.^2-26;
ydd=18*x;
plot(x,y,'-b', x,yd,'--r', x,ydd,':k')
```



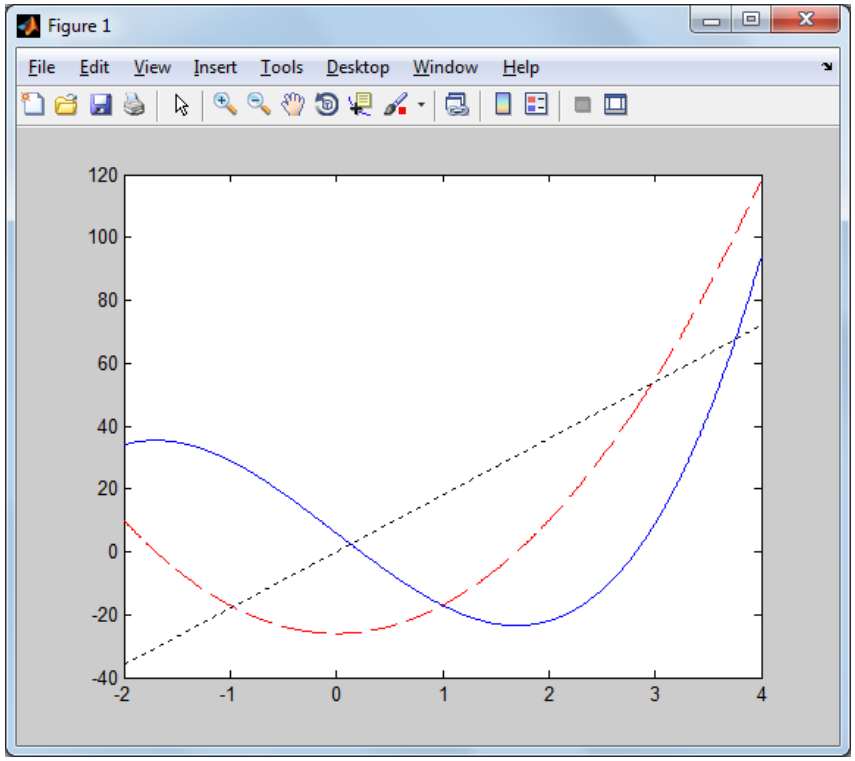
Multiple plots (2)

- To plot several graphs using the hold on, hold off commands, one graph is plotted first with the plot command. Then the hold on command is typed. This keeps the Figure Window with the first plot open, including the axis properties and formatting if any was done. Additional graphs can be added with plot commands that are typed next. Each plot command creates a graph that is added to that figure. The hold off command stops this process. It returns MATLAB to the default mode in which the plot command erases the previous plot and resets the axis properties.

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




```
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
```



line

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)
```

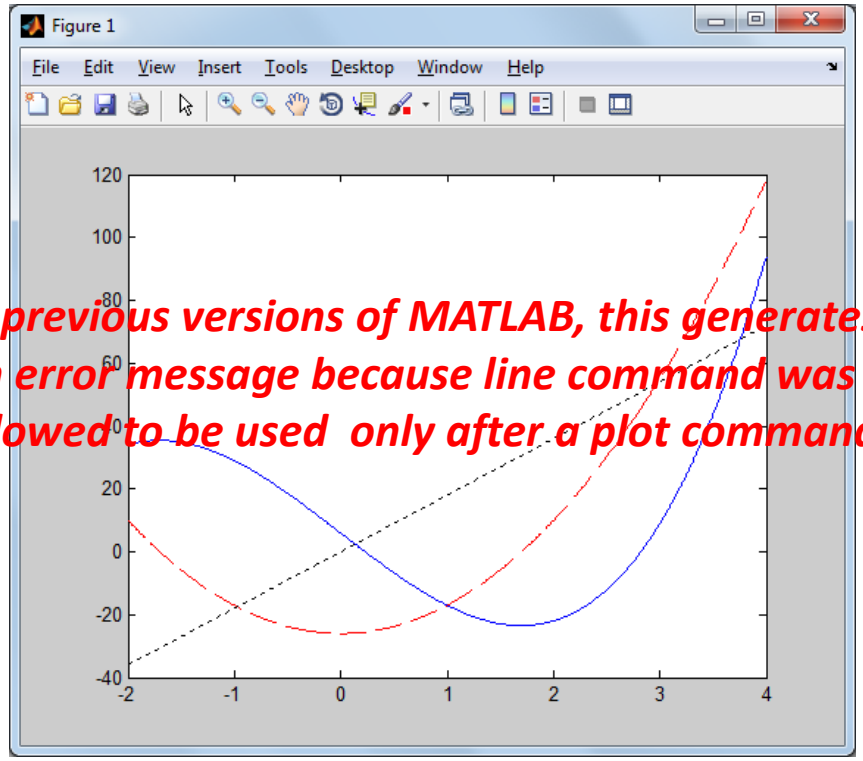
(Optional) Properties with values that can be used to specify the line style, color, and width, marker type, size, and edge and fill colors.

The format of the `line` command is almost the same as the `plot` command (see Section 5.1). 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 and if none are entered MATLAB uses default properties and values. For example, the command:

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

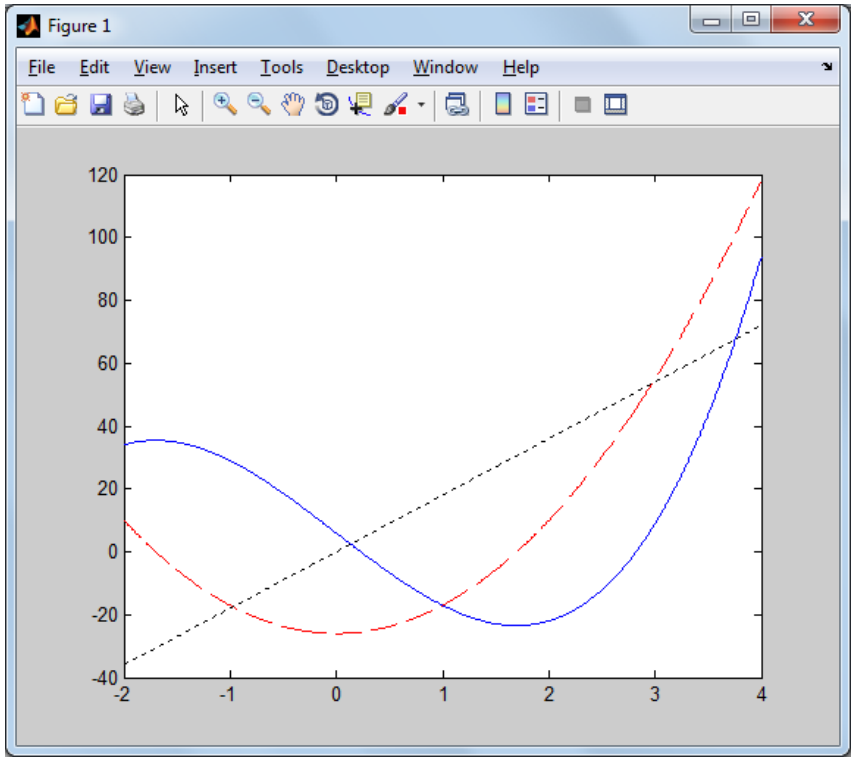
will add a dashed red line with circular markers to a plot that already exists.

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



in previous versions of MATLAB, this generates an error message because line command was allowed to be used only after a plot command.

```
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 (1)

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

The `xlabel` and `ylabel` commands:

Labels can be placed next to the axes with the `xlabel` and `ylabel` commands 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')
```

Formatting a plot (2)

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` (according to the axes of the figure). The `gtext` command places the text at a position specified by the user. When the command is executed, the Figure Window opens and the user specifies the position with the mouse.

Formatting a plot (3)

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 that the graphs were created. The `pos` is an optional number that specifies where in the figure the legend is 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 in plots

Modifier	Effect
\bf	bold font.
\it	italic style.
\rm	normal font.

Modifier	Effect
\fontname{fontname}	specified font is used.
\fontsize{fontsize}	specified font size is used.

Subscript by typing _ (the underscore) in front of the character or characters inside braces { }

Superscript by typing ^ in front of the character or characters inside braces { }

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

Formatting figures

- The ***axis*** command can be used to change the range and the appearance of the axes.

`axis([xmin,xmax,ymin,ymax])`	Sets the limits of both the `x` and `y` axes (`xmin`, `xmax`, `ymin`, and `ymax` are numbers).
`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.

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

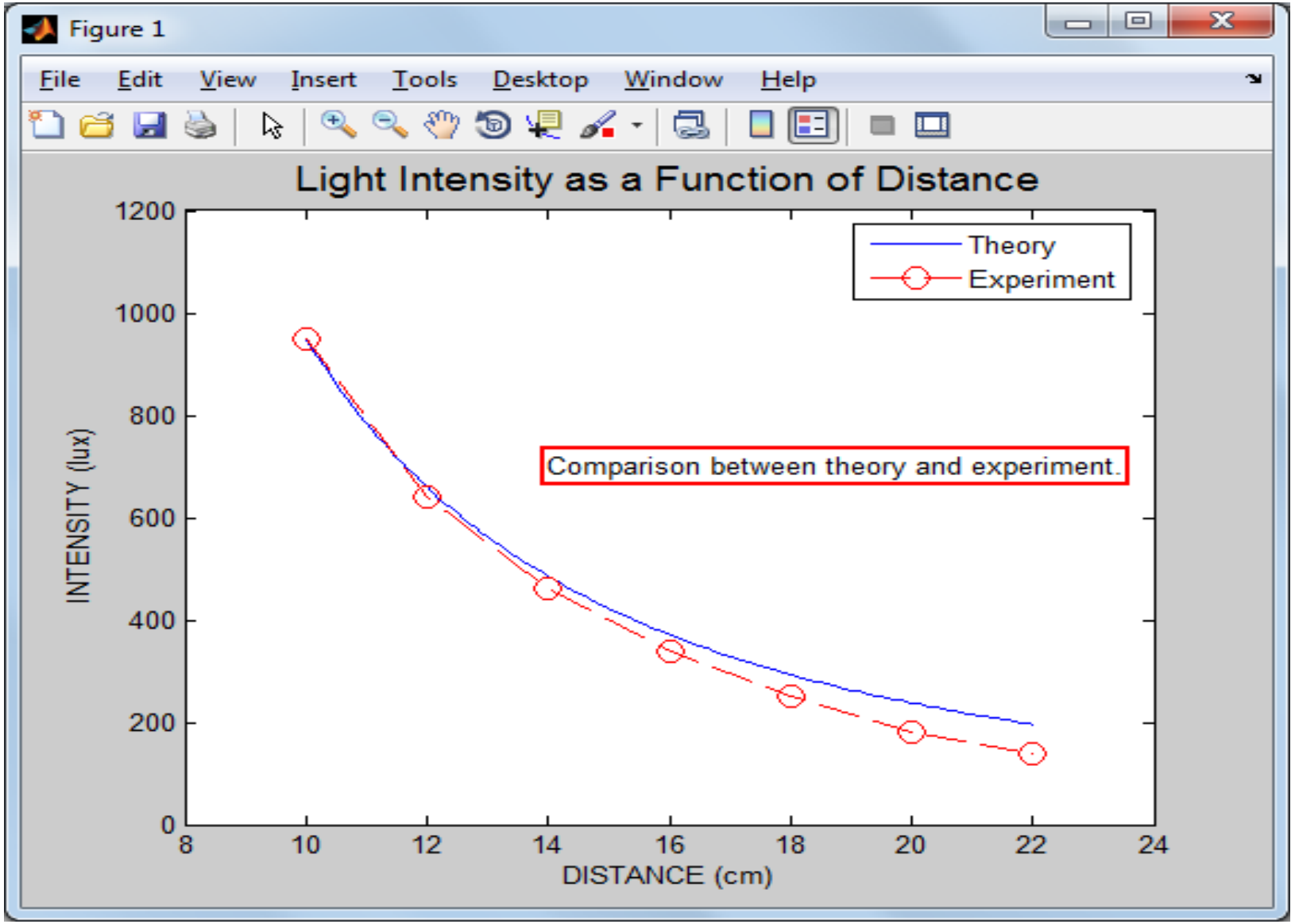
Formatting example

```
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 example (cont'd)

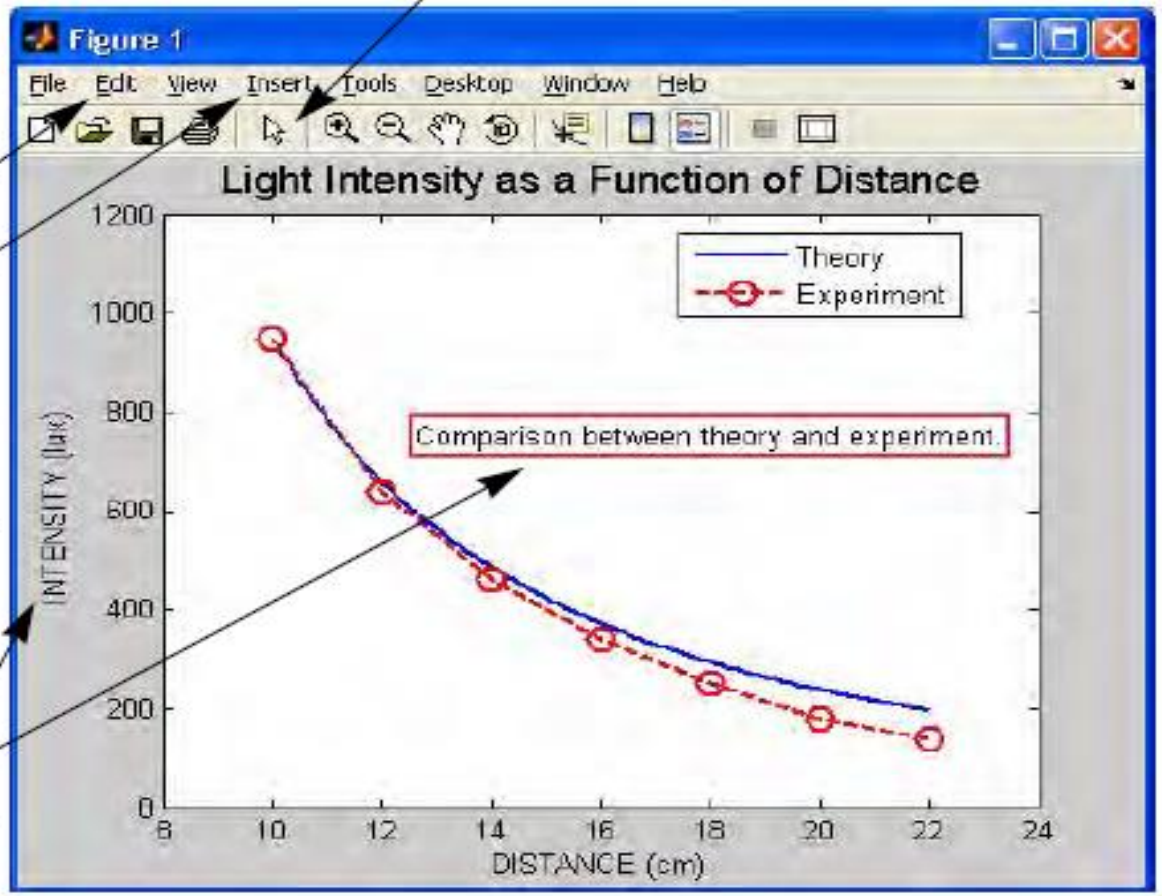


Formatting a plot using the plot editor

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 labels, legends and other objects by clicking on the object and dragging.

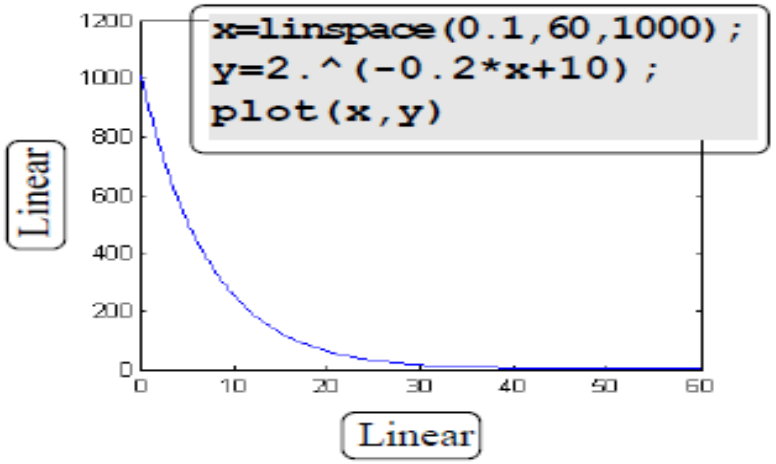


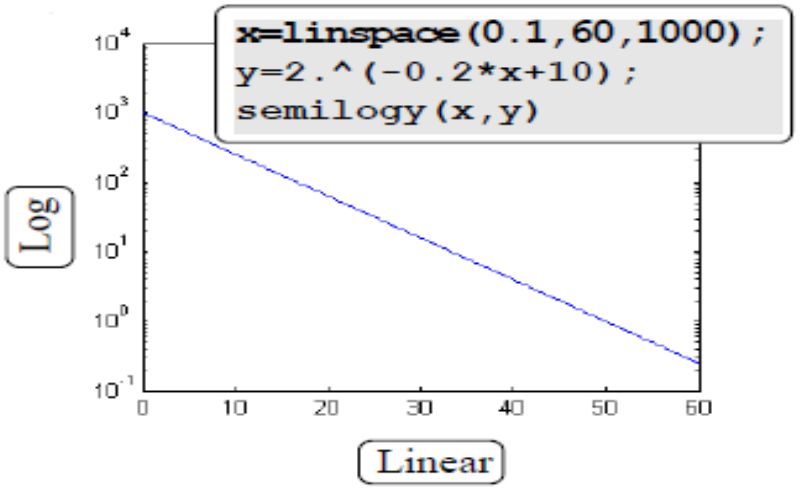
Plots with logarithmic axes

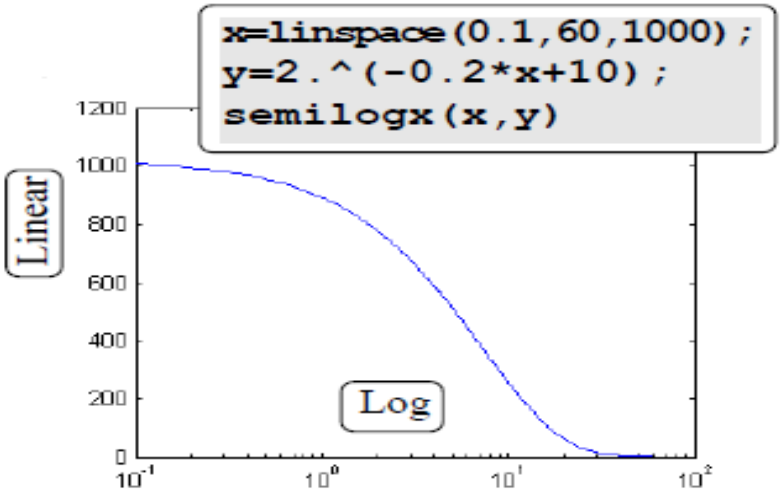
- Many science and engineering applications require plots in which one or both axes have a logarithmic (log) scale. Log scales provide means for presenting data over a wide range of values. It also provides a tool for identifying characteristics of data and possible forms of mathematical relationships that can be appropriate for modeling the data.
- ***semilogy(x,y)*** Plots y versus x with a log (base 10) scale for the y axis and linear scale for the x axis.
- ***semilogx(x,y)*** Plots y versus x with a log (base 10) scale for the x axis and linear scale for the y axis.
- ***loglog(x,y)*** Plots y versus x with a log (base 10) scale for both axes.

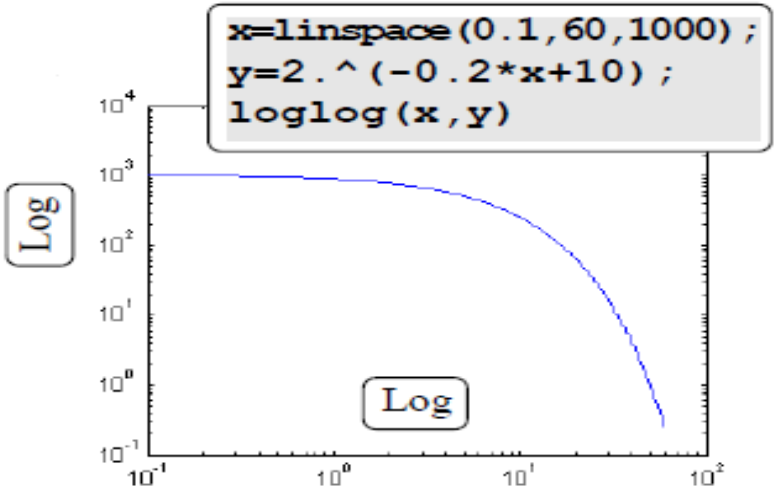
Logarithmic plot example

As an example, plot the function $y = 2^{(-0.2x+10)}$ for $0.1 \le x \le 60$.









Plots with error bars

- Experimental data that is measured and then displayed in plots frequently contains error and scatter. Even data that is generated by computational models includes error or uncertainty that depends on the accuracy of the input parameters and the assumptions in the mathematical models that are used. One method of plotting data that displays the error, or uncertainty, is by using error bars.
- Plots with error bars can be done in MATLAB with the ***errorbar*** command. Two forms of the command, one for making plots with symmetric error bars (with respect to the value of the data point), and the other for nonsymmetric error bars, at each point are presented.

Error bars

```
errorbar(x,y,e)
```

Vectors with horizontal and vertical coordinates of each point.

Vector with the values of the error at each point.

- The length of the three vectors `x`, `y`, and `e` must be the same.
- The length of the error bars is twice the value of `e`. At each point the error bar extends from `y(i)-e(i)` to `y(i)+e(i)`.

```
errorbar(x,y,d,u)
```

Vectors with horizontal and vertical coordinates of each point.

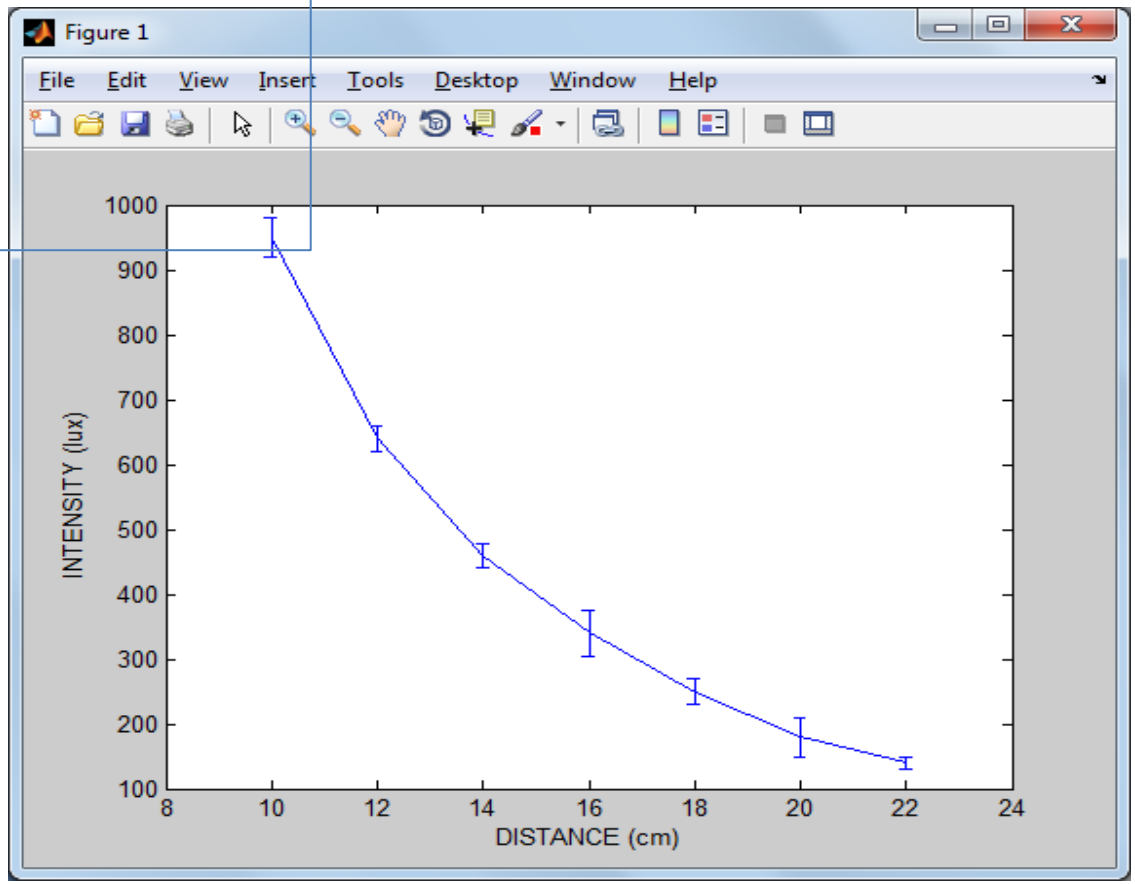
Vector with the upper-bound values of the error at each point.

Vector with the lower-bound values of the error at each point.

- The length of the three vectors `x`, `y`, `d`, and `u` must be the same.
- At each point the error bar extend from `y(i)-d(i)` to `y(i)+u(i)`.

Error bar example

```
xd=[10:2:22];
yd=[950 640 460 340 250 180 140];
ydErr=[30 20 18 35 20 30 10]
errorbar(xd,yd,ydErr)
xlabel('DISTANCE (cm)')
ylabel('INTENSITY (lux)')
```

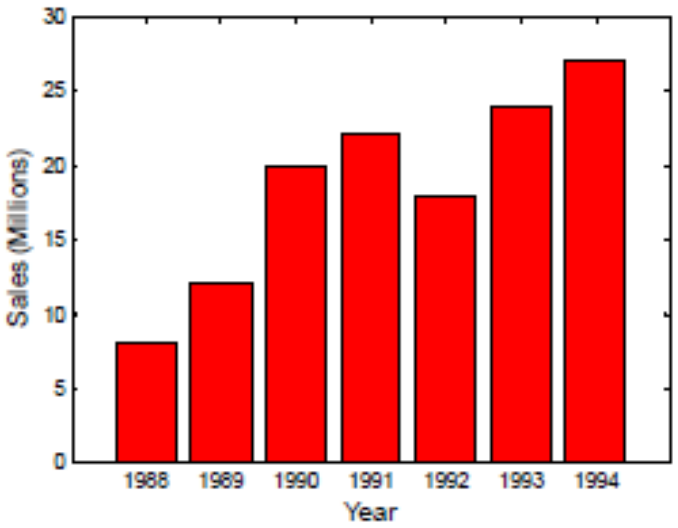


Special plots (1)

Vertical Bar Plot

Function format:

```
bar(x,y)
```



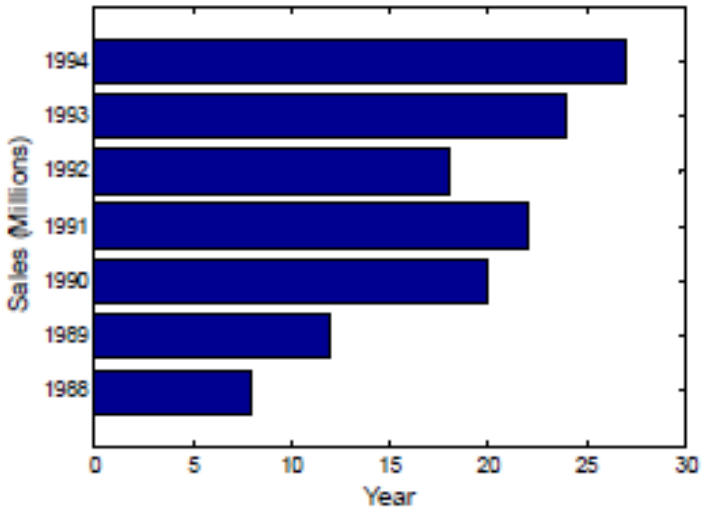

```
yr=[1988:1994];
sle=[8 12 20 22 18 24 27];
bar(yr,sle,'r')
xlabel('Year')
ylabel('Sales (Millions)')
```

The bars are in red.

Horizontal Bar Plot

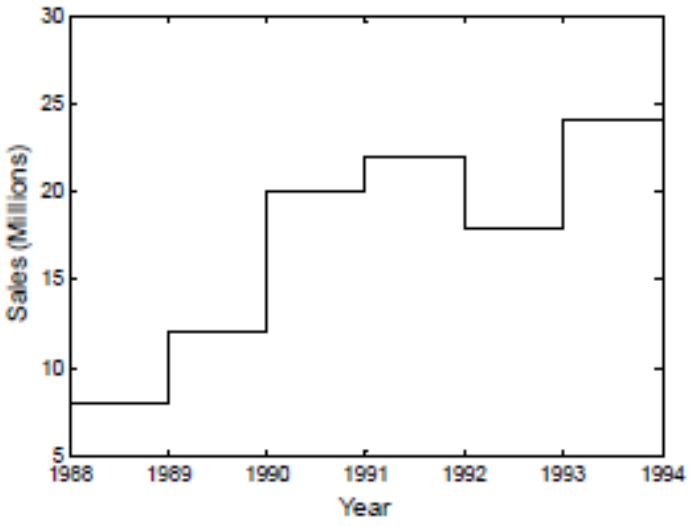
Function format:

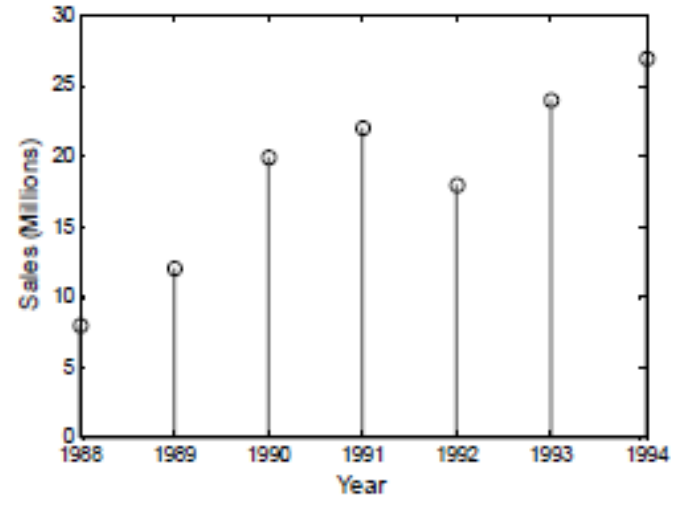
```
barh(x,y)
```




```
yr=[1988:1994];
sle=[8 12 20 22 18 24 27];
barh(yr,sle)
xlabel('Sales (Millions)')
ylabel('Year')
```

Special plots (2)

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

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

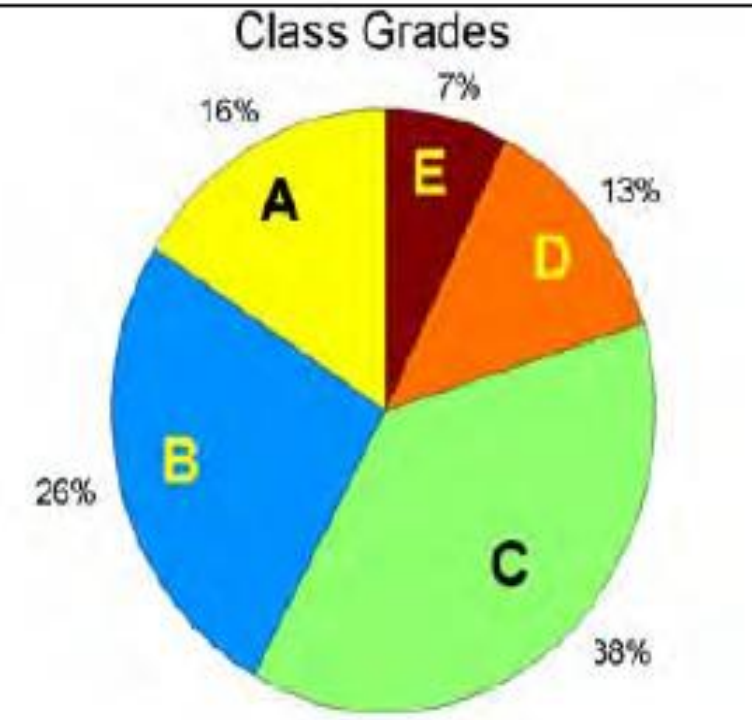
Special plots (3)

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.

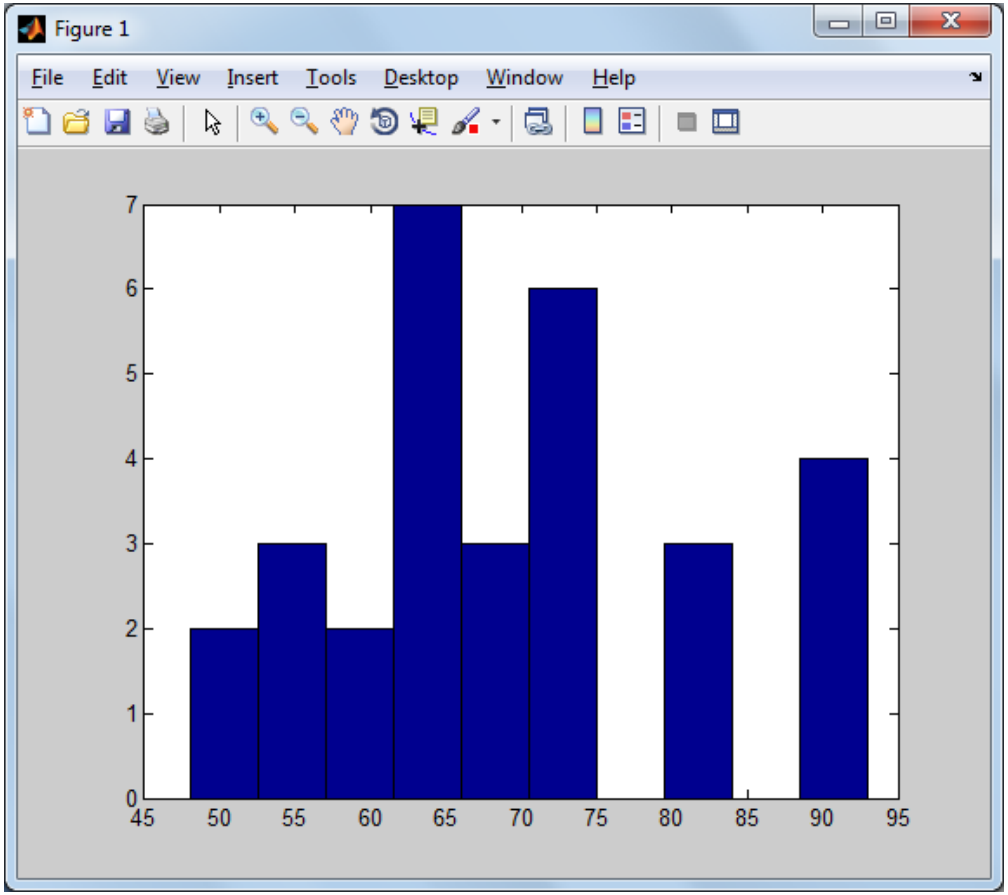
Histogram

- Histograms are plots that show the distribution of data. The overall range of a given set of data points is divided to smaller subranges (bins), and the histogram shows how many data points are in each bin. Histograms are created by using the ***hist*** command.

Histogram (cont'd)

```
>> 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)
```

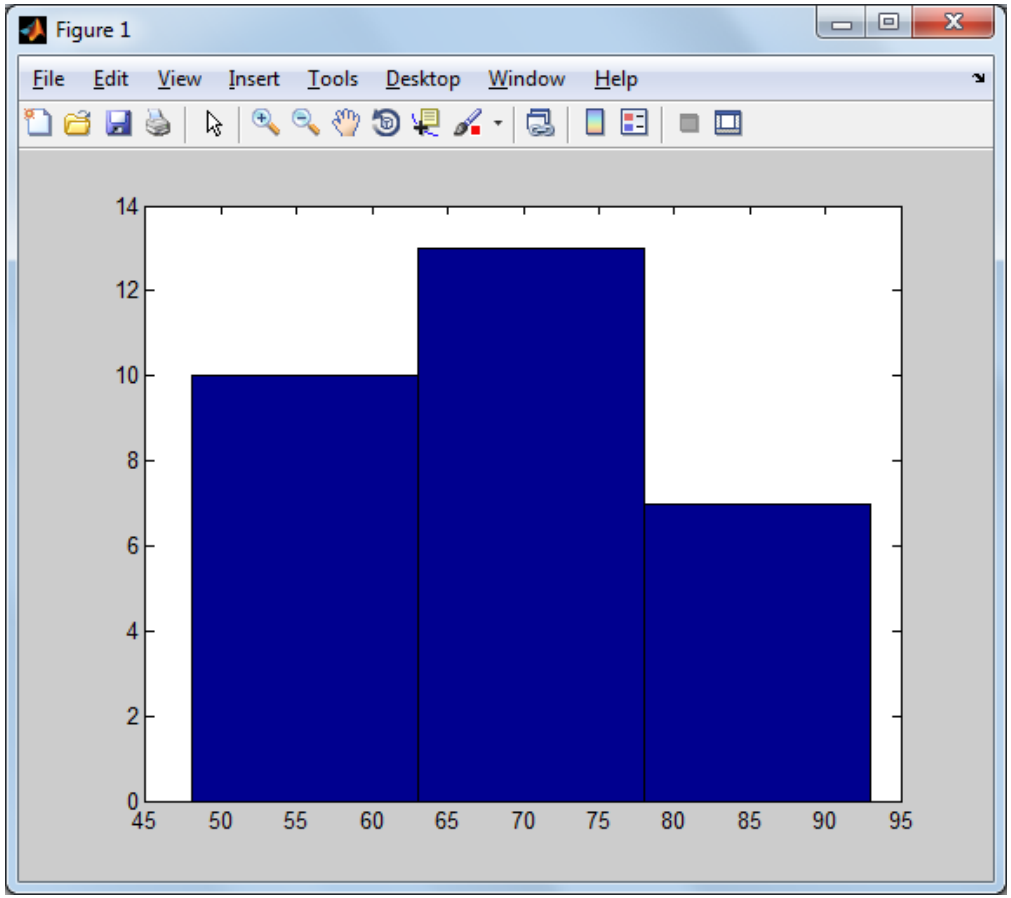
MATLAB divides the range of the data points into 10 equally spaced subranges (bins), and then plots the number of data points in each bin.



Histogram (cont'd)

```
>> 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,3)
```

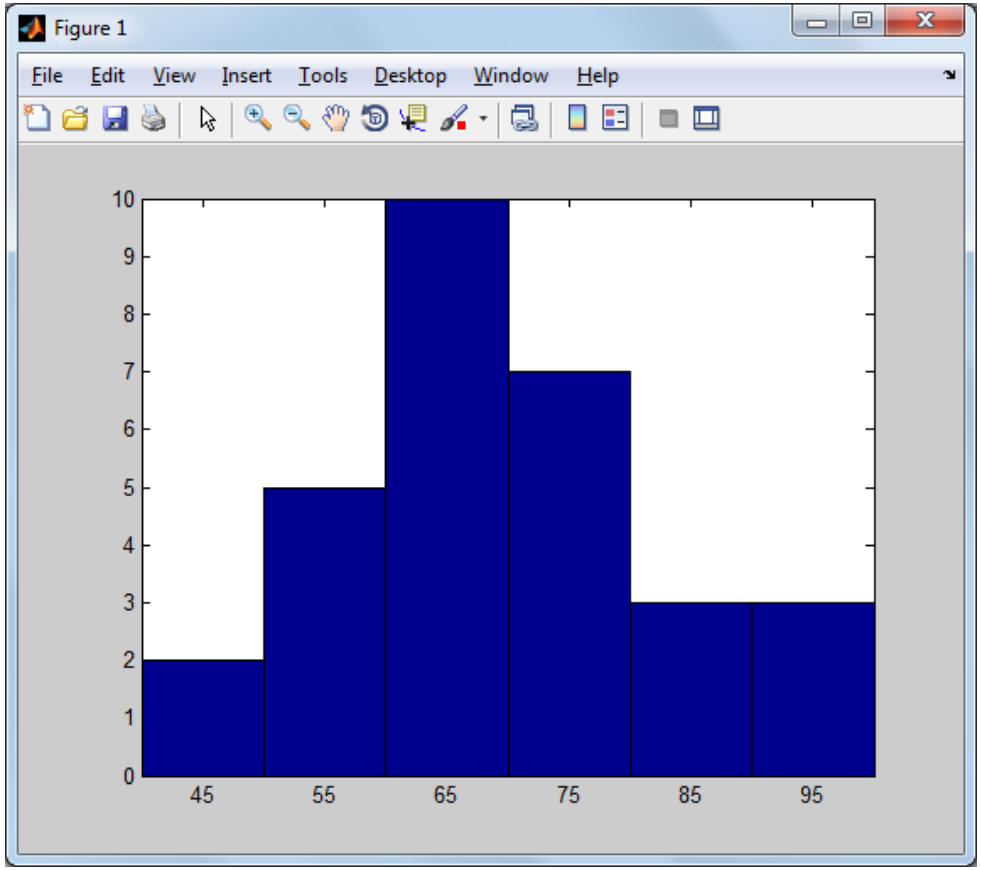
MATLAB divides the range of the data points into 3 equally spaced subranges (bins), and then plots the number of data points in each bin.



Histogram (cont'd)

```
>> 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];
>> x=[45:10:95];
>> hist(y,x)
```

MATLAB divides the range of the data points into 6 equally spaced subranges (bins), and then plots the number of data points in each bin. The center values of each bin is given by the vector of x.



Histogram (cont'd)

The `hist` command can be used with options that provide numerical output in addition to plotting a histogram. An output of the number of data points in each bin can be obtained with one of the following commands:

```
n=hist(y)
```

```
n=hist(y,nbins)
```

```
n=hist(y,x)
```

The output `n` is a vector. The number of elements in `n` is equal to the number of bins and the value of each element of `n` is the number of data points (frequency count) in the corresponding bin.

An additional optional numerical output is the location of the bins. This output can be obtained with one of the following commands:

```
[n xout]=hist(y)
```

```
[n xout]=hist(y,nbins)
```

`xout` is a vector in which the value of each element is the location of the center of the corresponding bin.

Histogram (cont'd)

```
>> n = hist(y)

n =
   2   3   2   7   3   6   0   3   0   4
```

The vector n shows how may elements are in each bin.

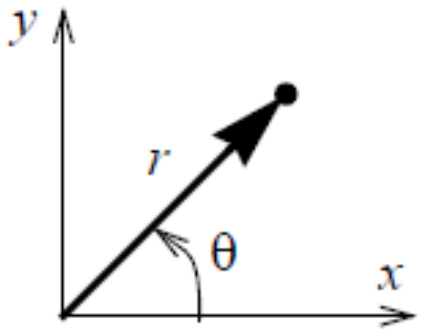
The vector n shows that the first bin has 2 data points, the second bin has 3 data points, and so on.

```
>> [n xout]=hist(y)

n =
   2   3   2   7   3   6   0   3   0   4

xout =
   50.2500   54.7500   59.2500   63.7500   68.2500   72.7500
77.2500   81.7500   86.2500   90.7500
```

The vector xout shows that the center of the first bin is at 50.25, the center of the second bin is at 54.75 and so on.

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 solution of science and engineering problems. The `polar` command is used to plot functions in polar coordinates. The command has the form:




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

Vector

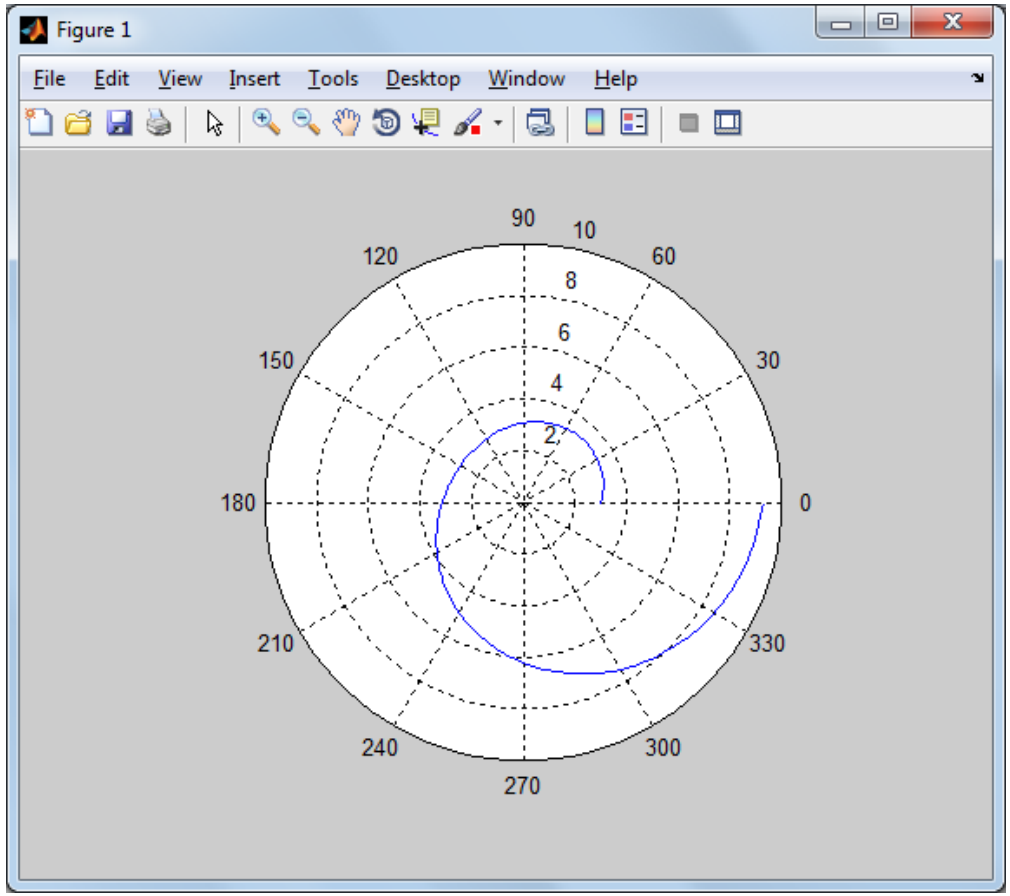
Vector

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

where `theta` and `radius` are vectors whose elements define the coordinates of the points to be plotted. The `polar` command plots the points and draws the polar grid. The line specifiers are the same as in the `plot` command.

Polar plots (cont'd)

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

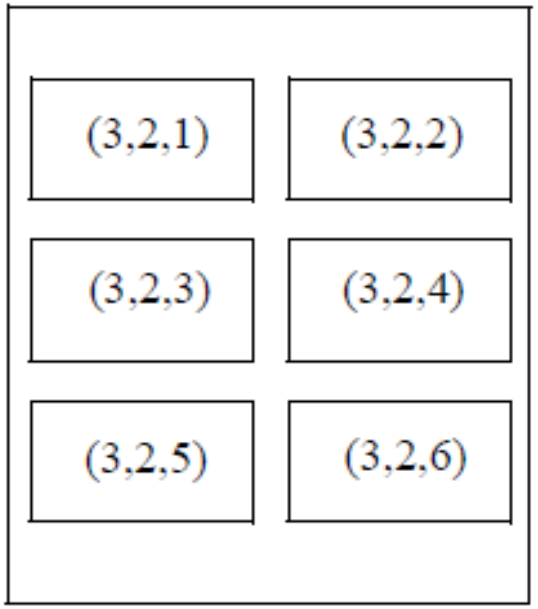


Multiple plots on the same figure

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

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

The command divides the Figure Window (page when printed) into $m \times n$ rectangular subplots where plots will be created. The subplots are arranged like elements in a $m \times n$ matrix where each element is a subplot. The subplots are numbered from 1 through $m \cdot n$. The upper left is 1 and the lower right is the number $m \cdot n$. The numbers increase from left to right within a row, from the first row to the last. The command `subplot(m,n,p)` makes the subplot `p` current. This means that the next plot command (and any formatting commands) will create a plot (with the corresponding format) in this subplot. For example, the command `subplot(3,2,1)` creates 6 areas arranged in 3 rows and 2 columns as shown, and makes the upper left subplot current.



Multiple figures

- When the plot or any other command that generates a plot is executed, the Figure Window opens (if not already open) and displays the plot. MATLAB labels the Figure Window as Figure 1 (see the top left corner of the Figure Window that is displayed in previous examples). If the Figure Window is already open when the plot or any other command that makes a plot is executed, a new plot is displayed in the Figure Window that is already open (replacing the existing plot). Commands that format plots are applied to the plot in the Figure Window that is open.

Multiple figures (cont'd)

- It is possible, however, to open additional Figure Windows and have several of them open (with plots) at the same time. This is done by typing the command figure. Every time the command figure is entered, MATLAB opens a new Figure Window. If a command that creates a plot is entered after a figure command, MATLAB generates and displays the new plot in the last Figure Window that was opened, which is called the active or current window. MATLAB labels the new Figure Windows successively; i.e., Figure 2, Figure 3, and so on.

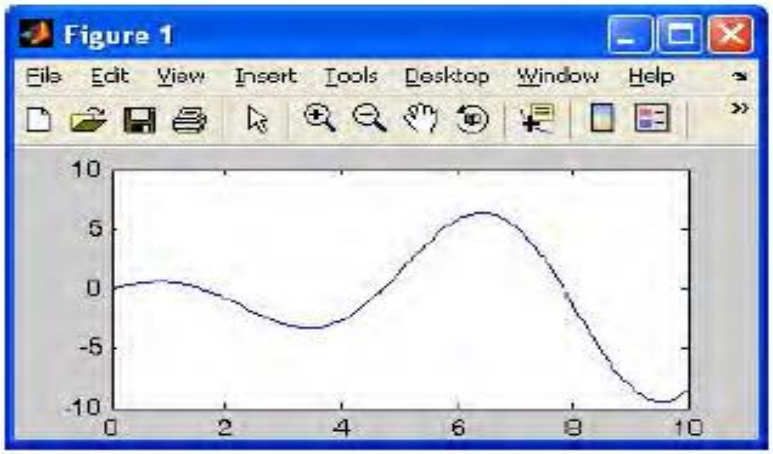
Multiple figures (cont'd)

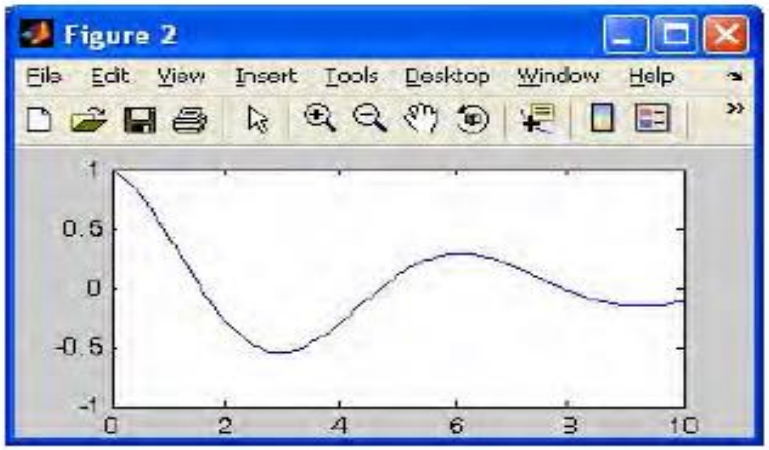
```
>> fplot('x*cos(x)',[0,10])
>> figure
>> fplot('exp(-0.2*x)*cos(x)',[0,10])
```

Plot displayed in Figure 1 Window.

Figure 2 Window opens.

Plot displayed in Figure 2 Window.





The `figure` command can also have an input argument that is a number (integer) `figure(n)`. The number corresponds to the number of a corresponding Figure Window. When the command is executed, Figure Window number `n` becomes the active Figure Window (if a Figure Window with this number does not exist, a new window with this number opens).

Figure Windows can be closed with the `close` command. Several forms of the command are:

`close` closes the active Figure Window.
`close(n)` closes the nth Figure Window.
`close all` closes all Figure Windows that are open.

Laboratory Session

Do both examples in this presentation and sample applications in Chapter 5 of the textbook.

Homework #7

Not later than the next week:

Solve problems 1, 4, 8, 10, 13, 15, and 19 from the Chapter 5 of the textbook using Matlab.