BIL 108E
Introduction to Scientific and Engineering Computing

ASST. PROF. DR. İPEK AKIN

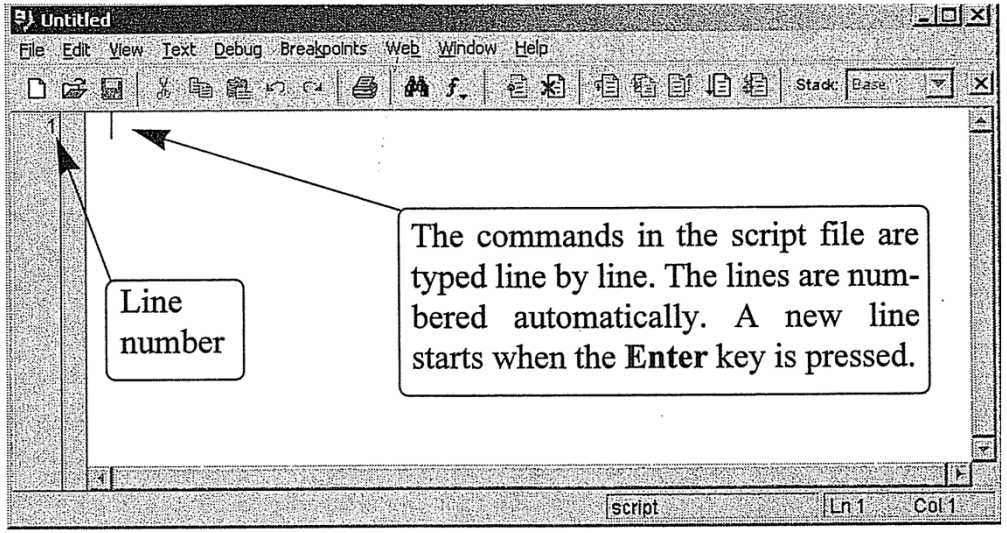
Script Files

INTRODUCTION

- Script file = M file (.m is used when they are saved)
- Sequence of MATLAB commands, called program
- When a script file runs, MATLAB executes the commands in the order they are written just as they were typed in the Command Window.
- Script files can be typed in any text editor and then pasted into the MATLAB editor.

Creating and Saving a Script File

- Use Editor/Debugger Window – opened from Command Window
- File – select New – select M-file



Creating and Saving a Script File

- Use Editor/Debugger Window – opened from Command Window
- File – select New – select M-file

$$F = \frac{Gm_1m_2}{r^2}$$ Newton's law of gravitation

```
>> G=6.67*10^(-11);
>> m1=10;
>> m2=100;
>> r=50;
```

Input to a Script File

When a script file is executed the variables that are used in calculations within the file must have assigned values:

The assignment of a variable can be done in 3 ways:

1) The variable is defined and assigned value in the script file

2) The variable is defined and assigned value in the Command Window

3) The variable is defined in the script file, but a specific value is entered in the Command Window when the script file is executed

1) The variable is defined and assigned value in the script file

The script file calculates the average points scored in 3 games.

```
% This script file calculates the average points scored in three games.
% The assignment of the values of the points is part of the script file.
game1 = 75;
game2 = 93;
game3 = 68;
ave_points = (game1 + game2 + game3)/3
```

The variables are assigned values within the script file.

```
>> Chapter4Example2

ave_points =
   78.6667
>>
```

The script file is executed by typing the name of the file.

The variable `ave_points` with its value is displayed in the Command Window.

2) The variable is defined and assigned value in the Command Window

```
% This script file calculates the average points scored in three games.
% The assignment of the values of the points to the variables
% game1, game2, and game3 is done in the Command Window.

ave_points = (game1+game2+game3)/3
```

The Command Window for running this file is:

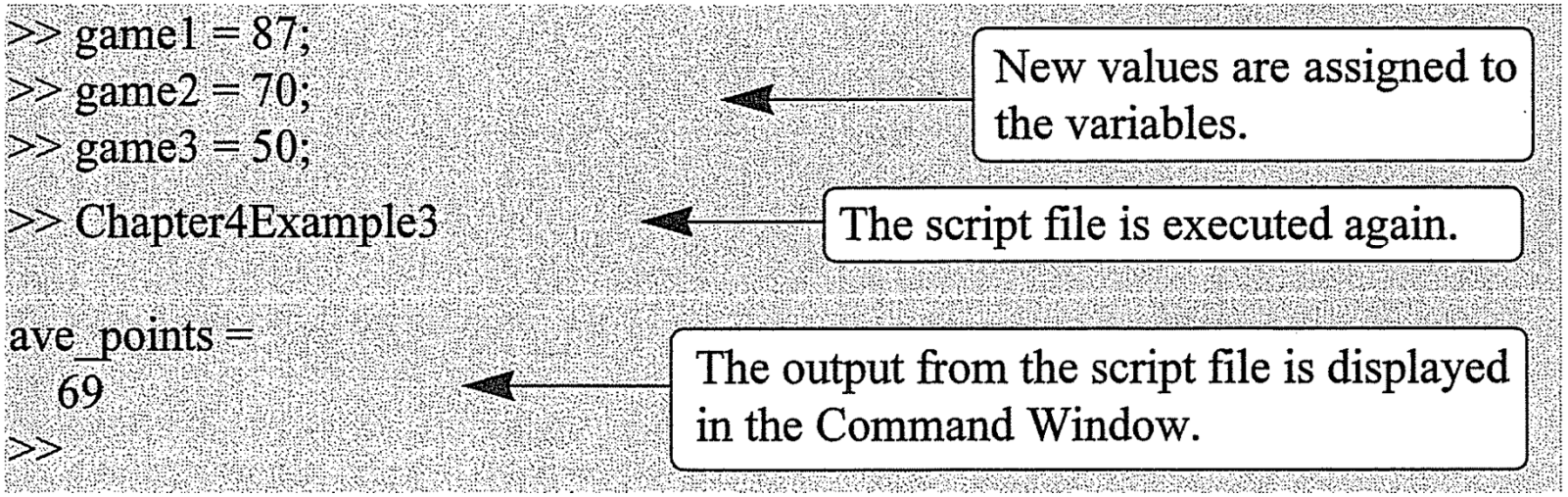
```
>> game1 = 67;
>> game2 = 90;
>> game3 = 81;
>> Chapter4Example3

ave_points =
   79.3333
```

The variables are assigned values in the Command Window.

The script file is executed.

The output from the script file is displayed in the Command Window.

2) The variable is defined and assigned value in the Command Window



3) The variable is defined in the script file, but a specific value is entered in the Command Window when the script file is executed

```
variable_name = input('string with a message that
                      is displayed in the Command Window')
```

Example

Consider the system of equations:

$x + 2y + 3z = 1$

$3x + 3y + 4z = 1$

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

Find the solution x to the system of equation.

Solution

- Use the MATLAB *editor* to create a file: **File** → **New** → **M-file**.
- Enter the following statements in the file:

```
A = [1 2 3; 3 3 4; 2 3 3];
b = [1; 1; 2];
x = A\b
```

- Save the file, for example, `example1.m`.

Run the file, in the command line, by typing:

```
>> example1
x    =
     -0.5000
      1.5000
     -0.5000
```

When execution completes, the variables (`A`, `b`, and `x`) remain in the workspace. To see a listing of them, enter `whos` at the command prompt.

There is another way to open the editor:
>> edit
or
>> edit filename.m
to open filename.m.

Output Commands

When a variable is assigned a value, or the name of a previously assigned variable is typed and the **Enter** key is pressed, MATLAB displays the variable and its value. This type of output is not displayed if a semicolon is typed at the end of the command.

MATLAB has several commands that can be used to generate displays.
The displays can be messages that provide information, numerical data, and plots.
Two commands that are frequently used to generate output are **disp** and **fprintf**.

disp Command

Used to display the elements of a variable without displaying the name of the variable, and to display text.

```
disp(name of a variable) or disp('text as string')
```

```
>> abc = [5  9  1;  7  2  4];
>> disp(abc)
     5     9     1
     7     2     4

>> disp('The problem has no solution.')

The problem has no solution.
>>
```

A 2×3 array is assigned to variable abc.

The disp command is used to display the abc array.

The array is displayed without its name.

The disp command is used to display a message.

```
% This script file calculates the average points scored in three games.
% The points from each game are assigned to the variables by
% using the input command.
% The disp command is used to display the output.

game1=input('Enter the points scored in the first game    ');
game2=input('Enter the points scored in the second game   ');
game3=input('Enter the points scored in the third game    ');
ave_points=(game1+game2+game3)/3;
disp(' ')
disp('The average of points scored in a game is:')
disp(' ')
disp(ave_points)
```

Display empty line.

Display text.

Display empty line.

Display the value of the variable `ave_points`.

When this file (saved as Chapter4Example5) is executed, the display in the Command Window

```
>> Chapter4Example5
Enter the points scored in the first game    89
Enter the points scored in the second game   60
Enter the points scored in the third game    82

The average of points scored in a game is:

   77
```

An empty line is displayed.

The text line is displayed.

An empty line is displayed.

The value of the variable `ave_points` is displayed.

fprintf Command

- Used to display output (text and data) on the screen or to save it to a file.
- Output can be formatted.

Using the `fprintf` command to display text:

To display text, the `fprintf` command has the form:

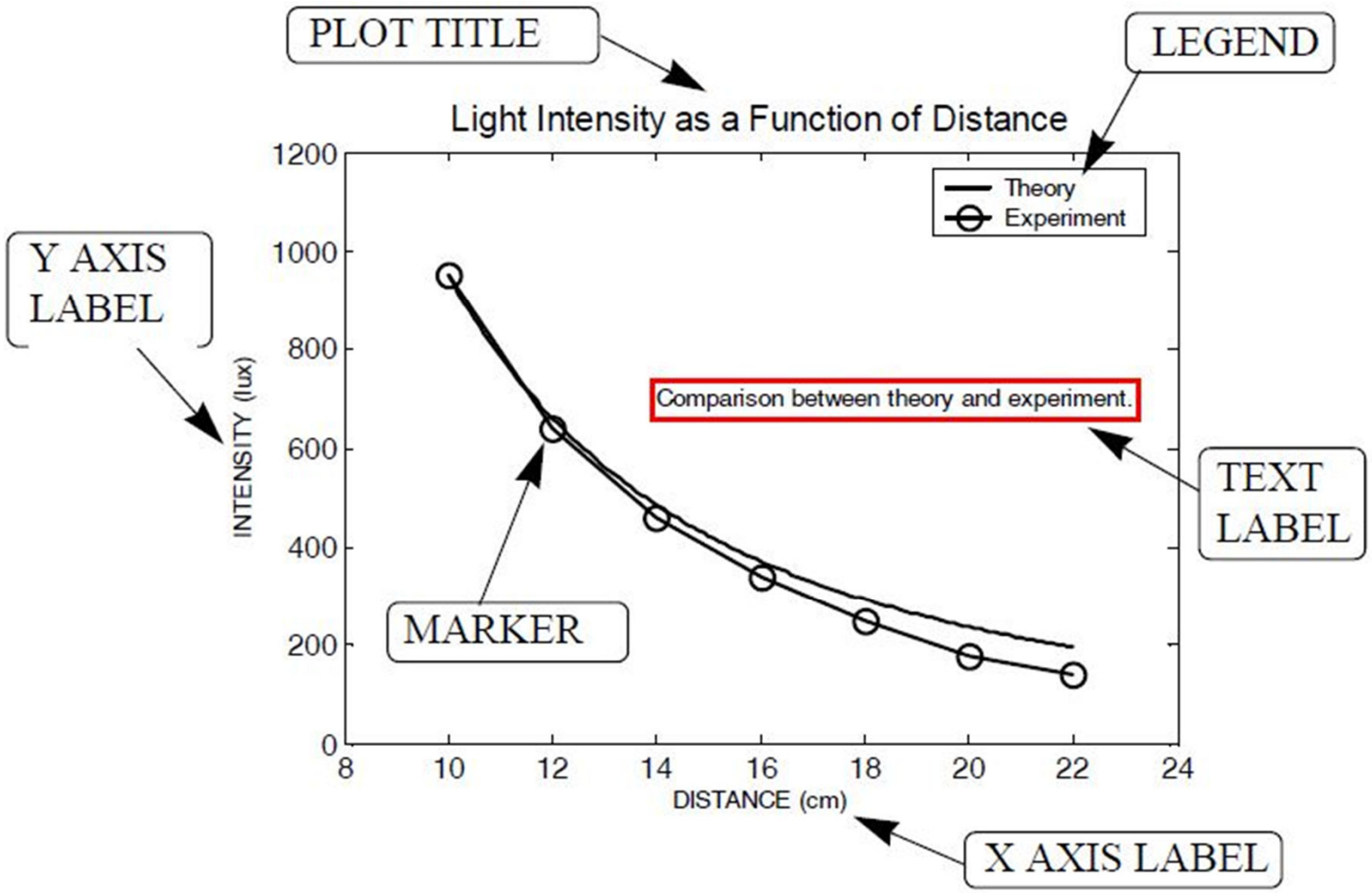
```
fprintf('text typed in as a string')
```

BIL 108E
Introduction to Scientific and Engineering Computing

ASST. PROF. DR. İPEK AKIN

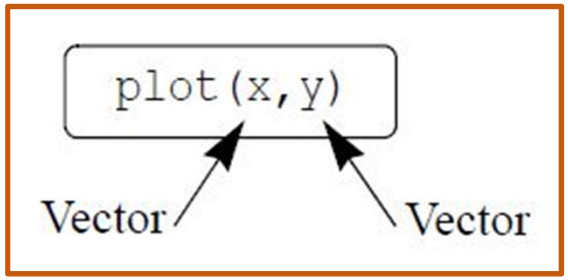
Two Dimensional Plots

Example of a formatted two-dimensional plot



plot command

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



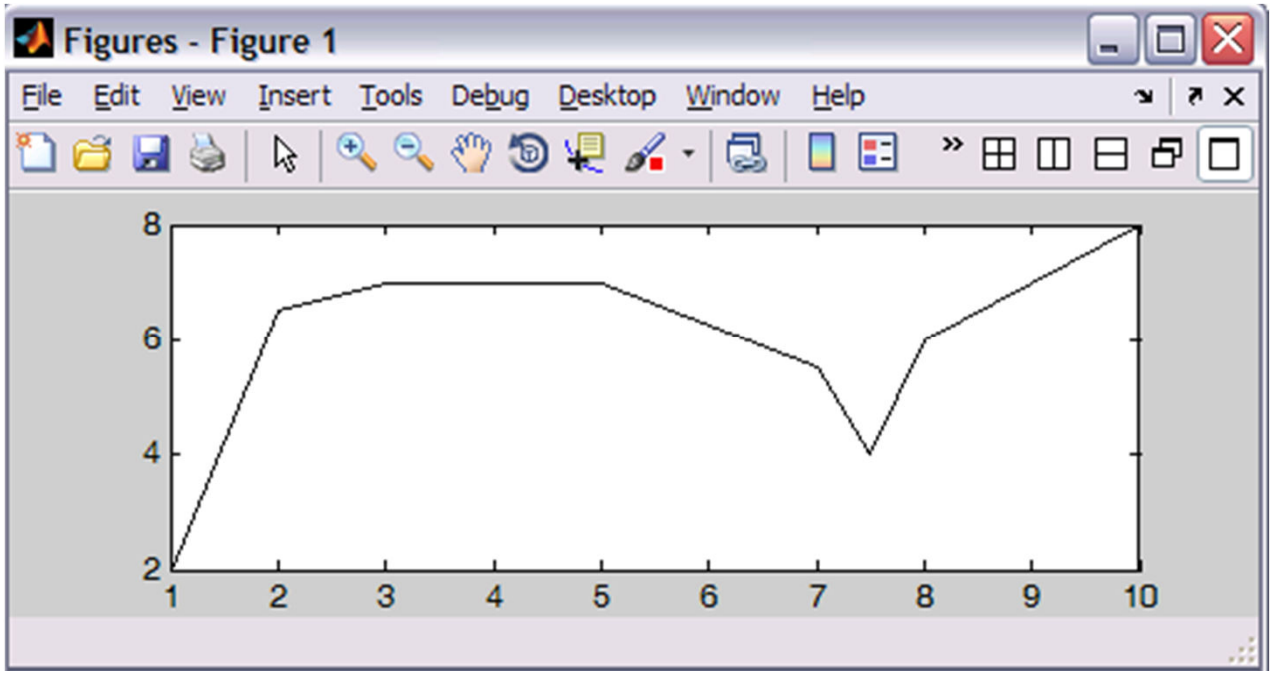
- x and y are each a vector (one-dimensional array).
- **The two vectors *must* have the same number of elements.**

x values on the abscissa (horizontal axis)

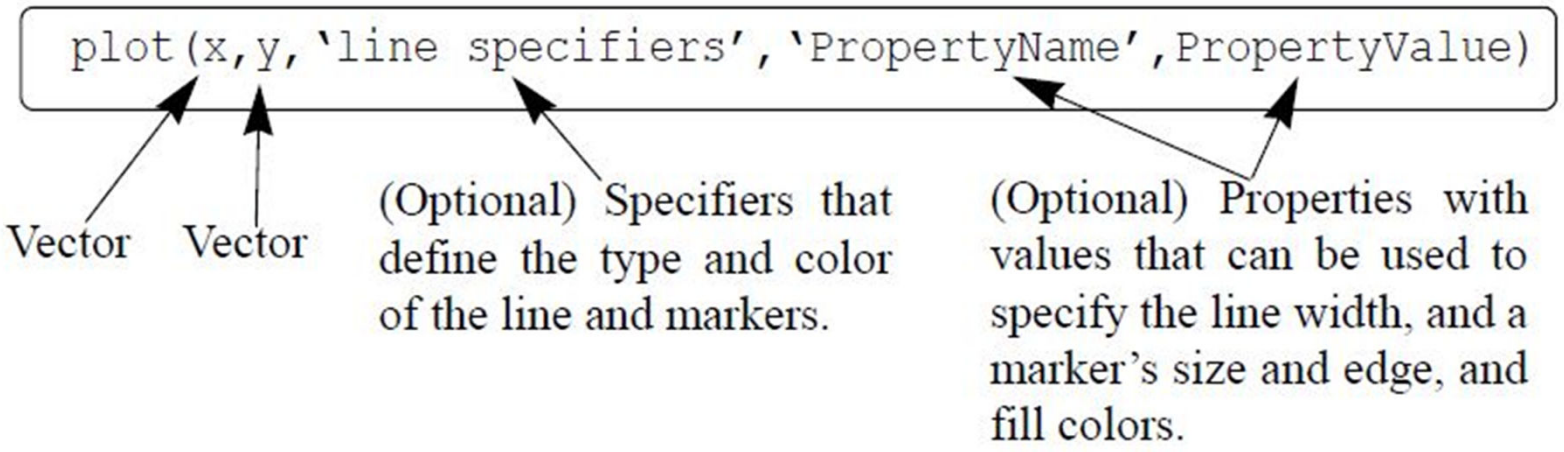
y values on the ordinate (vertical axis)

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 and the color and type of markers, if any are desired. With these options the command has the form:



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 specifiers are:

Line Color	Specifier
red	r
green	g
blue	b
cyan	c

Line Color	Specifier
magenta	m
yellow	y
black	k
white	w

Line Specifiers

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			

Notes About Using 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 (none, one, two, or all three types can be included in a command)

`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 ‘’,’’ and a value for the property, all inside the plot command.

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')
```

creates a plot that connects the points with a magenta solid line and circles as markers at the points.

The line width is 2 points and the size of the circle markers is 12 points.

The markers have a green edge line and yellow filling.

Note about line specifiers and properties

3 line specifiers

- Line Style
- Line Color
- Line Marker

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 specifier from the table above, typed as a string.
Marker	`marker` (or `Marker`)	Marker specifier from the table above, typed as a string.

Plot of Given Data

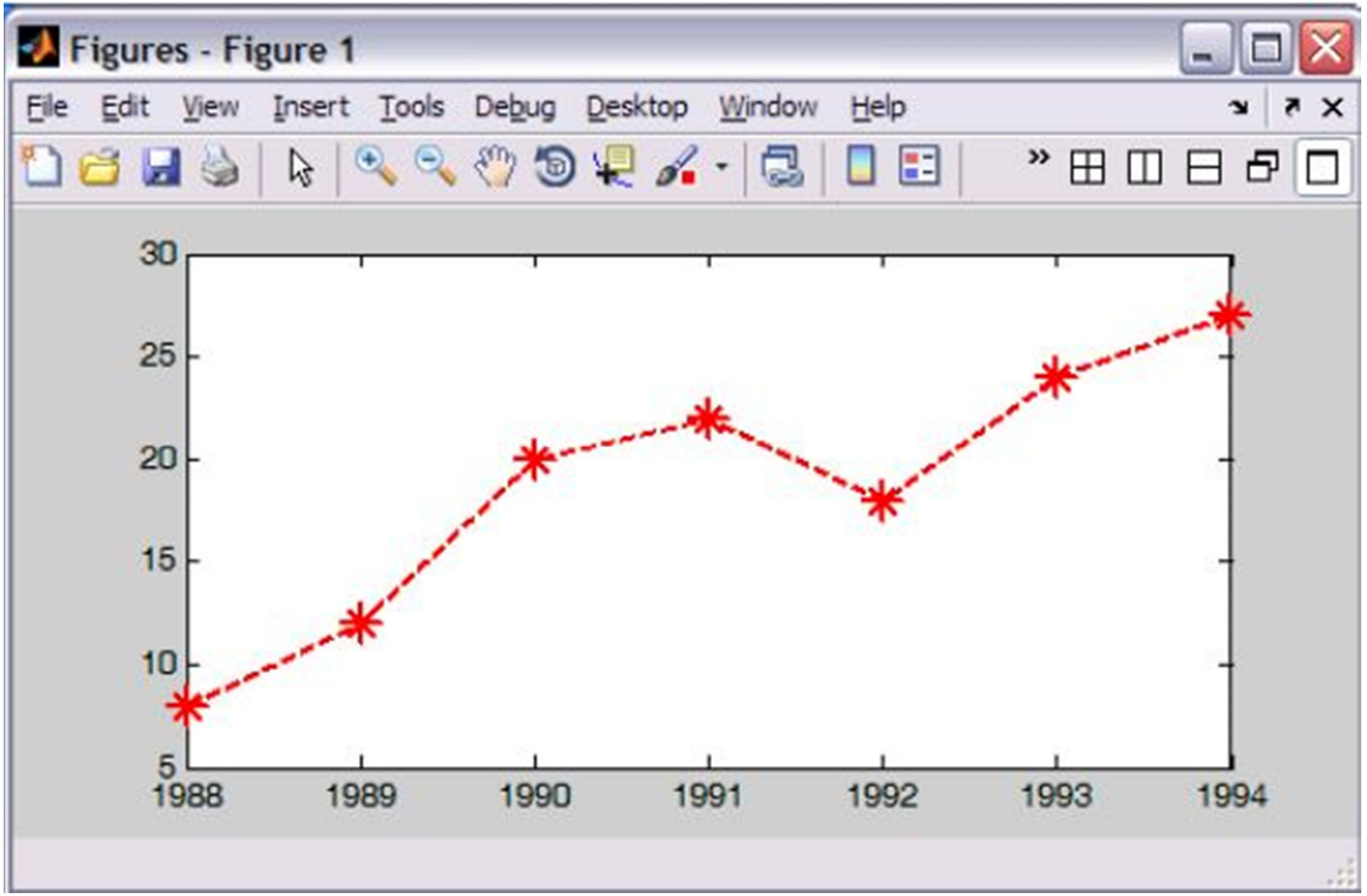
Year	1988	1989	1990	1991	1992	1993	1994
Sales (millions)	8	12	20	22	18	24	27

one vector → named yr
corresponding sales data → named sle

```
>> 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.


Figures - Figure 1
File Edit View Insert Tools Debug Desktop Window Help
30
25
20
15
10
5
1988
1989
1990
1991
1992
1993
1994

Plot of a Function

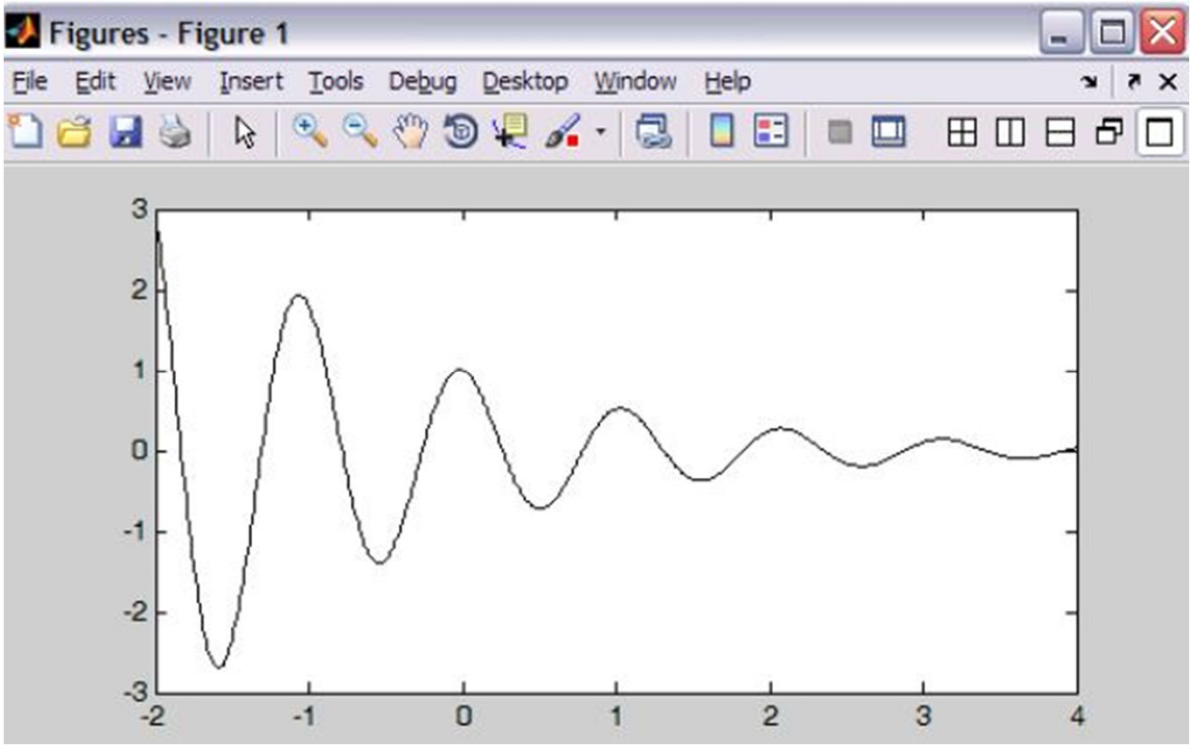
plot
fplot

```
%  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)
```

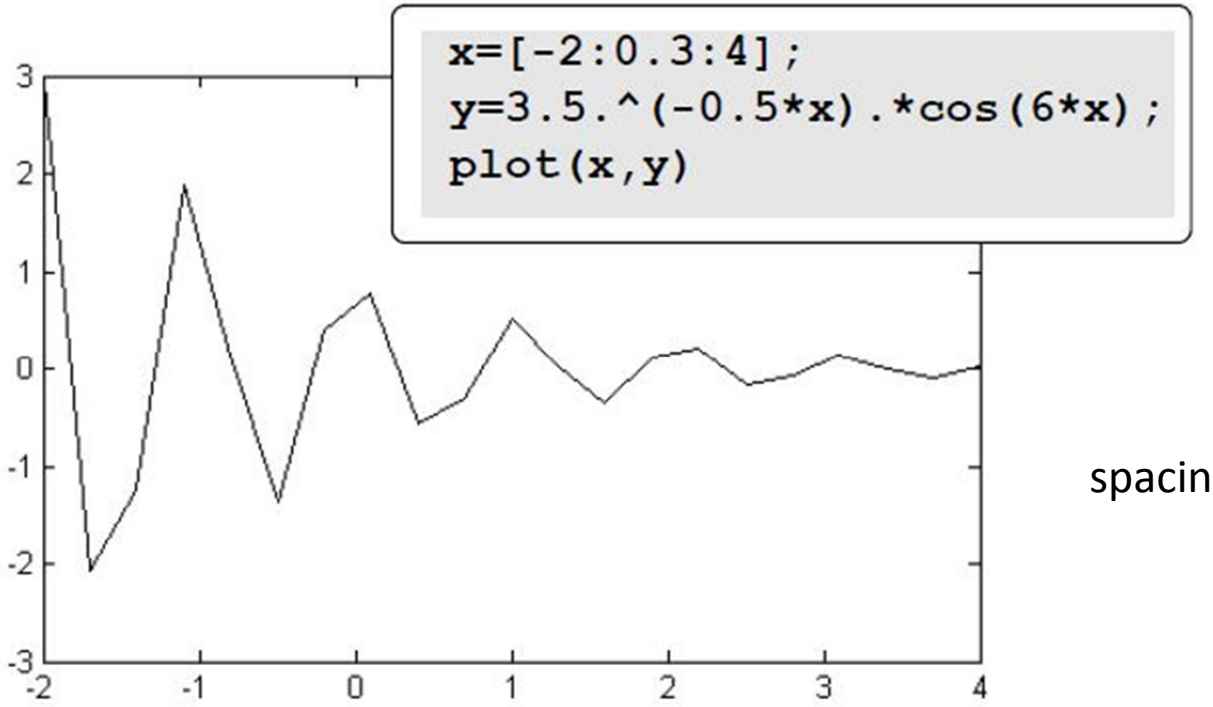
Create vector x with the domain of the function.

Create vector y with the function value at each x.

Plot y as a function of x.



spacing of 0.01



spacing of 0.3

gives a distorted picture of the function

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'

$$f(x) = 8x^2 + 5\cos(x)$$

```
'8*x^2+5*cos(x)'
```

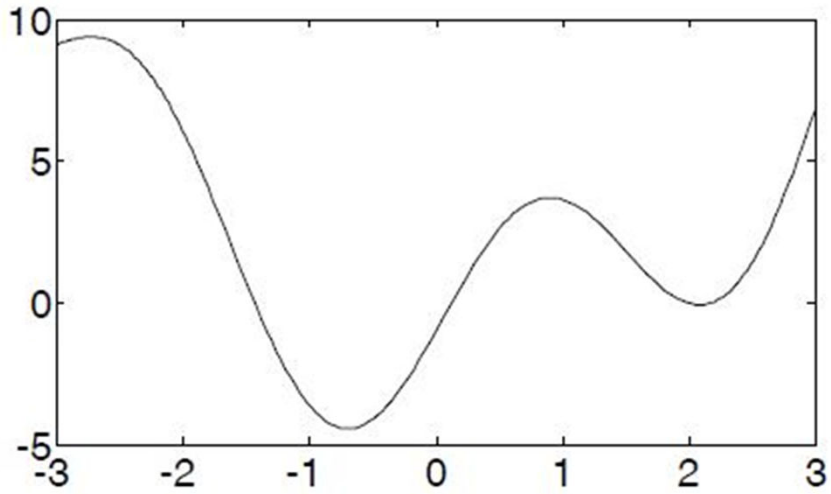
Can be typed function of any letter x can be t, z, etc.

The function cannot include previously defined variables. For example, in the function above it is not possible to assign 8 to a variable, and then use the variable when the function is typed in the `fplot` command.

limits: The limits argument 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].

line specifiers: The line specifiers are the same as in the plot command. 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])
```



Plotting Multiple Graphs in the Same Plot

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

1) using the plot command
2) using the hold on and hold off commands
3) 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.

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 (Plotting a function and its derivatives)

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

Solution

The first derivative of the function is: $y' = 9x^2 - 26$.

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

A script file that creates a vector x and calculates the values of y, y', and y'' is:

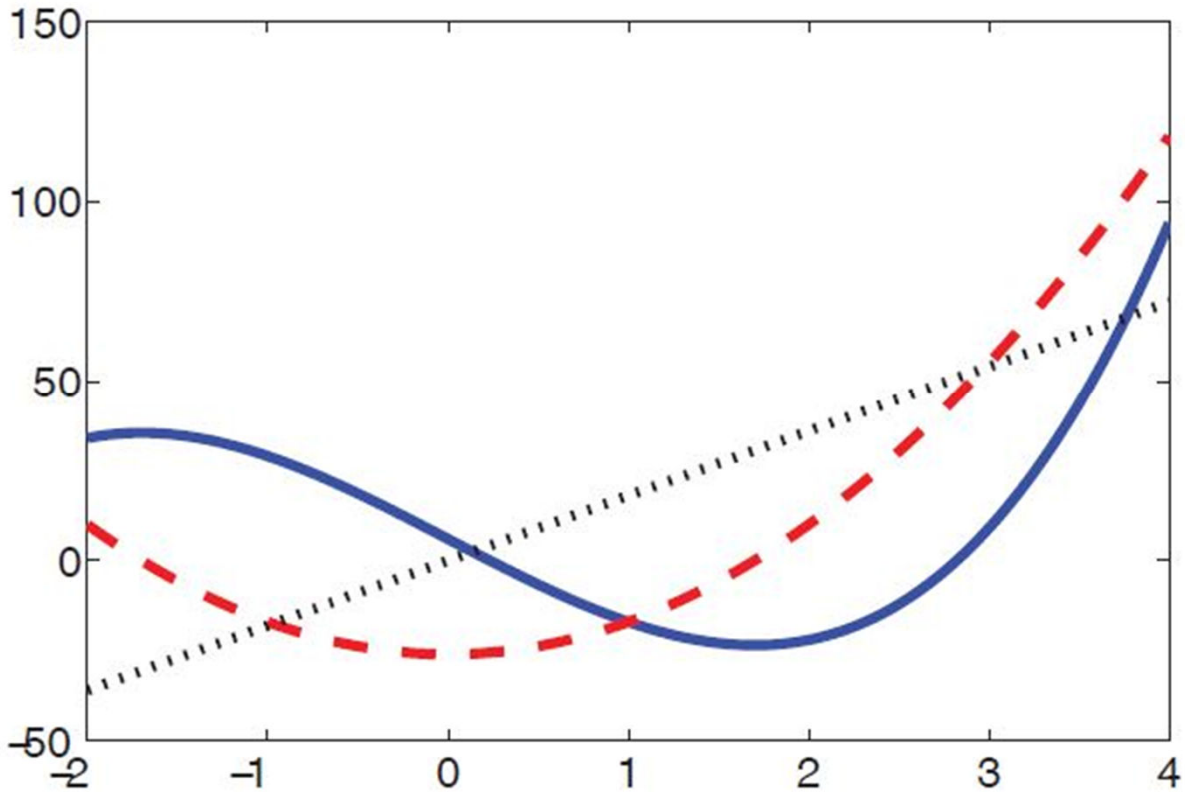
```
x=[-2:0.01:4];
y=3*x.^3-26*x+6;
yd=9*x.^2-26;
ydd=18*x;
plot(x,y,'-b',x,yd,'--r',x,ydd,':k')
```

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.


150
100
50
0
−50
−2
−1
0
1
2
3
4

hold on and 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

- 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.

Example (solving with hold on and hold off commands)

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

Solution

```
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.

line command

```
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 **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.

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

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

The major difference between the **plot** and **line** commands is :

plot command starts a new plot every time it is executed, while the **line** command adds lines to a plot that already exists.

Example

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

Solution

```
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

xlabel and **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')
```

title command

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

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

The text is placed at the top of the figure as a title.

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')
```

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

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

legend command

- 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.

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:

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

These modifiers affect the text from the point at which they are inserted until the end of the string. It is also possible to have the modifiers applied to only a section of the string by typing the modifier and the text to be affected inside braces { }.

A complete explanation of all the formatting features can be found in the Help Window under Text and Text Properties.

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

subscript and superscript

Subscript: _ (the underscore character)

Superscript: ^ in front of the character

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

Greek Characters

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

```
text(x,y,'text as string',PropertyName,PropertyValue)
```

Property name	Description	Possible property values
`Rotation`	Specifies the orientation of the text.	Scalar (degrees) Default: 0
`FontAngle`	Specifies italic or normal style characters.	`normal`, `italic` Default: `normal`
`FontName`	Specifies the font for the text.	Font name that is available in the system.
`FontSize`	Specifies the size of the font.	Scalar (points) Default: 10
`FontWeight`	Specifies the weight of the characters.	`light`, `normal`, `bold` Default: `normal`
`Color`	Specifies the color of the text.	Color specifiers (see Section 5.1).
`Background-Color`	Specifies the background color (rectangular area).	Color specifiers (see Section 5.1).
`EdgeColor`	Specifies the color of the edge of a rectangular box around the text.	Color specifiers (see Section 5.1). Default: none.
`LineWidth`	Specifies the width of the edge of a rectangular box around the text.	Scalar (points) Default: 0.5

axis command

- Axes are created 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.

`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 command

`grid on` Adds grid lines to the plot.

`grid off` Removes grid lines from the plot.

Example

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

```
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.

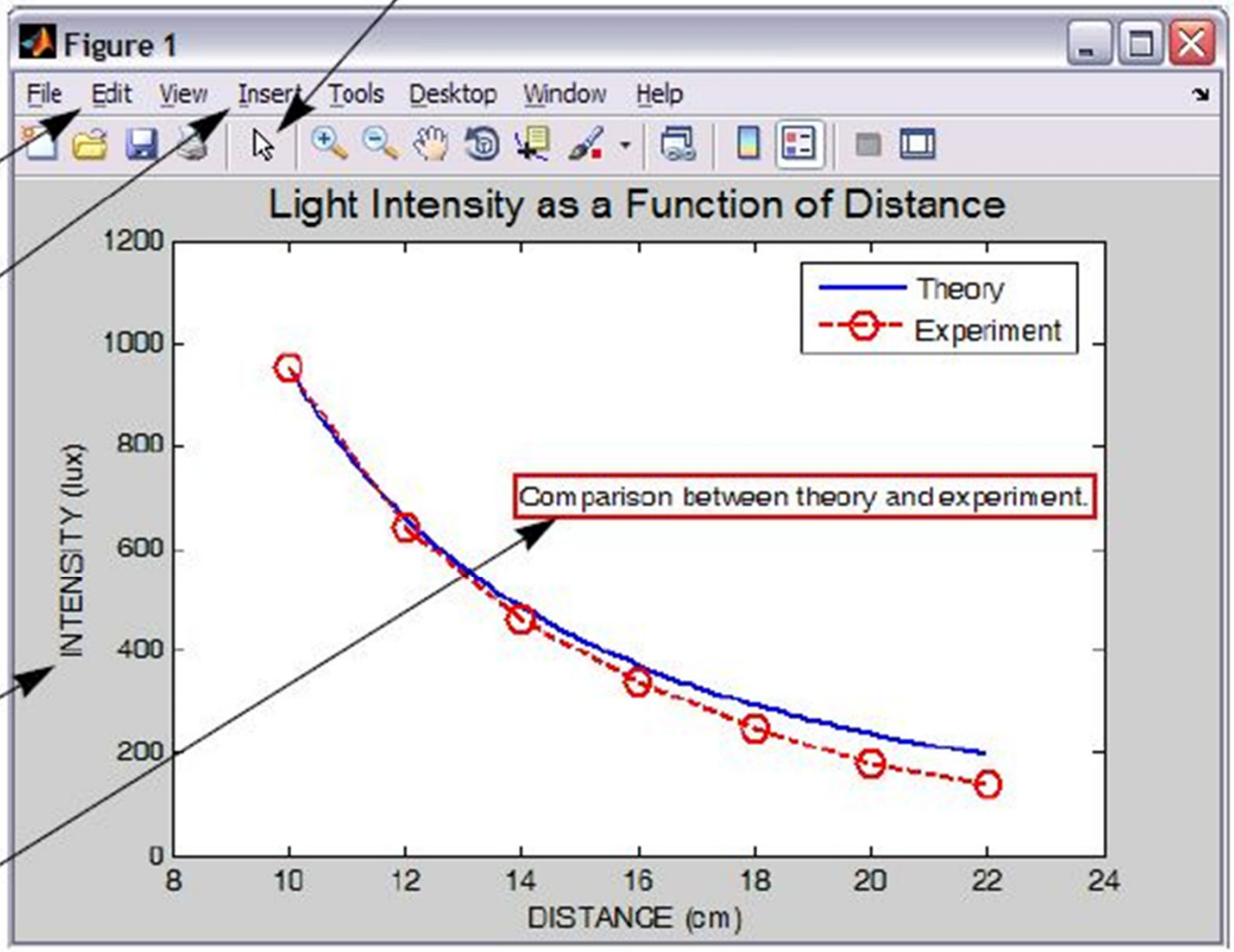
Formatting text inside the `text` command.

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.

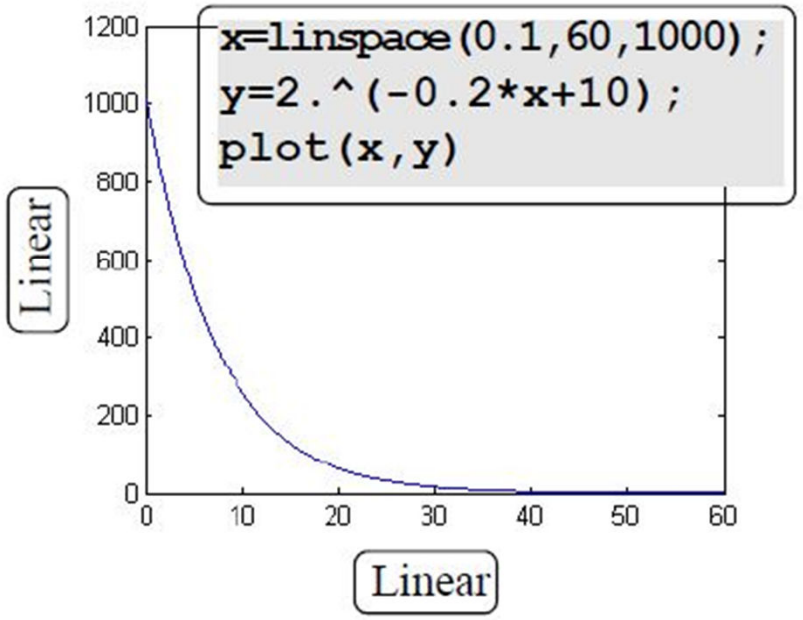
Change position of a label, legend or other object by clicking on the object and dragging.

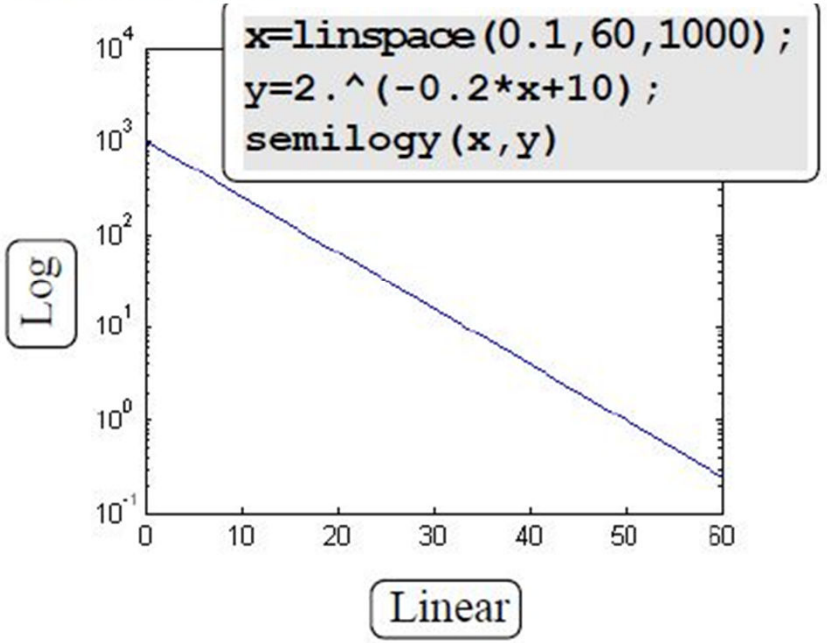


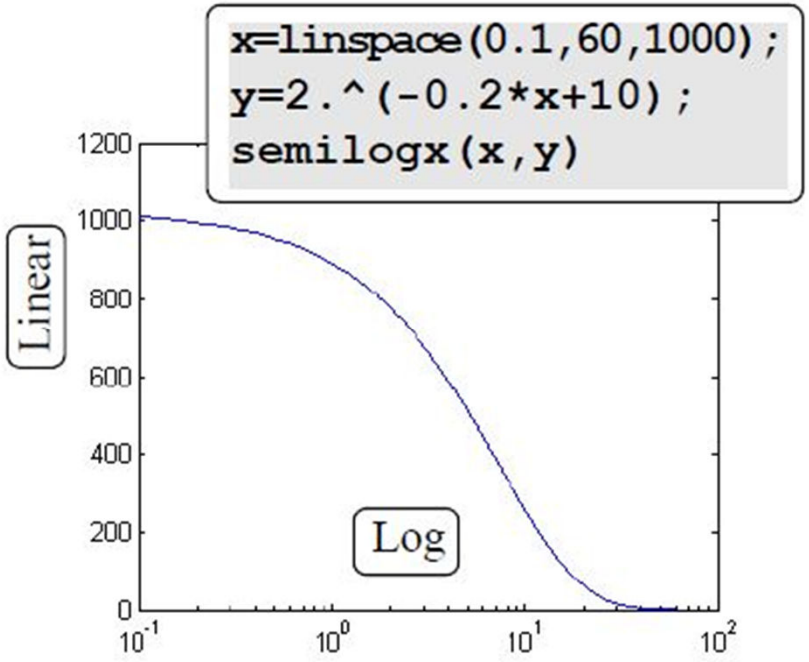
Plots with logarithmic axes

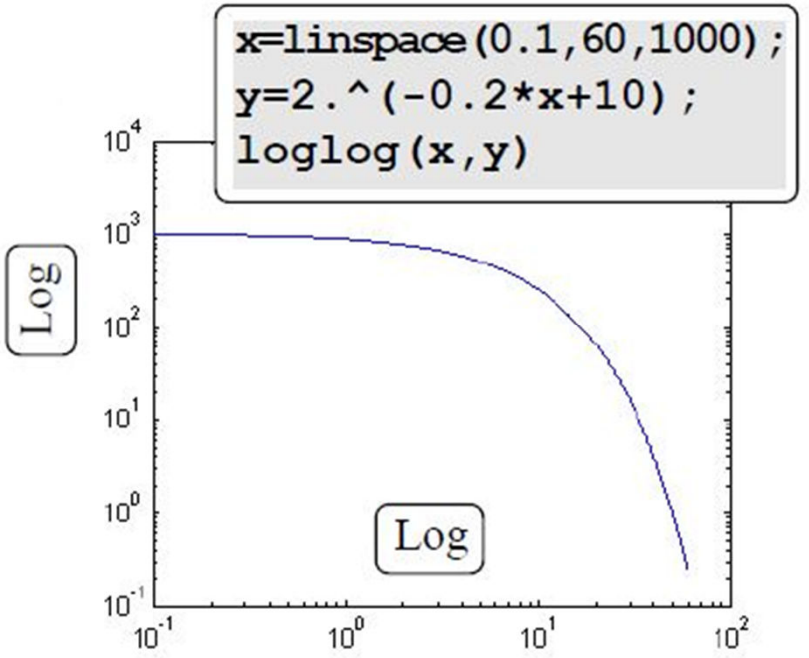
`semilogy(x,y)`	Plots `y` versus `x` with a log (base 10) scale for the y axis and linear scale for the x axis.
`semilogx(x,y)`	Plots `y` versus `x` with a log (base 10) scale for the x axis and linear scale for the y axis.
`loglog(x,y)`	Plots `y` versus `x` with a log (base 10) scale for both axes.

Plots of $y = 2^{(-0.2x+10)}$ with linear, semilog, and log-log scales.



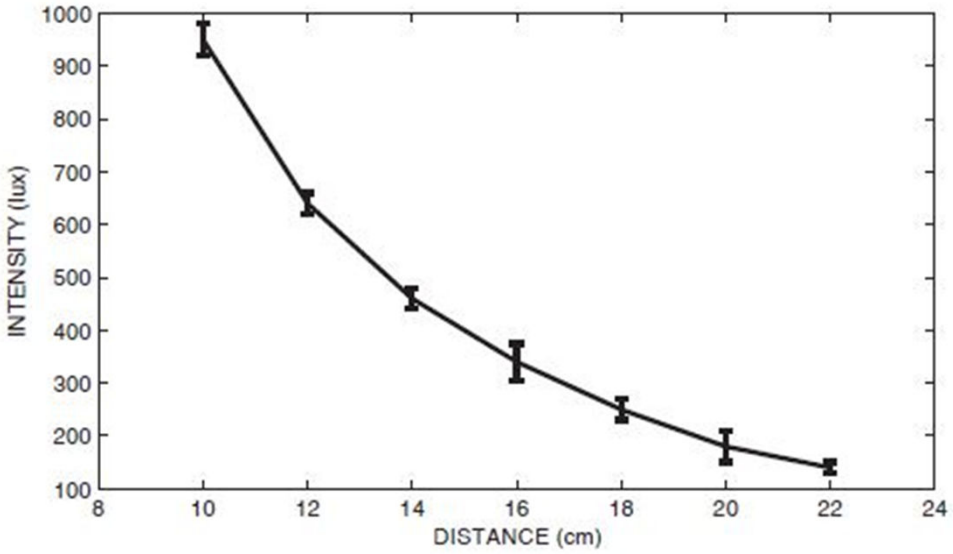






Plots with Error Bars

- Used on graphs to indicate the error, or uncertainty in a measurement



typically a short vertical line that is attached to a data point in a plot. It shows the magnitude of the error that is associated with the value that is displayed by the data point

errorbar
command

1) symmetric error bars
2) nonsymmetric error bars

1) symmetric error bars

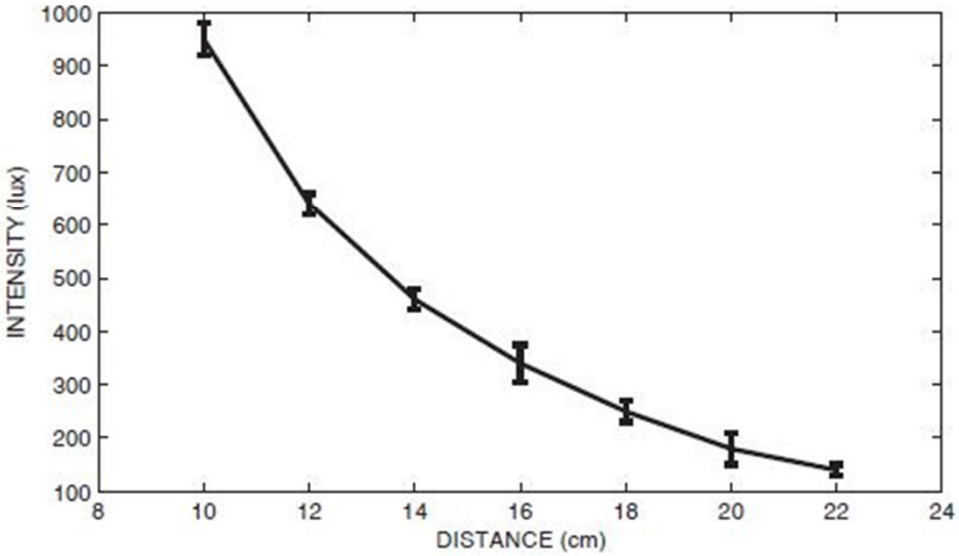
```
errorbar(x,y,e)
```

Vectors with horizontal and vertical coordinates of each point.

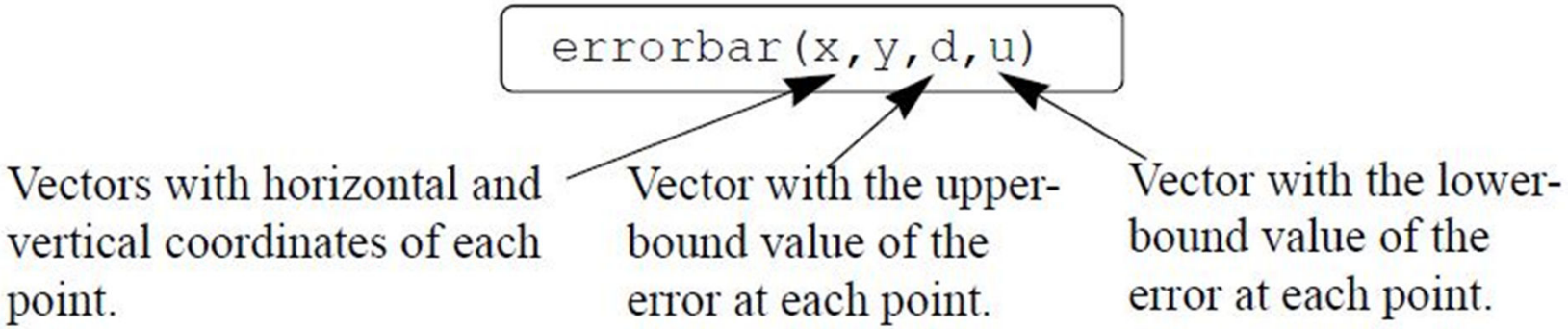
Vector with the value of the error at each point.

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

```
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)')
```



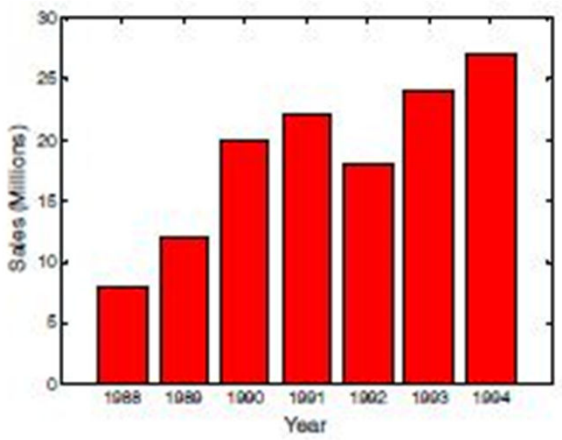
2) non-symmetric error bars

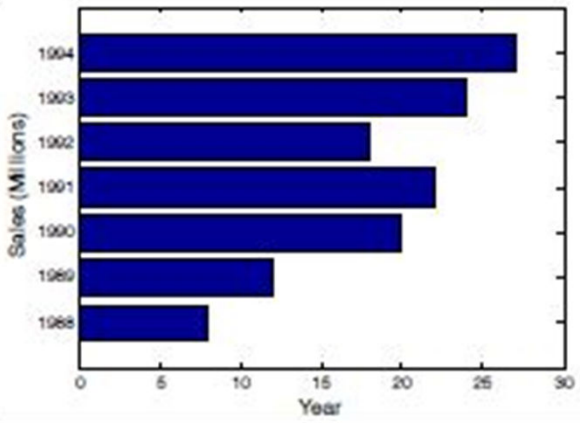


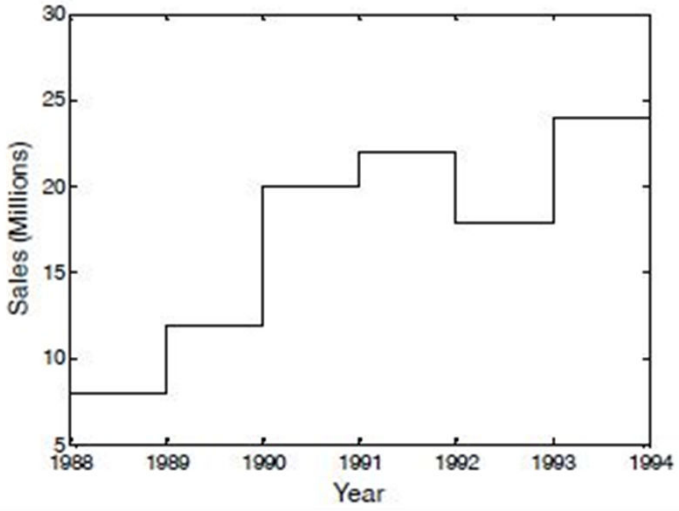
- The lengths of the four vectors `x`, `y`, `d`, and `u` must be the same.
- At each point the error bar extends from `y(i)-d(i)` to `y(i)+u(i)`.

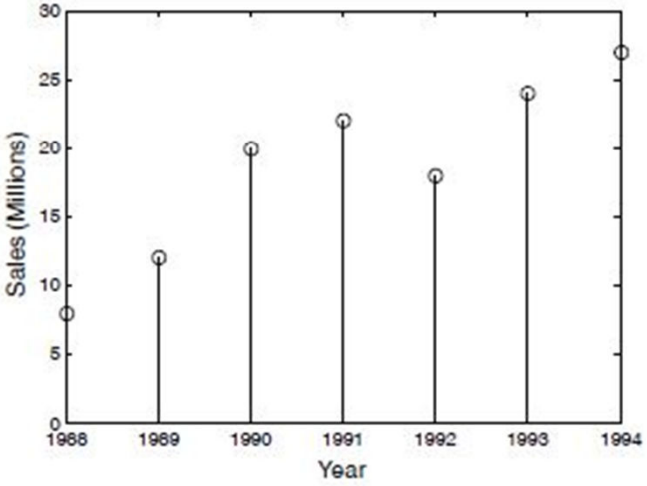
PLOTS WITH SPECIAL GRAPHICS

Help Window → "Functions by Category" → "Graphics" → "Basic Plots and Graphs" or "Specialized Plotting"

Vertical Bar Plot
 	
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 (Millions)')`

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')`

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)`

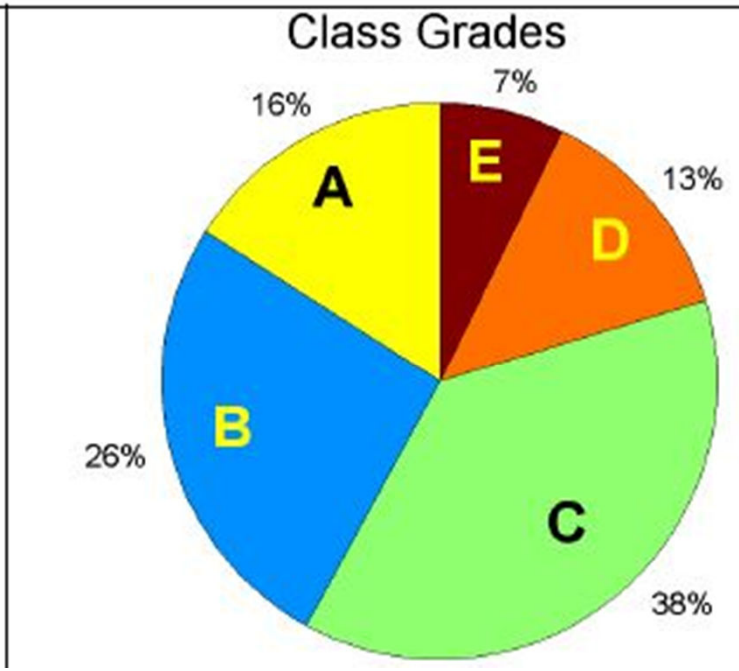
Pie Chart

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.

Histograms

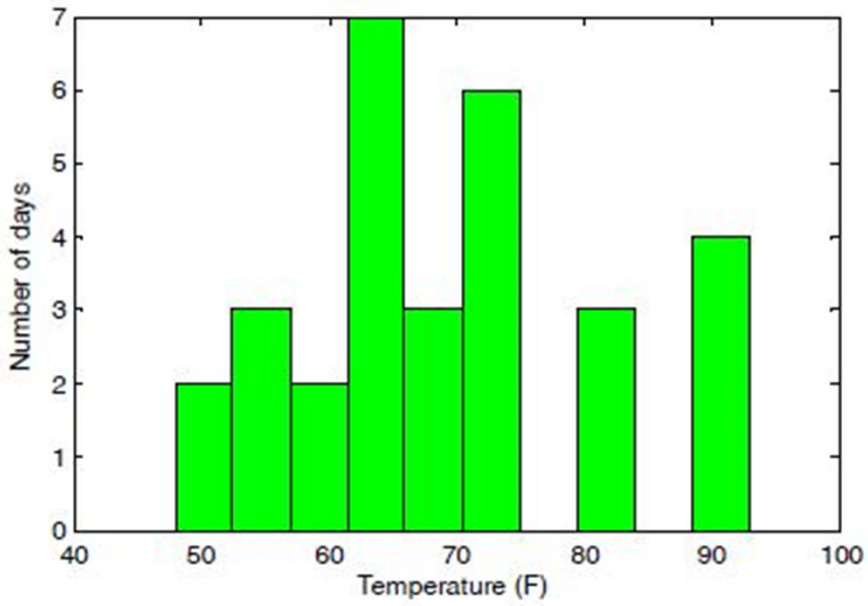
hist (y)

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

(y)

is a vector with the data points. 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.

```
>> 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 command:

hist (y,nbins) or **hist (y,x)**

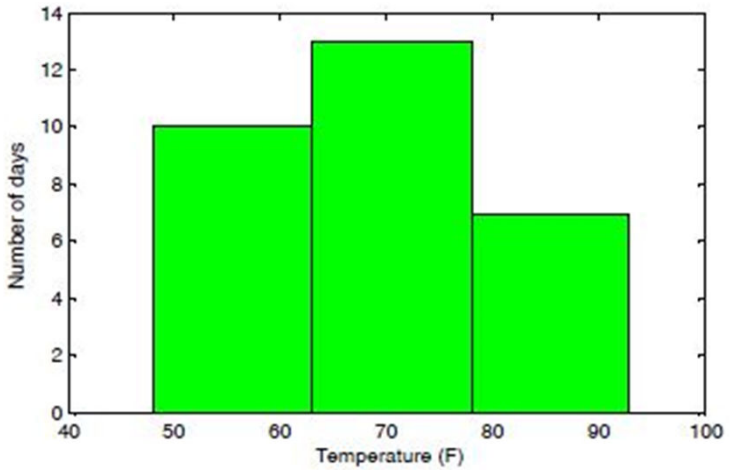
`nbins` is a scalar that defines the number of bins. MATLAB divides the range in equally spaced subranges.

`x` is a vector that specifies the location of the center of each bin (the distance between the centers does not have to be the same for all the bins). The edges of the bins are at the middle point between the centers.

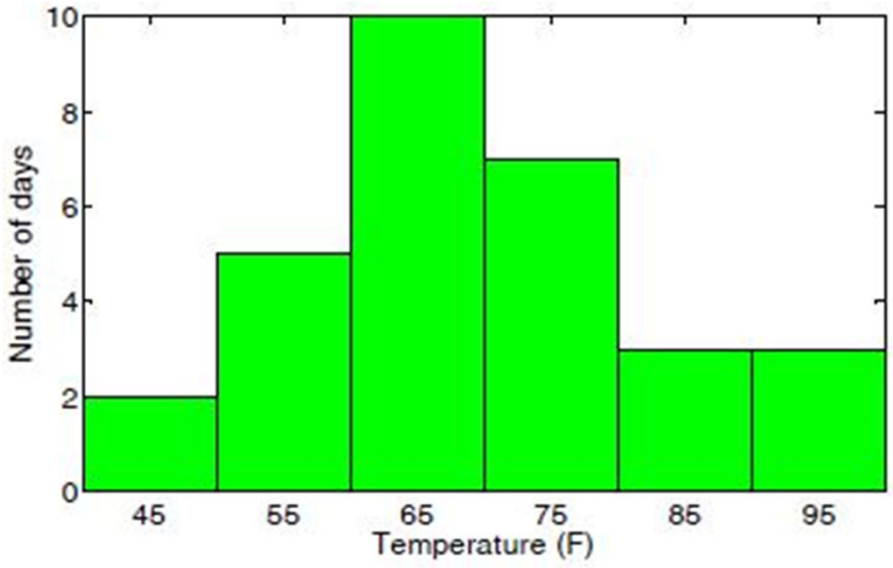
```
>> y=[58 73 73 53 50 48 56 73 73 66 69 63 74 82 84 91 93 89
91 80 59 69 56 64 63 66 64 74 63 69];
>> hist(y)
```

To divide the temperature range into three bins:

>> hist(y,3)



- Histogram displays the temp. data in 6 bins with an equal width of 10 degrees.
- The elements of the vector x for this plot are **45, 55, 65, 75, 85,** and **95**.
- The plot was obtained with the following commands:




```
>> x=[45:10:95]
x =
    45    55    65    75    85    95
>> hist(y,x)
```

`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.

```
>> n = hist(y)

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

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

- The vector n shows that the first bin has two data points, the second bin has three data points, and so on.

- 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)

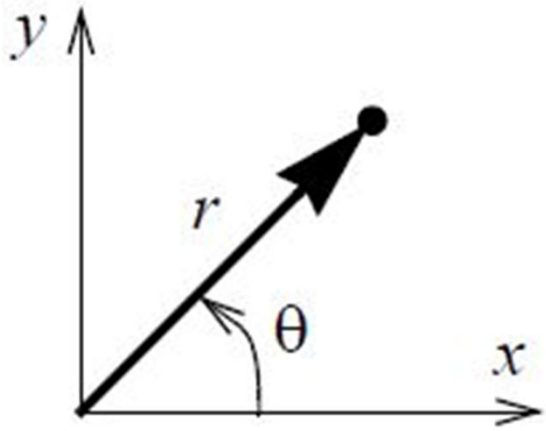
[n xout]=hist (y,nbins)

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

```
>> [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
```

- 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



- The position of a point in a plane is defined by the angle θ and the radius (distance) to the point.

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

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

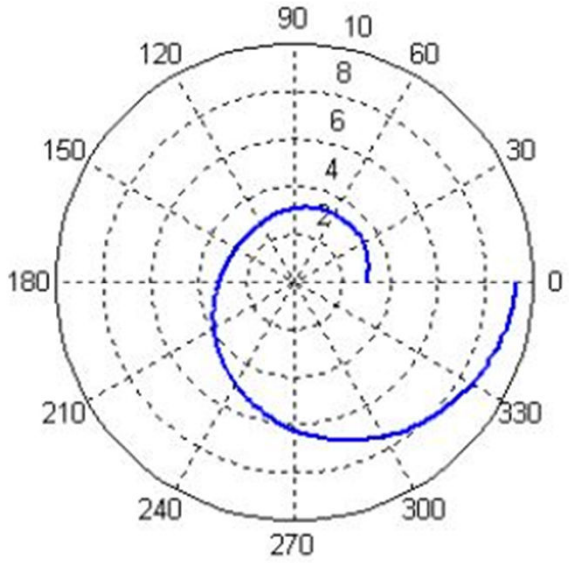
Vector — Vector — (Optional) Specifiers that define the type and color of the line and markers.

theta and **radius** are vectors whose elements define the coordinates of the points to be plotted

- A plot of the function

$$r = 3\cos^2(0.5\theta) + \theta \quad \text{for} \quad 0 \le \theta \le 2\pi$$

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

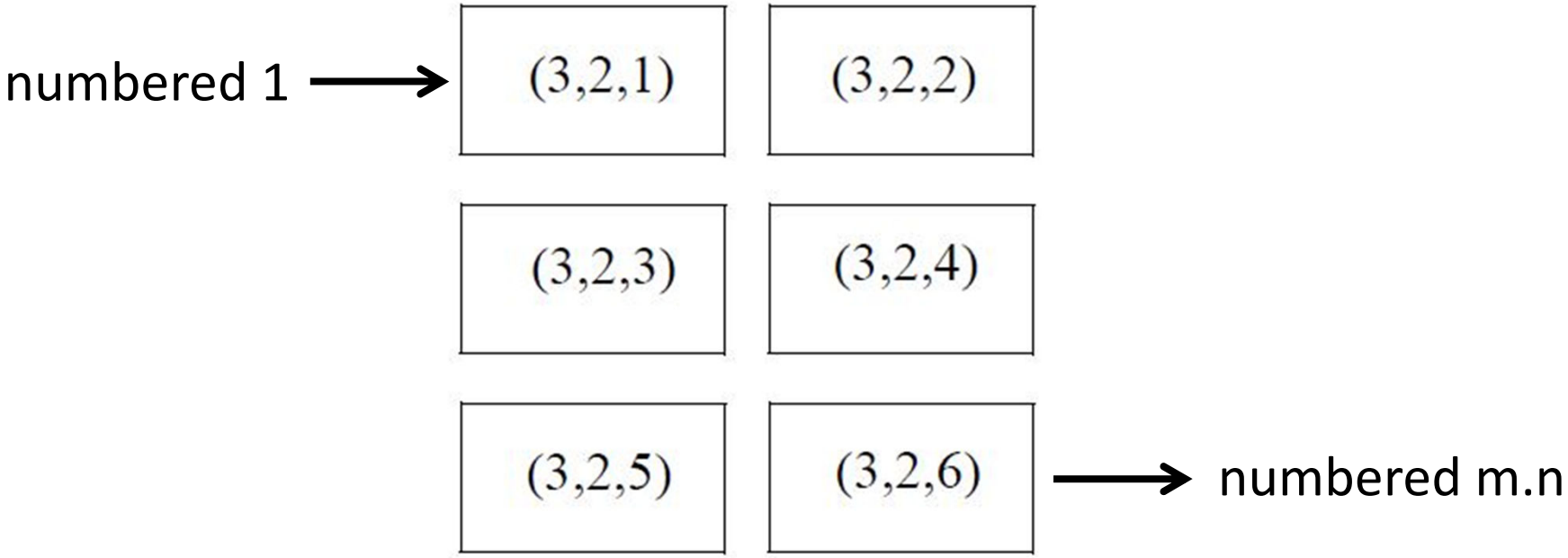


Putting Multiple Plots on the Same Page

`subplot (m,n,p)`

- The command divides the Figure Window (and the page when printed) into m x n rectangular subplots.
- The subplots are arranged like elements in an m x n matrix where each element is a subplot.
- The subplots are numbered from 1 to m.n

Putting Multiple Plots on the Same Page



- The command **subplot(m,n,p)** makes the subplot p current.
- Next plot command creates a plot in this subplot.
- The command **subplot(3,2,1)** creates six areas arranged in 3 rows and 2 columns as shown, and makes the upper left subplot current.

Multiple Figure Windows

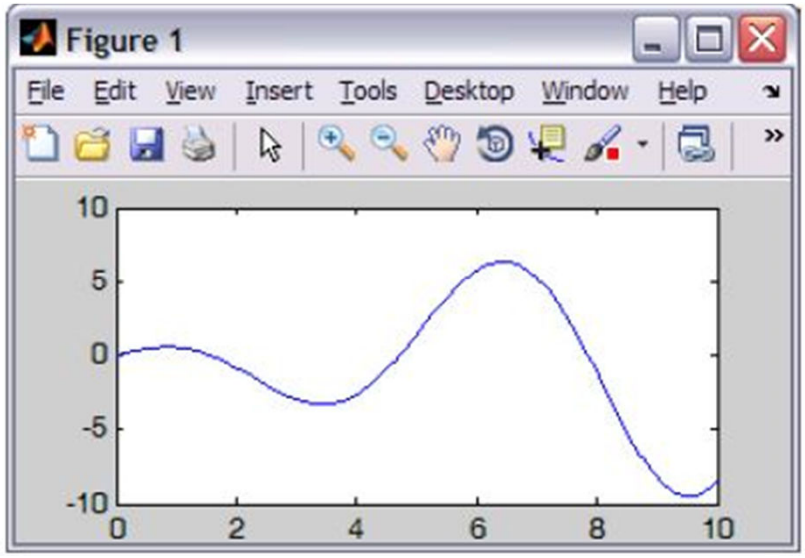
- Every time the command figure is entered, MATLAB opens a new Figure Window.
- If a command that creates a plot is entered after a figure command, MATLAB generates and displays the new plot in the last Figure Window that was opened, which is called the active or current window. MATLAB labels the new Figure Windows, i.e., Figure 2, Figure 3, and so on.

```
>> 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.



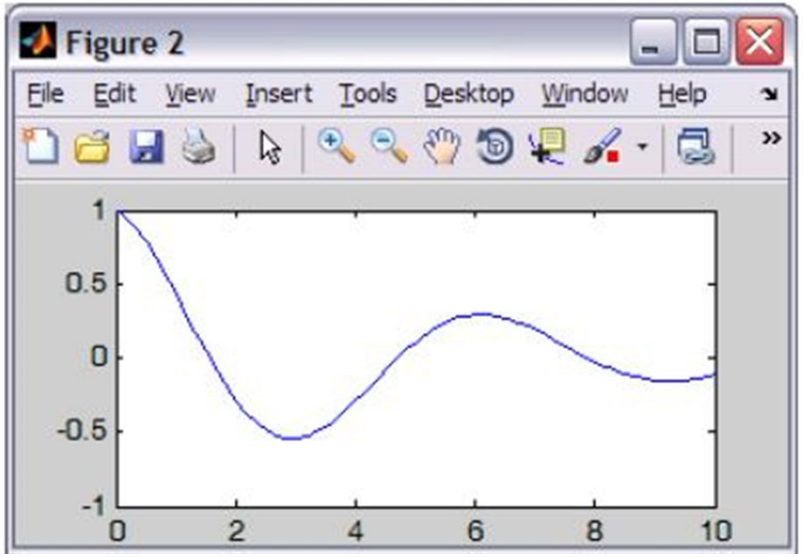


Figure Windows can be closed with the close command.
Several forms of the command are:

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