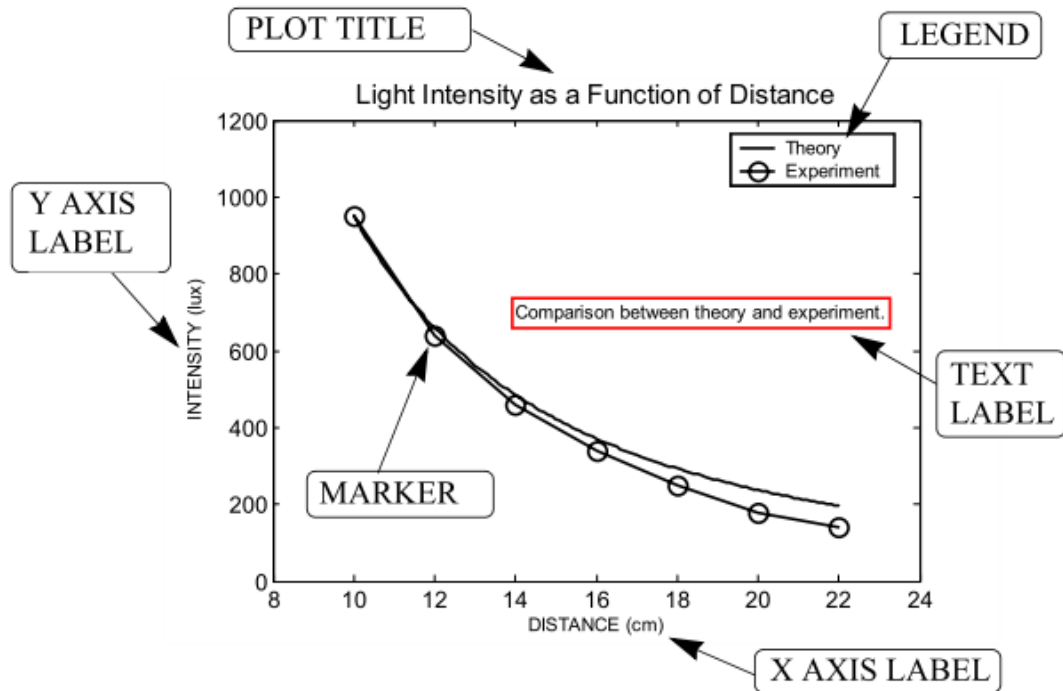


Two-Dimensional Plots

An example of a simple two-dimensional plot that was created with MATLAB is shown in Figure below.



The figure contains two curves that show the variation of light intensity with distance. The plot in Figure is formatted to have a title, axes' titles, a legend, markers, and a boxed text label.

The `plot` command

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

`plot(x,y)`
Vector Vector

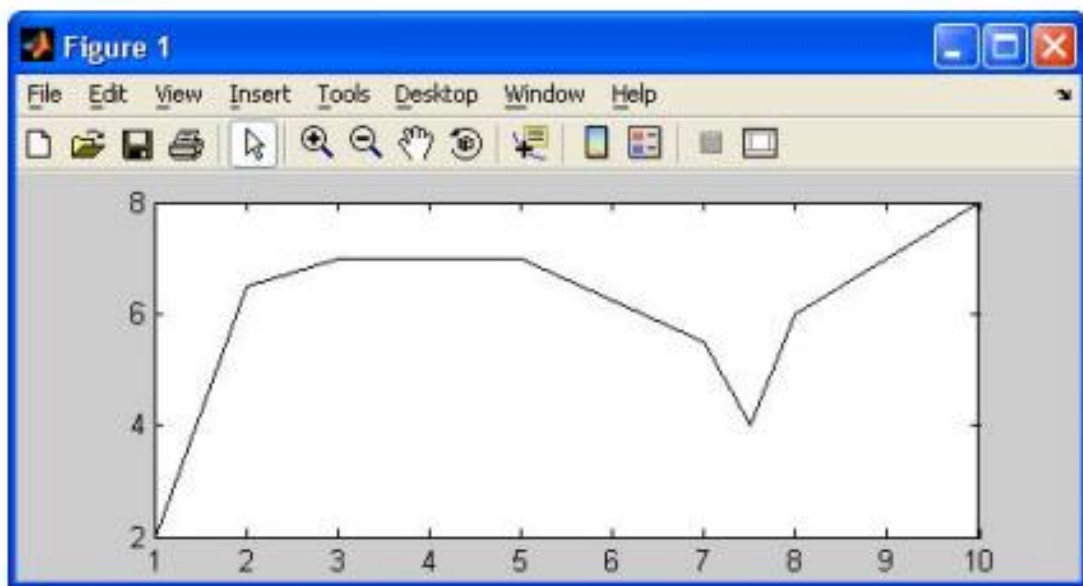
The arguments `x` and `y` are each a vector (one-dimensional array).

- Both vectors **must** have the same number of elements.
- When the plot command is executed a figure is created in the Figure Window.
- The curve is constructed of straight-line segments that connect the points whose coordinates are defined by the elements of the vectors x and y.
- The vectors can have any name.

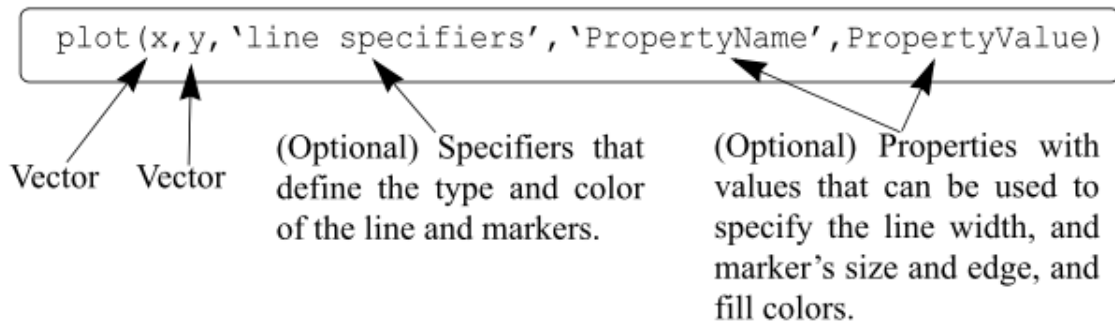
Example: If a vector x has the elements 1, 2, 3, 5, 7, 7.5, 8, 10, and a vector y has the elements 2, 6.5, 7, 7, 5.5, 4, 6, 8, a simple plot of y versus x can be created by typing:

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

Once the plot command is executed, the Figure Window opens and the plot is displayed.



- The plot appears on the screen in blue, which is the default line color.
- The plot command has additional optional arguments that can be used to specify the
 - Color and style of the line
 - Color and type of markers,



a. 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	-.

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			

- Within the string the specifiers can be typed in any order.
- The specifiers are optional. This means that none, one, two, or all the three can be included in a command.

Some examples:

<code>plot(x,y)</code>	A blue solid line connects the points with no markers (default).
<code>plot(x,y,'r')</code>	A red solid line connects the points.
<code>plot(x,y,'--y')</code>	A yellow dashed line connects the points.
<code>plot(x,y,'*')</code>	The points are marked with * (no line between the points).
<code>plot(x,y,'g:d')</code>	A green dotted line connects the points that are marked with diamond markers.

b. Property Name and Property Value:

- Properties are optional
- 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 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')
```

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.

A note about line specifiers and properties:

The three line specifiers, which are the style and 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:

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

Example 1

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

- The list of years is assigned to one vector (named `yr`),
- The corresponding sale data is assigned to a second vector (named `sale`).

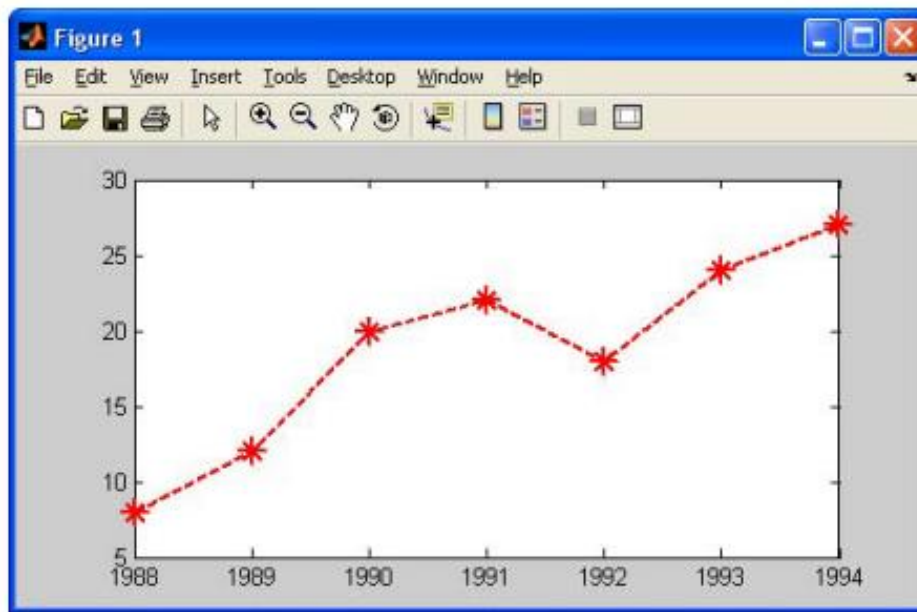
The vectors are created and the plot command is used is shown below:

```
>> 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 markers
size is 12 point.

Once the plot command is executed the Figure Window with the plot, as shown in Figure.



Example 2: Plot the function $y = 3.5^{-0.5x} \cos 6x$ for $-2 \leq x \leq 4$,

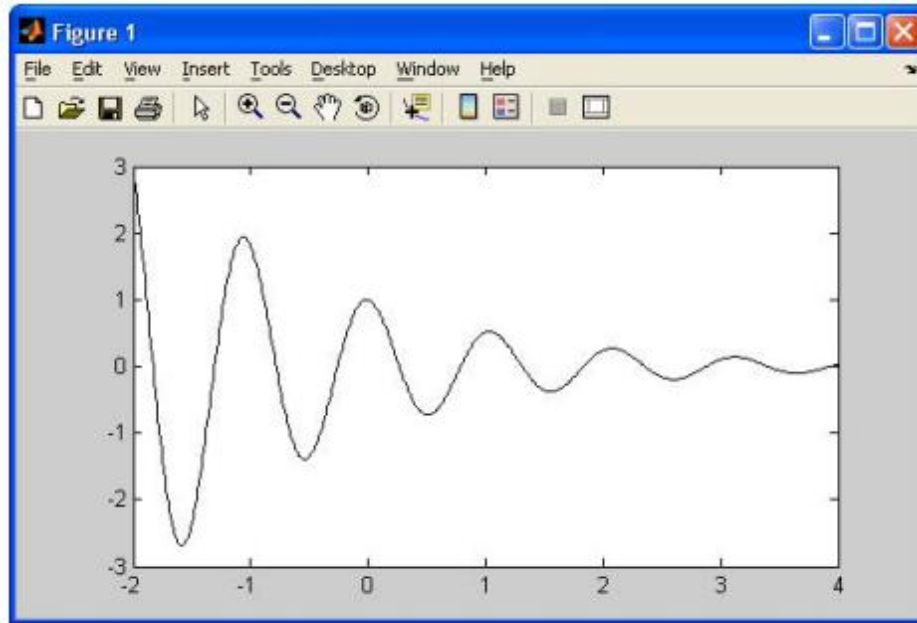
```
% A script file that creates a plot of
% the function: 3.5.^(-0.5*x).*cos(6*x)
x=[-2:0.01:4];
y=3.5.^(-0.5*x).*cos(6*x);
plot(x,y)
```

Create vector x with the domain of the function.

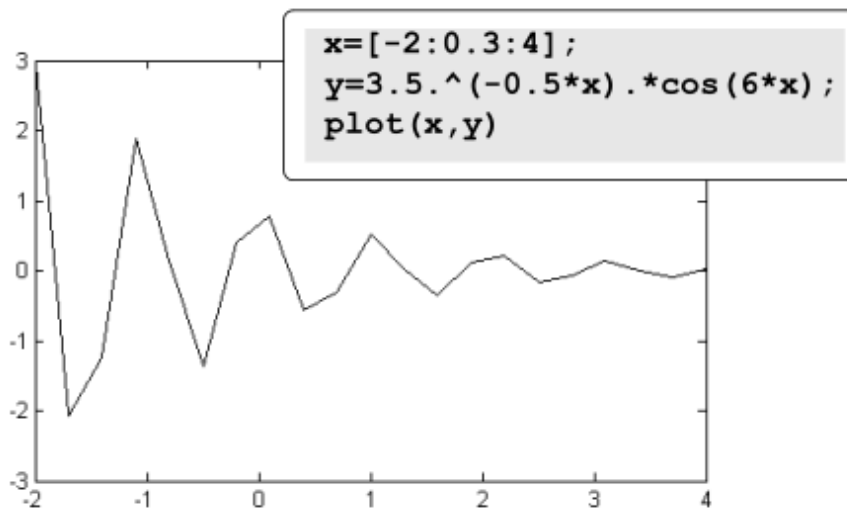
Create vector y with the function
value at each x.

Plot y as a function of x.

Once the script file is executed, the plot is created in the Figure Window, as shown in Figure.

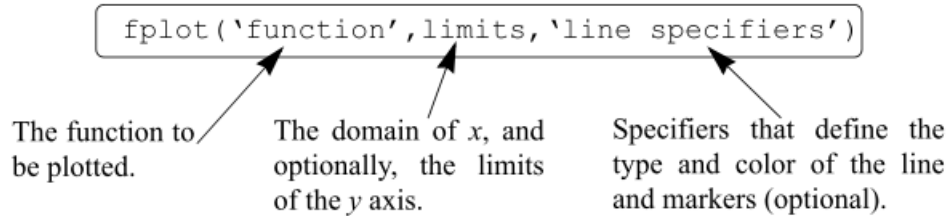


- To obtain an accurate plot of a function, the spacing between the elements of the vector x must be appropriate. Smaller spacing is needed for a function that changes rapidly. A small spacing of 0.01 produced the plot that is shown in Figure above.
- If the same function in the same domain is plotted with much larger spacing, for example 0.3, the plot that is obtained, shown in Figure below, gives a distorted picture of the function.



The `fplot` command

The `fplot` command plots a function with the form between specified limits.



❖ **'function'**: The function can be typed directly as a string inside the command.

For example, if the function that is being plotted is , it is typed as:

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

• The function to be plotted can be typed as a function of any letter.

❖ **limits** : The limits is a vector with two elements that specify the domain of x [x_{min}, x_{max}], or a vector with four elements that specifies the domain of x and the limits of the y -axis [$x_{min}, x_{max}, y_{min}, y_{max}$].

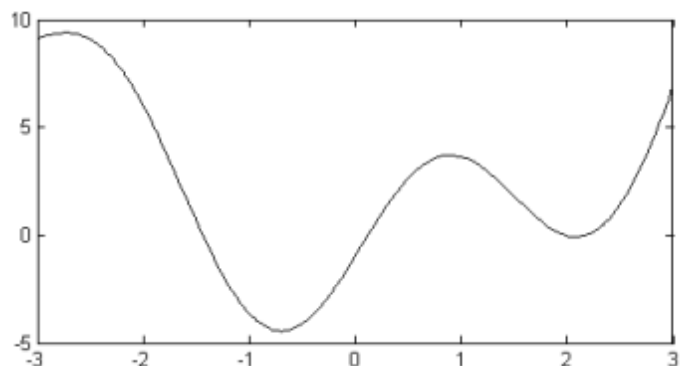
❖ **Line specifiers** : The line specifiers are the same as in the `plot` command. For example, a plot of the function

$$y = x^2 + 4\sin(2x) - 1 \text{ for } -3 \leq x \leq 3$$

Can be created with the `fplot` command by typing:

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

The figure that is obtained in the Figure Window is shown in Figure



Plotting multiple graphs in the same plot

In many situations there is a need to make several graphs in the same plot. for example, in Figure 5-1 where two graphs are plotted in the same figure.

- There are three methods to plot multiple graphs in one figure.
 1. By using the `plot` command,
 2. By using the `hold on`, `hold off` commands,
 3. 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 so that they can be identified.
- It is also possible to add line specifiers following each pair. For example:

```
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 1: Plotting a function and its derivatives

Plot the function $y=3x^3-26x+10$ And its first and second derivatives, for $-2 \leq x \leq 4$

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''

```
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.

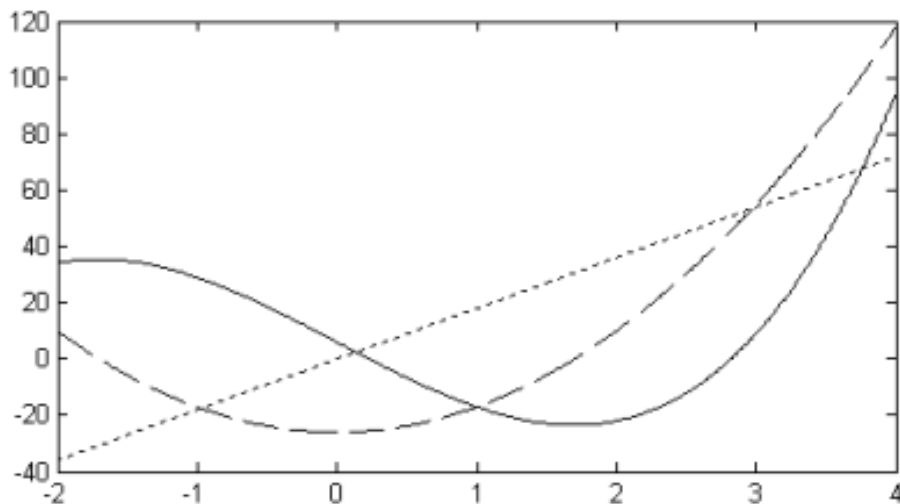
Create vector y with the function value at each x .

Create vector y_d with values of the first derivative.

Create vector y_{dd} with values of the second derivative.

Create three graphs, y vs. x , y_d vs. x , and y_{dd} vs. x in the same figure.

The plot that is created is shown in Figure



2. Using the hold on, hold off Commands

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

- One graph is plotted first with the `plot` command.

- b. 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.
- c. Additional graphs can be added with `plot` commands that are typed next. Each `plot` command creates a graph that is added to that figure.
- d. The `hold off` command stops this process. It returns MATLAB to the default mode in which the `plot` command erases the previous plot.

Example: Solve the previous example using `hold on`, `hold off` commands.

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.

3. Using the `line` Command

With the `line` command additional graphs (lines) can be added to a plot.

`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 `PropertyName` and `PropertyValue` features.

For example,

```
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,

line command adds lines to a plot that already exists.

- To make a plot that has several graphs,
 - a. A `plot` command is typed first
 - b. then `line` commands are typed for additional graphs.

(If a line command is entered before a plot command an error message is displayed.)

Example: Plotting a function and its derivatives (same aforementioned example)

Plot the function $y=3x^3-26x+10$ And its first and second derivatives, for $-2 \leq x \leq 4$ using `line` command.

Solution:

```
x=[-2:0.01:4];  
y=3*x.^3-26*x+10;  
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 figure can include specifying axis labels, plot title, legend, grid, range of custom axis, and text labels.
- Plots can be formatted by using:
 1. MATLAB commands that follow the `plot` or `fplot` commands: useful when a plot command is a part of a computer program
 2. The plot editor in the Figure Window: used after the plot has been created, it have to be repeated the next time the plot is created.

1. Formatting a Plot Using Commands

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

a. The `xlabel` and `ylabel` commands:

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

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

b. The `title` command:

A title can be added to the plot with the following command, The text is placed at the top of the figure as a title.

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

c. 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` (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.

d. 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. The `pos` is an optional number that specifies where in the figure the legend is placed. The options are:

<code>pos = -1</code>	Places the legend outside the axes boundaries on the right side.
<code>pos = 0</code>	Places the legend inside the axes boundaries in a location that interferes the least with the graphs.
<code>pos = 1</code>	Places the legend at the upper-right corner of the plot (default).
<code>pos = 2</code>	Places the legend at the upper-left corner of the plot.
<code>pos = 3</code>	Places the legend at the lower-left corner of the plot.
<code>pos = 4</code>	Places the legend at the lower-right corner of the plot.

Formatting the text in the `xlabel`, `ylabel`, `title`, `text` and `legend` commands:

The formatting can be used to define the font, size, position (superscript, subscript), style (italic, bold, etc.), and color of the characters, color of the background, and to define many other details of the display.

- A complete explanation of all the formatting features can be found in the Help Window under Text and Text Properties.
- The formatting can be done either by adding modifiers inside the string, or by adding to the command optional `PropertyName` and `PropertyValue` arguments following the string.

The modifiers are characters that are inserted within the string. Some of the modifiers that can be added are:

Modifier	Effect
<code>\bf</code>	bold font.
<code>\it</code>	italic style.
<code>\rm</code>	normal font.

Modifier	Effect
<code>\fontname{fontname}</code>	specified font is used.
<code>\fontsize{fontsize}</code>	specified font size is used.

- It is 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. Subscript and superscript:

A single character can be displayed as

subscript: By typing `_` (the underscore character) in front of the character

superscript: By typing `^` in front of the character.

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

b. 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.
- Some examples are:

Characters in the string	Greek Letter
<code>\alpha</code>	α
<code>\beta</code>	β
<code>\gamma</code>	γ
<code>\theta</code>	θ
<code>\pi</code>	π
<code>\sigma</code>	σ

Characters in the string	Greek Letter
<code>\Phi</code>	Φ
<code>\Delta</code>	Δ
<code>\Gamma</code>	Γ
<code>\Lambda</code>	Λ
<code>\Omega</code>	Ω
<code>\Sigma</code>	Σ

c. Text

Formatting of the text that is displayed by the `xlabel`, `ylabel`, `title`, and `text` commands can also be done by adding optional `PropertyName` and `PropertyValue` arguments following the string inside the command. for example:

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

- In the other three commands the `PropertyName` and `PropertyValue` arguments are added in the same way.
- The `PropertyName` is typed as a string, and the `PropertyValue` is typed as a number if the property value is a number and as a string if the property value is a word or a letter character.
- Some of the Property Names and corresponding possible Property Values are:

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
Background-Color	Specifies the background color (rectangular area).	Color specifiers
EdgeColor	Specifies the color of the edge of a rectangular box around the text.	Color specifiers Default: none.
LineWidth	Specifies the width of the edge of a rectangular box around the text.	Scalar (points) Default: 0.5

d. 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.
- The following are some of the possible forms of the axis command:

<code>axis([xmin,xmax,ymin,ymax])</code>	Sets the limits of both the x and y axes (xmin, xmax, ymin, and ymax are numbers).
<code>axis equal</code>	Sets the same scale for both axes.
<code>axis square</code>	Sets the axes region to be square.
<code>axis tight</code>	Sets the axis limits to the range of the data.

e. The grid command

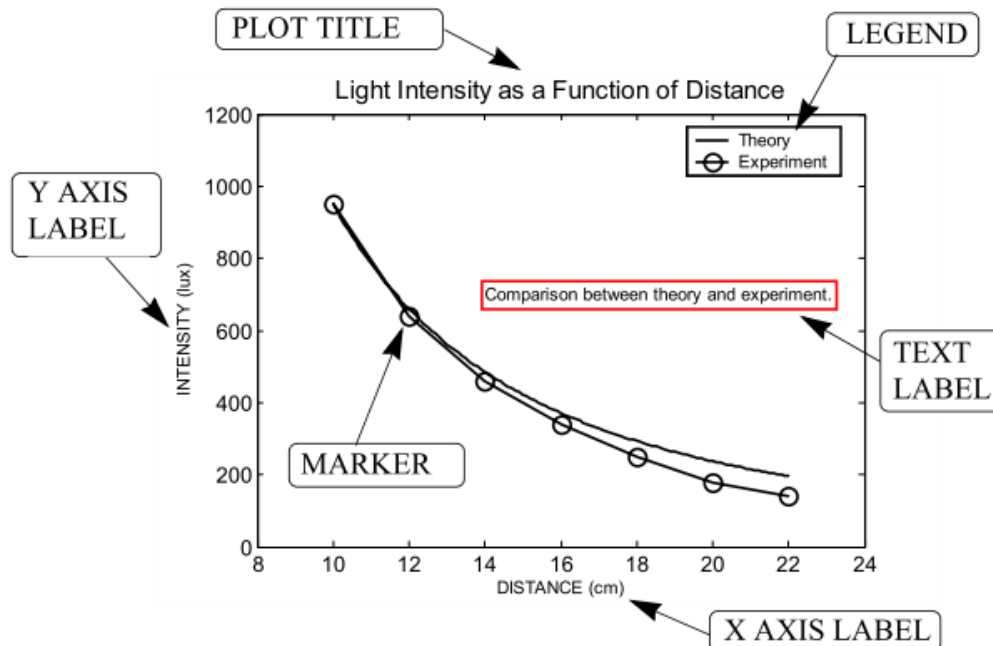
`grid on` → Adds grid lines to the plot.

`grid off` → Removes grid lines from the plot.

Example:

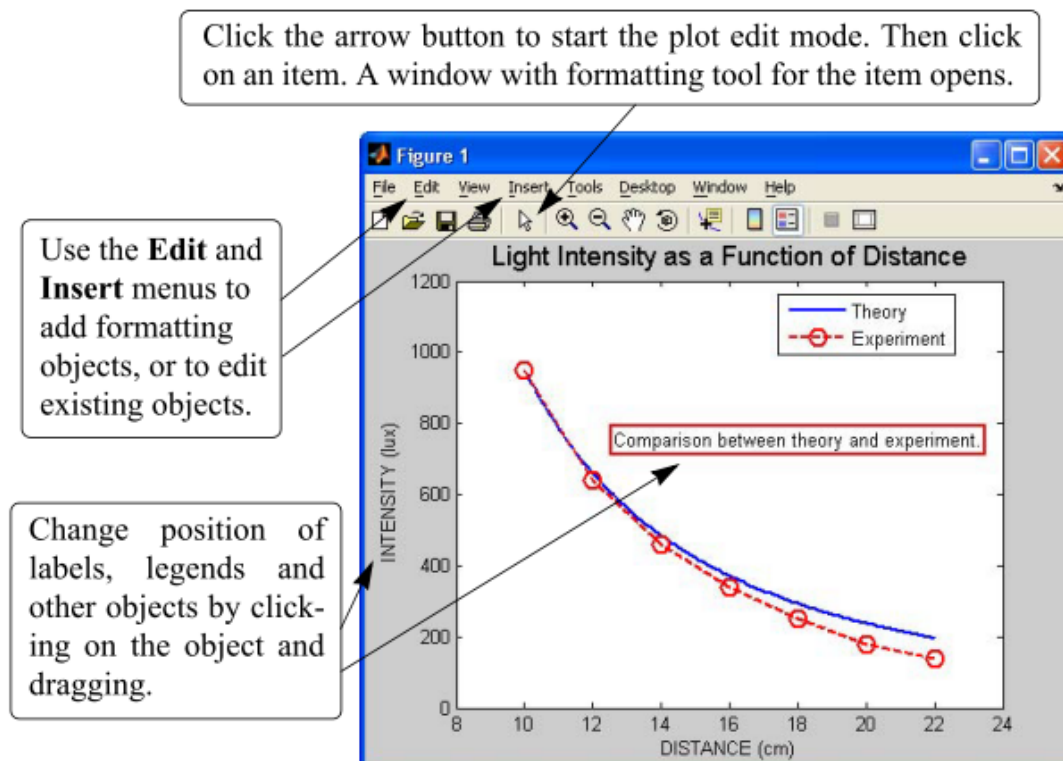
An example of formatting a plot by using commands is given in the following script file that was used to generate the formatted plot of the figure below

```
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
```



2. Formatting a Plot Using the Plot Editor

A plot can be formatted interactively in the Figure Window by clicking on the plot and/or using the menus. Figure below shows the Figure Window with the plot of figure above.



The Plot Editor can be used to introduce new formatting items, or to modify formatting that was initially introduced with the formatting commands.

PLOTS WITH LOGARITHMIC AXES

Log scales provide means for presenting data over a wide range of values. MATLAB commands for making plots with log axes are:

`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.

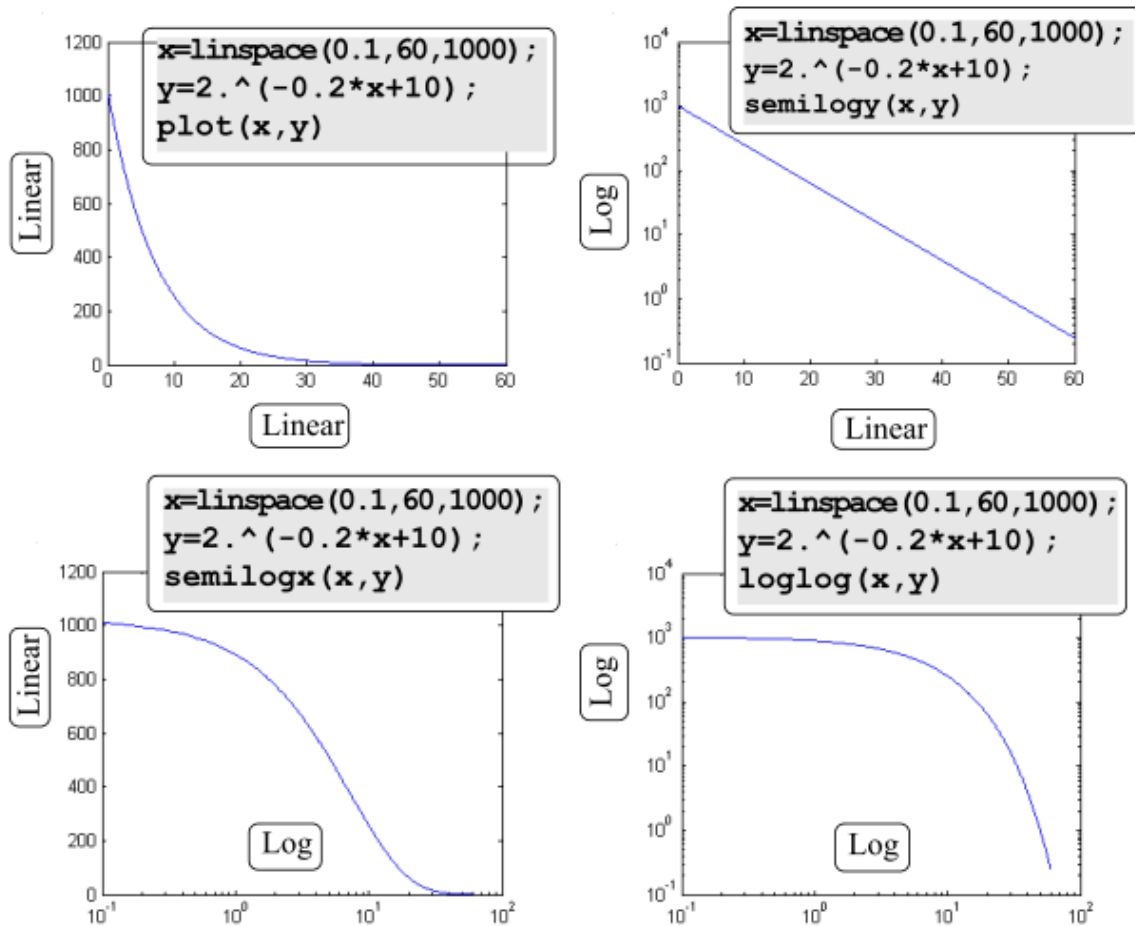
`loglog(x, y)` → Plots y versus x with a log (base 10) scale for both axes.

- Line specifiers and Property Name and property value can be added to the commands (optional) just as in the plot command.
- The number zero cannot be plotted on a log scale (since a log of zero is not defined).
- Negative numbers cannot be plotted on log scales (since a log of a negative number is not defined).

Example: The figure below shows four plots of the same function:

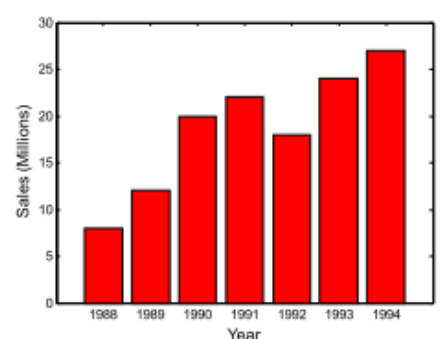
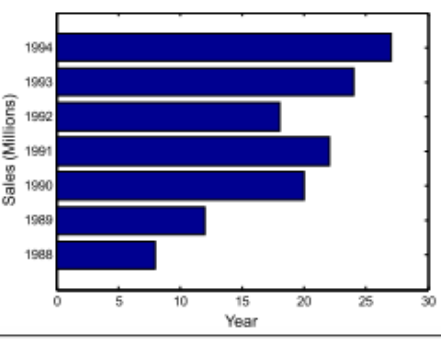
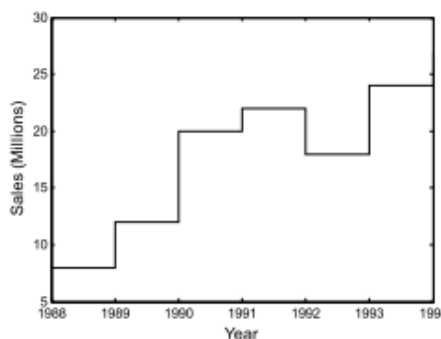
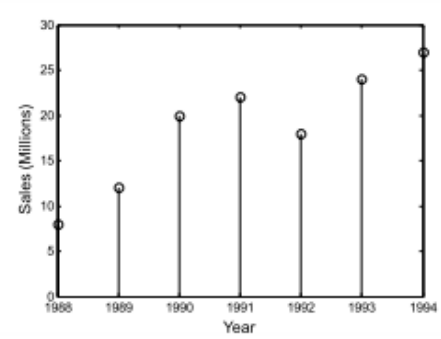
$$y = 2^{(-0.2x + 10)} \text{ for } 0.1 \leq x \leq 60$$

one with linear axes, one with log scale for the y-axis, one with log scale for the x-axis, and one with log scale on both axes.



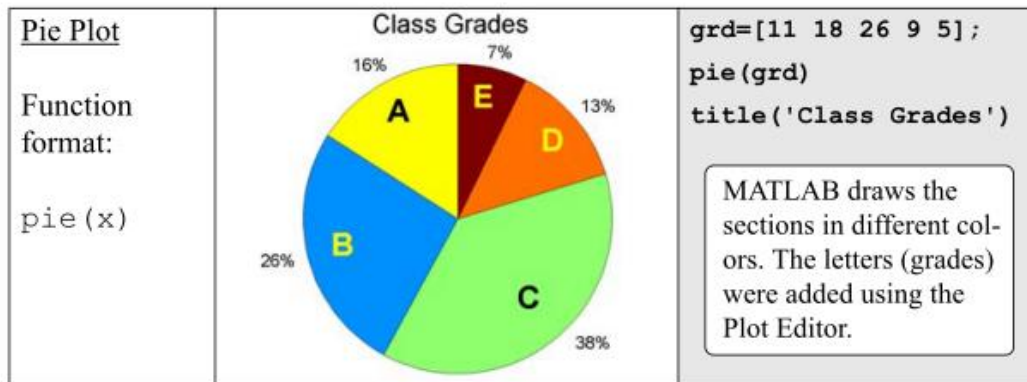
PLOTS WITH SPECIAL GRAPHICS

- MATLAB has many options for creating a wide variety of plots. These include bar, stairs, stem, pie plots and many more.
- A complete list of the plotting functions that MATLAB has and information on how to use them can be found in the Help Window.
- Bar (vertical and horizontal), stairs, and stem plots are presented as follows:

<u>Vertical Bar Plot</u> Function format: <code>bar(x,y)</code>		<pre>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)')</pre>
<u>Horizontal Bar Plot</u> Function format: <code>barh(x,y)</code>		<pre>yr=[1988:1994]; sle=[8 12 20 22 18 24 27]; barh(yr,sle) xlabel('Sales (Millions)') ylabel('Year')</pre>
<u>Stairs Plot</u> Function format: <code>stairs(x,y)</code>		<pre>yr=[1988:1994]; sle=[8 12 20 22 18 24 27]; stairs(yr,sle)</pre>
<u>Stem Plot</u> Function Format <code>stem(x,y)</code>		<pre>yr=[1988:1994]; sle=[8 12 20 22 18 24 27]; stem(yr,sle)</pre>

Pie charts are useful for visualizing the relative sizes of different but related quantities. For example, the table below shows the grades that were assigned to a class. The data below is used to create the pie chart that follows.

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



HISTOGRAMS

Histograms are plots that show the distribution of data.

- The histogram is a vertical bar plot in which
 - ➔ the width of each bar is equal to the range of the corresponding bin,
 - ➔ the height of the bar corresponds to the number of data points in the bin.
- Histograms are created in MATLAB with the `hist` command.

`hist(y)` `y` is a vector with the data points.

Example: the following data are the daily maximum temperature (in °F) for one month,

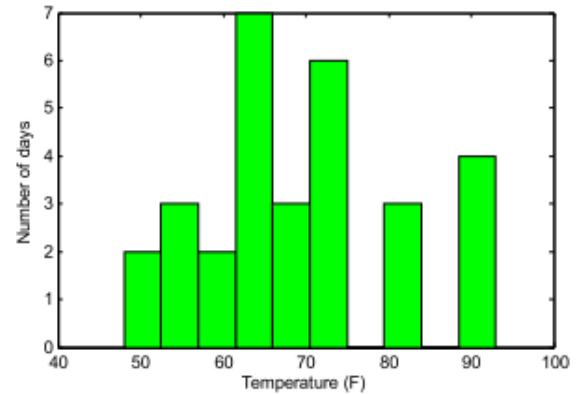
58 73 73 53 50 48 56 73 73 66 69 63 74 82 84 91 93 89 91 80 59 69 56 64 63 66 64 74 63 69

A histogram of this data is obtained with the commands:

```
>> 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)
```

The plot that is generated is shown in Figure below (the axes titles were added using the Plot Editor). The smallest value in the data set is 48 and the largest is 93, which means that the range is 45 and that the width of each bin is 4.5.

- The range of the first bin is from 48 to 52.5 and contains two points.
- The range of the second bin is from 52.5 to 57 and contains three points, and so on.
- Two of the bins (75 to 79.5 and 84 to 88.5) do not contain any points.



The histogram in Figure can also be created with the following command:

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

```
n =
```

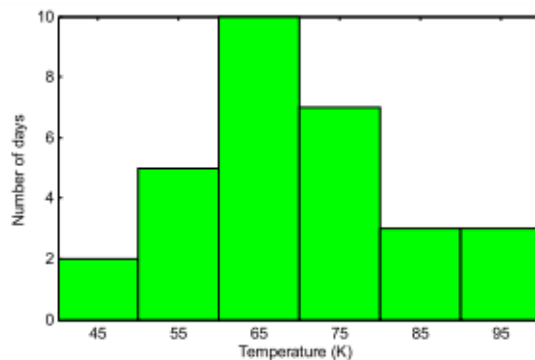
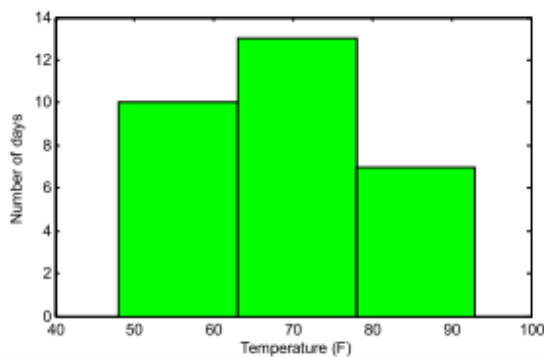
```
2 3 2 7 3 6 0 3 0 4
```

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

In the example above the user might prefer to divide the temperature range into 3 or 6 bins. This can be done with the command:

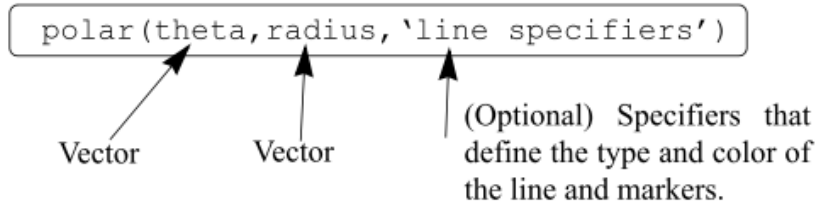
```
>> hist(y, 3)
```

```
>> hist(y, 6)
```



POLAR PLOTS

Polar coordinates are frequently used in the solution of science and engineering problems. The command has the form:



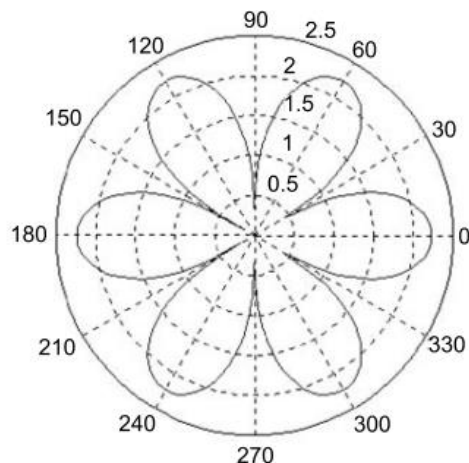
- the position of a point in a plane is defined by the angle θ (`theta`)
- the `radius` is the distance to the point,
- `theta` and `radius` are vectors whose elements define the coordinates of the points to be plotted.
- The `polar` command plots the points and draws the polar grid. The line specifiers are the same as in the `plot` command.
- To plot a function in a certain domain, a vector for values of θ is created first, and then a vector `r` with the corresponding values of $f(\theta)$ is created using element-by-element calculations. The two vectors are then used in the `polar` command.

Example: plot the following functions using polar plot

$$r = 5 \cos 3t; \quad 0 \leq t \leq 2\pi \text{ step} = 0.03$$

Solution:

```
% (a)
t = 0:0.03:2*pi;
r = sqrt(abs(5*cos(3*t)));
polar(t,r)
```



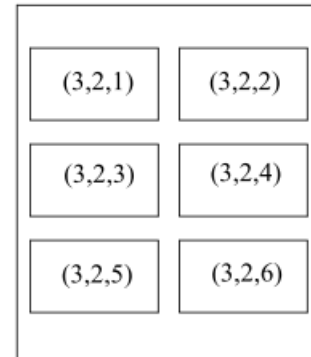
PLOTTING MULTIPLE PLOTS ON THE SAME PAGE

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 command `subplot(m,n,p)` makes the subplot p current.

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 FIGURE WINDOWS

- It is possible 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.

For example, after the following three commands are entered, two Figure Windows that are shown in Figure are displayed.

```
>> 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.

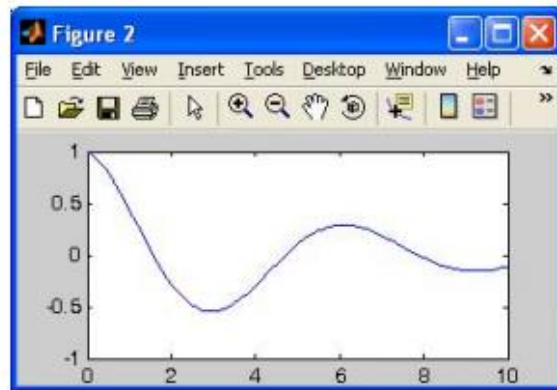
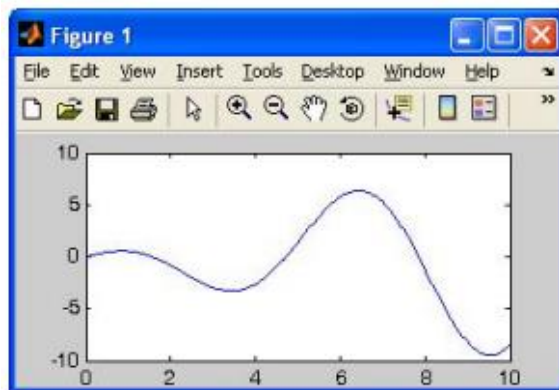


Figure Windows can be closed with the close command. Several forms of the command are:

`close` → closes the active Figure Window.

`close (n)` → closes the nth Figure Window, n is the Figure number.

`close all` → closes all Figure Windows that are open.

Examples

- 1) Generate an overlay plot for plotting three lines

$$\begin{aligned}y_1 &= \sin t \\y_2 &= t \\y_3 &= t - \frac{t^3}{3!} + \frac{t^5}{5!} - \frac{t^7}{7!};\end{aligned}$$

Where $t = 0 \leq t \leq 2\pi$, step 0.06. Using

- a. the plot command
- b. the hold command
- c. the line command

Solution:

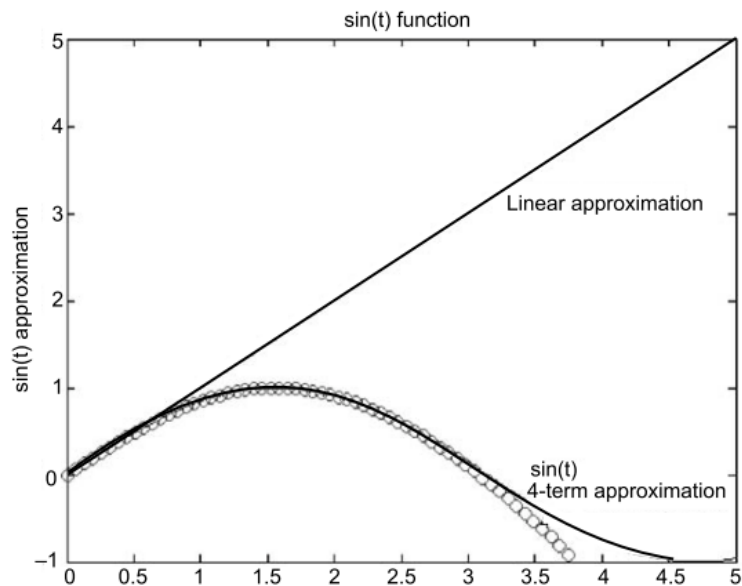
```
% a. using plot command
t=0:0.06:2*pi;
y1=sin(t);
y2=t;
y3=t-(t.^3)/6+(t.^5)/120-(t.^7)/5040;
plot(t,y1,t,y2,'-',t,y3,'o')
axis([0 5 -1 5])
```

```

xlabel('t')
ylabel('sin(t) approximation')
title('sin(t) function')
grid
text(3.5,0,'sin(t)')
gtext('Linear approximation')
gtext('4-term approximation')

```

The output is shown in Figure below:



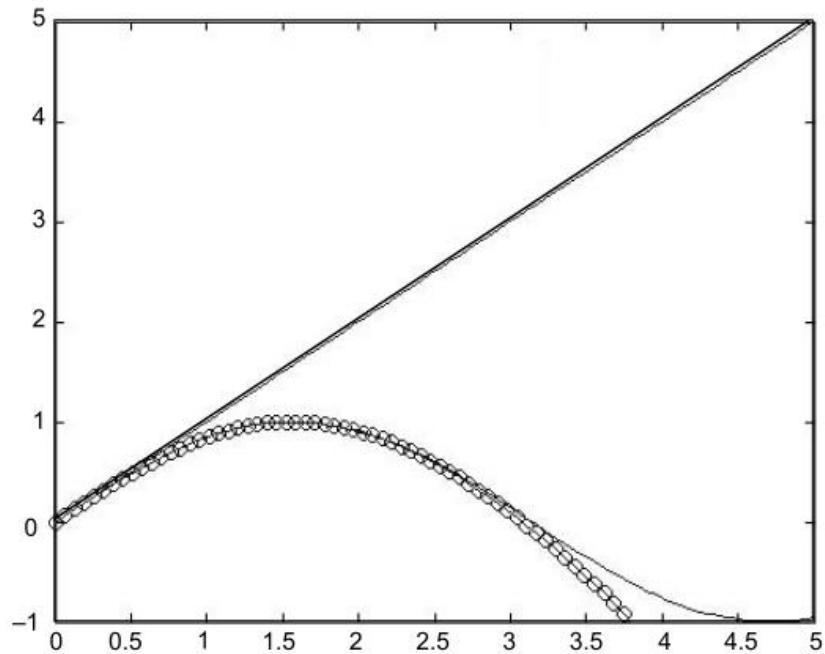
```

% b. Using the hold command
x=0:0.06:2*pi;
y1=sin(x);
plot(x,y1)
hold on
y2= x;
plot(x,y2,'-')
y3=x-(x.^3)/6+(x.^5)/120-(x.^7)/5040;
plot(x,y3,'o')

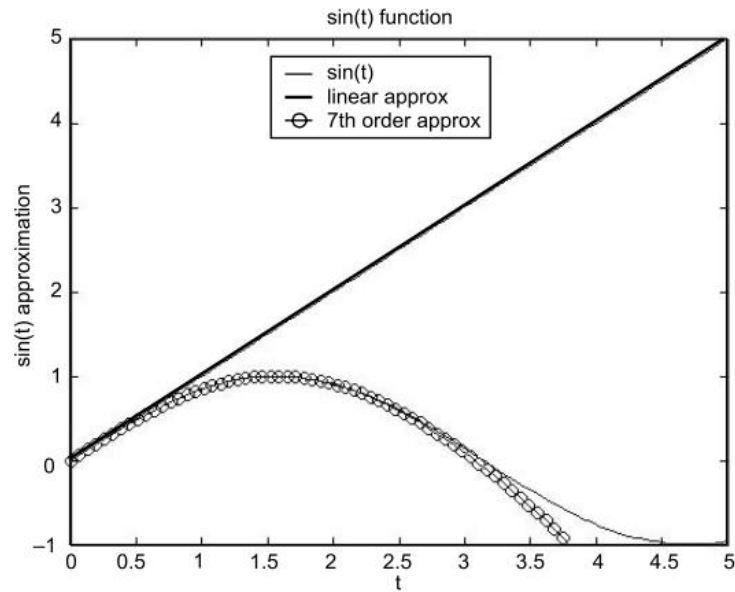
```

```
axis([0 5 -1 5])  
hold off
```

The output is shown in Figure below



```
% c. using the line command  
t=0:0.06:2*pi;  
y1=sin(t);  
y2=t;  
y3=t-(t.^3)/6+(t.^5)/120-(t.^7)/5040;  
plot(t,y1)  
line(t,y2,'linestyle','-')  
line(t,y3,'marker','o')  
axis([0 5 -1 5])  
xlabel('t')  
ylabel('sin(t) approximation')  
title('sin(t) function')  
legend('sin(t)', 'linear approx', '7th order approx')
```



2) Use the functions for plotting x-y data for plotting the following functions:

(i) $f(t) = t \cos(t)$ $0 \leq t \leq 10\pi$

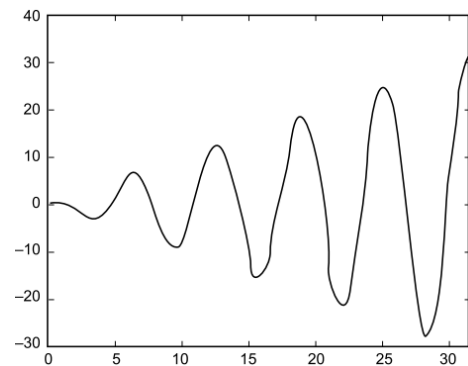
(ii) $x = e^t$

$y = 100 + e^{3t}$ $0 \leq t \leq 2\pi$ step 0.06

Solution:

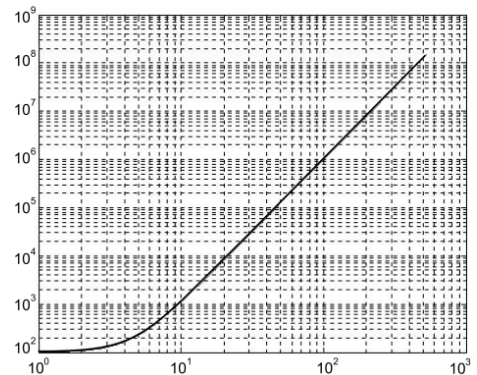
(i)

```
fplot('x*cos(x)', [0 10*pi])
```



(ii)

```
t=0:0.06:2*pi;
x=exp(t);
y=100+exp(3*t);
loglog(x, y)
grid
```

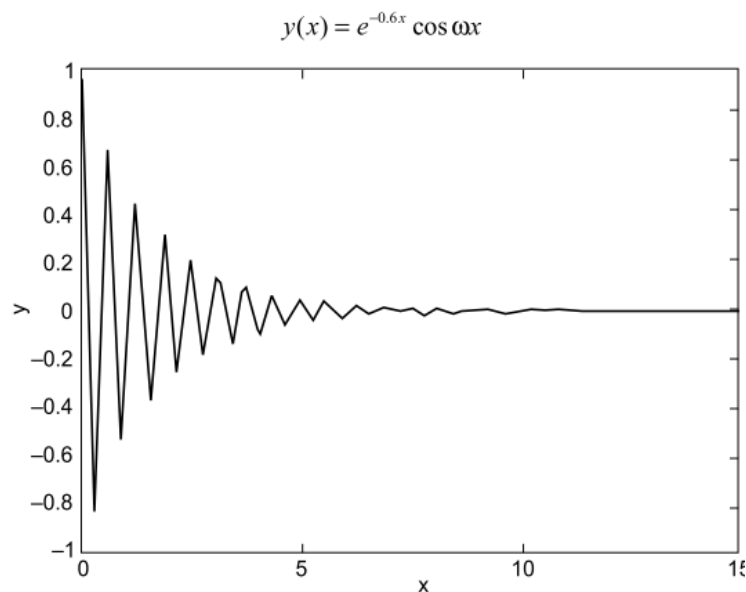


- 3) Generate a plot of $y(x) = e^{-0.6x} \cos \omega x$, where $\omega = 10$ rad/s, and $0 \leq x \leq 15$. Use the colon notation to generate the x vector in increments of 0.05.

Solution:

```
x=[0:0.1:15];  
  
w=10;  
  
y=exp(-0.6*x).*cos(w*x);  
  
plot(x,y)  
  
title('y(x)=e^-0.6^xcos\omegax')  
  
plot(x,y)  
  
title('y(x)=e^-0.6^xcos\omegax')  
  
xlabel('x')  
  
ylabel('y')
```

The output plot is as following:



Exercises

1. Make two separate plots of the function $f(x) = 0.01x^4 - 0.45x^2 + 0.5x - 2$; one plot for $-4 \leq x \leq 4$, and one for $-8 \leq x \leq 8$.
2. Plot the function $f(t) = \frac{5}{1 + e^{5.5 - 1.5x}} - \frac{x^2}{20}$ for $-10 \leq x \leq 10$.
3. Use the `fplot` command to plot the function:

$$f(x) = \frac{40}{1 + (x - 4)^2} + 5 \sin\left(\frac{20x}{\pi}\right)$$
in the domain $0 \leq x \leq 10$.
4. Plot the function $f(x) = \frac{x^2 - 4x - 5}{x - 2}$ for $-4 \leq x \leq 8$. Notice that the function has a vertical asymptote at $x = 2$. Plot the function by creating two vectors for the domain of x , the first vector (call it $x1$) with elements from -4 to 1.7 , and the second vector (call it $x2$) with elements from 2.3 to 8 . For each x vector create a y vector (call them $y1$ and $y2$) with the corresponding values of y according to the function. To plot the function make two curves in the same plot ($y1$ vs. $x1$, and $y2$ vs. $x2$).
5. Plot the function $f(x) = \frac{4x - 30}{x^2 - 3x - 10}$ for $-10 \leq x \leq 10$. Notice that the function has two vertical asymptotes. Plot the function by dividing the domain of x into three parts: one from -10 to near the left asymptote, one between the two asymptotes, and one from near the right asymptote to 10 . Set the range of the y -axis from -20 to 20 .
6. Plot the function $f(x) = 3x \cos^2 x - 2x$ and its derivative, both on the same plot, for $-2\pi \leq x \leq 2\pi$. Plot the function with a solid line, and the derivative with a dashed line. Add a legend and label the axes.
7. An electrical circuit that includes a voltage source v_S with an internal resistance r_S and a load resistance R_L is shown in the figure. The power P dissipated in the load is given by:

$$P = \frac{v_S^2 R_L}{(R_L + r_S)^2}$$

Plot the power P as a function of R_L for $1 \leq R_L \leq 10 \Omega$, given that $v_S = 12 \text{ V}$, and $r_S = 2.5 \Omega$.

