

6. Graphics

MATLAB includes good tools for visualization. Basic 2-D plots, fancy 3-D graphics with lighting and color-maps, complete user-control of the graphics objects through *Handle Graphics*, tools for design of sophisticated graphics user-interface, and animation are now part of MATLAB. What is special about MATLAB's graphics facility is its ease of use and expandability. Commands for most garden-variety plotting are simple, easy to use, and intuitive. If you are not satisfied with what you get, you can control and manipulate virtually everything in the graphics window. This, however, requires an understanding of the Handle Graphics, a system of low-level functions to manipulate graphics objects. In this section we take you through the main features of the MATLAB's graphics facilities.

6.1 Basic 2-D Plots

The most basic and perhaps the most useful command for producing a simple 2-D plot is

```
plot(xvalues, yvalues, 'style-option')
```

where *xvalues* and *yvalues* are vectors containing the *x*- and *y*-coordinates of points on the graph and the *style-option* is an optional argument that specifies the color, the line style (e.g. solid, dashed, dotted, etc.), and the point-marker style (e.g. o, +, *, etc.). All the three options can be specified together as the *style-option* in the general form:

color_linestyle_markerstyle

The two vectors *xvalues* and *yvalues* MUST have the same length. Unequal length of the two vectors is the most common source of error in the plot command. The `plot` function also works with a single vector argument, in

which case, the elements of the vector are plotted against row or column indices. Thus, for two column vectors x and y each of length n ,

<code>plot(x,y)</code>	plots y vs. x with a solid line (the default line style),
<code>plot(x,y,'--')</code>	plots y vs. x with a dashed line (more on this below),
<code>plot(x)</code>	plots the elements of x against their row index.

6.1.1 Style options

The *style-option* in the plot command is a character string that consists of 1, 2, or 3 characters that specify the color and/or the line style. There are several color, line-style, and marker-style options:

<i>Color Style-option</i>	<i>Line Style-option</i>	<i>Marker Style-option</i>
y yellow	- solid	+ plus sign
m magenta	-- dashed	o circle
c cyan	: dotted	* asterisk
r red	-. dash-dot	x x-mark
g green		. point
b blue		^ up triangle
w white		s square
k black		d diamond, etc.

The *style-option* is made up of either the color option, the line-style option, or a combination of the two.

Examples:

<code>plot(x,y,'r')</code>	plots y vs. x with a red solid line,
<code>plot(x,y,':')</code>	plots y vs. x with a dotted line,
<code>plot(x,y,'b--')</code>	plots y vs. x with a blue dashed line,
<code>plot(x,y,'+')</code>	plots y vs. x as unconnected points marked by +.

When no style option is specified, MATLAB uses the default option—a blue solid line.

6.1.2 Labels, title, legend, and other text objects

Plots may be annotated with `xlabel`, `ylabel`, `title`, and `text` commands.

The first three commands take string arguments, while the last one requires three arguments—`text(x-coordinate, y-coordinate, 'text')`, where the coordinate values are taken from the current plot. Thus,

```
xlabel('Pipe Length')      labels the x-axis with Pipe Length,
ylabel('Fluid Pressure')   labels the y-axis with Fluid Pressure,
```

`title('Pressure Variation')` titles the plot with `Pressure Variation`,
`text(2,6,'Note this dip')` writes 'Note this dip' at the location
 (2.0,6.0) in the plot coordinates.

We have already seen an example of `xlabel`, `ylabel`, and `title` in Fig. 3.10. An example of `text` appears in Fig. 6.2. The arguments of `text(x,y,'text')` command may be vectors, in which case x and y must have the same length and `text` may be just one string or a vector of strings. If `text` is a vector then it must have the same length as x and, of course, like any other string vector, must have each element of the same length. A useful variant of the `text` command is `gtext`, which only takes string argument (a single string or a vector of strings) and lets the user specify the location of the text by clicking the mouse at the desired location in the graphics window.

Legend:

The `legend` command produces a boxed legend on a plot, as shown, for example, in Fig. 6.3 on page 166. The `legend` command is quite versatile. It can take several optional arguments. The most commonly used forms of the command are listed below.

<code>legend(string1, string2, ...)</code>	produces legend using the text in <code>string1</code> , <code>string2</code> , etc. as labels.
<code>legend(LineStyle1, string1, ...)</code>	specifies the line-style of each label.
<code>legend(..., pos)</code>	writes the legend outside the plot-frame if <code>pos = -1</code> and inside the frame if <code>pos =</code> (There are other options for <code>pos</code> too.)
<code>legend off</code>	deletes the legend from the plot.

When MATLAB is asked to produce a legend, it tries to find a place on the plot where it can write the specified legend without running into lines, grid, and other graphics objects. The optional argument `pos` specifies the location of the legend box. `pos=1` places the legend in the upper right hand corner (default), 2 in the upper left hand corner, 3 in the lower left hand corner, and 4 in the lower right hand corner. The user, however, can move the legend at will with the mouse (click and drag). For more information, see the on-line help on `legend`.

6.1.3 Axis control, zoom-in, and zoom-out

Once a plot is generated you can change the axes limits with the `axis` command. Typing

`axis([xmin xmax ymin ymax])`

changes the current axes limits to the specified new values $xmin$ and $xmax$ for the x-axis and $ymin$ and $ymax$ for the y-axis.

Examples:

```
axis([-5 10 2 22]); sets the x-axis from -5 to 10, y-axis from 2 to 22.  
axy = [-5 10 2 22]; axis(axy); same as above.  
ax = [-5 10]; ay = [2 22]; axis([ax ay]); same as above.
```

The `axis` command may thus be used to zoom-in on a particular section of the plot or to zoom-out¹. There are also some useful predefined string arguments for the `axis` command:

<code>axis('equal')</code>	sets equal scale on both axes
<code>axis('square')</code>	sets the default rectangular frame to a square
<code>axis('normal')</code>	resets the axis to default values
<code>axis('axis')</code>	freezes the current axes limits
<code>axis('off')</code>	removes the surrounding frame and the tick marks.

The `axis` command must come after the `plot` command to have the desired effect.

Semi-control of axes

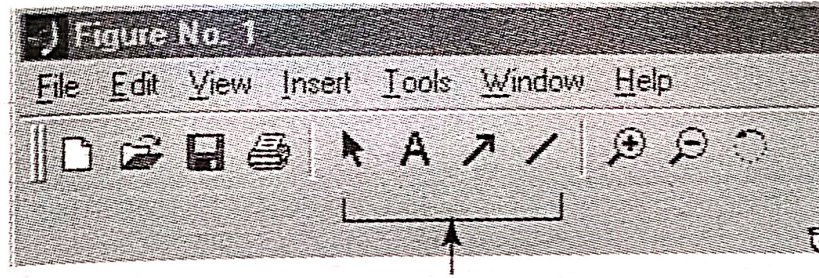
It is possible to control only part of the axes limits and let MATLAB set the other limits automatically. This is achieved by specifying the desired limits in the `axis` command along with `inf` as the values of the limits which you would like to be set automatically. For example,

<code>axis([-5 10 -inf inf])</code>	sets the x-axis limits at -5 and 10 and lets the y-axis limits be set automatically.
<code>axis([-5 inf -inf 22])</code>	sets the lower limit of the x-axis and the upper limit of the y-axis, and leaves the other two limits to be set automatically.

6.1.4 Modifying plots with *Plot Editor*

MATLAB 6 provides an enhanced (over previous versions) interactive tool for modifying an existing plot. To activate this tool, go to the **Figure** window and click on the left-leaning *arrow* in the menu bar (see Fig. 6.1). Now you can select and double (or right) click on any object in the current plot to edit it. Double clicking on the selected object brings up a **Property Editor** window where you can select and modify the current properties of the object. Other tools in the menu bar, e.g., text (marked by **A**), arrow, and line, lets you modify and annotate figures just like simple graphics packages do.

¹There is also a `zoom` command which can be used to zoom-in and zoom-out using the mouse in the figure window. See the on-line help on `zoom`.



Plot editing tools

Figure 6.1: MATLAB provides interactive plot editing tools in the Figure window menu bar. Select the first arrow (left-leaning) for activating plot editor. Select A, the right-leaning arrow, and the diagonal line for adding text, arrows, and lines, respectively, in the current plot.

You can also activate the plot editor in the figure window by typing `plottedit` on the command prompt. You can activate the property editor by typing `propedit` at the command prompt. However, to make good use of the property editor, you must have some understanding of Handle Graphics. See Section 6.4 on page 190 for details.

6.1.5 Overlay plots

There are three different ways of generating overlay plots in MATLAB: the `plot`, `hold`, and `line` commands.

Method-1: Using the `plot` command to generate overlay plots

If the entire set of data is available, `plot` command with multiple arguments may be used to generate an overlay plot. For example, if we have three sets of data— (x_1, y_1) , (x_2, y_2) , and (x_3, y_3) —the command `plot(x1,y1, x2,y2, ':', x3,y3, 'o')` plots (x_1, y_1) with a solid line, (x_2, y_2) with a dotted line, and (x_3, y_3) as unconnected points marked by small circles ('o'), all on the same graph (See Fig. 6.2 for example). Note that the vectors (x_i, y_i) must have the same length pairwise. If the length of all vectors is the same, then it is convenient to make a matrix of x vectors and a matrix of y vectors and then use the two matrices as the argument of the `plot` command. For example, if x_1, y_1, x_2, y_2, x_3 , and y_3 are all column vectors of length n then typing `X=[x1 x2 x3]; Y=[y1 y2 y3]; plot(X,Y)` produces a plot with three lines drawn in different colors. When `plot` command is used with matrix arguments, each column of the second argument matrix is plotted against the corresponding column of the first argument matrix.


```
>> t=linspace(0,2*pi,100);
>> y1=sin(t); y2=t;
>> y3=t-(t.^3)/6+(t.^5)/120;
>> plot(t,y1,t,y2,'--',t,y3,'o')

>> axis([0 5 -1 5])
>> xlabel('t')
>> ylabel('Approximations of sin(t)')
>> title('Fun with sin(t)')
>> text(3.5,0,'sin(t)')
>> gtext('Linear approximation')
>> gtext('First 3 terms')
>> gtext('in Taylor series')
```

```
% Generate vector t
% Calculate y1, y2, y3

% Plot (t,y1) with solid line
%- (t,y2) with dashed line and
%- (t,y3) with circles
% Zoom-in with new axis limits
% Put x-label
% Put y-label
% Put title
% Write 'sin(t)' at point (3.5,0)
```

gtext writes the specified string at a location clicked with the mouse in the graphics window. So after hitting return at the end of gtext command, go to the graphics window and click a location.

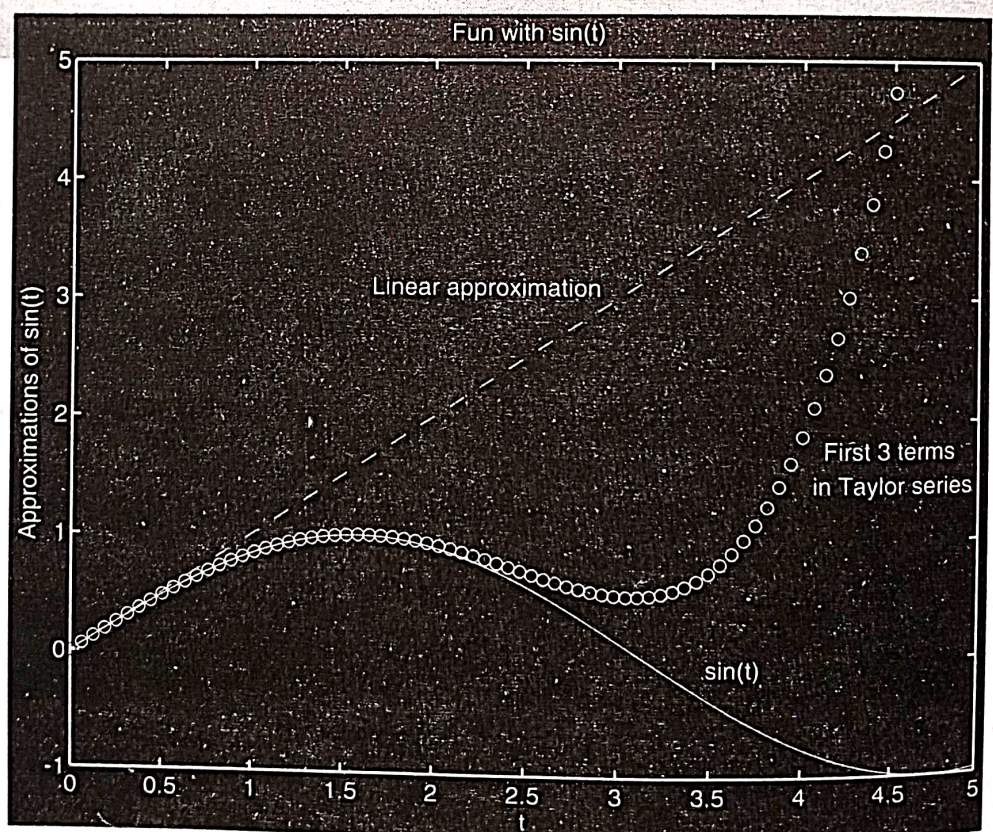


Figure 6.2: Example of an overlay plot along with examples of xlabel, ylabel, title, axis, text, and gtext commands. The three lines plotted are $y_1 = \sin t$, $y_2 = t$, and $y_3 = t - \frac{t^3}{3!} + \frac{t^5}{5!}$.

Method-2: Using the hold command to generate overlay plots

Another way of making overlay plots is with the hold command. Invoking hold on at any point during a session freezes the current plot in the graphics window. All subsequent plots generated by the plot command are simply added to the existing plot. The following script file shows how to generate the same plot as in Fig. 6.2 by using the hold command.

```
% - Script file to generate an overlay plot with the hold command -
x=linspace(0,2*pi,100);      % Generate vector x
y1=sin(x);                   % Calculate y1
plot(x,y1)                   % Plot (x,y1) with solid line
hold on                       % Invoke hold for overlay plots
y2=x; plot(x,y2,'--')        % Plot (x,y2) with dashed line
y3=x-(x.^3)/6+(x.^5)/120;    % Calculate y3
plot(x,y3,'o')               % Plot (x,y3) as pts. marked by 'o'
axis([0 5 -1 5])             % Zoom-in with new axis limits
hold off                      % Clear hold command
```

The hold command is useful for overlay plots when the entire data set to be plotted is not available at the same time. You should use this command if you want to keep adding plots as the data becomes available. For example, if a set of calculations done in a for loop generates vectors x and y at the end of each loop and you would like to plot them on the same graph, hold is the way to do it.

Method-3: Using the line command to generate overlay plots

The line is a low-level graphics command which is used by the plot command to generate lines. Once a plot exists in the graphics window, additional lines may be added by using the line command directly. The line command takes a pair of vectors (or a triplet in 3-D) followed by *parameter name/parameter value* pairs as arguments:

`line(xdata, ydata, ParameterName, ParameterValue)`

This command simply adds lines to the existing axes. For example, the overlay plot created by the above script file could also be created with the following script file, which uses the line command instead of the hold command. As a bonus to the reader, we include an example of the legend command (see page 161).

```
% -- Script file to generate an overlay plot with the line command --
t=linspace(0,2*pi,100);      % Generate vector t
y1=sin(t);                   % Calculate y1, y2, y3
y2=t;
y3=t-(t.^3)/6+(t.^5)/120;
```



```

plot(t,y1)                % Plot (t,y1) with (default) solid line
line(t,y2,'linestyle','--') % Add line (t,y2) with dahed line and
line(t,y3,'marker','o')    % Add line (t,y3) plotted with circles

axis([0 5 -1 5])          % Zoom-in with new axis limits

xlabel('t')                % Put x-label
ylabel('Approximations of sin(t)') % Put y-label
title('Fun with sin(t)')   % Put title

legend('sin(t)', 'linear approx.', '5th order approx.')
                                % add legend

```

The output generated by the above script file is shown in Fig. 6.3. After generating the plot, click and hold the mouse on the legend rectangle and see if you can drag the legend to some other position. Alternatively, you could specify an option in the legend command to place the legend rectangle in any of the four corners of the plot. See the on-line help on `legend`.

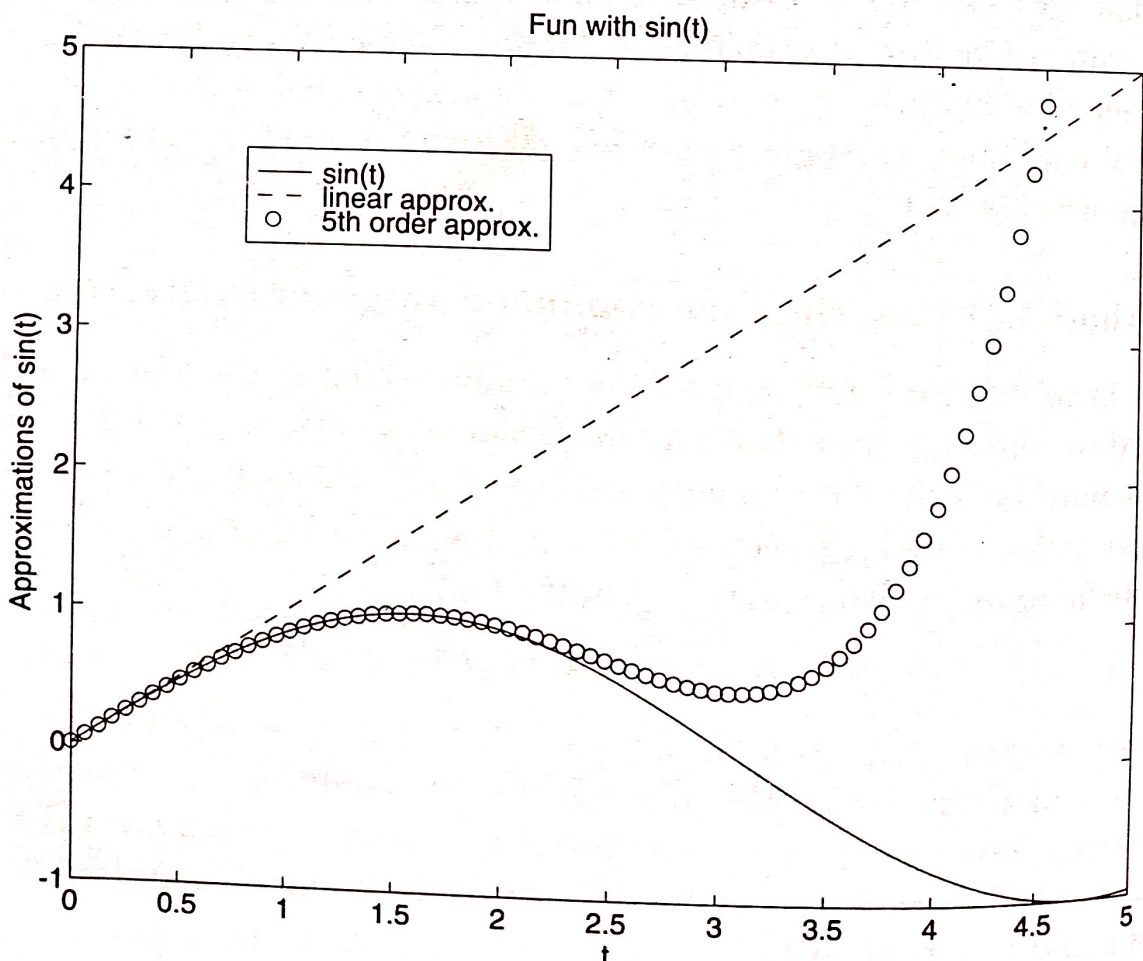


Figure 6.3: Example of an overlay plot produced by using the `line` command. The legend is produced by the `legend` command. See the script file for details.

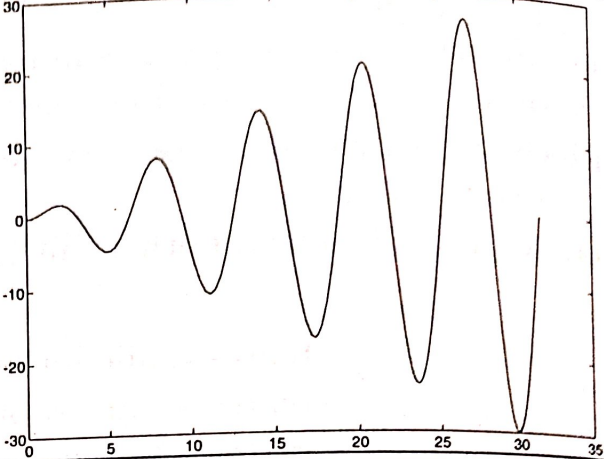
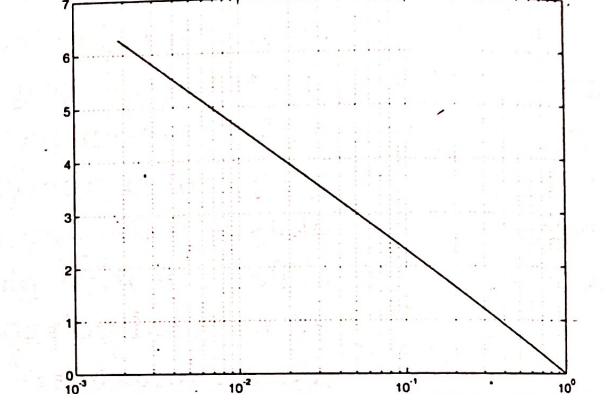
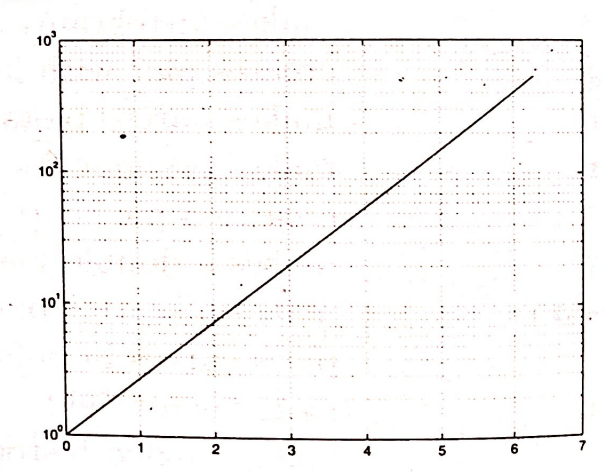
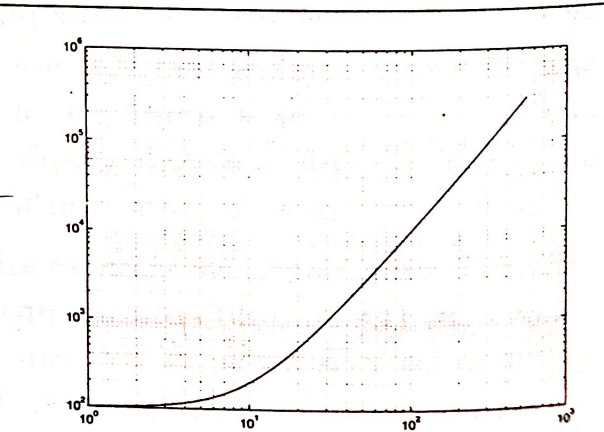
6.1.6 Specialized 2-D plots

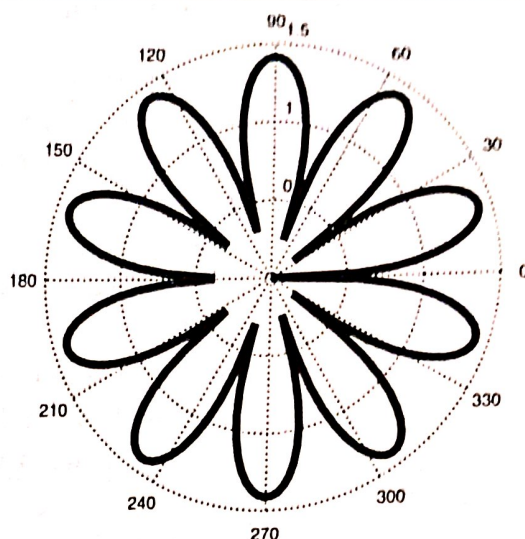
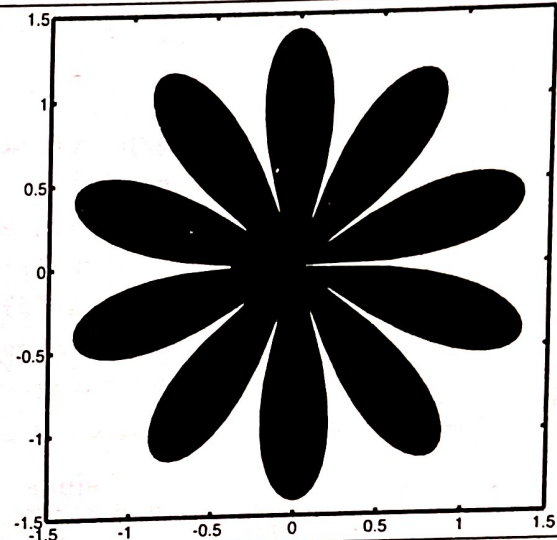
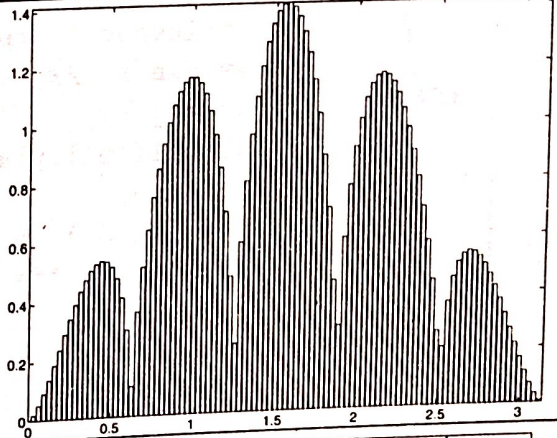
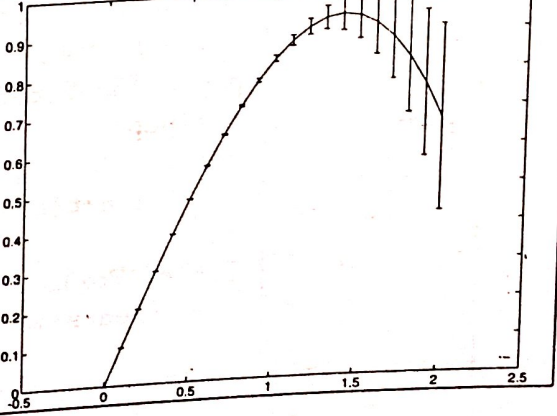
There are many specialized graphics functions for 2-D plotting. They are used as alternatives to the `plot` command we have just discussed. There is a whole suite of *ezplotter* functions, such as `ezplot`, `ezpolar`, `ezcontour`, etc., which are truly easy to use. See Section 3.6 for a discussion and examples of these functions.

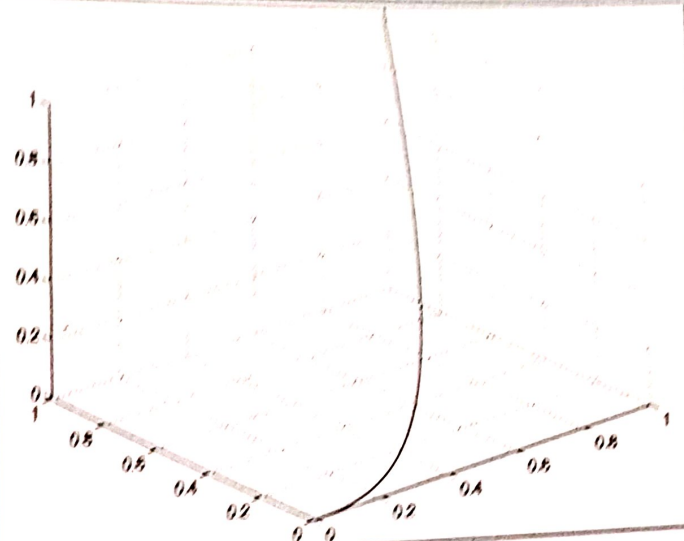
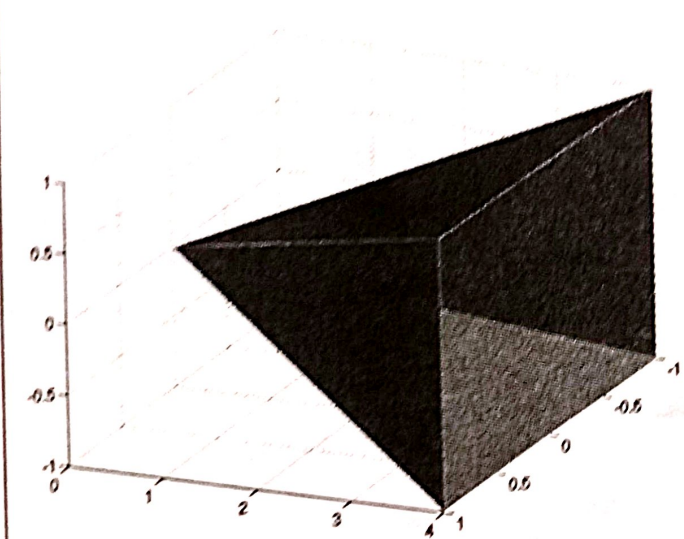
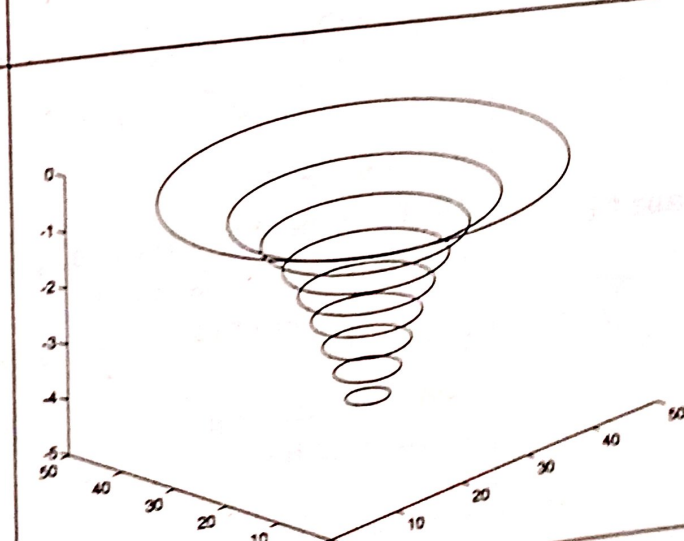
Here, we provide a list of other functions commonly used for plotting x - y data:

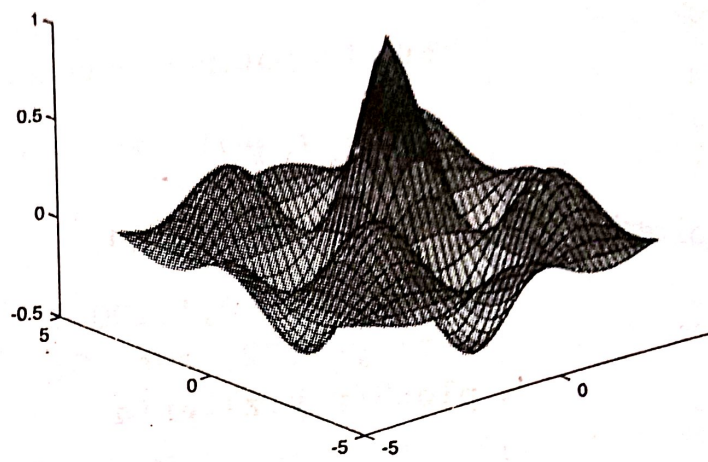
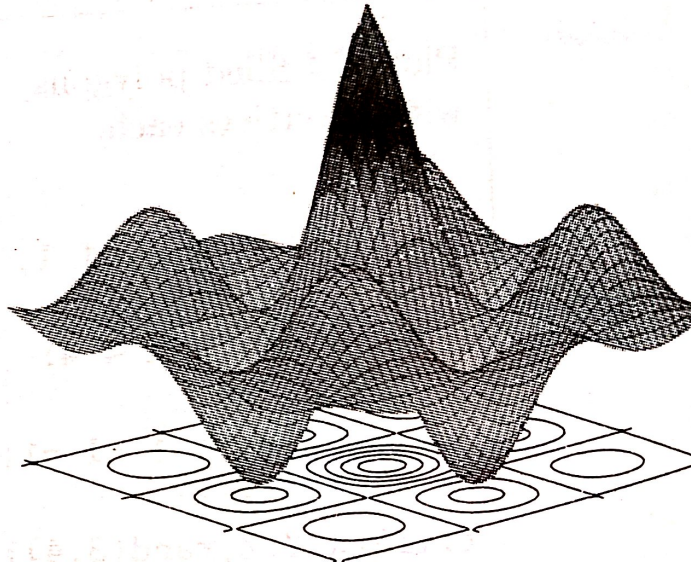
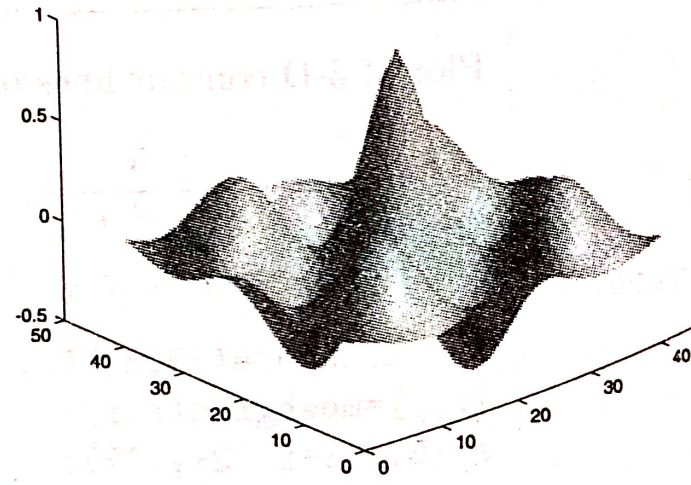
<code>area</code>	creates a filled area plot
<code>bar</code>	creates a bar graph
<code>barh</code>	creates a horizontal bar graph
<code>comet</code>	makes an animated 2-D plot
<code>compass</code>	creates arrow graph for complex numbers
<code>contour</code>	makes contour plots
<code>contourf</code>	makes filled contour plots
<code>errorbar</code>	plots a graph and puts error bars
<code>feather</code>	makes a feather plot
<code>fill</code>	draws filled polygons of specified color
<code>fplot</code>	plots a function of a single variable
<code>hist</code>	makes histograms
<code>loglog</code>	creates plot with log scale on both x and y axes
<code>pareto</code>	makes pareto plots
<code>pcolor</code>	makes pseudocolor plot of a matrix
<code>pie</code>	creates a pie chart
<code>plotyy</code>	makes a double y -axis plot
<code>plotmatrix</code>	makes a scatter plot of a matrix
<code>polar</code>	plots curves in polar coordinates
<code>quiver</code>	plots vector fields
<code>rose</code>	makes angled histograms
<code>scatter</code>	creates a scatter plot
<code>semilogx</code>	makes semilog plot with log scale on the x -axis
<code>semilogy</code>	makes semilog plot with log scale on the y -axis
<code>stairs</code>	plots a stair graph
<code>stem</code>	plots a stem graph

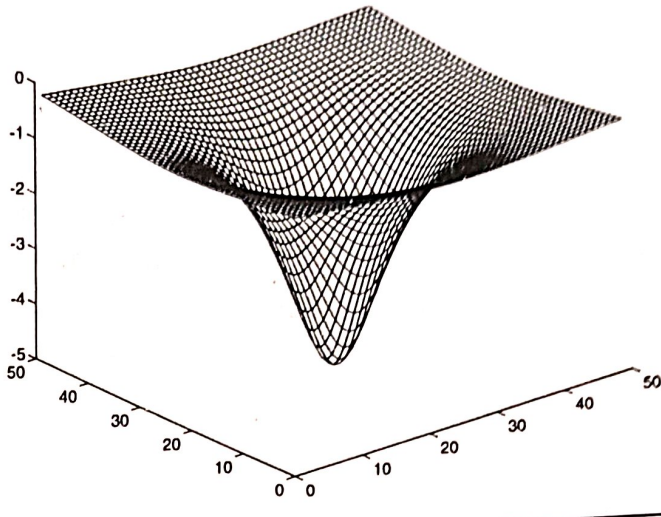
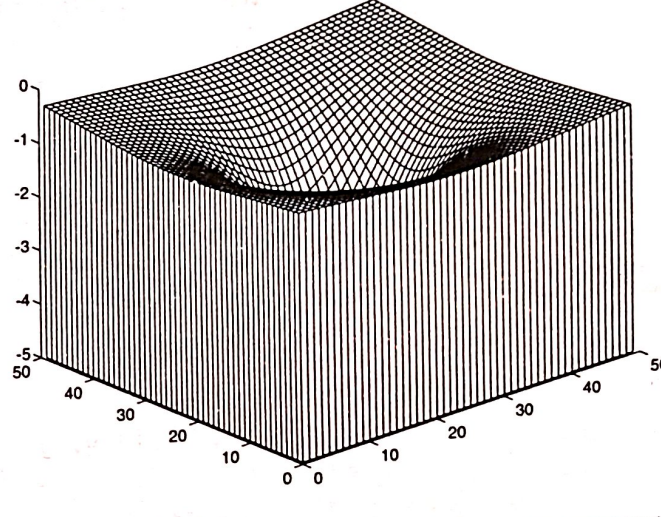
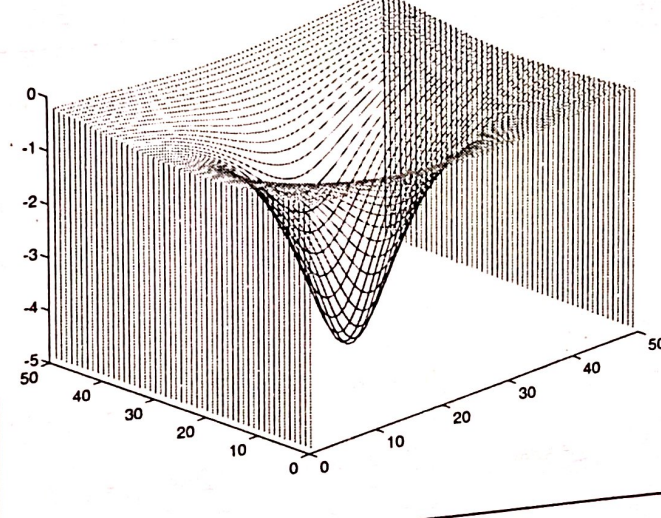
On the following pages we show examples of these functions. The commands shown in the middle column produce the plots shown in the right column. There are several ways you can use these graphics functions. Also, many of them take optional arguments. The following examples should give you a basic idea of how to use these functions and what kind of plot to expect from them. For more information on any of these functions see the on-line help.

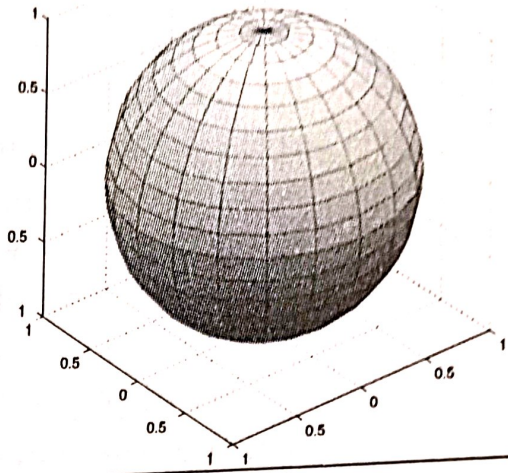
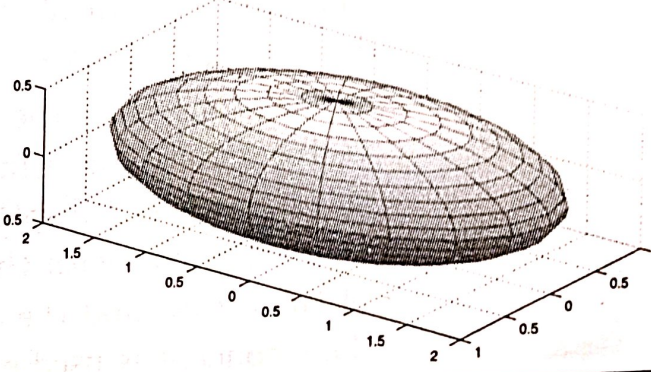
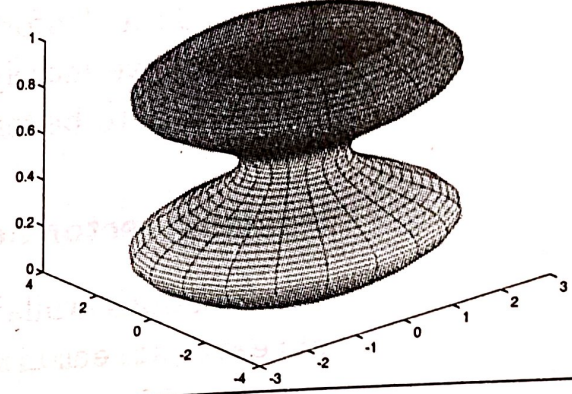
Function	Example Script	Output
fplot	$f(t) = t \sin t, \quad 0 \leq t \leq 10\pi$ <pre>fplot('x.*sin(x)',[0 10*pi])</pre> <p>Note that the function to be plotted must be written as a function of x.</p>	
semilogx	$x = e^{-t}, \quad y = t, \quad 0 \leq t \leq 2\pi$ <pre>t=linspace(0,2*pi,200); x = exp(-t); y = t; semilogx(x,y), grid</pre>	
semilogy	$x = t, \quad y = e^t, \quad 0 \leq t \leq 2\pi$ <pre>t=linspace(0,2*pi,200); semilogy(t,exp(t)) grid</pre>	
loglog	$x = e^t, \quad y = 100 + e^{2t}, \quad 0 \leq t \leq 2\pi$ <pre>t=linspace(0,2*pi,200); x = exp(t); y = 100+exp(2*t); loglog(x,y), grid</pre>	

Function	Example Script	Output
polar	$r^2 = 2 \sin 5t, \quad 0 \leq t \leq 2\pi$ <pre> t=linspace(0,2*pi,200); r=sqrt(abs(2*sin(5*t))); polar(t,r) </pre>	
fill	$r^2 = 2 \sin 5t, \quad 0 \leq t \leq 2\pi$ $x = r \cos t, \quad y = r \sin t$ <pre> t=linspace(0,2*pi,200); r=sqrt(abs(2*sin(5*t))); x=r.*cos(t); y=r.*sin(t); fill(x,y,'k'); axis('square') </pre>	
bar	$r^2 = 2 \sin 5t, \quad 0 \leq t \leq 2\pi$ $y = r \sin t$ <pre> t=linspace(0,2*pi,200); r=sqrt(abs(2*sin(5*t))); y=r.*sin(t); bar(t,y) axis([0 pi 0 inf]); </pre>	
errorbar	$f_{\text{approx}} = x - \frac{x^3}{3!}, \quad 0 \leq x \leq 2$ $\text{error} = f_{\text{approx}} - \sin x$ <pre> x=0:.1:2; aprx2=x-x.^3/6; er=aprx2-sin(x); errorbar(x,aprx2,er) </pre>	

Function	Example Script	Output
plot3	<p>Plot of a parametric space curve:</p> $x(t) = t, y(t) = t^2, z(t) = t^3.$ $0 \leq t \leq 1.$ <pre> t=linspace(0,1,100); x=t; y=t.^2; z=t.^3; plot3(x,y,z),grid </pre>	
fill3	<p>Plot of 4 filled polygons with 3 vertices each.</p> <pre> X=[0 0 0 0; 1 1 -1 1; 1 -1 -1 -1]; Y=[0 0 0 0; 4 4 4 4; 4 4 4 4]; Z=[0 0 0 0; 1 1 -1 -1; -1 1 1 -1]; fill3(X,Y,Z,rand(3,4)) view(120,30) </pre>	
contour3	<p>Plot of 3-D contour lines of</p> $z = -\frac{5}{1+x^2+y^2},$ $ x \leq 3, y \leq 3.$ <pre> r = linspace(-3,3,50); [x,y]=meshgrid(r,r); z=-5./(1+x.^2+y.^2); contour3(z) </pre>	

Function	Example Script	Output
surf	$z = \cos x \cos y e^{\frac{-\sqrt{x^2+y^2}}{4}}$ $ x \leq 5, \quad y \leq 5$ <pre> u = -5:.2:5; [X,Y] = meshgrid(u, u); Z = cos(X).*cos(Y).*... exp(-sqrt(X.^2+Y.^2)/4); surf(X,Y,Z) </pre>	
surfc	$z = \cos x \cos y e^{\frac{-\sqrt{x^2+y^2}}{4}}$ $ x \leq 5, \quad y \leq 5$ <pre> u = -5:.2:5; [X,Y] = meshgrid(u, u); Z = cos(X).*cos(Y).*... exp(-sqrt(X.^2+Y.^2)/4); surfc(Z) view(-37.5,20) axis('off') </pre>	
surfl	$z = \cos x \cos y e^{\frac{-\sqrt{x^2+y^2}}{4}}$ $ x \leq 5, \quad y \leq 5$ <pre> u = -5:.2:5; [X,Y] = meshgrid(u, u); Z = cos(X).*cos(Y).*... exp(-sqrt(X.^2+Y.^2)/4); surfl(Z) shading interp colormap hot </pre>	

Function	Example Script	Output
mesh	$z = -\frac{5}{1+x^2+y^2}$ $ x \leq 3, \quad y \leq 3$ <pre> x = linspace(-3,3,50); y = x; [x,y] = meshgrid(x,y); z=-5./(1+x.^2+y.^2); mesh(z) </pre>	
meshz	$z = -\frac{5}{1+x^2+y^2}$ $ x \leq 3, \quad y \leq 3$ <pre> x = linspace(-3,3,50); y = x; [x,y] = meshgrid(x,y); z=-5./(1+x.^2+y.^2); meshz(z) view(-37.5, 50) </pre>	
waterfall	$z = -\frac{5}{1+x^2+y^2}$ $ x \leq 3, \quad y \leq 3$ <pre> x = linspace(-3,3,50); y = x; [x,y] = meshgrid(x,y); z=-5./(1+x.^2+y.^2); waterfall(z) hidden off </pre>	

Function	Example Script	Output
sphere	<p>A unit sphere centered at the origin and generated by 3 matrices x, y, and z of size 21×21 each.</p> <pre>sphere(20) or [x,y,z]=sphere(20); surf(x,y,z)</pre>	
ellipsoid	<p>An ellipsoid of radii $rx = 1$, $ry = 2$ and $rz = 0.5$, centered at the origin.</p> <pre>cx=0; cy=0; cz=0; rx=1; ry=2; rz=0.5; ellipsoid(cx,cy,cz,rx,ry,rz)</pre>	
cylinder	<p>A cylinder generated by</p> $r = \sin(3\pi z) + 2$ $0 \leq z \leq 1, \quad 0 \leq \theta \leq 2\pi.$ <pre>z=[0:.02:1]'; r=sin(3*pi*z)+2; cylinder(r)</pre>	
slice	<p>Slices of the volumetric function $f(x,y,z) = x^2 + y^2 - z^2$ $x \leq 3$, $y \leq 3$, $z \leq 3$ at $x = -2$ and 2.5, $y = 2.5$, and $z = 0$.</p> <pre>v = [-3:.2:3]; [x,y,z]=meshgrid(v,v,v); f=(x.^2+y.^2-z.^2); xv=[-2 2.5]; yv=2.5; zv=0; slice(x,y,z,f,xv,yv,zv); view([-30 30])</pre> <p>The value of the function is indicated by the color intensity.</p>	