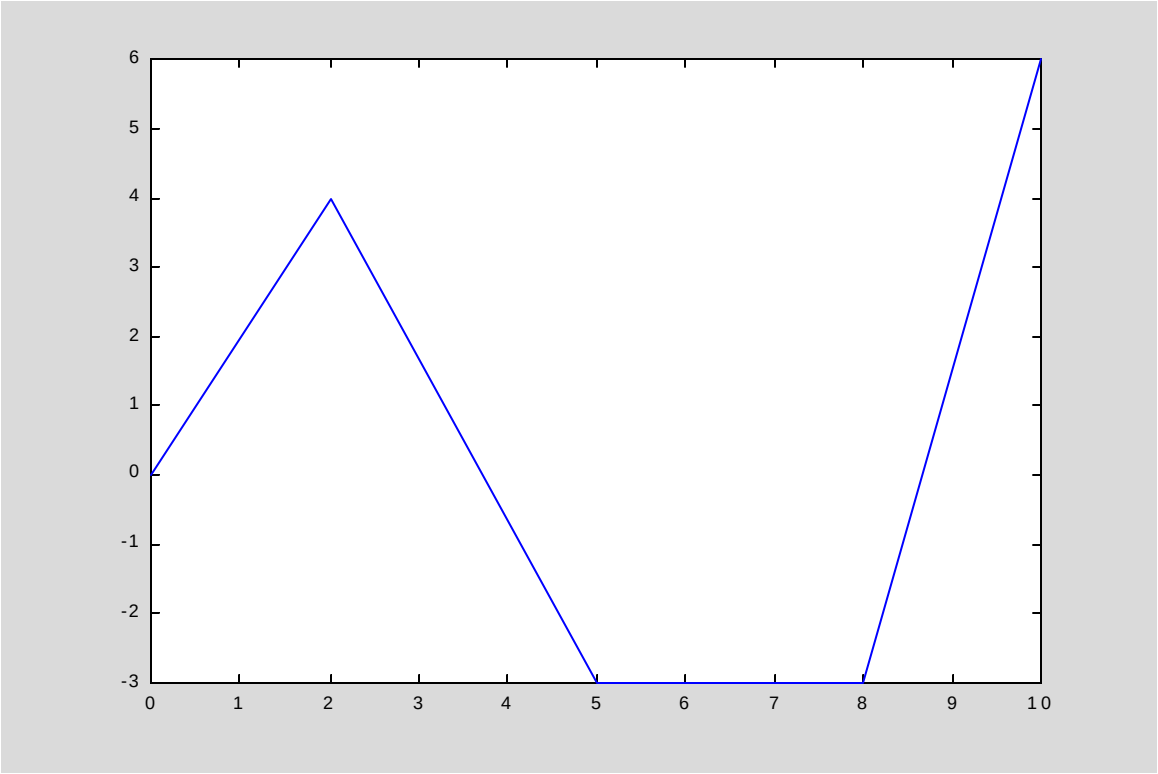
Chapter 26 2-D Graphics

26.1 The `plot` Function

The basic function for producing 2-D graphs is the MATLAB '`plot`' function. It is generally called with two identically sized row (or column) vectors, one for the set of abscissa values and the second for the ordinate axis. The plotted points are ordered pairs of values, one from the first vector and one from the second vector. For example, to plot the points $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ the arrays `x=` $[x_1, x_2, \ldots, x_n]$ and `y=`$[y_1, y_2, \ldots, y_n]$ are created and then passed to the plot function for graphing.

Example 26.1.1

```
x=[0 2 5 8 10]; % Create x vector
y=[0 4 -3 -3 6]; % Create y vector
plot(x,y) % Plot (0,0), (2,4), (5,-3), (8,-3), (10,6)
```

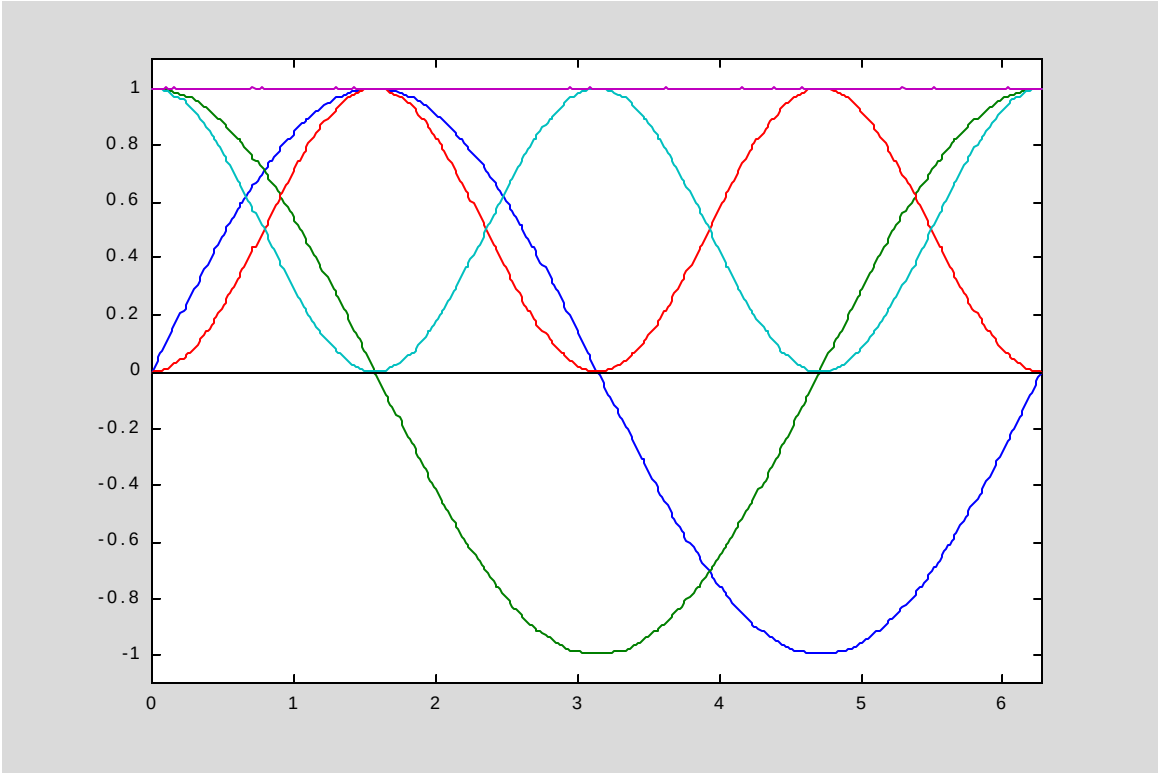


When the '`plot`' command is issued within an m-File or from the Command window, MATLAB opens a new Figure window, draws a set of axes automatically scaled to fit the data, plots the points in the order in which they appear in the two vectors, and then connects all the points by straight lines. Numerical scales appear on both axes as do tick marks.

The `plot'` command will clear the current Figure window (if one is open) before plotting the new data. Several data sets can be plotted from one `'plot'` command by adding additional arguments.

Example 26.1.2

```
x=linspace(0,2*pi,500);
u=sin(x);
y=cos(x);
z1=sin(x).^2;
z2=cos(x).^2;
z=z1+z2;
v=[0 2*pi -1.1 1.1];
plot(x,u,x,y,x,z1,x,z2,x,z)
axis(v)
hold on
plot([0 2*pi],[0 0],'k')
```



Note, if separate `'plot'` commands were issued, i.e. `plot(x,u)`, `plot(x,y)`, etc. only the last data set of points $(x_1,z_1),(x_2,z_2),\ldots..,(x_{500},z_{500})$ would appear in the Figure window.

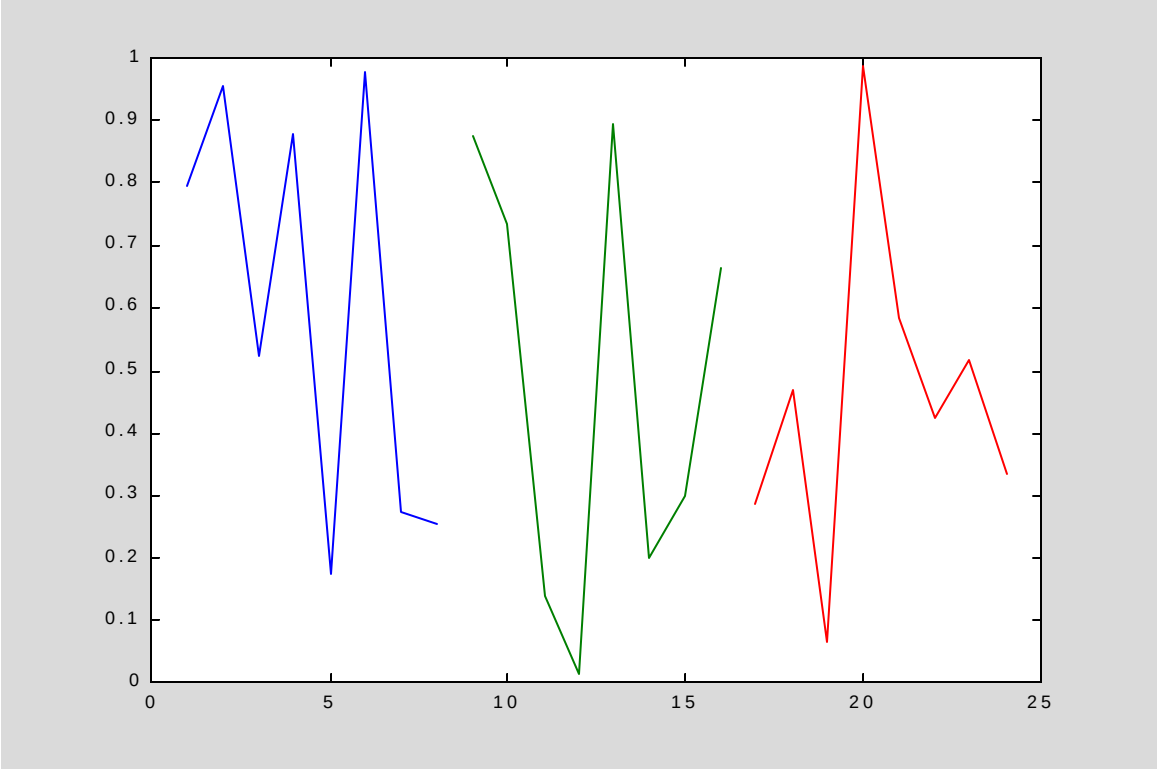
The arguments containing the data points to be plotted are not restricted to row vectors. They may be rectangular arrays provided the number of rows and columns are

the same. The corresponding columns are paired and the data points plotted in the same way when the arguments are one dimensional row (or column) vectors.

Example 26.1.3

```
X=[(1:8)', (9:16)', (17:24)'] % X is a 8 by 3 array
Y=rand(8,3) % Y is a 8 by 3 array of random numbers
plot(X,Y) % Plot 1st column of Y vs 1st column of X, plot 2nd column
 % of Y vs 2nd column of X, Plot 3rd column of Y vs 3rd column of X
```

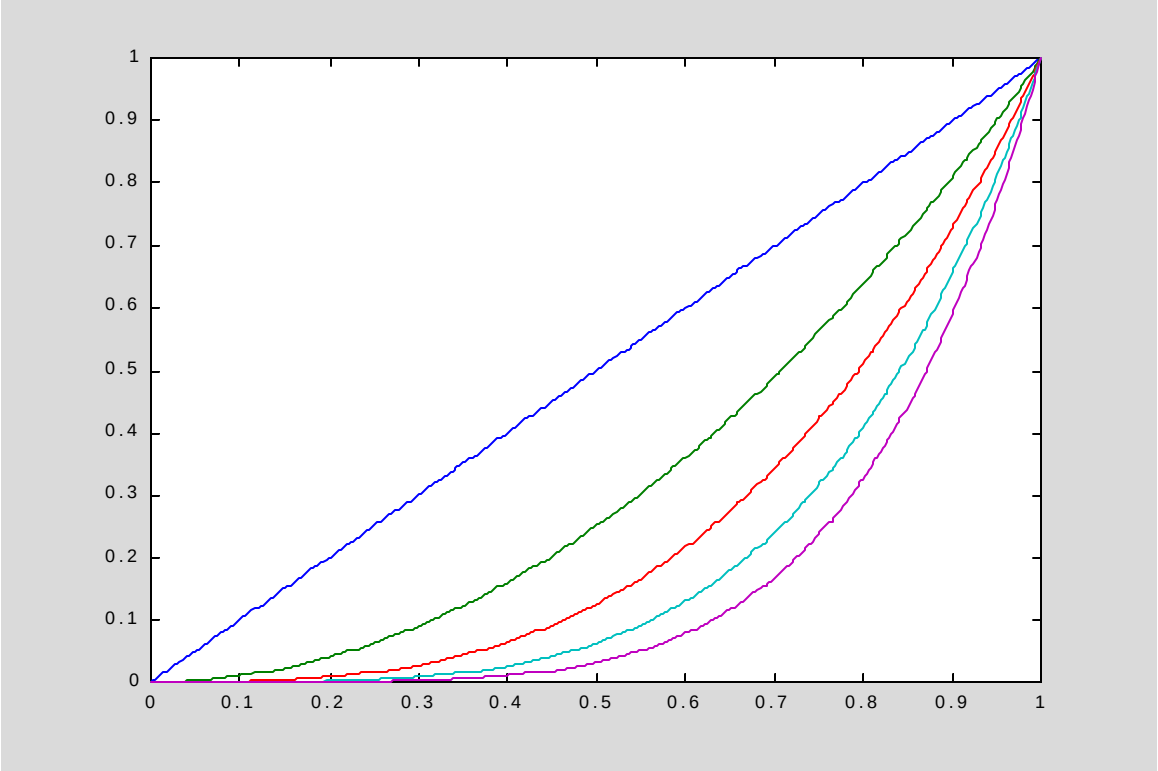
```
X =
     1     9    17
     2    10    18
     3    11    19
     4    12    20
     5    13    21
     6    14    22
     7    15    23
     8    16    24
Y =
    0.7948    0.8757    0.2844
    0.9568    0.7373    0.4692
    0.5226    0.1365    0.0648
    0.8801    0.0118    0.9883
    0.1730    0.8939    0.5828
    0.9797    0.1991    0.4235
    0.2714    0.2987    0.5155
    0.2523    0.6614    0.3340
```



MATLAB also accepts a column vector as the first argument and a recatangular array for the second argument as long as both have the same number of rows. In this case, each column of data in the array is paired with the column vector (first argument) and there is one plot for each combination.

Example 26.1.4

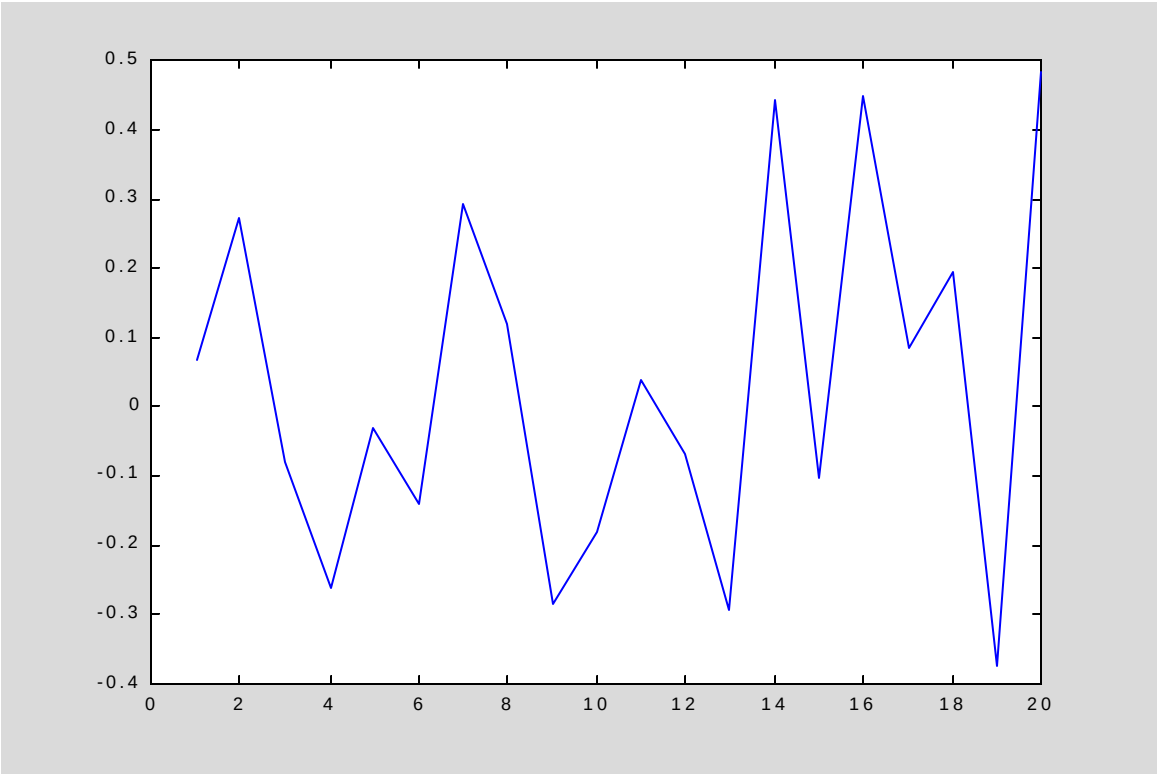
```
x=[linspace(0,1,500)]'; % Create 500´1 column vector x
Y=[x,x.^2,x.^3,x.^4,x.^5]; % Create 500´5 array Y
plot(x,Y) % Plot columns of Y vs column vector x
```



'Plot' can be called with a single argument, say a column vector y of real numbers, and the result is a graph of the elements of y vs. the integers 1, 2, 3, ..., length(y).

Example 26.1.5

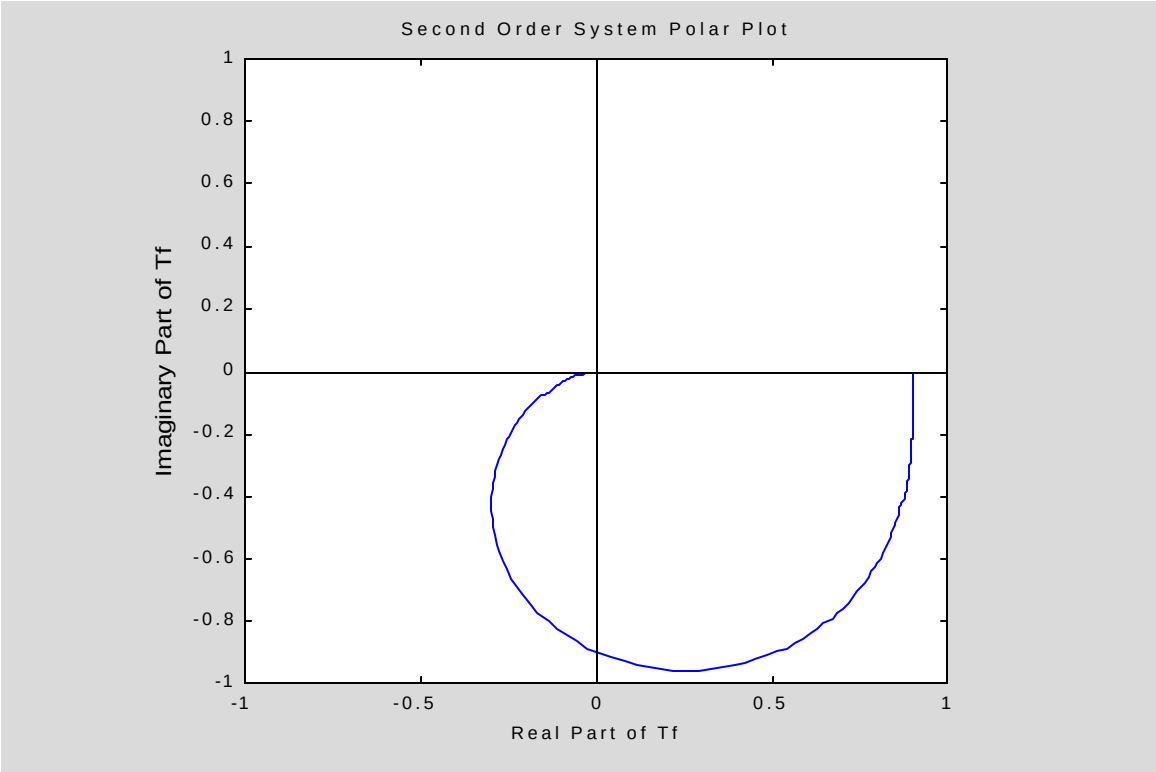
```
y=0.5-rand(20,1); % Create 20´1 column vector y
plot(y) % Plot {1,y(1)}, {2,y(2)}, ....., {20,y(20)}
```



If y is complex, the result is a plot of the imaginary parts of y vs. its real parts.

Example 26.1.5

```
w=logspace(-2,2,500);
s=j*w;
rho=0.5; % Dampling Ratio
wn=10; % Natural frequency
Tf=90./(s.^2 + 2*rho*wn*s + wn*wn); % Second Order System Transfer
                                    % Function
plot(Tf) % Polar Plot
axis([-1 1 -1 1])
axis square
hold on
plot([0,0],[-1,1],'k',[-1,1],[0,0],'k') %
xlabel('Real Part of Tf')
ylabel('Imaginary Part of Tf')
title('Second Order System Polar Plot')
```


Second Order System Polar Plot
Imaginary Part of Tf
Real Part of Tf
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1

26.2 Linestyles, Markers, and Colors

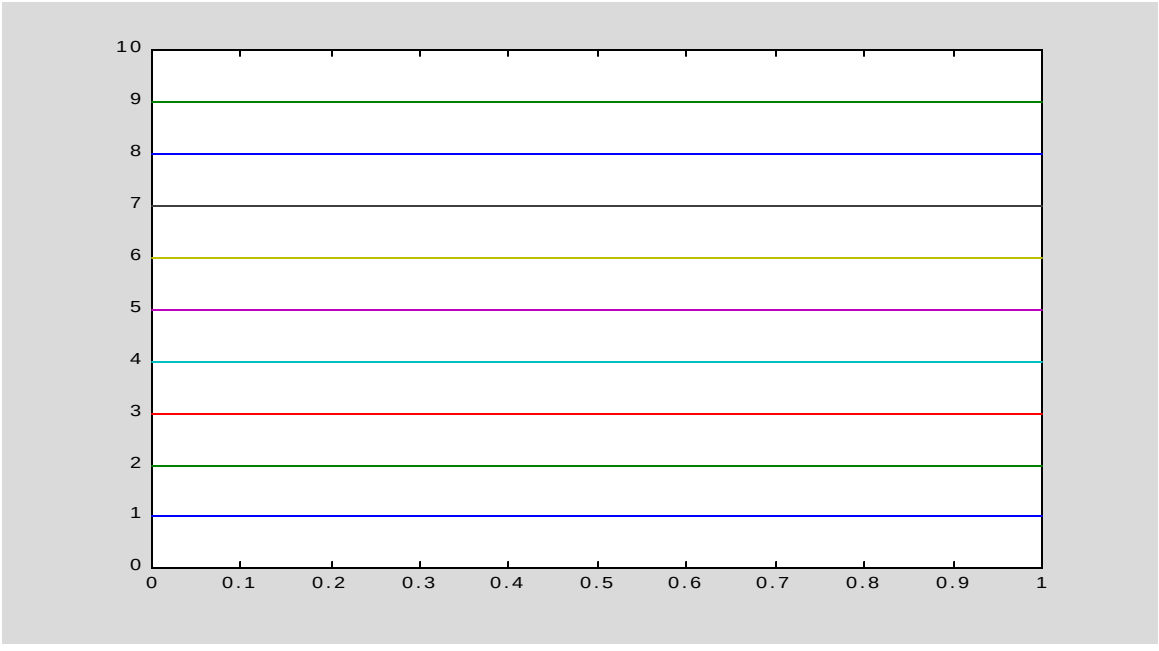
Linestyles, markers and colors of MATLAB plots are controllable by specifying a string argument immediately following the data. The choices are

	Color		Marker		Linestyle
y	yellow	.	point	-	solid
m	magenta	o	circle	:	dotted
c	cyan	x	x-mark	-.	dashdot
r	red	+	plus	--	dashed
g	green	*	star		
b	blue	s	square		
w	white	d	diamond		
k	black	v	triangle (down)		
		^	triangle (up)		
		<	triangle (left)		
		>	triangle (right)		
		p	pentagram		
		h	hexagram		

MATLAB uses a default color scheme to draw multiple curves on the same plot. The order is easily determined as evidenced by the following example.

Example 26.2.1

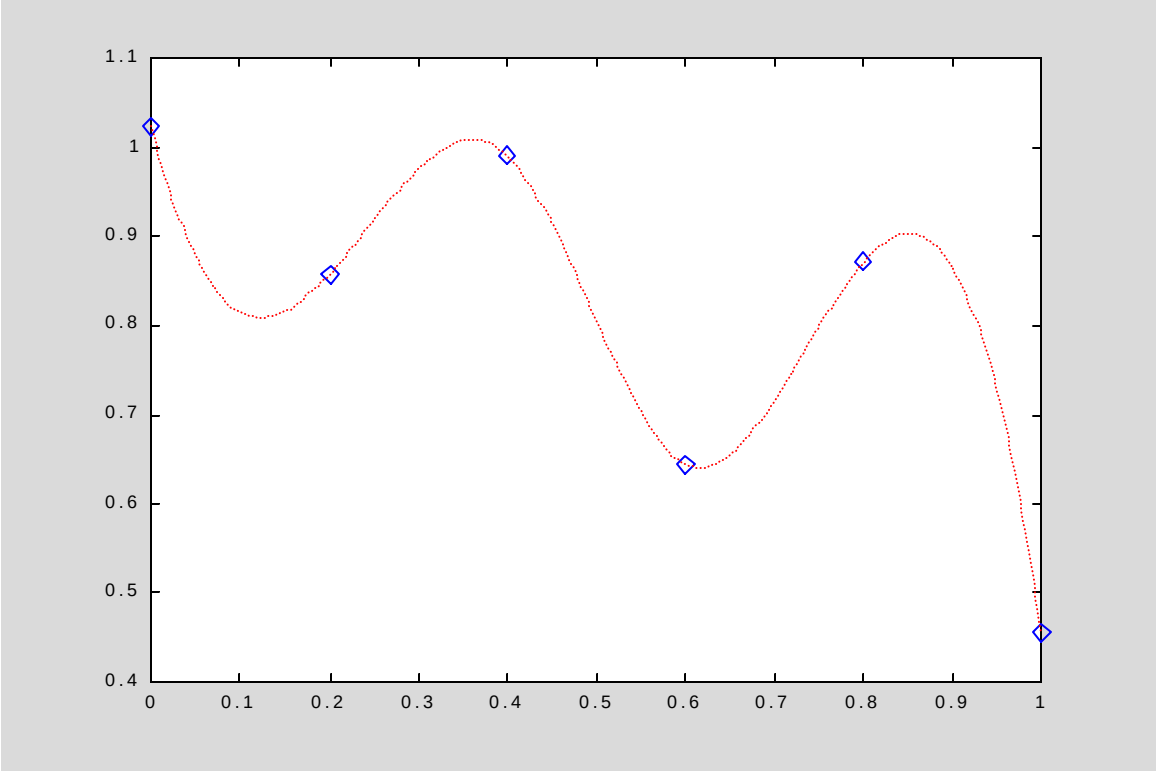
```
x=[0;1];
Y=[[1;1],[2;2],[3;3],[4;4],[5;5],[6;6],[7;7],[8;8],[9;9]];
plot(x,Y)
axis([0 1 0 10])
```



In the next example, a set of data points is shown as blue diamond symbols and the points are connected using cubic splines drawn as a continuous red dotted curve.

Example 26.2.1

```
x=0:0.2:1; % Create x data
y=exp(-x)+rand(1,6)/2; % Create y data
plot(x,y,'bd') % Plot data points as blue diamonds
hold on
xi=linspace(0,1,500);
yi=interp1(x,y,xi,'spline');
plot(xi,yi,'r:') % Plot cubic spline fit as red dotted curve
```



The default line style is a solid line which will be used to connect the points (with straight lines) when there is no marker specified. There is no default marker and none will be drawn unless one is specified. However, when a marker is specified for placement at the data points, the markers are not connected by a straight line unless a line style is given.

26.3 Plotting Styles

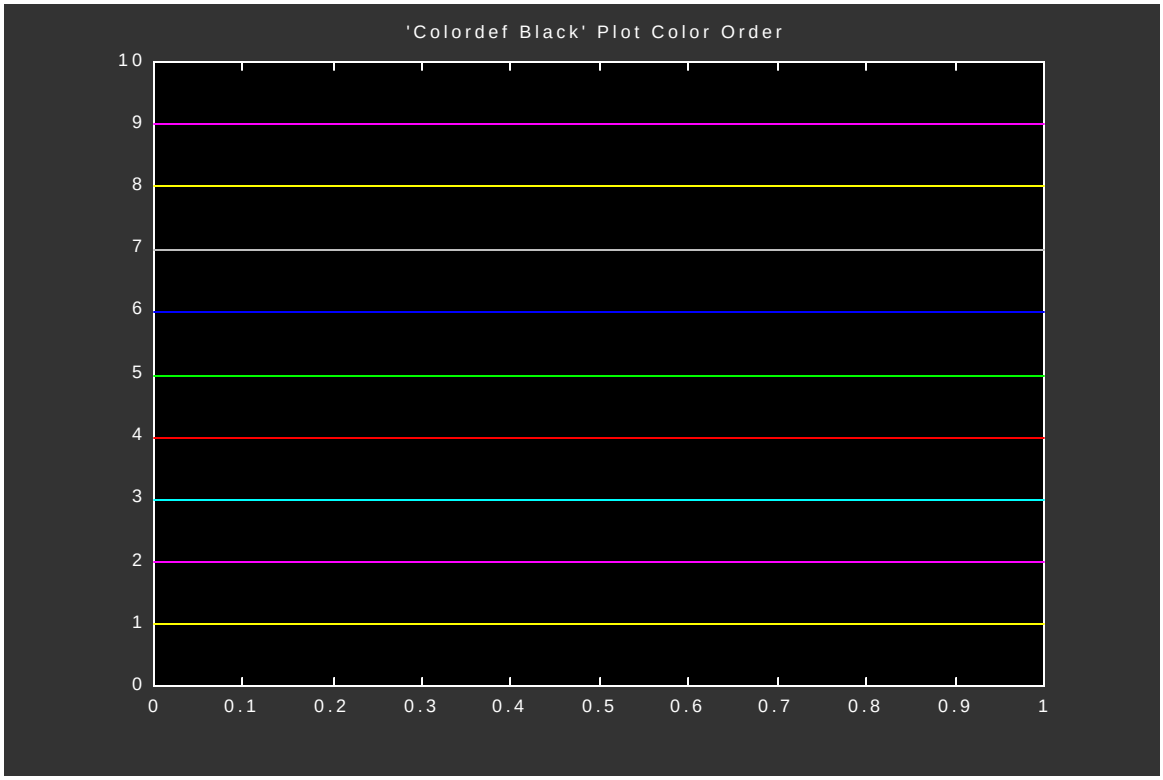
Different background colors are available for the axes and the figure in which it is embedded. The '`colordef`' command controls this feature.

```
COLORDEF Set color defaults.
    COLORDEF WHITE or COLORDEF BLACK changes the color defaults on the
    root so that subsequent figures produce plots with a white or
    black axes background color.  The figure background color is
    changed to be a shade of gray and many other defaults are changed
    so that there will be adequate contrast for most plots.
```

Note, that when using a black background color on the axes, the default color order for drawing line plots is different than when the default '`colordef white`' is in effect. Example 26.2.1 is run again with the '`colordef black`' command to show the new ordering.

Example 26.3.1

```
colordef black
x=[0;1];
Y=[[1;1],[2;2],[3;3],[4;4],[5;5],[6;6],[7;7],[8;8],[9;9]];
plot(x,Y)
axis([0 1 0 10])
title(' ''Colordef Black'' Plot Color Order ')
```

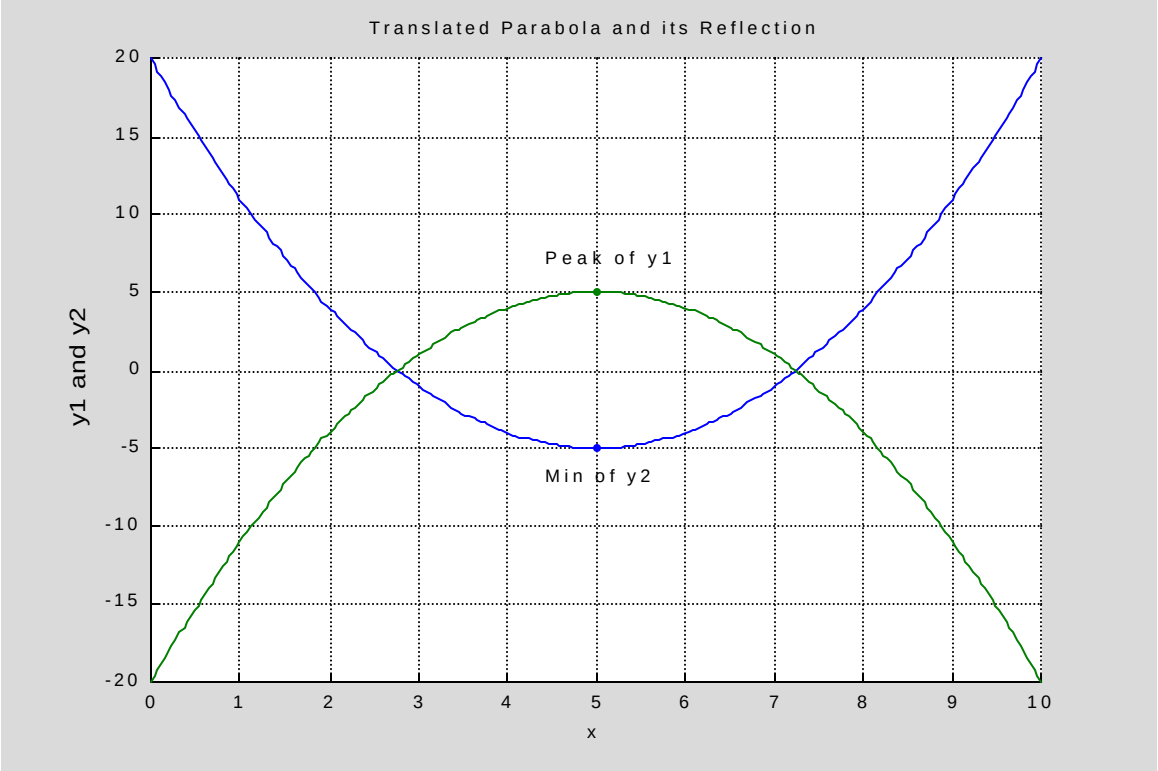


26.4 Plot Grids, Axes Box, and Labels

Horizontal and vertical grid lines can be superimposed on a graph by issuing a '`grid on`' command. '`Box off`' will remove the box which borders the axes. Labels on the `x` and `y` axes and a title are obtained by enclosing text strings inside the '`xlabel`', '`ylabel`' and '`title`' commands respectively. Finally text can be placed at point (`x`,`y`) on the plot when the '`text(x,y)`' syntax is used. The center left edge of the text is positioned at the point (`x`,`y`).

Example 26.4.1

```
x=linspace(0,10,250);
y1=(x-5).^2 - 5;
y2=-y1;
plot(x,y1,x,y2)
hold on
x=5;
y1=(x-5).^2 - 5;
y2=-y1;
plot(x,y1,'.',x,y2,'.')
box off
grid on
xlabel('x')
ylabel('y1 and y2')
title('Translated Parabola and its Reflection')
text(4.4,7,'Peak of y1')
text(4.4,-7,'Min of y2')
```


Translated Parabola and its Reflection
y1 and y2
x
Peak of y1
Min of y2
20
15
10
5
0
-5
-10
-15
-20
0
1
2
3
4
5
6
7
8
9
10

26.5 Customizing Plot Axes

The axes on graphs can be customized in a number of ways. The general procedure involves issuing a '`plot`' command followed by one or more '`axis`' commands. The axis features under MATLAB's control are listed below.

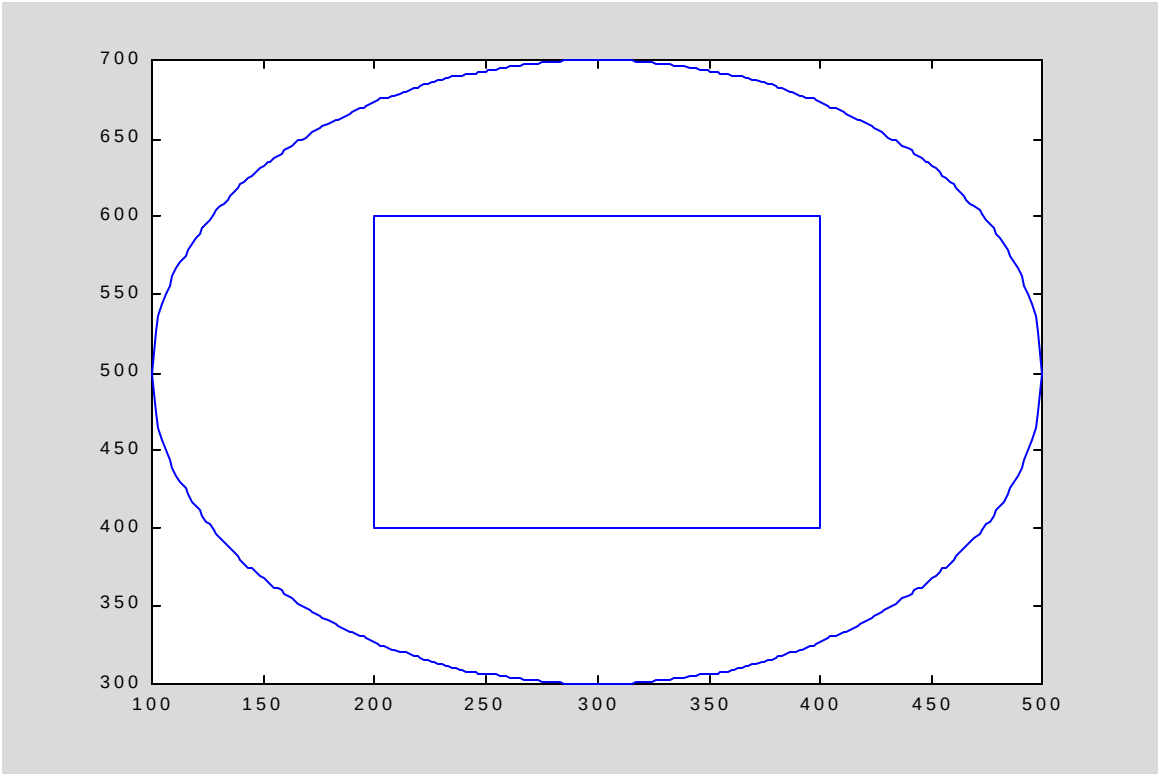
```
AXIS  Control axis scaling and appearance.

    AXIS([XMIN XMAX YMIN YMAX]) sets scaling for the x- and y-axes
       on the current plot.
    AXIS([XMIN XMAX YMIN YMAX ZMIN ZMAX]) sets the scaling for the
       x-, y- and z-axes on the current 3-D plot.
    V = AXIS returns a row vector containing the scaling for the
       current plot.  If the current view is 2-D, V has four
       components; if it is 3-D, V has six components.
    AXIS AUTO  returns the axis scaling to its default, automatic
       mode where, for each dimension, 'nice' limits are chosen based
       on the extents of all line, surface, patch, and image children.
    AXIS MANUAL  freezes the scaling at the current limits, so that if
       HOLD is turned on, subsequent plots will use the same limits.
    AXIS TIGHT  sets the axis limits to the range of the data.
    AXIS FILL  sets the axis limits and PlotBoxAspectRatio so that
       the axis fills the position rectangle.  This option only has
       an effect if PlotBoxAspectRatioMode or DataAspectRatioMode are
       manual.
     AXIS IJ  puts MATLAB into its "matrix" axes mode.  The coordinate
       system origin is at the upper left corner.  The i axis is
       vertical and is numbered from top to bottom.  The j axis is
       horizontal and is numbered from left to right.
    AXIS XY  puts MATLAB into its default "Cartesian" axes mode.  The
       coordinate system origin is at the lower left corner.  The x
       axis is horizontal and is numbered from left to right.  The y
       axis is vertical and is numbered from bottom to top.
    AXIS EQUAL  sets the aspect ratio so that equal tick mark
       increments on the x-,y- and z-axis are equal in size. This
       makes SPHERE(25) look like a sphere, instead of an ellipsoid.
    AXIS IMAGE  is the same as AXIS EQUAL except that the plot
       box fits tightly around the data.
    AXIS SQUARE  makes the current axis box square in size.
    AXIS NORMAL  restores the current axis box to full size and
       removes any restrictions on the scaling of the units.
       This undoes the effects of AXIS SQUARE and AXIS EQUAL.
    AXIS VIS3D  freezes aspect ratio properties to enable rotation of
       3-D objects and overrides stretch-to-fill.
    AXIS OFF  turns off all axis labeling, tick marks and background.
    AXIS ON  turns axis labeling, tick marks and background back on.
```

Examples 26.5.1 through 26.5.5 illustrate the use of some axes control features. In the first example, a circle and a square are plotted without any subsequent axis control commands. Neither shape appears correct because of the screen's 4:3 aspect ratio. The scales are also set automatically to contain the plotted data points.

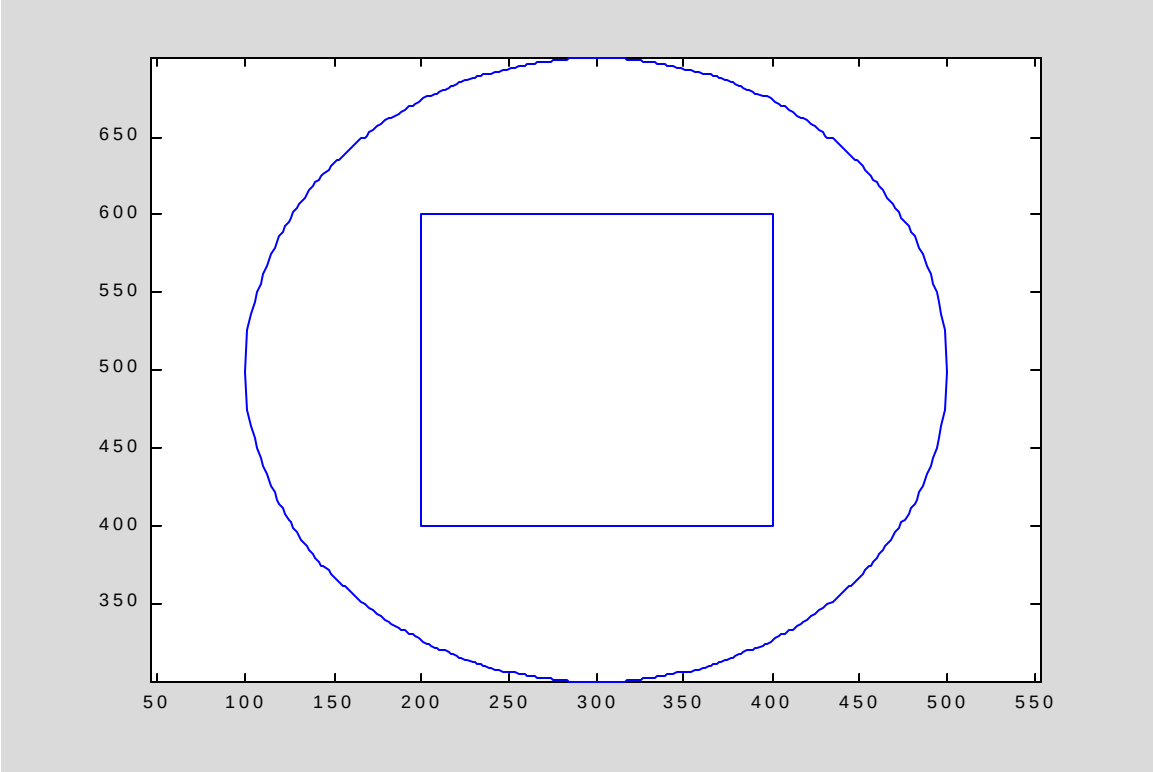
Example 26.5.1

```
h=300; k=500; r=200; % Set circle parameters
xi=linspace(100,500,250);
yui=k + sqrt(r^2 - (xi-h).^2); % Find upper half-circle y values
yli=k - sqrt(r^2 - (xi-h).^2); % Find lower half-circle y values
plot(xi,yui,'b',xi,yli,'b') % Plot circle
hold on
xa=200;xb=200;xc=400;xd=400;
ya=400;yb=600;yc=600;yd=400;
x=[xa xb xc xd xa]; % Fix x locations of square's corners
y=[ya yb yc yd ya]; % Fix y locations of square's corners
plot(x,y) % Plot square
```



In the second example, the MATLAB code is identical to Example 26.5.1 except for the addition of the last command '`axis equal`' which forces the aspect ratio to 1:1 changing the appearance of the the circle and square.

```
h=300; k=500; r=200; % Set circle parameters
xi=linspace(100,500,250);
yui=k + sqrt(r^2 - (xi-h).^2); % Find upper half-circle y values
yli=k - sqrt(r^2 - (xi-h).^2); % Find lower half-circle y values
plot(xi,yui,'b',xi,yli,'b') % Plot circle
hold on
xa=200;xb=200;xc=400;xd=400;
ya=400;yb=600;yc=600;yd=400;
x=[xa xb xc xd xa]; % Fix x locations of square's corners
y=[ya yb yc yd ya]; % Fix y locations of square's corners
plot(x,y) % Plot square
axis equal
```

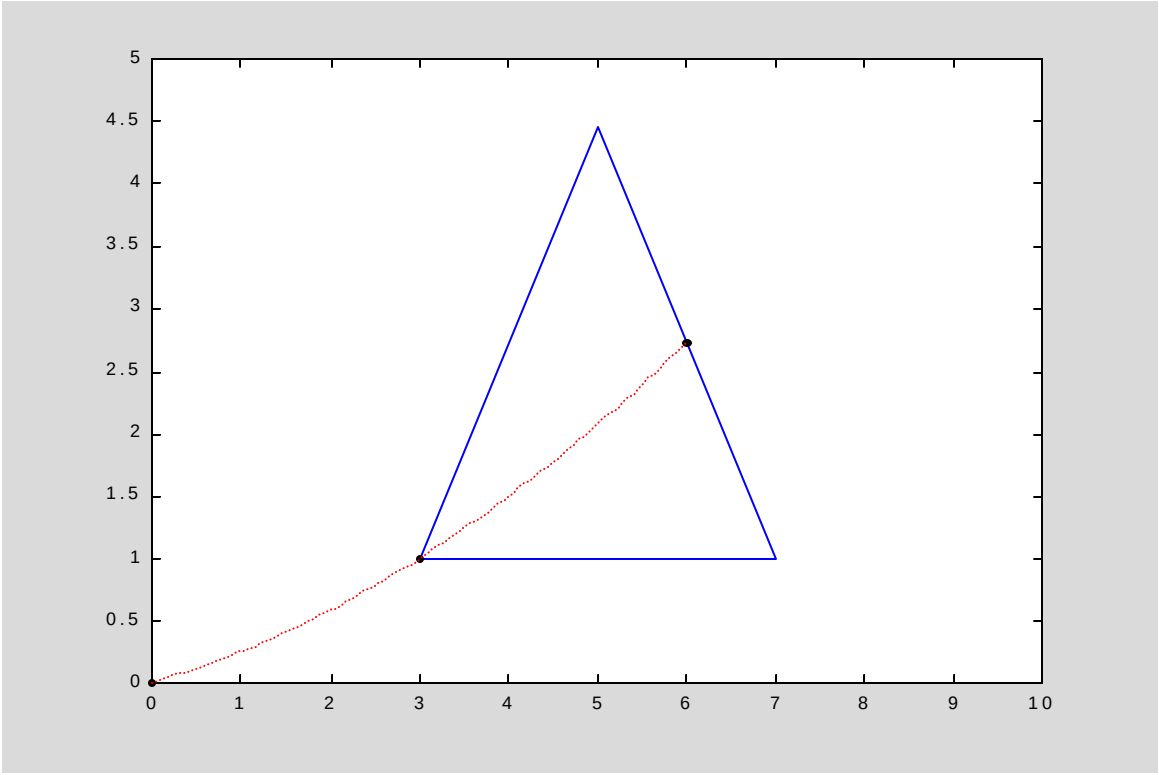


Note, that the plot is still enclosed in a rectangle and the scaling is still chosen by default to assure the entire circle is visible.

In the next example, the scales along both axes are set with the appropriate axis scaling command. The figure is an equilateral triangle which appears to be an isosceles triangle.

Example 26.5.3

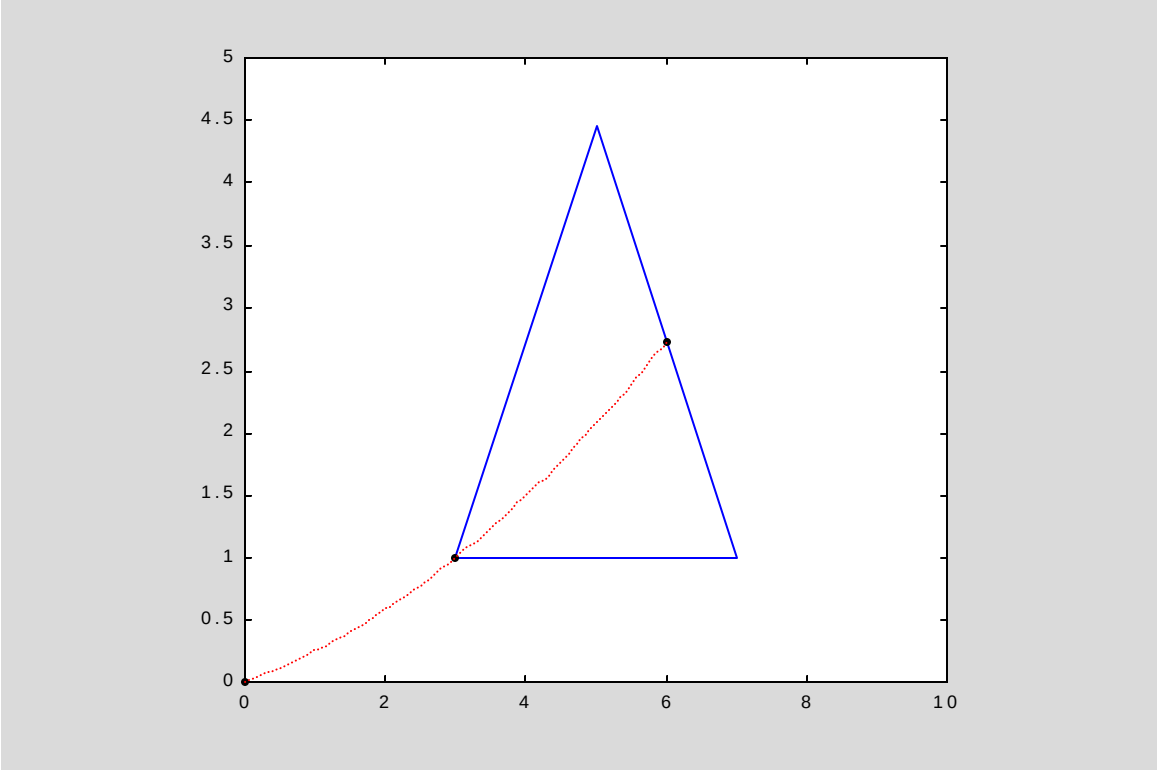
```
L=2; % Equilateral triangle with sides of length 2L
xA=3; xB=xA+L; xC=xA+2*L; % Set x corners of equilateral triangle
yA=1; yB=yA+sqrt(3)*L; yC=yA; % Set y corners of equilateral triangle
x=[xA xB xC xA];
y=[yA yB yC yA];
v=[0 10 0 5]; % v=[xmin xmax ymin ymax]
plot(x,y) % Plot equilateral triangle
hold on
axis(v) % Set x and y scales to values in vector v
xdata=[0 xA (xB+xC)/2];
ydata=[0 yA (yB+yC)/2];
p=polyfit(xdata, ydata, 2); % Fit quadratic thru (xdata, ydata)
xi=linspace(0,(xB+xC)/2,100);
yi=polyval(p,xi);
plot(xdata, ydata, '.k', xi, yi, 'r:')
```



Making the border of the plot square using '`axis square`' does not correct the problem as illustrated in the next example.

Example 26.5.4

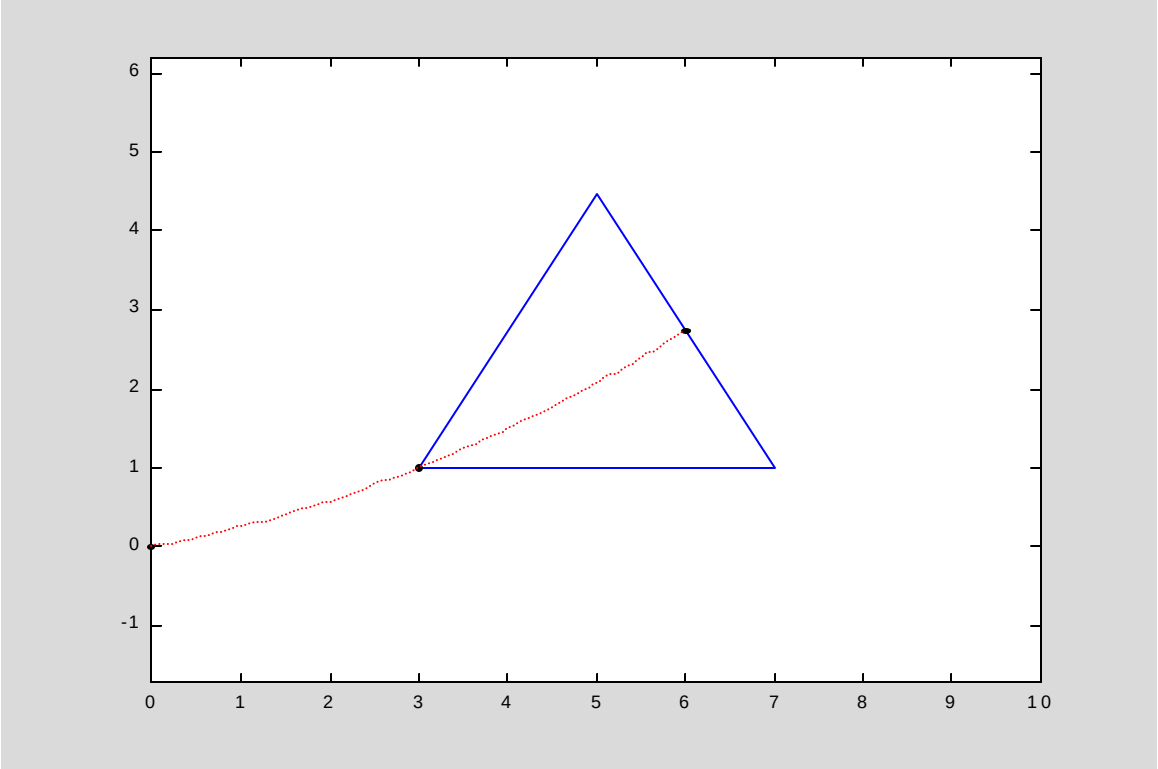
```
L=2; % Equilateral triangle with sides of length 2L
xA=3; xB=xA+L; xC=xA+2*L; % Set x corners of equilateral triangle
yA=1; yB=yA+sqrt(3)*L; yC=yA; % Set y corners of equilateral triangle
x=[xA xB xC xA];
y=[yA yB yC yA];
v=[0 10 0 5];
plot(x,y)
hold on
axis(v) % Set x and y scales to values in vector v
xdata=[0 xA (xB+xC)/2];
ydata=[0 yA (yB+yC)/2];
p=polyfit(xdata, ydata, 2); % Fit quadratic thru (xdata, ydata)
xi=linspace(0,(xB+xC)/2,100);
yi=polyval(p,xi);
plot(xdata, ydata, '.k', xi, yi, 'r:')
axis square
```



To render the equilateral triangle with equal sides on the screen, the axis square command is replaced by axis equal as shown in Example 26.5.5.

Example 26.5.5

```
L=2; % Equilateral triangle with sides of length 2L
xA=3; xB=xA+L; xC=xA+2*L; % Set x corners of equilateral triangle
yA=1; yB=yA+sqrt(3)*L; yC=yA; % Set y corners of equilateral triangle
x=[xA xB xC xA];
y=[yA yB yC yA];
v=[0 10 0 5];
plot(x,y)
hold on
axis(v) % Set x and y scales to values in vector v
xdata=[0 xA (xB+xC)/2];
ydata=[0 yA (yB+yC)/2];
p=polyfit(xdata, ydata, 2); % Fit quadratic thru (xdata, ydata)
xi=linspace(0,(xB+xC)/2,100);
yi=polyval(p,xi);
plot(xdata, ydata, '.k', xi, yi, 'r:')
axis equal
```



Note, the ymin=0 and ymax=5 values are no longer in effect.

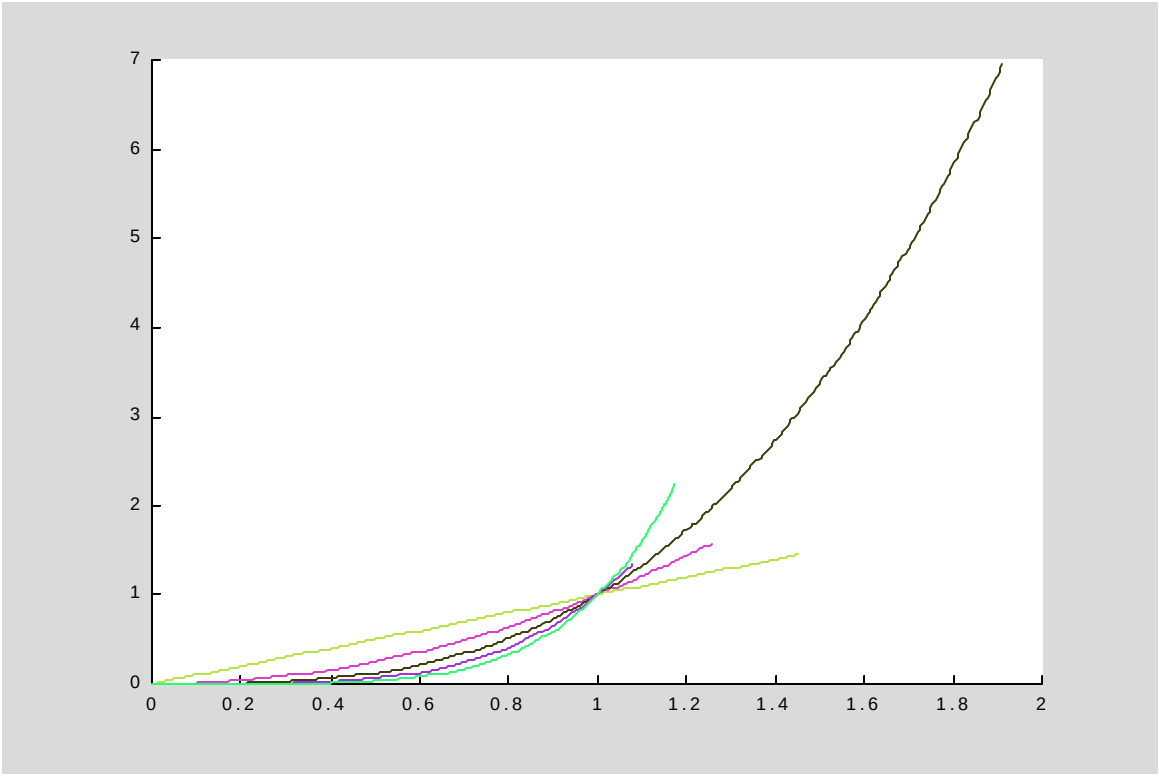
26.6 Multiple Plots

As we have already seen, its possible to include multiple plots on the same set of axes. The '`hold on`' command retains previous plots in the current set of axes and remains in effect until a '`hold off`' command is issued. Subsequent plots are drawn on an empty set of axes and cleared prior to the next plot until '`hold on`' is reissued. The current hold state, i.e. `on` or `off`, can be ascertained using the '`ishold`' command which returns a 1 if '`hold on`' is in effect and 0 if not.

Example 26.6.1

```
ishold
if ishold==0 % Check if hold is off
hold on
ishold
end % if ishold==0
for i=1:5
x=linspace(0,1+rand,500);
y=x.^i;
plot(x,y,'Color',[rand rand rand])
end % for i=1:5
hold off
ishold
```

```
ans = 0
ans = 1
ans = 0
```



26.7 Multiple Figures

More than one Figure window can be used for plotting sets of data. The '`figure`' command issued in the Command window or from within an m-File creates a new Figure window with an integer 'handle' to identify it. The last Figure window created is the active or current figure. To make a different Figure window active, either click on it with the mouse or issue the command '`figure(h)`' where `h` is the handle of the Figure window to be activated.

The current Figure window is closed by a mouse click in the upper right hand corner or as a result of the MATLAB command '`close`'. Any existing Figure window can be closed using `close(h)'` where `h` is the handle of the Figure window to be closed. All Figure windows are closed with '`close all`'. Finally, '`clf`' will erase the contents of the current Figure window without deleting the Figure window.

The following example contains a script m-File which opens three Figure windows and plots different sets of data in each. It should be run from the Command window and not the m-Book.

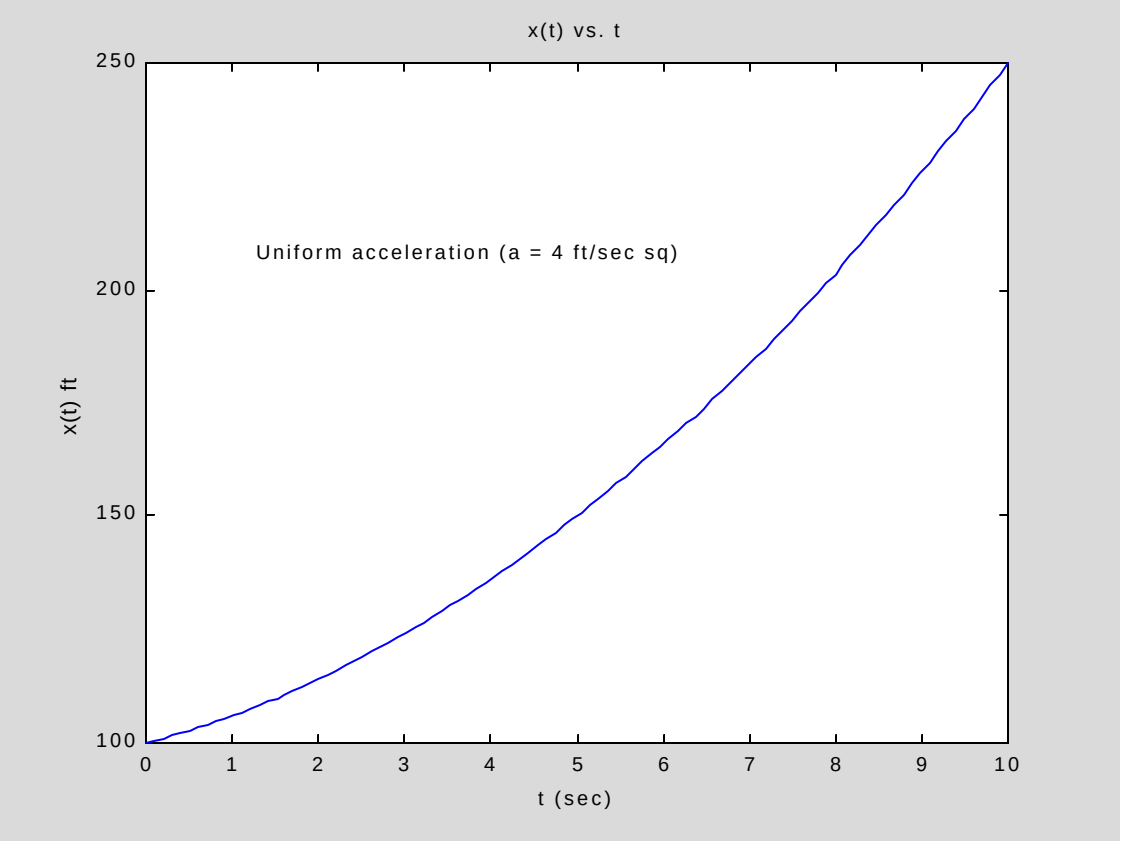
Example 26.7.1

```
% Script file uniform_acc.m
% This file will accept initial conditions for a uniform acceleration
% trajectory and plot in separate Figure windows the dispacement,
% velocity and acceleration profiles

close all % Close all figures
a0=input('Enter constant acceleration in ft/sec sq:  '); % Accept
                                    % uniform acceleration value
x0=input('Enter initial displacement in ft:  '); % Accept initial
                                                 % displacement
v0=input('Enter initial velocity in ft/sec:  ') ; % Accept initial
                                                  % velocity
x=[a0/2 v0 x0]; % Set displacement polynomial coefficient vector
v=polyder(x); % Find velocity polynomial coefficient vector
a=polyder(v); % Find acceleration polynomial vector
ti=linspace(0,10,100);
xi=polyval(x,ti);
vi=polyval(v,ti);
ai=polyval(a,ti);
plot(ti,xi) % Plot displacement data points in Figure 1
xlabel('t (sec)')
ylabel('x(t) ft')
title('x(t) vs. t')
figure % Open Figure 2
plot(ti,vi) % Plot velocity data points in Figure 2
xlabel('t (sec)')
ylabel('v(t) ft/sec')
title('v(t) vs. t')
figure % Open Figure 3
plot(ti,ai) % Plot acceleration data points in Figure 3
```

```
xlabel('t (sec)')
ylabel('a(t) ft/sec sq')
title('a(t) vs. t')
```

After running the script file, Figure 3 is on the screen and Figures 1 and 2 are minimized on the task bar. Clicking on Figure 1 in the task bar will make it the active Figure window. The plot of displacement, x(t) vs t in Figure 1 is shown below for values of a0=2, x0=100 and v0=5.



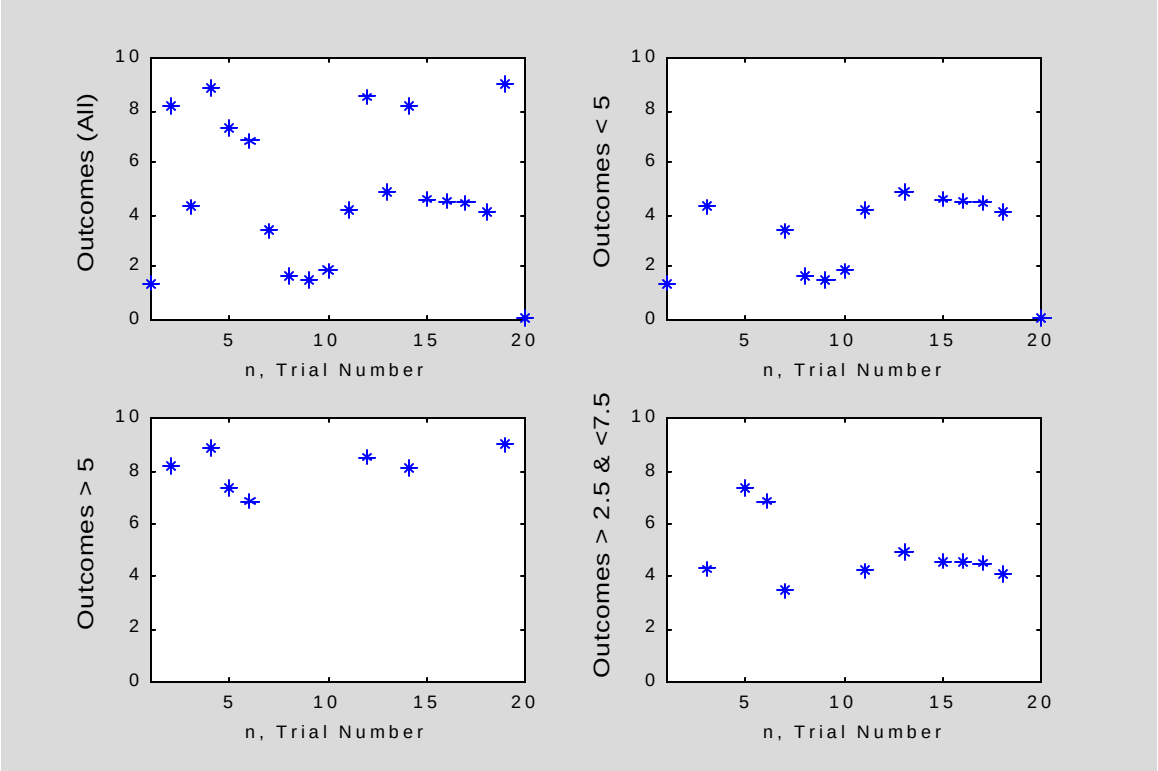
26.8 Subplots

More than one set of axes can be postioned in a Figure window. In fact, the window can be segmented so that several rows and columns of axes will appear simultaneously. To configure the current Figure window for an `m×n` arrangement of subplots, the MATLAB command is '`subplot(m,n,p)`' where `p` refers to the active subplot, i.e. the area where the next data set is to be plotted. The numbering of subplots is from left to right starting in the top row.

The following example simulates 20 outcomes of an experiment. It graphs all outcomes in the first plot (upper left corner) and then continues to graph subsets of the outcomes in succeeding plots.

Example 26.8.1

```
x=10*rand(1,20);
subplot(2,2,1) % Activate upper left corner of 2´2 subplot window
plot(x,'*')
axis([1 20 0 10])
xlabel('n, Trial Number')
ylabel('Outcomes (All)')
subplot(2,2,2) % Activate upper right corner of 2´2 subplot window
xlt5_indices=find(x<5);
xlt5=x(xlt5_indices);
plot(xlt5_indices,xlt5,'*')
axis([1 20 0 10])
xlabel('n, Trial Number')
ylabel('Outcomes < 5')
subplot(2,2,3) % Activate lower left corner of 2´2 subplot window
xgt5_indices=find(x>5);
xgt5=x(xgt5_indices);
plot(xgt5_indices,xgt5,'*')
axis([1 20 0 10])
xlabel('n, Trial Number')
ylabel('Outcomes > 5')
subplot(2,2,4) % Activate lower right corner of 2´2 subplot window
xbtwn_indices=find(x>2.5 & x<7.5);
xbtwn=x(xbtwn_indices);
plot(xbtwn_indices,xbtwn,'*')
axis([1 20 0 10])
xlabel('n, Trial Number')
ylabel('Outcomes > 2.5 & <7.5')
```

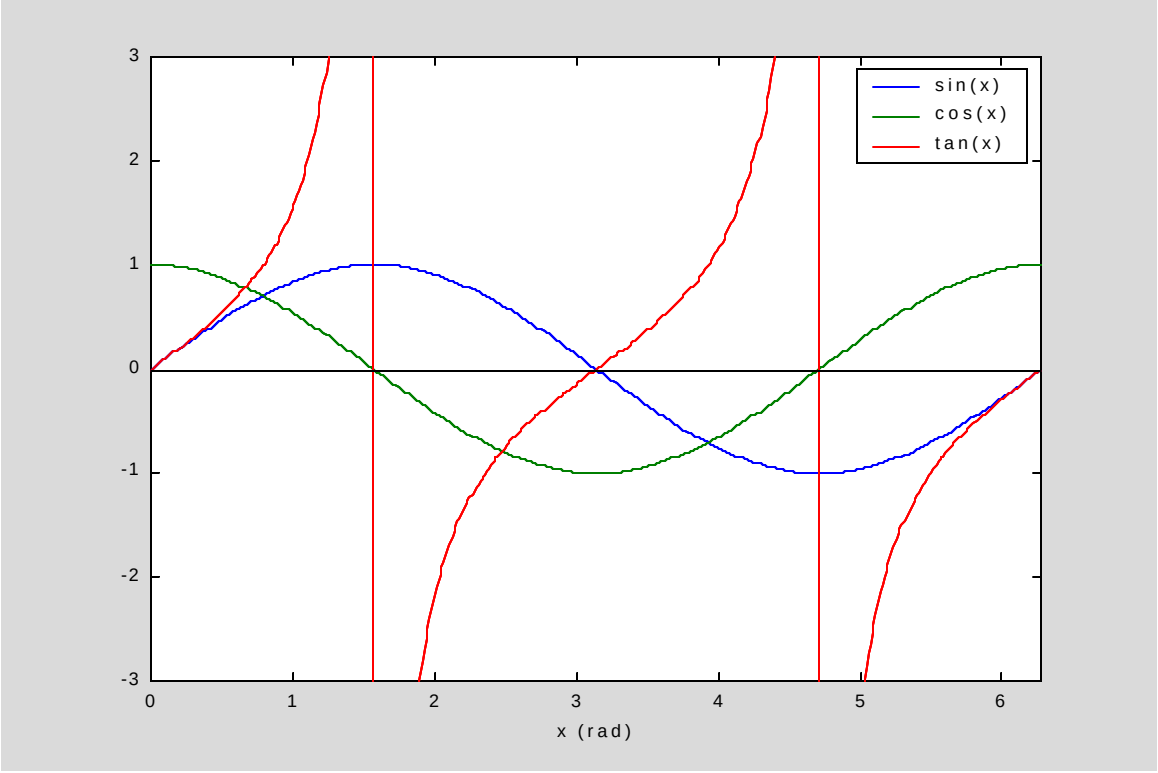

Outcomes (All)
n, Trial Number
Outcomes < 5
n, Trial Number
Outcomes > 5
n, Trial Number
Outcomes > 2.5 & <7.5
n, Trial Number

26.9 Interactive Plotting Tools

The '`legend`' command places string labels in a legend box to help distinguish between the different curves within a set of axes. Example 26.9.1 includes plots of several trigonometric functions and a legend to help identify each one.

Example 26.9.1

```
x=linspace(0,2*pi,500);
ysin=sin(x);
ycos=cos(x);
ytan=tan(x);
v=[0 2*pi -3 3];
plot(x,ysin,x,ycos,x,ytan)
axis(v)
legend('sin(x)','cos(x)','tan(x)')
hold on
plot([0 2*pi],[0 0],'k') % Plot x axis
xlabel('x (rad)')
```



The legend box can be dragged to another location if the default position is unsatisfactory.

A '`zoom`' function in MATLAB allows you to zoom in or out inside a Figure window. It can be activated (and deactivated) by typing '`zoom on`' and '`zoom off`' in

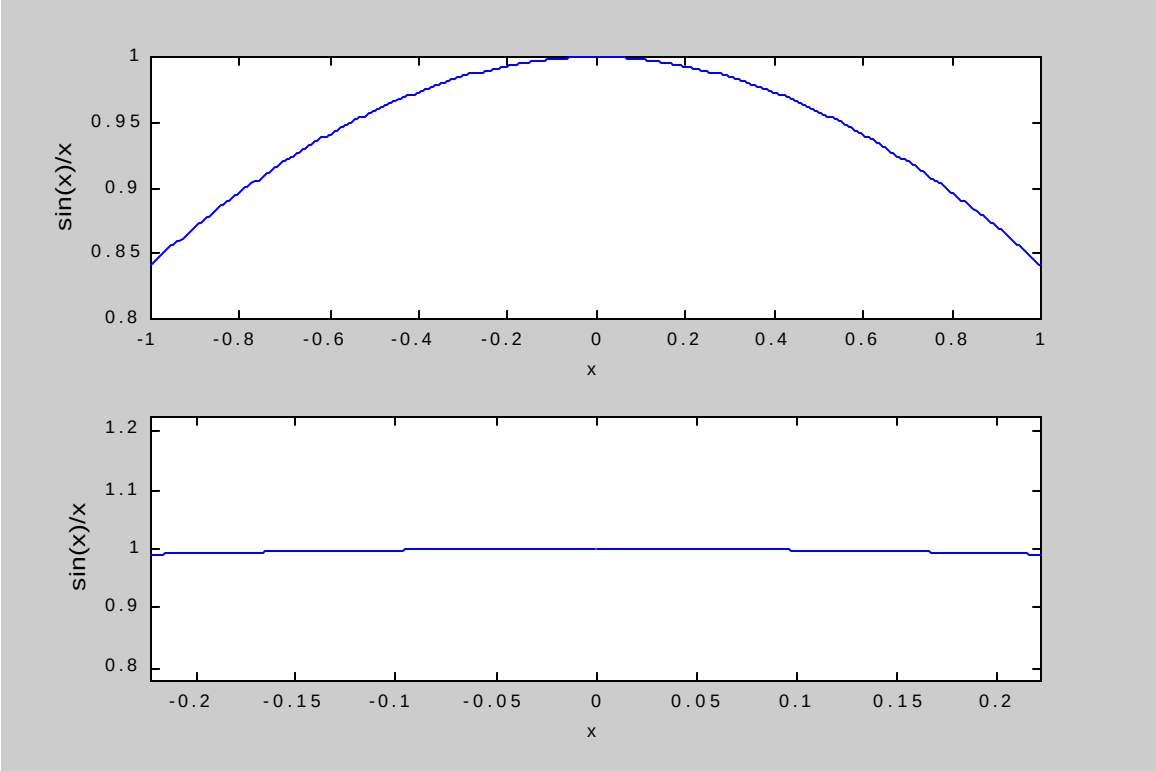
the Command window or the statements can be included in an m-File. When '`zoom`' is on, left and right mouse clicks zoom in and out from the mouse point by a factor of 2. The '`zoom`' feature can also be turned on by selecting the magnifying glass icons in the figure window tool bar.

Other features of zooming include the capability of zooming in either the `x` or `y` directions only as well as specifying a zoom factor. Type '`help zoom`' to learn more about zooming.

Example 26.9.2

```
x=-1:0.001:1;
y=sin(x)./x;
subplot(2,1,1)
plot(x,y)
xlabel('x'),ylabel('sin(x)/x')
subplot(2,1,2)
plot(x,y)
axis([-1 1 0 2])
xlabel('x'),ylabel('sin(x)/x')
zoom(3) % Zoom in by 2 and 2 again
```

```
Warning: Divide by zero.
```



In the above example, the zoom(3) command zooms in on both axes by a factor of 2, two times. That is, the x-axis goes from (-1,1) to (-0.5,0.5) to (-0.25,0.25) and the y-axis changes from (0,2) to (0.5,1.5) to (0.75,1.25). Also note the 'hole' at x=0 in the magnified plot and the 'NaN' warning.

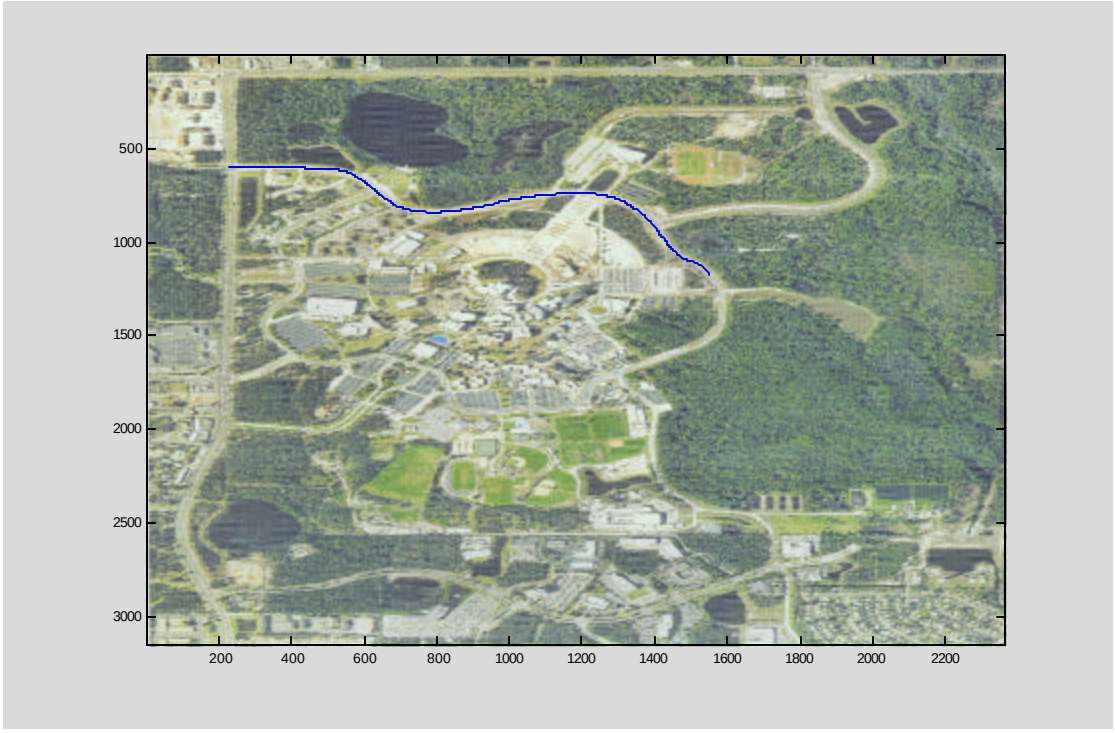
MATLAB has a rudimentary digitizing capability which is described below.

```
GINPUT Graphical input from mouse.
    [X,Y] = GINPUT(N) gets N points from the current axes and returns
the X- and Y-coordinates in length N vectors X and Y. The cursor can be
positioned using a mouse. Data points are entered by pressing a mouse
button. A carriage return terminates input before N points are entered.
    [X,Y] = GINPUT gathers an unlimited number of points until the
return key is pressed.
```

In the following example, an image of the UCF campus is read and displayed in a Figure window. The '`ginput`' function is used to extract the coordinates of points along Gemini Blvd (to the North). A spline function is fitted to the points and then plotted as an overlay to the image.

Example 36.10.3

```
X=imread('campusaerial','JPEG'); % Read image as JPEG file
image(X) % Display image
[xroad,yroad]=ginput  % Collect (x,y) coordinates of points along road
xi=linspace(200,1600,500); % x values along road for spline evaluation
yi=interp1(xroad,yroad,xi,'spline');
plot(xi,yi,'b')
```



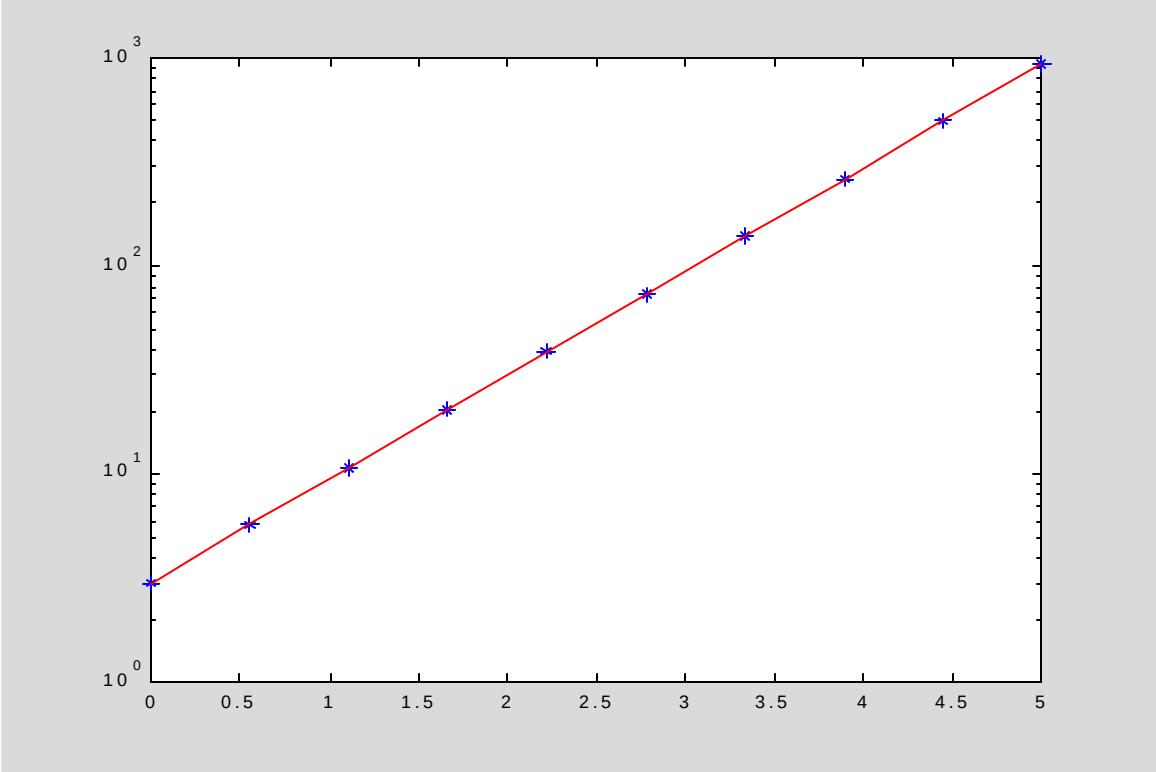

```
x = 1.0e+003 *
    0.2357    0.3459    0.4679    0.5900    0.6644    0.7448    0.8788
    1.0277    1.1765    1.3135    1.4028    1.4624    1.5428    1.5815
y = 1.0e+003 *
    0.5983    0.5983    0.6041    0.6615    0.7765    0.8340    0.8225
     0.7650    0.7363    0.7823    0.9260    1.0639    1.1501    1.2996
```

26.11 Specialized 2-D Plots

Data can be plotted on logarithmic scales by using either '`semilogx`', '`semilogy`' or '`loglog`' instead of the conventional '`plot`' command.

Example 16.11.1

```
x=linspace(0,5,10);
y=3*10.^(x./2);
semilogy(x,y,'*b',x,y,'r')
```



The function used to generate the data points was $y = 3 \cdot 10^{x/2}$. Starting with the plotted data points, the function can be estimated using the end points as follows:

$y = A \cdot 10^{Bx}$ thru $(x = 0,\ y = 3)$ and $(x = 5,\ y = 950)$

$\log 3 = \log A + B(0) \quad \Rightarrow \quad A = 3$

$$\log 950 = \log 3 + B(5) \quad \Rightarrow \quad B = \frac{\log 950 - \log 3}{5} = \frac{2.9777 - 0.4771}{5} = 0.5001$$

The 'comet' function provides an eye-appealing way of plotting connecting data points. The following example must be run from the Command window or a script file to demonstrate the '`comet`' plot feature.

Example 26.11.2

```
% Script file Comet_Plot.m
% This file plots two comet style graphs
x=linspace(0,2*pi,500);
y=sin(x);
comet(x,y) % Plot sine function
pause(2)
close(1)
theta=linspace(0,2*pi,500);
r=2*sin(2*theta);
[x,y]=pol2cart(theta,r);
comet(x,y) % Plot rose curve
pause(2)
close(1)
theta=linspace(0,10*pi,500);
r=2*theta;
[x,y]=pol2cart(theta,r);
comet(x,y) % Plot spiral
```

A stacked area plot permits a cumulative function and the contribution of each component to be graphed with the areas between the components filled in. For example, if

$$f(x) = f_1(x) + f_2(x) + f_3(x)$$

$$f_1(x) = 2x+1$$

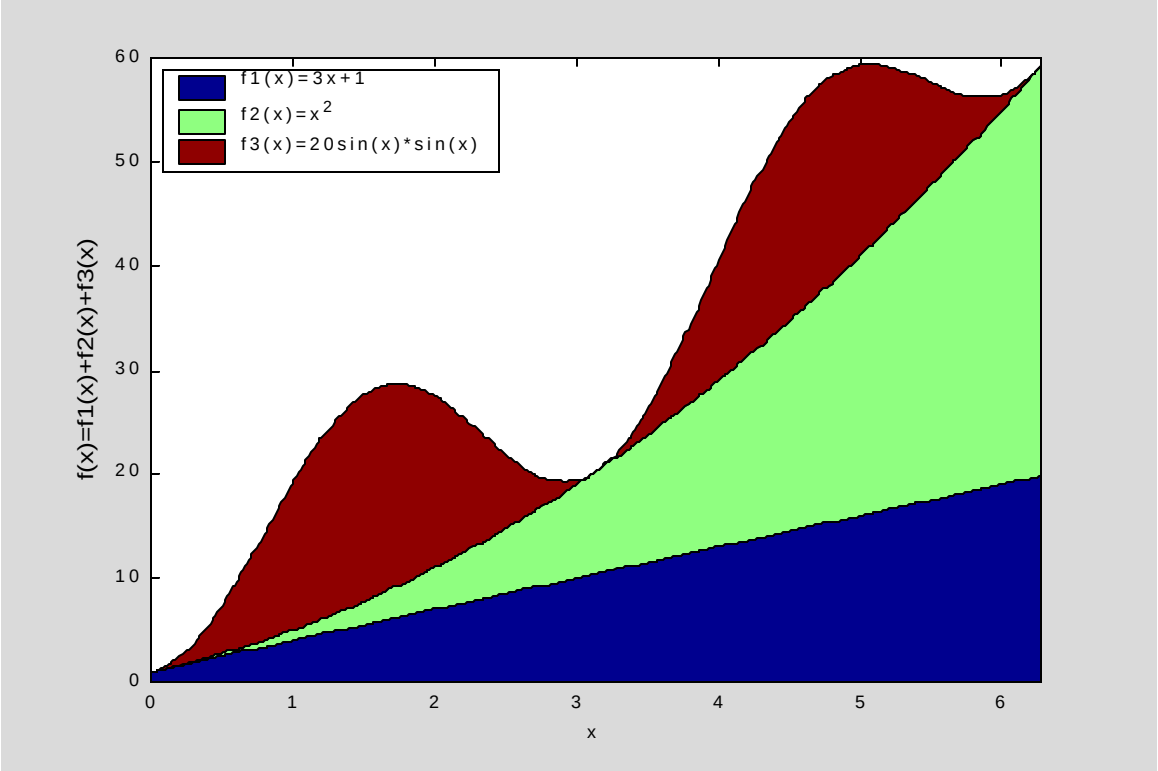
$$f_2(x) = x^2$$

$$f_3(x) = \sin(p\,x)$$

the next example plots the stacked components of $f(x)$ using the MATLAB '`area`' function.

Example 26.11.3

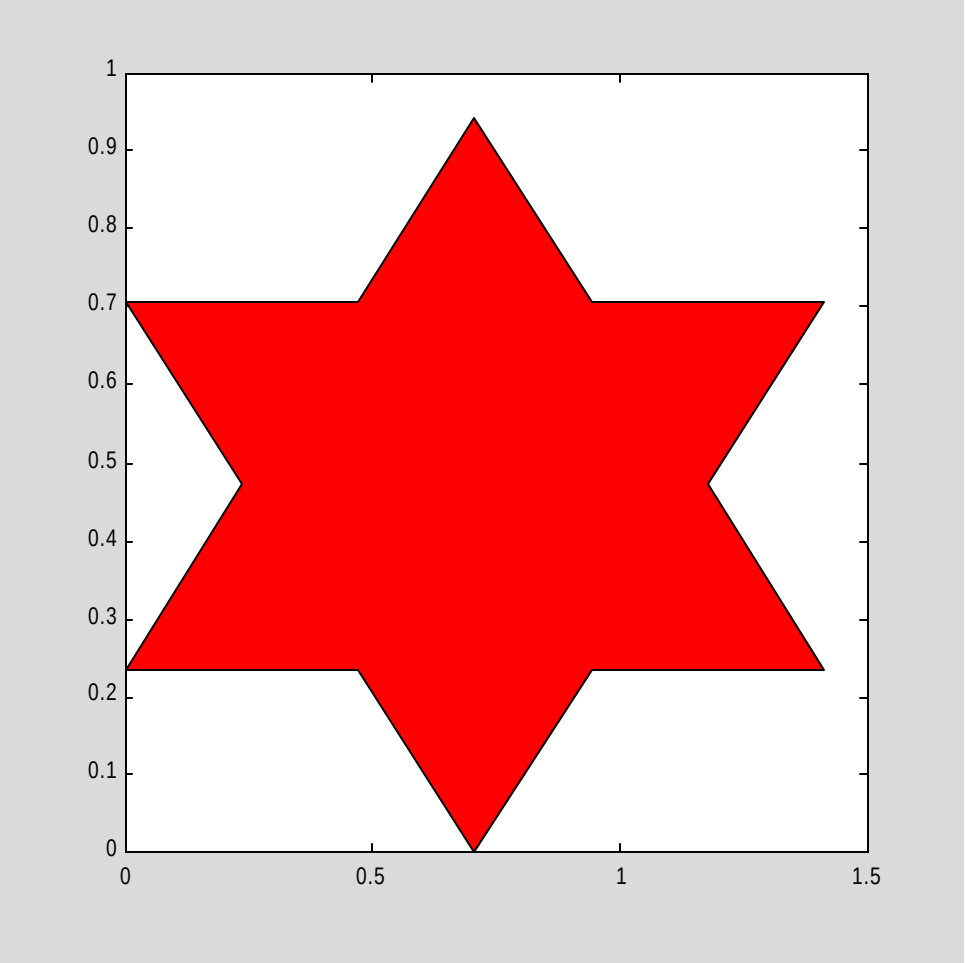
```
x=[linspace(0,2*pi,500)]';
f1=3*x+1;
f2=x.^2;
f3=20*sin(x).^2;
F=[f1 f2 f3];
area(x,F)
xlabel('x')
ylabel('f(x)=f1(x)+f2(x)+f3(x)')
legend('f1(x)=3x+1','f2(x)=x^2','f3(x)=20sin(x)*sin(x)',2)
```



A polygon described by a set of vertices can be drawn and filled in using the '`fill`' function. An example follows.

Example 26.11.4

```
x1=sqrt(2)/2; y1=0;
x5=sqrt(2); y5=sqrt(2)/2;
m1=(y5-y1)/(x5-x1);
y2=sqrt(2)/6;
x2=x1+(y2-y1)/m1;
y4=sqrt(2)/3;
x4=x1+(y4-y1)/m1;
x3=x5;y3=y2;
x6=x2;y6=y5;
x7=x1;
y7=2*sqrt(2)/3;
x8=x7-(x6-x7);
y8=y6;
x9=0; y9=y8;
x10=x1-(x4-x1);
y10=y4;
x11=0;y11=y3;
x12=x8;y12=y11;
x=[x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12 x1];
y=[y1 y2 y3 y4 y5 y6 y7 y8 y9 y10 y11 y12 y1];
axis equal
fill(x,y,'r')
```

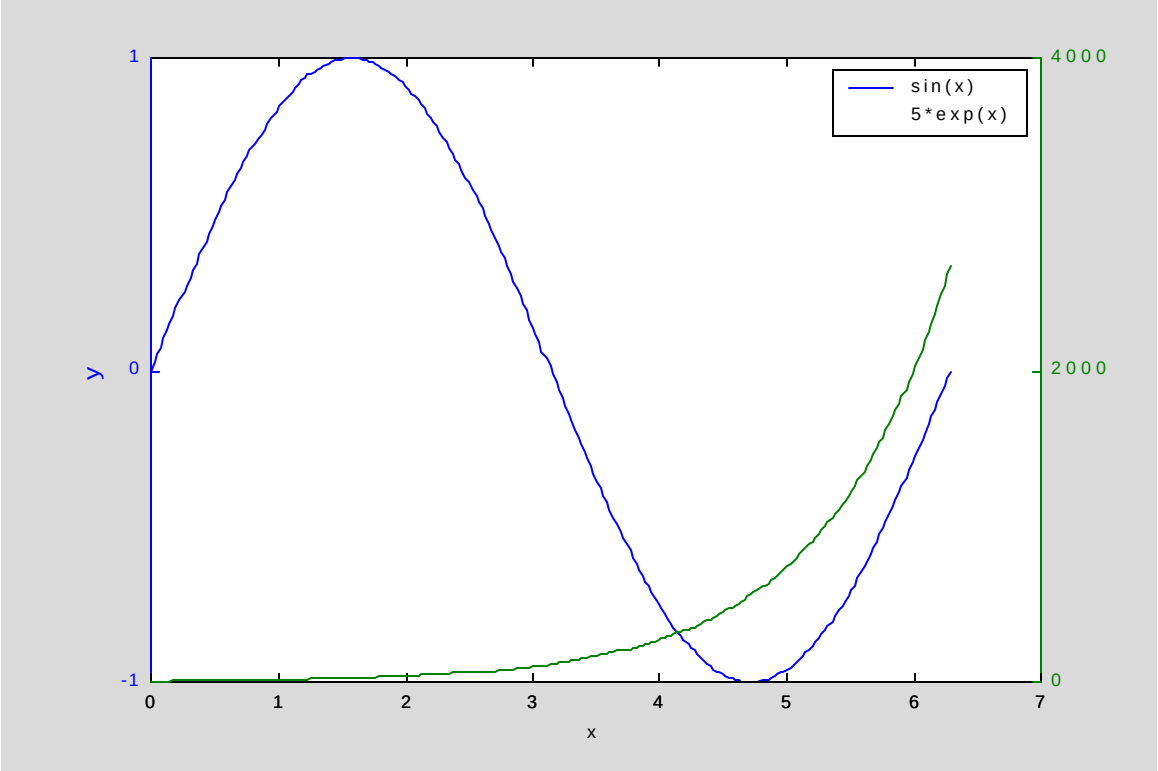


When two different sets of data require different y-scales, the solution is to use MATLAB's '`plotyy`' function.

Example 26.11.5

```
x=linspace(0,2*pi,250);
y=sin(x);
z=5*exp(x);
plotyy(x,y,x,z)
xlabel('x')
ylabel('y')
legend('sin(x)','5*exp(x)')
```

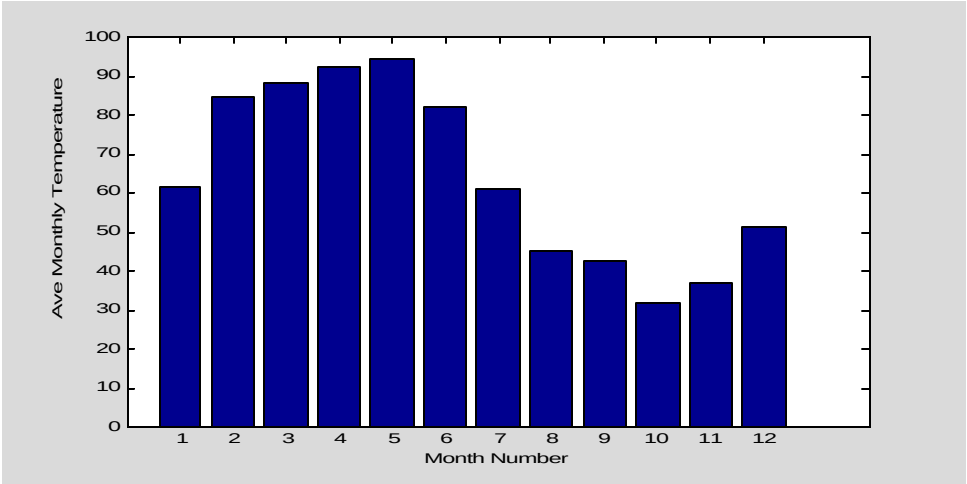
Since '`plotyy`' actually creates two sets of axes inside a Figure window, its ill-advised to issue any commands which alter the appearance of axes since they will affect only one of the two sets of axes and produce unintended results. In fact, in the previous example, the '`legend`' command was not recognized for the x-z axes. Try adding the command '`axis tight`' after the '`plotyy`' command to the code in Example 26.11.5 and observe the results.



There are several other specialized 2-D plots available in MATLAB. Bar charts, stem plots, and polar plots are shown in the following examples.

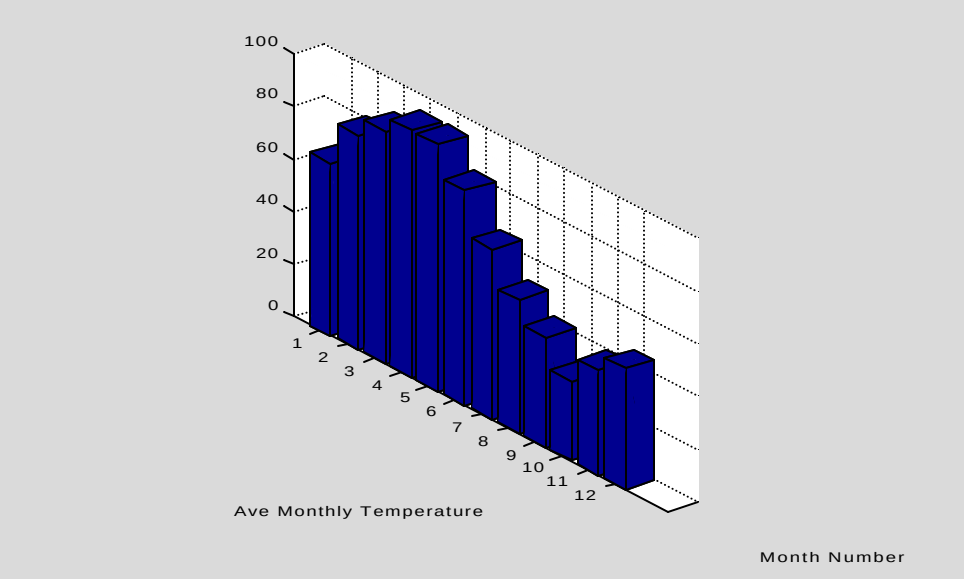
Example 26.11.6

```
x=1:12;
Monthly_temp=60+10*rand(1,12) + 30*sin(pi*(x-1)/6);
bar(x,Monthly_temp) % Plot bar chart
xlabel('Month Number')
ylabel('Ave Monthly Temperature')
```



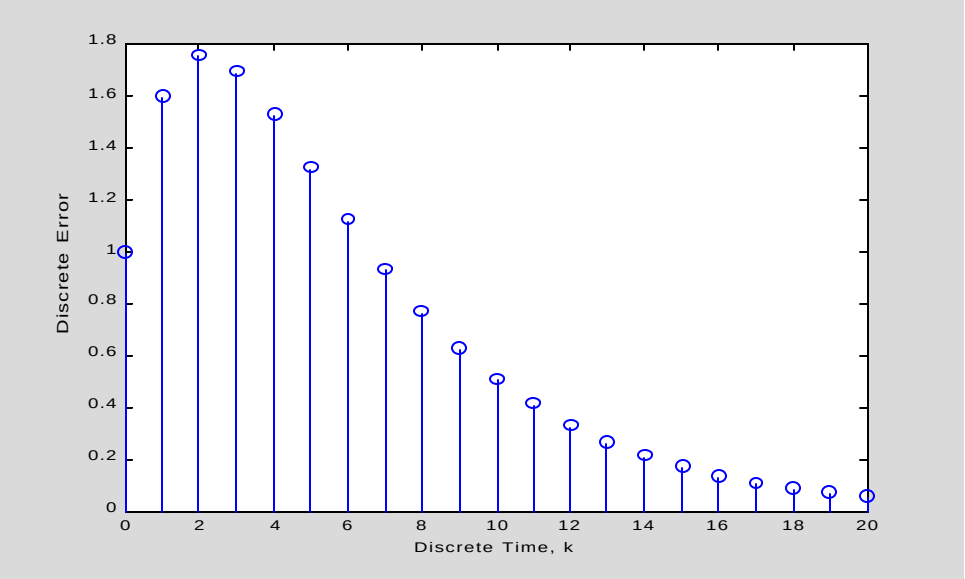
Example 26.11.7

```
x=1:12;
Monthly_temp=60+10*rand(1,12) + 30*sin(pi*(x-1)/6);
bar3(x,Monthly_temp) % Plot 3-D bar chart
xlabel('Month Number')
ylabel('Ave Monthly Temperature')
```



Example 26.11.8

```
k=0:20;
ek=-4*(0.6).^k +5*(0.8).^k ;
stem(k,ek) % Plot discrete data
xlabel('Discrete Time, k')
ylabel('Discrete Error')
```



Example 26.11.9

```
theta=linspace(0, 2*pi, 250);
r=1+2*sin(theta);
polar(theta,r)
title('Polar Plot of a Limacon')
```