Chapter 5
Two-Dimensional Plots

Plots are a very useful tool for presenting information. This is true in any field, but especially in science and engineering, where MATLAB is mostly used. MATLAB has many commands that can be used for creating different types of plots. These include standard plots with linear axes, plots with logarithmic and semi-logarithmic axes, bar and stairs plots, polar plots, three-dimensional contour surface and mesh plots, and many more. The plots can be formatted to have a desired appearance. The line type (solid, dashed, etc.), color, and thickness can be prescribed, line markers and grid lines can be added, as can titles and text comments. Several graphs can be created in the same plot, and several plots can be placed on the same page. When a plot contains several graphs and/or data points, a legend can be added to the plot as well.

This chapter describes how MATLAB can be used to create and format many types of two-dimensional plots. Three-dimensional plots are addressed separately in Chapter 9. An example of a simple two-dimensional plot that was created with MATLAB is shown in Figure 5-1. The figure contains two curves that show the variation of light intensity with distance. One curve is constructed from data points measured in an experiment, and the other curve shows the variation of light as predicted by a theoretical model. The axes in the figure are both linear, and different types of lines (one solid and one dashed) are used for the curves. The theoretical curve is shown with a solid line, while the experimental points are connected with a dashed line. Each data point is marked with a circular marker. The dashed line that connects the experimental points is actually red when the plot is displayed in the Figure Window. As shown, the plot in Figure 5-1 is formatted to have a title, axis titles, a legend, markers, and a boxed text label.

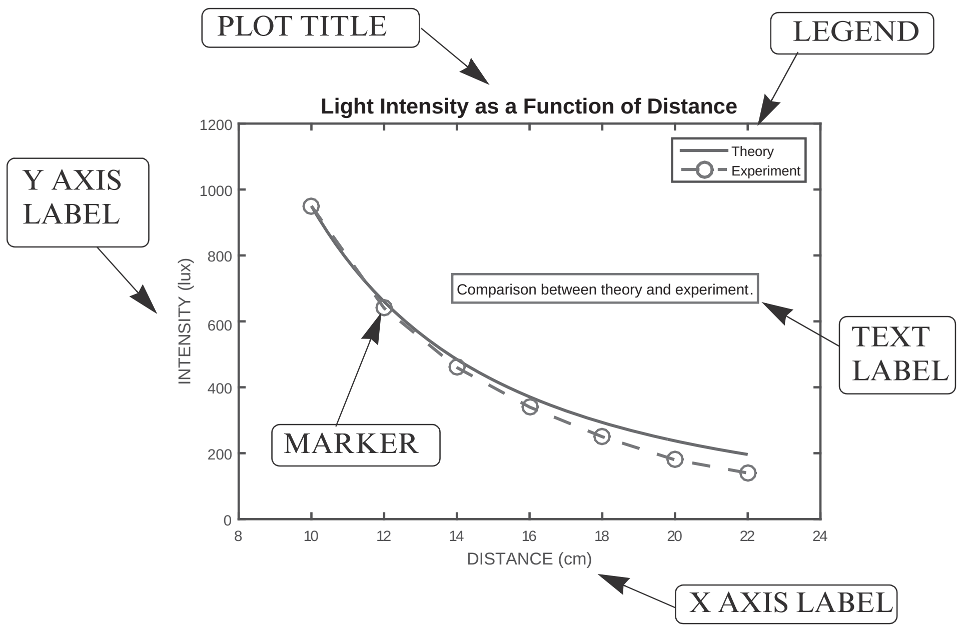


Figure 5-1: Example of a formatted two-dimensional plot.

5.1 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). The two vectors *must* have the same number of elements. When the `plot` command is executed, a figure is created in the Figure Window. If not already open, the Figure Window opens automatically when the command is executed. The figure has a single curve with the `x` values on the abscissa (horizontal axis) and the `y` values on the ordinate (vertical axis). 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`. Each of the vectors, of course, can have any name. The vector that is typed first in the `plot` command is used for the horizontal axis, and the vector that is typed second is used for the vertical axis. If only one vector is entered as an input argument in the `plot` command (for example `plot(y)`) than the figure will show a plot of the values of the elements of the vector ($y(1)$, $y(2)$, $y(3)$, …) versus the element number (1, 2, 3, …).

The figure that is created has axes with a linear scale and default range. For 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 the following in the Command Window:

```
>> x=[1.1  1.8  3.2  5.5  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, as shown in Figure 5-2.

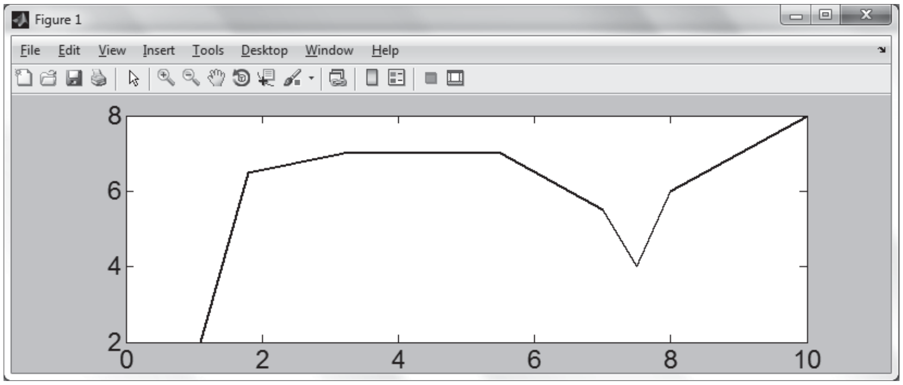


Figure 5-2: The Figure Window with a simple plot.

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

```
plot(x,y,'line specifiers','PropertyName',PropertyValue)
```

Vector Vector

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

(Optional) Properties with values that can be used to specify the line width, and a marker's size and edge, and fill colors.

Line Specifiers:

Line specifiers are optional and can be used to define the style and color of the line and the type of markers (if markers are desired). The line style specifiers are:

Line Style	Specifier
solid (default)	-
dashed	--

Line Style	Specifier
dotted	:
dash-dot	-.

The line color specifiers are:

Line Color	Specifier
red	`r`
green	`g`
blue	`b`
cyan	`c`

Line Color	Specifier
magenta	`m`
yellow	`y`
black	`k`
white	`w`

The marker type specifiers are:

Marker Type	Specifier		Marker Type	Specifier
plus sign	`+`		square	`s`
circle	`o`		diamond	`d`
asterisk	`*`		five-pointed star	`p`
point	`.`		six-pointed star	`h`
cross	`x`		triangle (pointed left)	`<`
triangle (pointed up)	`^`		triangle (pointed right)	`>`
triangle (pointed down)	`v`			

Notes about using the specifiers:

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

Some examples:

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

`Property Name and Property Value:`

Properties are optional and can be used to specify the thickness of the line, the size of the marker, and the colors of the marker's edge line and fill. The Property Name is typed as a string, followed by a comma and a value for the property, all inside the `plot` command.

Four properties and their 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 2 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 indicate 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	`linestyle` (or `LineStyle`)	Line style specifier from the table above, typed as a string.
Line color	`color` (or `Color`)	Color specifier from the table above, typed as a string.
Marker	`marker` (or `Marker`)	Marker specifier from the table above, typed as a string.

As with any command, the `plot` command can be typed in the Command Window, or it can be included in a script file. It also can be used in a function file (explained in Chapter 7). It should also be remembered that before the `plot` command can be executed, the vectors `x` and `y` must have assigned elements.

This can be done, as was explained in Chapter 2, by entering values directly, by using commands, or as the result of mathematical operations. The next two subsections show examples of creating simple plots.

5.1.1 Plot of Given Data

In this case given data is used to create vectors that are then used in the `plot` command. 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

To plot this data, the list of years is assigned to one vector (named `yr`), and the corresponding sales data is assigned to a second vector (named `sle`). The Command Window where 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 marker size is 12 points.

Once the `plot` command is executed, the Figure Window with the plot, as shown in Figure 5-3, opens. The plot appears on the screen in red.

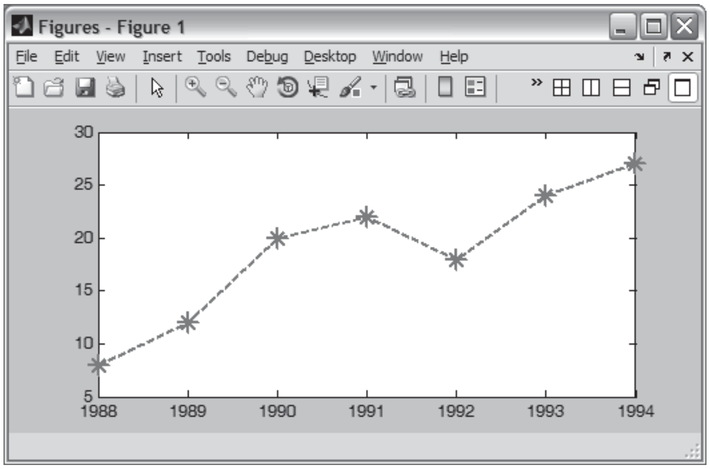


Figure 5-3: The Figure Window with a plot of the sales data.

5.1.2 *Plot of a Function*

In many situations there is a need to plot a given function. This can be done in MATLAB by using the `plot` or the `fplot` command. The use of the `plot` command is explained below. The `fplot` command is explained in detail in the next section.

In order to plot a function $y = f(x)$ with the `plot` command, the user needs to first create a vector of values of x for the domain over which the function will be plotted. Then a vector y is created with the corresponding values of $f(x)$ by using element-by-element calculations (see Chapter 3). Once the two vectors are defined, they can be used in the `plot` command.

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

```
%  A script file that creates a plot of
%  the function: 3.5.^(-0.5*x).*cos(6x)
x=[-2:0.01:4];            Create vector x with the domain of the function.
y=3.5.^(-0.5*x).*cos(6*x);       Create vector y with the func-
                                 tion value at each x.
plot(x,y)                            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 5-4. Since the plot is made up of segments of straight lines that connect the points, 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

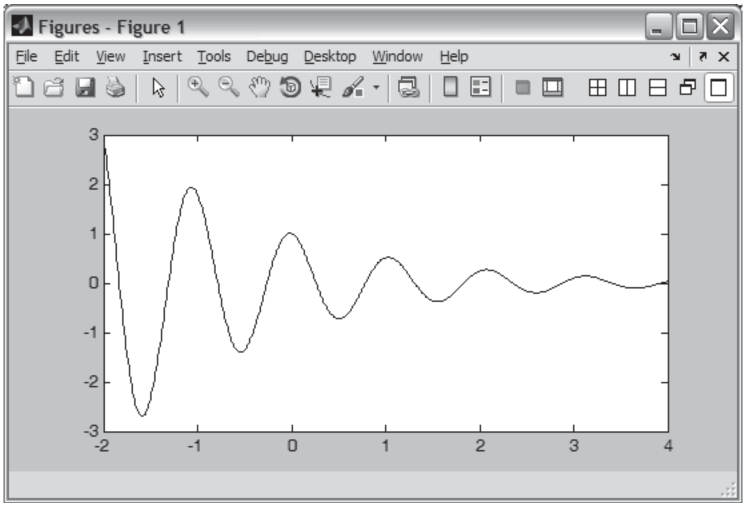
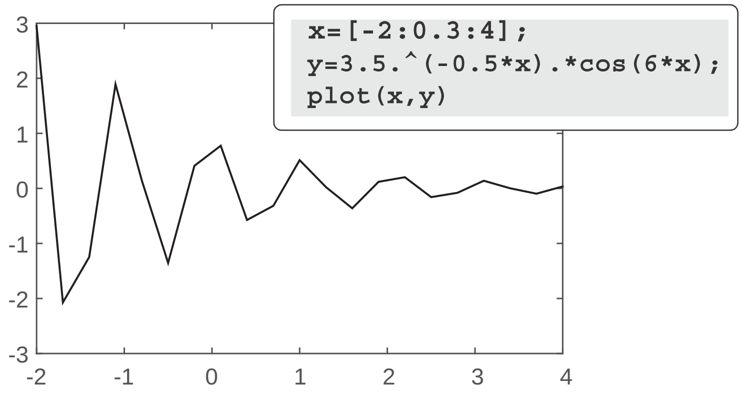


Figure 5-4: The Figure Window with a plot of the function $y = 3.5^{-0.5x}\cos(6x)$.

function that changes rapidly. In the last example a small spacing of 0.01 produced the plot that is shown in Figure 5-4. However, 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 5-5, gives a distorted picture of the function.



```
x=[-2:0.3:4];
y=3.5.^(-0.5*x).*cos(6*x);
plot(x,y)
```

Figure 5-5: A plot of the function $y = 3.5^{-0.5x}\cos(6x)$ with large spacing.

Note also that in Figure 5-4 the plot is shown with the Figure Window, while in Figure 5-5 only the plot is shown. The plot can be copied from the Figure Window (in the **Edit** menu, select **Copy Figure**) and then pasted into other applications.

5.2 *THE* `fplot` *COMMAND*

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

```
fplot(function,limits,'line specifiers')
```

The function to be plotted.

The domain of x typed as two-element vector.

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

`'function':` The function should be typed in the form of an anonymous function (covered in detail in Section 7.8). The form of an anonymous function is: `@ (x) f(x)`. For example, if the function $f(x) = 8x^2 + 5\cos(x)$ is to be plotted, it is typed as: `@ (x) 8*x.^2+5*cos(x)`. The functions can include MATLAB built-in functions and functions that are created by the user (covered in Chapter 7).

- The function should be typed using element-by-element operations, and can include previously defined variables. For example, in the function above it is possible to assign 8 to a variable, and then use the variable when the function is typed in the `fplot` command.

- The function to be plotted can be typed as a function of any letter. For example, the function in the previous paragraph can be typed as `@ (t) 8*t.^2+5*cos(t)` or `@ (z) 8*z.^2+5*cos(z)'`.

`limits:` The limits argument is a vector with two elements that specify the domain of x [`xmin,xmax`], or a vector with four elements that specifies the domain of x and the limits of the y-axis [`xmin,xmax,ymin,ymax`].

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

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

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

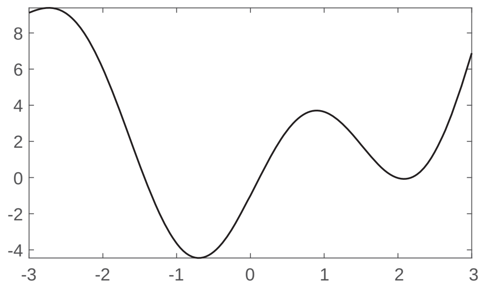


Figure 5-6: A plot of the function $y = x^2 + 4\sin(2x) - 1$.

5.3 *PLOTTING MULTIPLE GRAPHS IN THE SAME PLOT*

In many situations, there is a need to make several graphs in the same plot. This is shown, 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. One is by using the `plot` command, the second is by using the `hold on` and `hold off` commands, and the third is by using the `line` command.

5.3.1 *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 the command

```
plot(x,y,‘-b’,u,v,‘--r’,t,h,‘g:’)
```

plots `y` vs. `x` with a solid blue line, `v` vs.`u` with a dashed red line, and `h` vs. `t` with a dotted green line.

Sample Problem 5-1: Plotting a function and its derivatives

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

Solution

The first derivative of the function is: $y' = 9x^2 - 26$.
The second derivative of the function is: $y'' = 18x$.
A script file that creates a vector x and calculates the values of y, y', and y'' is:

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

Create vector `x` with the domain of the function.

Create vector `y` with the function value at each `x`.

Create vector `yd` with values of the first derivative.

Create vector `ydd` with values of the second derivative.

Create three graphs, `y` vs. `x`, `yd` vs. `x`, and `ydd` vs. `x`, in the same figure.

The plot that is created is shown in Figure 5-7.

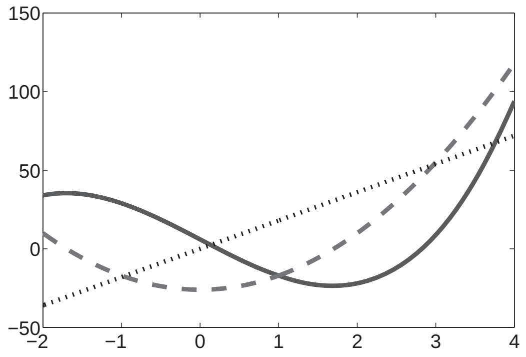


Figure 5-7: A plot of the function $y = 3x^3 - 26x + 10$ and its first and second derivatives.

5.3.2 *Using the* `hold on` *and* `hold off` *Commands*

To plot several graphs using the `hold on` and `hold off` commands, one graph is plotted first with the `plot` command. Then the `hold on` command is typed. This keeps the Figure Window with the first plot open, including the axis properties and formatting (see Section 5.4) if any was done. Additional graphs can be added with `plot` commands that are typed next. Each `plot` command creates a graph that is added to that figure. The `hold off` command stops this process. It returns MATLAB to the default mode, in which the `plot` command erases the previous plot and resets the axis properties.

As an example, a solution of Sample Problem 5-1 using the `hold on` and `hold off` commands is shown in the following script file:

```
x=[-2:0.01:4];
y=3*x.^3-26*x+6;
yd=9*x.^2-26;
ydd=18*x;
plot(x,y,'-b')                    The first graph is created.
hold on
plot(x,yd,'--r')          Two more graphs are added to the figure.
plot(x,ydd,':k')
hold off
```

5.3.3 *Using the* `line` *Command*

With the `line` command additional graphs (lines) can be added to a plot that already exists. The form of the line command is:

```
line(x,y,'PropertyName',PropertyValue)
```

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

The format of the `line` command is almost the same as the `plot` command (see Section 5.1). The `line` command does not have the line specifiers, but the line style, color, and marker can be specified with the Property Name and property value features. The properties are optional, and if none are entered MATLAB uses default properties and values. For example, the command:

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

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

The major difference between the `plot` and `line` commands is that the `plot` command starts a new plot every time it is executed, while the `line` command adds lines to a plot that already exists. To make a plot that has several graphs, a plot command is typed first and then line commands are typed for additional graphs. (If a line command is entered before a plot command, an error message is displayed.)

The solution to Sample Problem 5-1, which is the plot in Figure 5-7, can be obtained by using the `plot` and `line` commands as shown in the following script file:

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

5.4 *Formatting a Plot*

The `plot` and `fplot` commands create bare plots. Usually, however, a figure that contains a plot needs to be formatted to have a specific look and to display information in addition to the graph itself. This can include specifying axis labels, plot title, legend, grid, range of custom axis, and text labels.

Plots can be formatted by using MATLAB commands that follow the `plot` or `fplot` command, or interactively by using the plot editor in the Figure Window. The first method is useful when a `plot` command is a part of a computer program (script file). When the formatting commands are included in the program, a formatted plot is created every time the program is executed. On the other hand, formatting that is done in the Figure Window with the plot editor after a plot has been created holds only for that specific plot, and will have to be repeated the next time the plot is created.

5.4.1 Formatting a Plot Using Commands

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

The `xlabel` and `ylabel` commands:

Labels can be placed next to the axes with the `xlabel` and `ylabel` command which have the form:

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

The `title` command:

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

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

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

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.

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', ..... ,'Location','pos')
```

The strings are the labels that are placed next to the line sample. Their order corresponds to the order in which the graphs were created. The `'Location','pos'` are optional strings that specifies where in the figure the legend is to be placed. Several options are:

`NE`	Places the legend at the upper-right corner of the plot (default).
`NW`	Places the legend at the upper-left corner of the plot.
`SE`	Places the legend at the lower-right corner of the plot.
`SW`	Places the legend at the lower-left corner of the plot.
`B`	Places the legend inside the plot in a location that interferes the least with the graphs.
`BO`	Places the legend in a least unused space outside the plot.

To read about other options for the position of the legend type `help legend` in the Command Window.

Formatting the text within the `xlabel`, `ylabel`, `title`, `text`

and `legend` commands:

The text in the string that is included in the command and is displayed when the command is executed can be formatted. The formatting can be used to define the font, size, position (superscript, subscript), style (italic, bold, etc.), and color of the characters, the color of the background, and many other details of the display. Some of the more common formatting possibilities are described below. 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
`\bf`	bold font
`\it`	italic style
`\rm`	normal font

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

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

Subscript and superscript:

A single character can be displayed as a subscript or a superscript by typing _ (the underscore character) or ^ in front of the character, respectively. Several consecutive characters can be displayed as a subscript or a superscript by typing the characters inside braces { } following the _ or the ^.

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
`\alpha`	α
`\beta`	β
`\gamma`	γ
`\theta`	θ
`\pi`	π
`\sigma`	σ

Characters in the string	Greek letter
`\Phi`	Φ
`\Delta`	Δ
`\Gamma`	Γ
`\Lambda`	Λ
`\Omega`	Ω
`\Sigma`	Σ

Formatting 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.

With this option, the `text` command, for example, has the form:

```
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 (see Section 5.1).
`Background-Color`	Specifies the background color (rectangular area).	Color specifiers (see Section 5.1).
`EdgeColor`	Specifies the color of the edge of a rectangular box around the text.	Color specifiers (see Section 5.1). Default: none.
`LineWidth`	Specifies the width of the edge of a rectangular box around the text.	Scalar (points) Default: 0.5

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. In many situations, a graph looks better if the range of the axes extend beyond the range of the data. The following are some of the possible forms of the `axis` command:

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

The `grid` command:

`grid on`	Adds grid lines to the plot.
`grid off`	Removes grid lines from the plot.

An example of formatting a plot by using commands is given in the following script file that was used to generate the formatted plot in Figure 5-1.

```
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.','Edge-
Color','r','LineWidth',2)
hold on
plot(xd,yd,'ro--','linewidth',1.0,'markersize'
legend('Theory','Experiment',0)
hold off
```

Formatting text inside the `title` command.

Formatting text inside the `text` command.

5.4.2 *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 5-8 shows the Figure Window with the plot of Figure 5-1. The Plot Editor can be used to introduce new formatting items or to modify formatting that was initially introduced with the formatting commands.

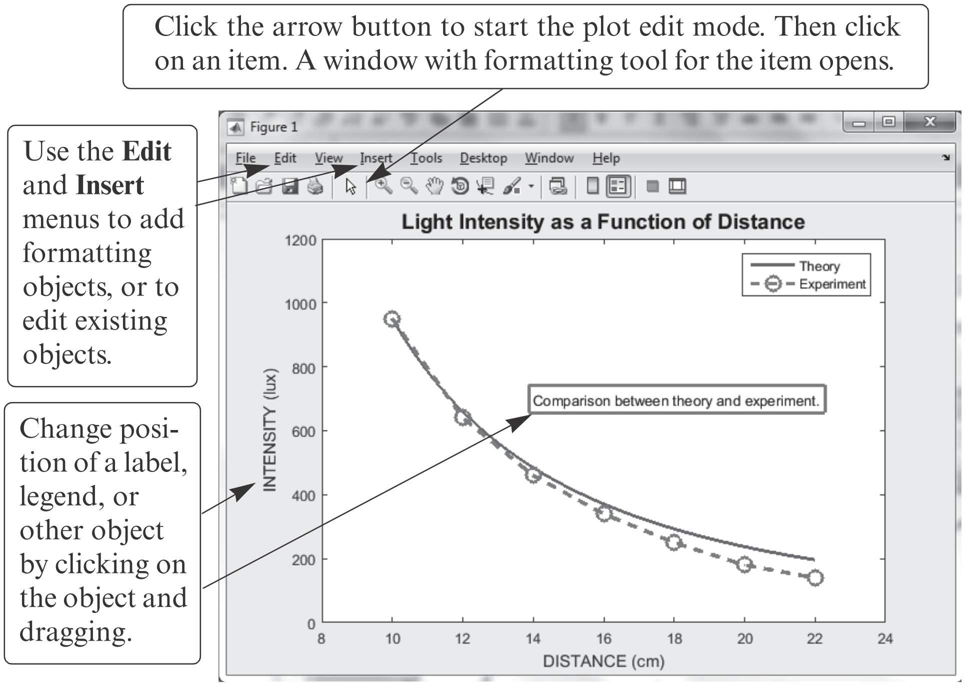

Click the arrow button to start the plot edit mode. Then click on an item. A window with formatting tool for the item opens.
Use the **Edit** and **Insert** menus to add formatting objects, or to edit existing objects.
Change position of a label, legend, or other object by clicking on the object and dragging.
Figure 1
File Edit View Insert Tools Desktop Window Help
Light Intensity as a Function of Distance
Theory
Experiment
Comparison between theory and experiment.
INTENSITY (lux)
DISTANCE (cm)
0
200
400
600
800
1000
1200
8
10
12
14
16
18
20
22
24

Figure 5-8: Formatting a plot using the Plot Editor.

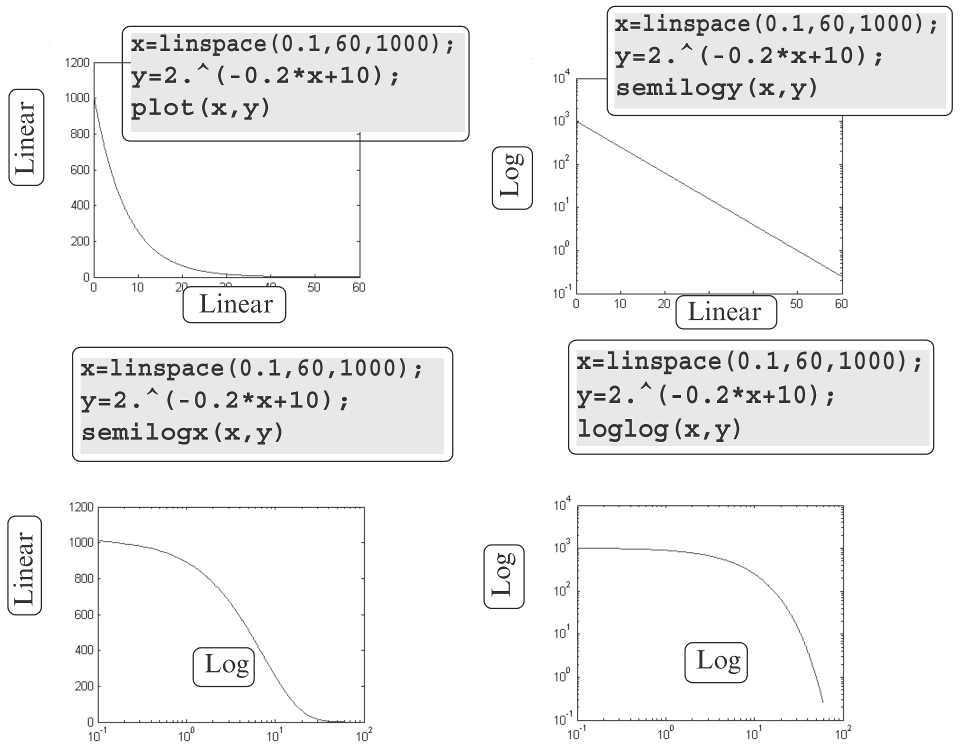
5.5 *Plots with Logarithmic Axes*

Many science and engineering applications require plots in which one or both axes have a logarithmic (log) scale. Log scales provide means for presenting data over a wide range of values. It also provides a tool for identifying characteristics of data and possible forms of mathematical relationships that can be appropriate for modeling the data (see Section 8.2.2).

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 arguments can be added to the commands (optional) just as in the `plot` command. As an example, Figure 5-9 shows a plot of the function $y = 2^{(-0.2x+10)}$ for $0.1 \le x \le 60$. The figure shows four plots of the same function: one with linear axes, one with a log scale for the y axis, one with a log scale for the x axis, and one with a log scale on both axes.



```
x=linspace(0.1,60,1000);
y=2.^(-0.2*x+10);
plot(x,y)
```

```
x=linspace(0.1,60,1000);
y=2.^(-0.2*x+10);
semilogy(x,y)
```

```
x=linspace(0.1,60,1000);
y=2.^(-0.2*x+10);
semilogx(x,y)
```

```
x=linspace(0.1,60,1000);
y=2.^(-0.2*x+10);
loglog(x,y)
```

Figure 5-9: Plots of $y = 2^{(-0.2x+10)}$ with linear, semilog, and log-log scales.

Notes for plots with logarithmic axes:

- 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).

5.6 Plots with Error Bars

Experimental data that is measured and then displayed in plots frequently contains error and scatter. Even data that is generated by computational models includes error or uncertainty that depends on the accuracy of the input parameters and the assumptions in the mathematical models that are used. One method of plotting data that displays the error, or uncertainty, is by using error bars. An error bar is typically a short vertical line that is attached to a data point in a plot. It shows the magnitude of the error that is associated with the value that is displayed by the data point. For example, Figure 5-10 shows a plot with error bars for the experimental data from Figure 5-1.

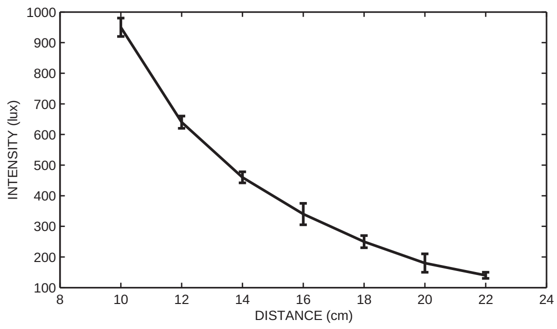


Figure 5-10: A plot with error bars.

Plots with error bars can be done in MATLAB with the `errorbar` command. Two forms of the command, one for making plots with symmetric error bars (with respect to the value of the data point) and the other for nonsymmetric error bars at each point, are presented. When the error is symmetric, the error bar extends the same length above and below the data point, and the command has the form:

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

Vectors with horizontal and vertical coordinates of each point.

Vector with the value of the error at each point.

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

The plot in Figure 5-10, which has symmetric error bars, was done by executing the following code:

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

The command for making a plot with error bars that are not symmetric is:

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

Vectors with horizontal and vertical coordinates of each point.

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

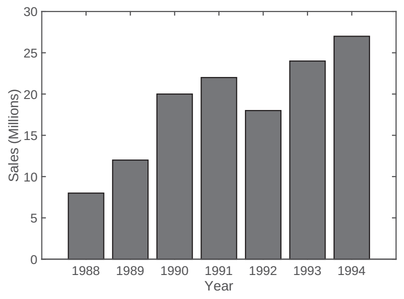
Vector with the lower-bound value of the error at each point.

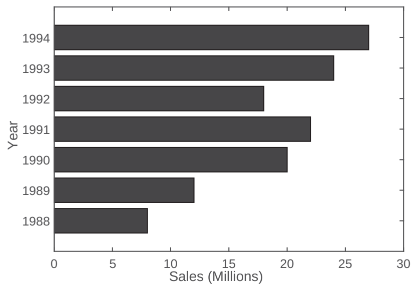
- The lengths of the four vectors `x`, `y`, `d`, and `u` must be the same.
- At each point the error bar extends from `y(i)-d(i)` to `y(i)+u(i)`.

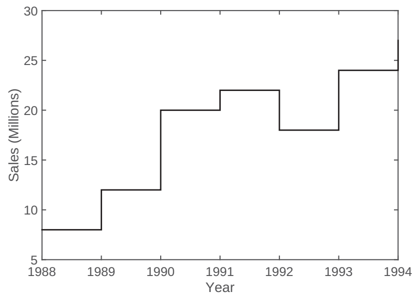
5.7 *Plots with Special Graphics*

All the plots that have been presented so far in this chapter are line plots in which the data points are connected by lines. In many situations plots with different graphics or geometry can present data more effectively. MATLAB has many options for creating a wide variety of plots. These include bar, stairs, stem, and pie plots and many more. Following are some of the special graphics plots that can be created with MATLAB. A complete list of the plotting functions that MATLAB offers and information on how to use them can be found in the Help Window. In this window first choose "Functions by Category," then select "Graphics" and then select "Basic Plots and Graphs" or "Specialized Plotting."

Bar (vertical and horizontal), stairs, and stem plots are presented in the following charts using the sales data from Section 5.1.1.

Vertical Bar Plot
 	
Function format:
 	
`bar(x,y)`
		`yr=[1988:1994];`
		`sle=[8 12 20 22 18 24 27];`
		`bar(yr,sle,'r')` ← The bars are in red.
		`xlabel('Year')`
		`ylabel('Sales (Millions)')`

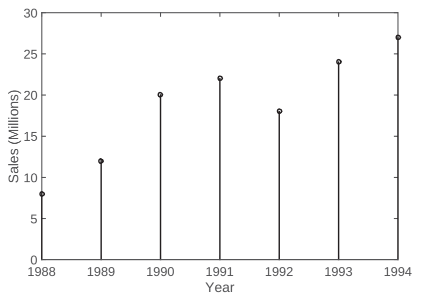
Horizontal Bar Plot
 	
Function format:
 	
`barh(x,y)`
		`yr=[1988:1994];`
		`sle=[8 12 20 22 18 24 27];`
		`barh(yr,sle)`
		`xlabel('Sales (Millions)')`
		`ylabel('Year')`

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

Stem Plot

Function Format

`stem(x,y)`



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

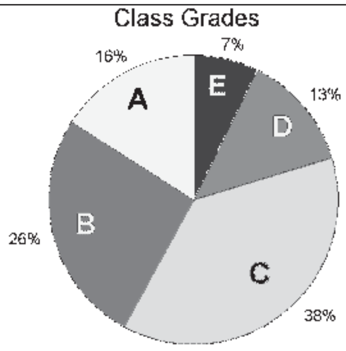
Pie charts are useful for visualizing the relative sizes of different but related quantities. For example, the table below shows the grades that were assigned to a class. The data is used to create the pie chart that follows.

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

Pie Plot

Function format:

`pie(x)`



```
grd=[11 18 26 9 5];
pie(grd)
title('Class Grades')
```

MATLAB draws the sections in different colors. The letters (grades) were added using the Plot Editor.

5.8 *HISTOGRAMS*

Histograms are plots that show the distribution of data. The overall range of a given set of data points is divided into subranges (bins), and the histogram shows how many data points are in each bin. The histogram is a vertical bar plot in which the width of each bar is equal to the range of the corresponding bin and the height of the bar corresponds to the number of data points in the bin. Histograms are created in MATLAB with the `hist` command. The simplest form of the command is:

```
hist(y)
```

`y` is a vector with the data points. MATLAB divides the range of the data points into 10 equally spaced subranges (bins) and then plots the number of data points in each bin.

For example, the following data points are the daily maximum temperature (in °F) in Washington, DC, during the month of April 2002: 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 (data from the U.S. National Oceanic and Atmospheric Administration). 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 5-11 (the axis 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 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.

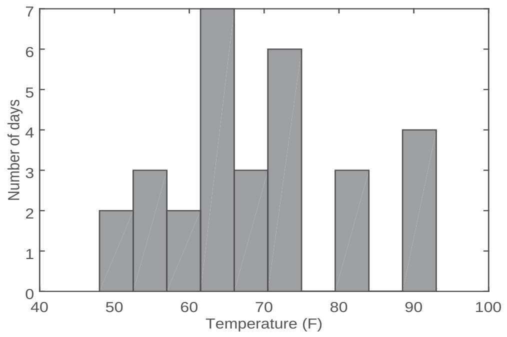


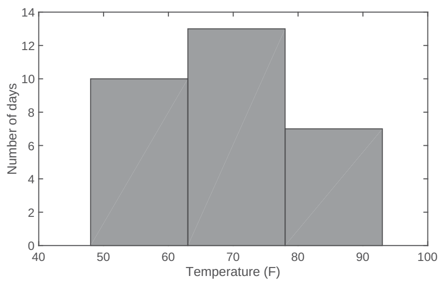
Figure 5-11: Histogram of temperature data.

Since the division of the data range into 10 equally spaced bins might not be the division that is preferred by the user, the number of bins can be defined to be different than 10. This can be done either by specifying the number of bins, or by specifying the center point of each bin as shown in the following two forms of the `hist` command:

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

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

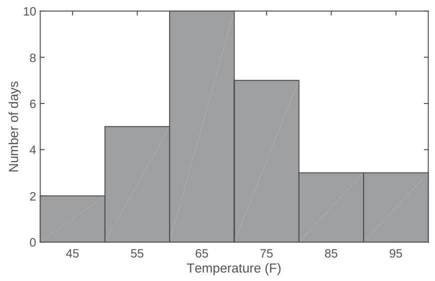
`x` is a vector that specifies the location of the center of each bin (the distance between the centers does not have to be the same for all the bins). The edges of the bins are at the middle point between the centers.



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

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

As shown in the top graph, the histogram that is generated has three equally spaced bins.



The number and width of the bins can also be specified by a vector x whose elements define the centers of the bins. For example, shown in the lower graph is a histogram that displays the temperature data from above in six bins with an equal width of 10 degrees. The elements of the vector x for this plot are 45, 55, 65, 75, 85, and 95. The plot was obtained with the following commands:

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

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

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

The output n is a vector. The number of elements in n is equal to the number of bins, and the value of each element of n is the number of data points (frequency count) in the corresponding bin. For example, the histogram in Figure 5-11 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.

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

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

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

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

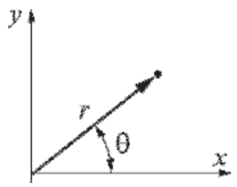
`xout` is a vector in which the value of each element is the location of the center of the corresponding bin. For example, for the histogram in Figure 5-11:

```
>> [n xout]=hist(y)
n =
   2   3   2   7   3   6   0   3   0   4
xout =
    50.2500   54.7500   59.2500   63.7500   68.2500   72.7500
77.2500  81.7500  86.2500  90.7500
```

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

5.9 *Polar Plots*

Polar coordinates, in which the position of a point in a plane is defined by the angle θ and the radius (distance) to the point, are frequently used in the solution of science and engineering problems. The `polar` command is used to plot functions in polar coordinates. The command has the form:



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

Vector

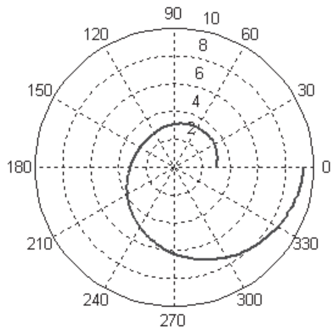
Vector

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

where `theta` and `radius` are vectors whose elements define the coordinates of the points to be plotted. The `polar` command plots the points and draws the polar grid. The line specifiers are the same as in the `plot` command. To plot a function $r = f(\theta)$ 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.

For example, a plot of the function $r = 3\cos^2(0.5\theta) + \theta$ for $0 \le \theta \le 2\pi$ is shown below.

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

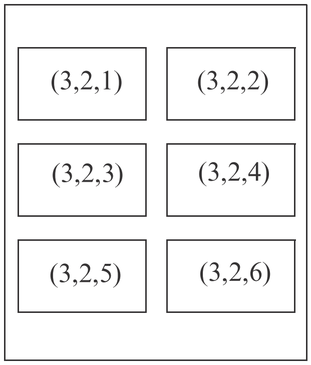


5.10 Putting Multiple Plots on the Same Page

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

$$\boxed{\texttt{subplot(m,n,p)}}$$

The command divides the Figure Window (and the page when printed) into $m \times n$ rectangular subplots. The subplots are arranged like elements in an $m \times n$ matrix where each element is a subplot. The subplots are numbered from 1 through $m \cdot n$. The upper left subplot is numbered 1, and the lower right subplot is numbered $m \cdot n$. The numbers increase from left to right within a row, from the first row to the last. The command `subplot(m,n,p)` makes the subplot `p` current. This means that the next plot command (and any formatting commands) will create a plot (with the corresponding format) in this subplot. For example, the command `subplot(3,2,1)` creates six areas arranged in three rows and two columns as shown, and makes the upper left subplot current. An example of using the `subplot` command is shown in the solution of Sample Problem 5-2.



5.11 Multiple Figure Windows

When `plot` or any other command that generates a plot is executed, the Figure Window opens (if not already open) and displays the plot. MATLAB labels the Figure Window as Figure 1 (see the top left corner of the Figure Window that is displayed in Figure 5-4). If the Figure Window is already open when the `plot` or any other command that generates a plot is executed, a new plot is displayed in the Figure Window (replacing the existing plot). Commands that format plots are applied to the plot in the Figure Window that is open.

It is possible, however, to open additional Figure Windows and have several of them open (with plots) at the same time. This is done by typing the command `figure`. Every time the command `figure` is entered, MATLAB opens a new Figure Window. If a command that creates a plot is entered after a `figure` command, MATLAB generates and displays the new plot in the last Figure Window that was opened, which is called the active or current window. MATLAB labels the new Figure Windows successively; i.e., Figure 2, Figure 3, and so on. For example, after the following three commands are entered, the two Figure Windows that are shown in Figure 5-12 are displayed.

```
>> fplot(@ (x) x.*cos(x),[0,10])          Plot displayed in Figure 1 window.
>> figure                                  Figure 2 window opens.
>> fplot(@ (x)exp(-0.2*x).*cos(x),[0,10])  Plot displayed in Figure 2 window.
```

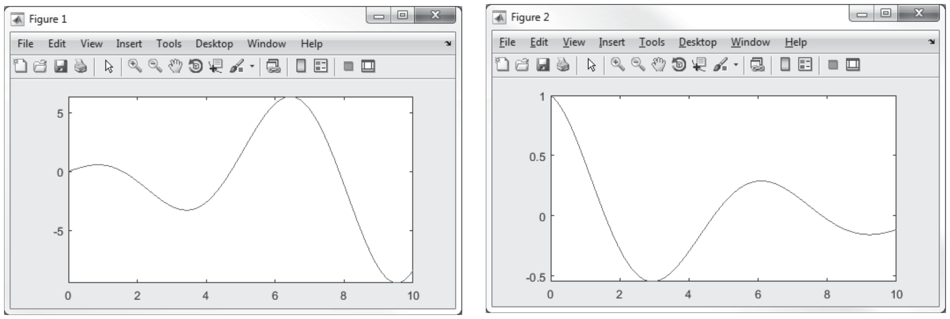


Figure 1
File Edit View Insert Tools Desktop Window Help
5
0
-5
0 2 4 6 8 10
Figure 2
File Edit View Insert Tools Desktop Window Help
1
0.5
0
-0.5
0 2 4 6 8 10

Figure 5-12: Two open Figure Windows.

The `figure` command can also have an input argument that is a number (integer), of the form `figure(n)`. The number corresponds to the number of the corresponding Figure Window. When the command is executed, window number `n` becomes the active Figure Window (if a Figure Window with this number does not exist, a new window with this number opens). When commands that create new plots are executed, the plots that they generate are displayed in the active Figure Window. In the same way, commands that format plots are applied to the plot in the active window. The `figure(n)` command provides means for having a program in a script file that opens and makes plots in a few defined Figure Windows. (If several `figure` commands are used in a program instead, new Figure Windows will open every time the script file is executed.)

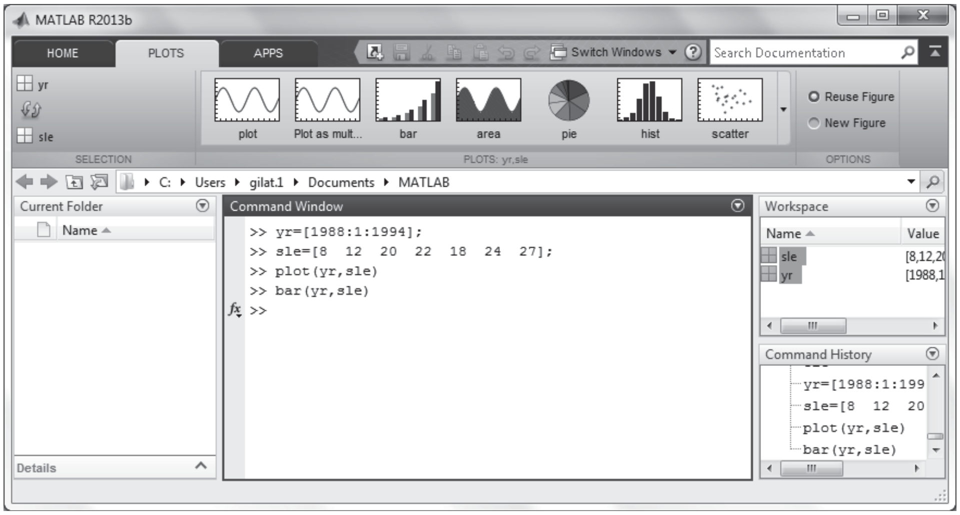
Figure Windows can be closed with the `close` command. Several forms of the command are:

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

5.12 PLOTTING USING THE PLOTS TOOLSTRIP

Plots can also be constructed interactively by using the PLOTS Toolstrip in the Command Window. The PLOTS Toolstrip, as shown in Fig. 5-13, is displayed when the PLOTS tab is selected. To make a two-dimensional plot, the vectors with the data points that will be used for the plot have to be already assigned and displayed in the Workspace Window (see Section 4.1). To make a plot, select a variable in the Workspace Window and then, holding the CTRL key, select any additional variables needed. Once a selection of variables has been made, the Toolstrip shows icons with images of plot types that can be created with the selected variables (e.g. line graph, scatter plot, bar graph, pie chart, etc.). Clicking on an icon opens a Figure Window with the corresponding figure displayed. In addition, the MATLAB command that created the plot is displayed in the Command Window. The user can then copy the command and paste it into a script file such that in the future the same figure will be created when the script file is executed. On the right side of the Toolstrip the user can choose to view different plot types in the same Figure Window (Reuse Figure), or to view a new figure type in a new Figure Window (New Figure), such that figure types can be compared side by side.

Using the Plots Toolstrip is useful when the user wants to examine different plot options for given data. For example, Figure 5-13 shows the default layout of MATLAB with the PLOTS Toolstrip displayed. In the Command Window, the sales data from Section 5.1.1 are assigned to two vectors `yr` and `sle`. The vectors are also displayed (and selected) in the Workspace Window. Icons with images of various type of plots that can be created are displayed in the PLOTS Toolstrip at the top. Additional types of plots can be displayed by clicking on the down-arrow on the right.



```
>> yr=[1988:1:1994];
>> sle=[8  12  20  22  18  24  27];
>> plot(yr,sle)
>> bar(yr,sle)
>>
```

Figure 5-13: Using the PLOTS Toolstrip.

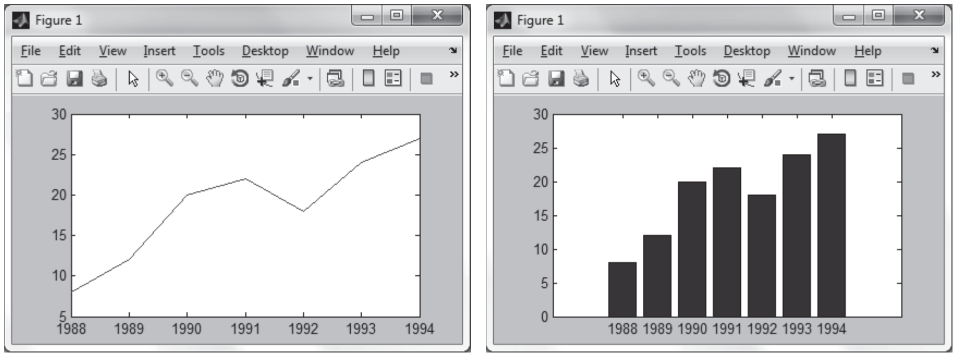
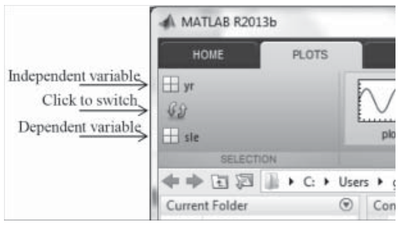


Figure 5-14: Using the PLOTS Toolstrip.

As an example, two different figures, one with line plot and the other with bar plot, were created using the two vectors `yr` and `sle`. The two figures are displayed in Figure 5-14 and the commands that created the plots are displayed in the Command Window in Figure 5-13.

Additional notes:

- When selecting variables for the plot (in the Workspace Window), the first to be selected will be the independent variable (horizontal axis) and the second will be the dependent variable (vertical axis). After the selection, the variables can be switched by clicking on the Switch icon.

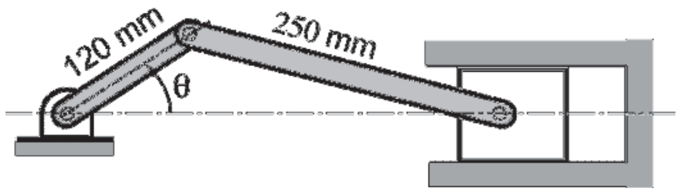


- If only one variable (vector) is selected for a figure, the values of the vector elements will be plotted versus the number of the element.

5.13 *Examples of MATLAB Applications*

Sample Problem 5-2: Piston-crank mechanism

The piston-rod-crank mechanism is used in many engineering applications. In the mechanism shown in the following figure, the crank is rotating at a constant speed of 500 rpm.



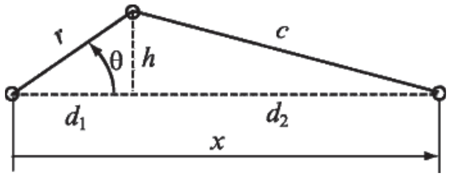
Calculate and plot the position, velocity, and acceleration of the piston for one revolution of the crank. Make the three plots on the same page. Set $\theta = 0°$ when $t = 0$.

Solution

The crank is rotating with a constant angular velocity $\dot{\theta}$. This means that if we set $\theta = 0°$ when $t = 0$, then at time t the angle θ is given by $\theta = \dot{\theta} t$, and means that $\ddot{\theta} = 0$ at all times.

The distances d_1 and h are given by:

$d_1 = r\cos\theta$ and $h = r\sin\theta$



With h known, the distance d_2 can be calculated using the Pythagorean Theorem:

$$d_2 = (c^2 - h^2)^{1/2} = (c^2 - r^2\sin^2\theta)^{1/2}$$

The position x of the piston is then given by:

$$x = d_1 + d_2 = r\cos\theta + (c^2 - r^2\sin^2\theta)^{1/2}$$

The derivative of x with respect to time gives the velocity of the piston:

$$\dot{x} = -r\dot{\theta}\sin\theta - \frac{r^2\dot{\theta}\sin 2\theta}{2(c^2 - r^2\sin^2\theta)^{1/2}}$$

The second derivative of x with respect to time gives the acceleration of the piston:

$$\ddot{x} = -r\dot{\theta}^2\cos\theta - \frac{4r^2\dot{\theta}^2\cos 2\theta\,(c^2 - r^2\sin^2\theta) + (r^2\dot{\theta}\sin 2\theta)^2}{4(c^2 - r^2\sin^2\theta)^{3/2}}$$

In the equation above, $\ddot{\theta}$ was taken to be zero.

A MATLAB program (script file) that calculates and plots the position, velocity, and acceleration of the piston for one revolution of the crank is shown below.

```
THDrpm=500; r=0.12; c=0.25;          Define θ̇, r, and c.
THD=THDrpm*2*pi/60;                  Change the units of θ̇ from rpm to rad/s.
tf=2*pi/THD;                         Calculate the time for one revolution of the crank.
t=linspace(0,tf,200);                Create a vector for the time with 200 elements.
TH=THD*t;                            Calculate θ for each t.
d2s=c^2-r^2*sin(TH).^2;              Calculate d2 squared for each θ.
x=r*cos(TH)+sqrt(d2s);               Calculate x for each θ.
xd=-r*THD*sin(TH)-(r^2*THD*sin(2*TH))./(2*sqrt(d2s));
```

```
xdd=-r*THD^2*cos(TH)-(4*r^2*THD^2*cos(2*TH).*d2s+
(r^2*sin(2*TH)*THD).^2)./(4*d2s.^(3/2));
```

Calculate $\dot{x}$ and $\ddot{x}$ for each θ.

```
subplot(3,1,1)
plot(t,x)
```

Plot x vs. t.

```
grid
xlabel('Time (s)')
ylabel('Position (m)')
```

Format the first plot.

```
subplot(3,1,2)
plot(t,xd)
```

Plot $\dot{x}$ vs. t.

```
grid
xlabel('Time (s)')
ylabel('Velocity (m/s)')
```

Format the second plot.

```
subplot(3,1,3)
plot(t,xdd)
```

Plot $\ddot{x}$ vs. t.

```
grid
xlabel('Time (s)')
ylabel('Acceleration (m/s^2)')
```

Format the third plot.

When the script file runs it generates the three plots on the same page as shown in Figure 5-13. The figure nicely shows that the velocity of the piston is zero at the end points of the travel range where the piston changes the direction of the motion. The acceleration is maximum (directed to the left) when the piston is at the right end.

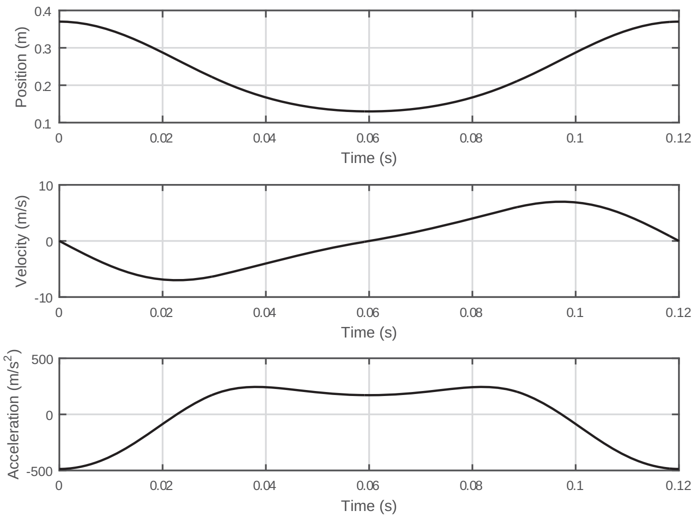
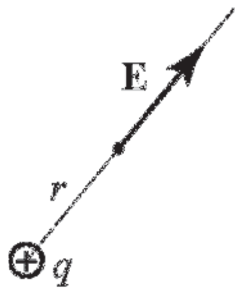


Figure 5-15: Position, velocity, and acceleration of the piston vs. time.

Sample Problem 5-3: Electric Dipole

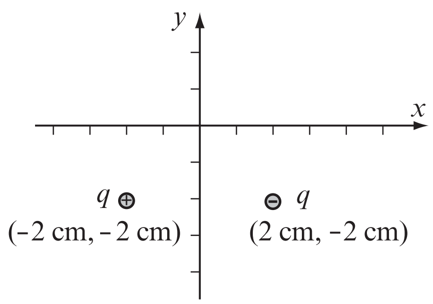
The electric field at a point due to a charge is a vector **E** with magnitude E given by Coulomb's law:

$$E = \frac{1}{4\pi\varepsilon_0}\frac{q}{r^2}$$



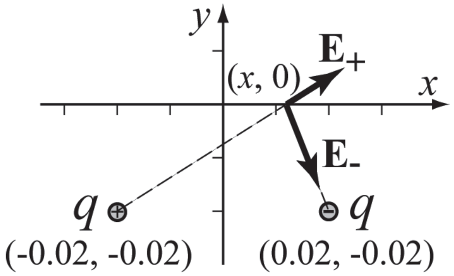
where $\epsilon_0 = 8.8541878 \times 10^{-12}\ \frac{\text{C}^2}{\text{N}\cdot\text{m}^2}$ is the permittivity constant, q is the magnitude of the charge, and r is the distance between the charge and the point. The direction of **E** is along the line that connects the charge with the point. **E** points outward from q if q is positive, and toward q if q is negative. An electric dipole is created when a positive charge and a negative charge of equal magnitude are placed some distance apart. The electric field, **E**, at any point is obtained by superposition of the electric field of each charge.

An electric dipole with $q = 12 \times 10^{-9}$ C is created, as shown in the figure. Determine and plot the magnitude of the electric field along the x axis from $x = -5$ cm to $x = 5$ cm.



Solution

The electric field **E** at any point (x, 0) along the x axis is obtained by adding the electric field vectors due to each of the charges.



$$\mathbf{E} = \mathbf{E}\_ + \mathbf{E}_+$$

The magnitude of the electric field is the length of the vector **E**.

The problem is solved by following these steps:

Step 1: Create a vector x for points along the x axis.

Step 2: Calculate the distance (and distance squared) from each charge to the points on the x axis.

$$r_{\text{minus}} = \sqrt{(0.02 - x)^2 + 0.02^2} \qquad r_{plus} = \sqrt{(0.02 + x)^2 + 0.02^2}$$

Step 3: Write unit vectors in the direction from each charge to the points on the x axis.

$$\mathbf{E}_{minusUV} = \frac{1}{r_{minus}}[(0.02 - x)\mathbf{i} - 0.02\mathbf{j}] \qquad \mathbf{E}_{plusUV} = \frac{1}{r_{plus}}[(0.02 + x)\mathbf{i} + 0.02\mathbf{j}]$$

Step 4: Calculate the magnitude of the vector $\mathbf{E}_-$ and $\mathbf{E}_+$ at each point by using Coulomb's law.

$$E_{minusMAG} = \frac{1}{4\pi\epsilon_0}\frac{q}{r_{minus}^2} \qquad E_{plusMAG} = \frac{1}{4\pi\epsilon_0}\frac{q}{r_{plus}^2}$$

Step 5: Create the vectors $\mathbf{E}_-$ and $\mathbf{E}_+$ by multiplying the unit vectors by the magnitudes.

Step 6: Create the vector $\mathbf{E}$ by adding the vectors $\mathbf{E}_-$ and $\mathbf{E}_+$.

Step 7: Calculate E, the magnitude (length) of $\mathbf{E}$.

Step 8: Plot E as a function of x.

A program in a script file that solves the problem is:

```
q=12e-9;
epsilon0=8.8541878e-12;
x=[-0.05:0.001:0.05]';
rminusS=(0.02-x).^2+0.02^2;
rminus=sqrt(rminusS);
rplusS=(x+0.02).^2+0.02^2;
rplus=sqrt(rplusS);
EminusUV=[((0.02-x)./rminus), (-0.02./rminus)];
EplusUV=[((x+0.02)./rplus), (0.02./rplus)];
EminusMAG=(q/(4*pi*epsilon0))./rminusS;
EplusMAG=(q/(4*pi*epsilon0))./rplusS;
Eminus=[EminusMAG.*EminusUV(:,1), EminusMAG.*EminusUV(:,2)];
Eplus=[EplusMAG.*EplusUV(:,1), EplusMAG.*EplusUV(:,2)];
E=Eminus+Eplus;
EMAG=sqrt(E(:,1).^2+E(:,2).^2);
plot(x,EMAG,'k','linewidth',1)
xlabel('Position along the x-axis (m)','FontSize',12)
ylabel('Magnitude of the electric field (N/C)','FontSize',12)
title('ELECTRIC FIELD DUE TO AN ELECTRIC DIPOLE','FontSize',12)
```

Create a column vector x.

Step 2. Each variable is a column vector.

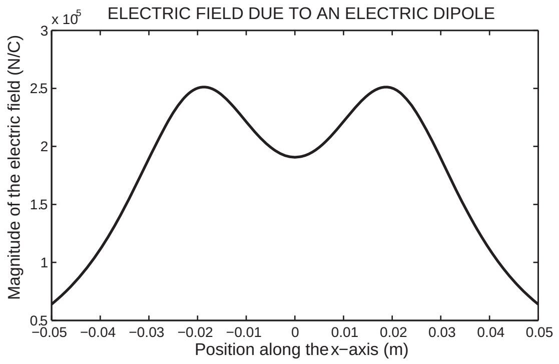
Steps 3 & 4. Each variable is a two column matrix. Each row is the vector for the corresponding x.

Step 6.

Step 7.

Step 5.

When this script file is executed in the Command Window, the following figure is created in the Figure Window:



5.14 Problems

1. Use the plot command to plot the function $f(x) = x^2 - 10\sqrt{x} + 2$ for $0 \le x \le 5$.

2. Use the plot command to plot the function $f(x) = (0.5x^4 + 1.1x^3 - 0.9x^2)e^{-0.7x}$ for $-3 \le x \le 10$.

3. Use the plot command to plot the function $f(x) = 3\cos(1.7x)e^{-0.3x} + 2\sin(1.4x)e^{0.3x}$ for $-7 \le x \le 7$.

4. Plot the function $f(x) = x^2e^{-x}$ and its derivative for $0 \le x \le 10$ in one figure. Plot the function with a solid line, and the derivative with a dashed line. Add a legend and label the axes.

5. Make two separate plots of the function $f(x) = x^4 - 2x^3 + 1.3x^2 - 0.3x + 0.02$, one plot for $-3 \le x \le 4$ and one for $0 \le x \le 1$.

6. Use the fplot command to plot the function $f(x) = 5(e^{-0.5x} - e^{-0.8x})$ for $0 \le x \le 10$.

7. Plot the function $f(x) = \sin(2x)\cos^2(0.5x)$ and its derivative, both on the same plot, for $-\pi \le x \le 2\pi$. Plot the function with a solid line and the derivative with a dashed line. Add a legend and label the axes.

8. The orbit of the planet Mercury around the sun can be approximated by the equation $r = \dfrac{3.44 \times 10^7}{1 - 0.206\cos\theta}$ miles. Make a plot of the orbit.

9. A parametric equation is given by

$$x = 0.7\sin(10t), \quad y = 1.2\sin(8t)$$

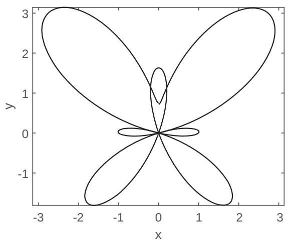
Plot the function for $0 \le t \le \pi$. Format the plot such that both axes will range from –1.5 to 1.5.

10. The butterfly curve (Fay, T. H. "The Butterfly Curve." Amer. Math. Monthly 96, pp. 442-443, 1989) is given by the following parametric equations:

$$x = \sin t\left[e^{\cos t} - 2\cos(4t) + \sin^5(t/12)\right]$$

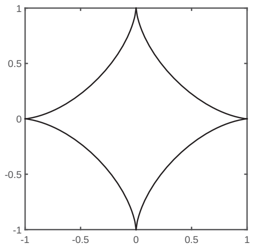
$$y = \cos t\left[e^{\cos t} - 2\cos(4t) + \sin^5(t/12)\right]$$

On one page make two plots of butterfly curves. One for $0 \le t \le 2\pi$ and the other for $0 \le t \le 10\pi$.



11. A plot of an astroid is shown in the figure on the right. Make the plot using the Cartesian equation:

$$x^{2/3} + y^{2/3} = 1$$



12. Make the plot of the astroid that is shown in the previous problem by using the parametric equation:

$x = \cos^3(t)$ and $y = \sin^3(t)$ for $-\pi \le x \le \pi$.

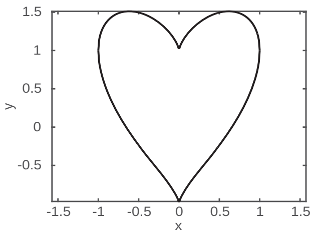
13. Plot the function $f(x) = \dfrac{x^2 - 6x + 7}{x^3 - 8}$ in the domain $0 \le x \le 4$. Notice that the function has a vertical asymptote at $x = 2$. Plot the function by creating two vectors for the domain of x. The first vector (name it $x1$) includes elements from 0 to 1.9, and the second vector (name it $x2$) includes elements from 2.1 to 4. For each x vector create a y vector (name 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. $x2$).

14. Plot the function $f(x) = x + \dfrac{1}{x^2 - 1}$ for $-4 \le x \le 4$. Notice that the function has two vertical asymptotes. Plot the function by dividing the domain of x into three parts: one from –4 to near the left asymptote, one between the two asymptotes, and one from near the right asymptote to 4. Set the range of the y axis from –15 to 15.

15. The shape of the heart shown in the figure is given by the equation:

$$x^2 + \left(y - \sqrt[3]{x^2}\right)^2 = 1$$

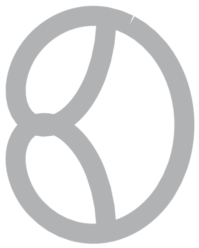
Make a plot of the heart.



16. The shape of the pretzel shown is given by the following parametric equations:

$$x = (3.3 - 0.3t^2)\cos t \quad y = (3.3 - 0.4t^2)\sin t$$

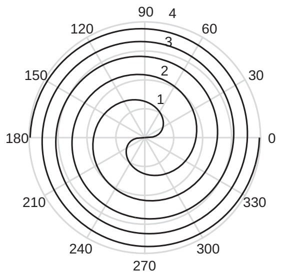
where $-4 \le t \le 4$. Make a plot of the pretzel.



17. Make a polar plot of the function:

$$r = \pm\sqrt{\theta} \quad \text{for} \quad 0 \le \theta \le 5\pi$$

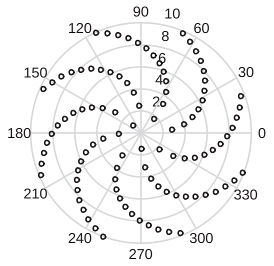
The plot, shown in the figure, is Fermat's spiral.



18. Make a polar plot of the function:

$$\theta = n\,135.7° \qquad r = \sqrt{n} \quad \text{for} \quad n = 1, 2, 3, \ldots, 100$$

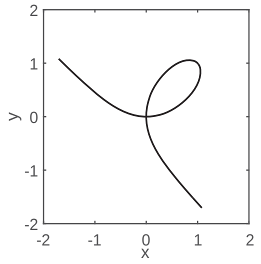
The plot is shown on the right.



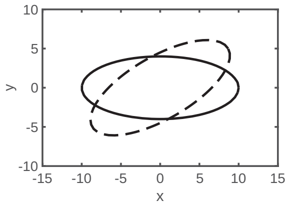
19. Make a plot (shown) of the function:

$$x^3 + y^3 = 2xy$$

(Hint: Rewrite the function in a polar form.)



20. Plot two ellipses is one figure (shown). The ellipse with the solid line has major axes of $a = 10$ and $b = 4$. The ellipse with the dashed line is the solid-line ellipse rotated by 30º.



21. The following data gives the height (in inches) of a sunflower plant as a function of time (days after it was planted).

Time (*days*)	10	20	30	40	50	60	70
Height (*in.*)	9	22	44	63	80	94	97

The height can be modeled by the logistic function:

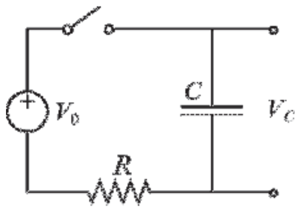
$$H = \frac{100.8}{1+23e^{-0.093t}}$$

where H is the height (in.) and t is the time (days). Make a plot of the height versus time. The figure should show the data from the table above as points and the height modeled by the equation as a solid line. Add a legend, and label the axes.

22. The voltage V_C t seconds after closing the switch in the circuit shown is given by:

$$V_C = V_0\left(1-e^{-t/RC}\right)$$

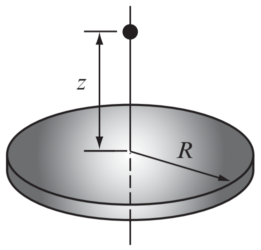
Plot V_C as a function of t for $0 \le t \le 15$ s. Label the axes. $V_0 = 36$ V, $R = 2,500\ \Omega$, and $C = 1,200\ \mu$F.



23. The force F (in N) acting between a particle with a charge q and a round disk with a radius R and a charge Q is given by the equation:

$$F = \frac{Qqz}{2\epsilon_0}\left(1-\frac{z}{\sqrt{z^2+R^2}}\right)$$

where $\epsilon_0 = 0.885 \times 10^{-12}$ C^2/(N-m^2) is the permittivity constant and z is the distance to the particle. Consider the case where $Q = 9.4 \times 10^{-6}$ C, $q = 2.4 \times 10^{-5}$ C, and $R = 0.1$ m. Make a plot of F as a function of z for $0 \le z \le 0.3$ m. Use MATLAB's built-in function `max` to find the maximum value of F and the corresponding distance z.



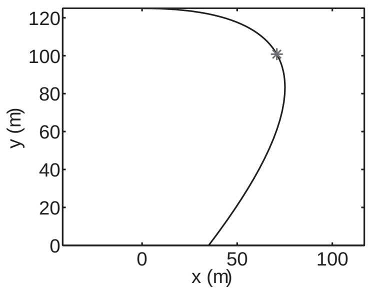
24. The curvilinear motion of a particle is defined by the following parametric equations:

$x = 52t - 9t^2$ m $\qquad y = 125 - 5t^2$ m

The velocity of the particle is given by

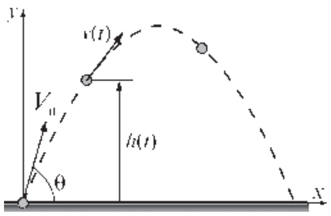
$v = \sqrt{v_x^2 + v_y^2}$, where $v_x = \frac{dx}{dt}$ and $v_y = \frac{dy}{dt}$.

For $0 \le t \le 5$ s make one plot that shows the position of the particle (y versus x) and a second plot (on the same page) of the velocity of the particle as a function of time. In addition, by using MAT-



LAB's `min` function, determine the time at which the velocity is the lowest, and the corresponding position of the particle. Using an asterisk marker, show the position of the particle in the first plot. For time use a vector with spacing of 0.1 s.

25. The height and speed of a projectile shoot at a speed v_0 at an angle θ as a function of time are given by:



$$h(t) = v_0 t \sin\theta - gt^2/2$$
$$v(t) = \sqrt{v_0^2 - 2v_0 g t \sin\theta + g^2 t^2}$$

where $g = 9.81$ m/s^2. Determine the time that the projectile will hit the ground and plot the height and the speed as a function of time (two plots on one page) for the case that $v_0 = 200$ m/s and $\theta = 70°$. Add titles and label the axes.

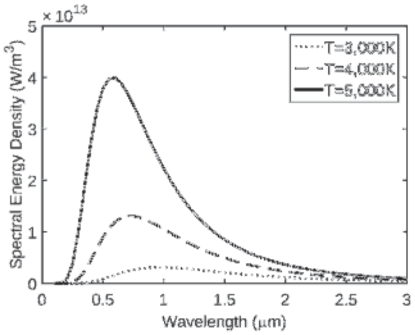
26. The position x as a function of time of a particle that moves along a straight line is given by:

$$x(t) = 8 - 4t^3 e^{-0.4t} + 2t^2 \text{ ft}$$

The velocity $v(t)$ of the particle is determined by the derivative of $x(t)$ with respect to t, and the acceleration $a(t)$ is determined by the derivative of $v(t)$ with respect to t.

Derive the expressions for the velocity and acceleration of the particle, and make plots of the position, velocity, and acceleration as functions of time for $0 \le t \le 8$ s. Use the `subplot` command to make the three plots on the same page with the plot of the position on the top, the velocity in the middle, and the acceleration at the bottom. Label the axes appropriately with the correct units.

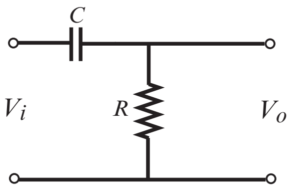
27. According to Planck's law of blackbody radiation, the spectral energy density R as a function of wavelength λ (m) and temperature T (K) is given by:



$$R = \frac{2\pi c^2 h}{\lambda^5}\frac{1}{e^{hc/\lambda kT} - 1}$$

where $c = 3\times10^8$ m/s is the speed of light, $h = 6.626\times10^{-34}$ J-s is the Planck constant, and $k = 1.38\times10^{-23}$ J/K is Boltzmann constant. Make the shown figure that contains plots of R as a function of λ for $0.1 \le \lambda \le 3$ μm for three temperatures T=3,000 K, T=4,000 K, and T=5,000 K.

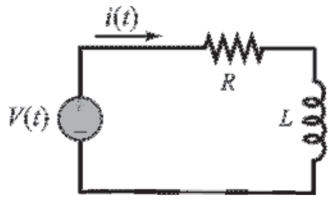
28. A high-pass filter passes signals with frequencies that are higher than a certain cutoff frequency. In this filter the ratio of the magnitudes of the voltages is given by:



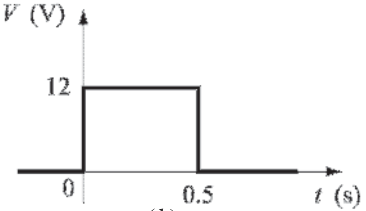
$$\left|\frac{V_o}{V_i}\right| = \frac{\omega RC}{\sqrt{1+\omega^2 R^2 C^2}}$$

where $\omega = 2\pi f$ is the frequency of the input signal. Given $R = 2,000$ and $C = 0.2$ μF, plot $\left|\frac{V_o}{V_i}\right|$ as a function of *f* for $10 \le f \le 50,000$ Hz. Use logarithmic scale for the horizontal (*f*) axis and linear scale for the vertical axis.

29. A resistor, $R = 4\ \Omega$, and an inductor, $L = 1.3$ H, are connected in a circuit to a voltage source as shown in Figure (*a*) (an *RL* circuit). When the voltage



(*a*)



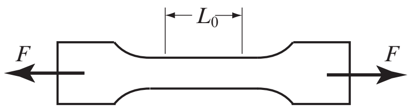
(*b*)

source applies a rectangular voltage pulse with an amplitude of $V = 12$ V and a duration of 0.5 s, as shown in Figure (*b*), the current *i*(*t*) in the circuit as a function of time is given by:

$$i(t) = \frac{V}{R}(1 - e^{-Rt/L}) \quad \text{for} \quad 0 \le t \le 0.5 \text{ s}$$
$$i(t) = e^{-Rt/L}\frac{V}{R}(e^{0.5R/L} - 1) \quad \text{for} \quad 0.5 \le t \text{ s}$$

Make a plot of the current as a function of time for $0 \le t \le 2$ s.

30. In a typical tension test a dog-bone shaped specimen is pulled in a machine. During the test, the force *F* needed to pull the specimen and the length *L* of a gauge section are measured. This data is used for plotting a stress-strain diagram of the material. Two definitions, engineering and true, exist for stress and strain. The engineering stress σ_e and strain ϵ_e are defined by $\sigma_e = \frac{F}{A_0}$ and $\epsilon_e = \frac{L-L_0}{L_0}$, where L_0 and A_0 are the initial gauge length and the initial cross-sectional area of the specimen, respectively. The true stress σ_t and strain ϵ_t are defined by $\sigma_t = \frac{F}{A_0}\frac{L}{L_0}$ and $\epsilon_t = \ln\frac{L}{L_0}$.



The following are measurements of force and gauge length from a ten-

sion test with an aluminum specimen. The specimen has a round cross section with a radius of 0.25 in. (before the test). The initial gauge length is 0.5 in. Use the data to calculate and generate the engineering and true stress-strain curves, both on the same plot. Label the axes and use a legend to identify the curves.
Units: When the force is measured in pounds (lb) and the area is calculated in in.^{2}, the unit of the stress is psi (pounds per square inch).

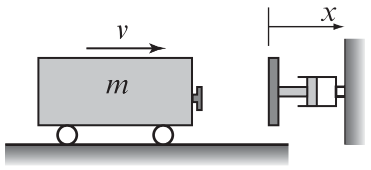
F (lb)	0	4,390	7,250	10,780	11,710	12,520	12,800	13,340
L (in.)	0.5	0.50146	0.50226	0.50344	0.50423	0.50577	0.50693	0.51138
F (lb)	13,740	13,820	13,850	13,910	13,990	14,020	14,130	
L (in.)	0.52006	0.52169	0.52362	0.52614	0.53406	0.54018	0.56466	

31. According to special relativity, a rod of length L moving at velocity v will shorten by an amount δ, given by:

$$\delta = L\left(1-\sqrt{1-\frac{v^2}{c^2}}\right)$$

where c is the speed of light (about 300×10^6 m/s). Consider a rod of 2 m long, and make three plots of δ as a function of v for $0 \le v \le 300 \times 10^6$ m/s. In the first plot use linear scale for both axes. In the second plot use logarithmic scale for v and linear scale for δ, and in the third plot use logarithmic scale for both v and δ. Which of the plots is the most informative?

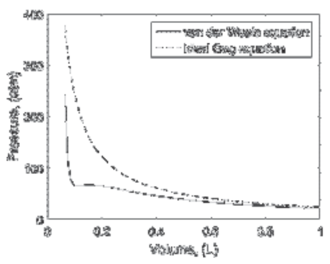
32. A railroad bumper is designed to slow down a rapidly moving railroad car. After a 20,000-kg railroad car traveling at 20 m/s engages the bumper, its displacement x (in meters) and velocity v (in m/s) as a function of time t (in seconds) is given by:



$$x(t) = 4.219(e^{-1.58t} - e^{-6.32t}) \quad \text{and} \quad v(t) = 26.67\,e^{-6.32t} - 6.67\,e^{-1.58t}$$

Plot the displacement and the velocity as a function of time for $0 \le t \le 4$ s. Make two plots on one page.

33. The ideal gas equation states that $\frac{PV}{RT} = n$, where P is the pressure, V is the volume, T is the temperature, $R = 0.08206$ (L atm)/(mol K) is the gas constant, and n is the number of moles. Real gases, especially at high pressures, deviate from this behavior. Their response can be modeled with the

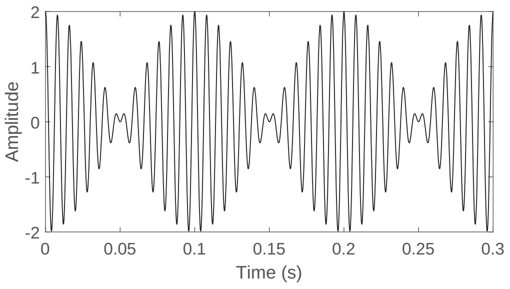


van der Waals equation $P = \frac{nRT}{V-nb} - \frac{n^2a}{V^2}$, where a and b are material constants. For CO_2 $a = 3.592$ L^2atm/mol^2, and $b = 0.04267$ L/mol. Make the shown figure that displays two plots of P versus V for $0.065 \le V \le 1$ L. In one plot the pressure is calculated by using the ideal gas equation and the other by using the van der Waals equation. Label the axes and display a legend.

34. Two sound waves of slightly different frequencies f_1 and f_2:

$$y_1 = \cos(2\pi f_1 t)$$
$$y_2 = \cos(2\pi f_2 t)$$

produce sound that is alternating loud and soft. This phenomenon, which is called beating, is described by the equation:

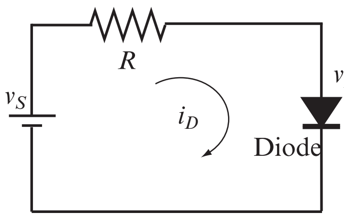


$$y = 2\cos\left(2\pi\frac{f_1+f_2}{2}t\right)\cos\left(2\pi\frac{f_1-f_2}{2}t\right)$$

Make a plot of the beating sound (shown) for $0 \le t \le 0.3$ s for the case that $f_1 = 130$ Hz and $f_2 = 120$ Hz.

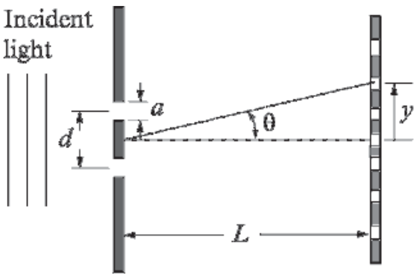
35. Consider the diode circuit shown in the figure. The current i_D and the voltage v_D can be determined from the solution of the following system of equations:

$$i_D = I_0\left(e^{\frac{qv_D}{kT}} - 1\right), \quad i_D = \frac{v_S - v_D}{R}$$



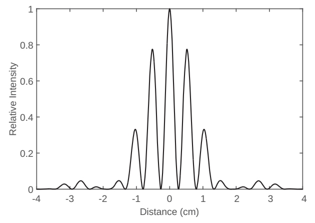
The system can be solved numerically or graphically. The graphical solution is found by plotting i_D as a function of v_D from both equations. The solution is the intersection of the two curves. Make the plots and estimate the solution for the case where $I_0 = 10^{-14}$ A, v_S = 1.5 V, R = 1200 Ω, and $\frac{kT}{q} = 30$ mV.

36. A monochromatic light that passes through a double slit produces on a screen a diffraction pattern consisting of bright and dark fringes. The intensity of the bright fringes, I, as a function of θ can be calculated by:

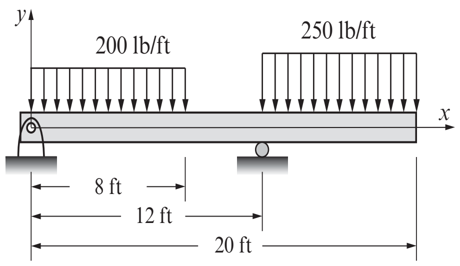


$$I = I_{\max}(\cos\beta)^2\left(\frac{\sin\alpha}{\alpha}\right)^2$$

where $\alpha = \frac{\pi a}{\lambda}\sin\theta$ and $\beta = \frac{\pi d}{\lambda}\sin\theta$, λ is the light wave length, a is the width of the slits, and d is the distance between the slits. Plot (as shown) the relative intensity $I / I_{\max}$ as a function of y (distance to fringes on the screen) for $-4 \le y \le 4$ cm given $\lambda = 480$ nm, $a = 0.025$ mm, and $d = 0.09$ mm.



37. A simply supported beam is loaded as shown. The shear force V and bending moment M as a function of x are given by the following equations:



$V(x) = 400 - 200x$ lb

$M(x) = -100x^2 + 400x$ lb-ft

for $0 \le x \le 8$ ft.

$V(x) = -1{,}200$ lb, $M(x) = -1{,}200x + 6{,}400$ lb-ft for $8 \le x \le 12$ ft

$V(x) = -250x + 5{,}000$ lb, $M(x) = -125(x-12)^2 + 2{,}000x - 32{,}000$ lb-ft for $12 \le x \le 20$ ft.

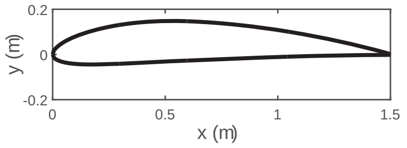
Plot the shear force and the bending moment as a function of x (two figures on one page such that the shear force diagram is displayed above the bending moment diagram).

38. Biological oxygen demand (BOD) is a measure of the relative oxygen depletion effect of a waste contaminant and is widely used to assess the amount of pollution in a water source. The BOD in the effluent (L_c in mg/L) of a rock filter without recirculation is given by:

$$L_c = \frac{L_0}{1 + \frac{(2.5D^{2/3})}{\sqrt{Q}}}$$

where L_0 is the influent BOD (mg/L), D is the depth of the filter (m), and Q is the hydraulic flow rate [L/(m^2-day)]. Assuming Q = 300 L/(m^2-day) plot the effluent BOD as a function of the depth of the filter ($100 \le D \le 2000$ m) for $L_0 = 5, 10,$ and 20 mg/L. Make the three plots in one figure and estimate the depth of filter required for each of these cases to obtain drinkable water. Label the axes and display a legend.

39. The shape of a asymmetric four-digit series NACA airfoil is described by the equations:



$x_U = x - y_t \sin\theta \quad y_U = y_c + y_t \cos\theta$
$x_L = x + y_t \sin\theta \quad y_L = y_c - y_t \cos\theta$

where the subscripts U and L corresponds to the upper and lower airfoil surface, respectively. y_t is half the thickness of the foil given by:

$$y_t = 5tc\left[0.2969\sqrt{\frac{x}{c}} - 0.126\left(\frac{x}{c}\right) - 0.3516\left(\frac{x}{c}\right)^2 + 0.2843\left(\frac{x}{c}\right)^3 - 0.1015\left(\frac{x}{c}\right)^4\right]$$

where c is the cord length, t is the maximum thickness (as a fraction of the cord length), and x is the position along the cord. y_c is the coordinate of the camber line given by:

$y_c = m\frac{x}{p^2}\left(2p - \frac{x}{c}\right)$ for $0 \le x \le pc$, and $y_c = m\frac{c-x}{(1-p)^2}\left(1 + \frac{x}{c} - 2p\right)$ for $pc \le x \le c$

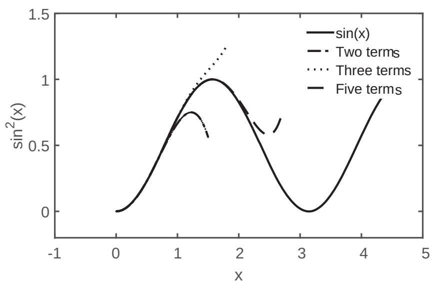
where m and p are constants. The angle θ is given by:

$\theta = \arctan\left[\frac{2m}{p^2}\left(p - \frac{x}{c}\right)\right]$ for $0 \le x \le pc$, and $\theta = \arctan\left[\frac{2m}{(1-p)^2}\left(p - \frac{x}{c}\right)\right]$ for $pc \le x \le c$. Plot the airfoil shown in the figure (NACA 4412) for which $t = 0.12$, $p = 0.4$, $m = 0.04$, and $c = 1.5$ m.

40. The Taylor series expansion for $\sin^2 x$ is:

$$x^2 - \frac{2^3x^4}{4!} + \frac{2^5x^6}{6!} - \frac{2^5x^6}{6!} + \frac{2^7x^8}{8!}$$

Plot the figure on the right, which shows, for $0 \le x \le 4.5$, the plot of the function $\sin^2 x$ and plots of the Taylor series expansion of $\sin^2 x$ with two, three, and five terms. Label the axes and display a legend.


sin(x)
Two terms
Three terms
Five terms
sin²(x)
x