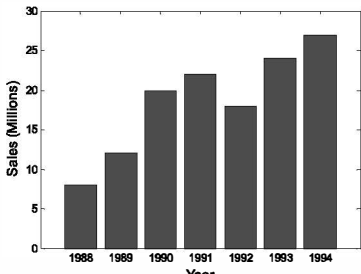
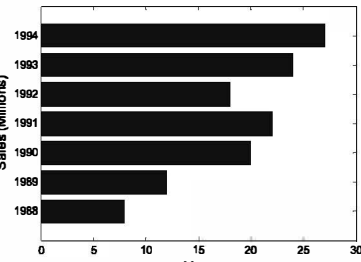


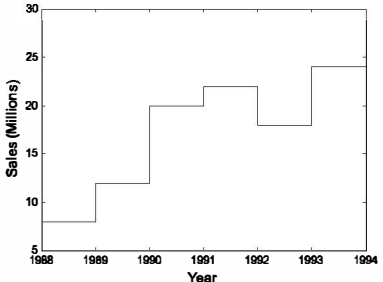
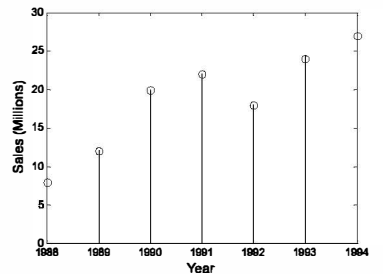
- The lengths of the four vectors x , y , d , and u must be the same.
- At each point the error bar extends from $y(i) - d(i)$ to $y(i) + u(i)$.

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.”

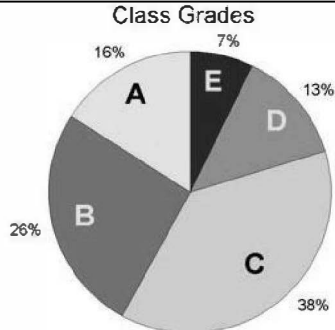
Bar (vertical and horizontal), stairs, and stem plots are presented in the following charts using the sales data from Section 5.1.1.

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

Stairs Plot Function format: <code>stairs(x,y)</code>		<pre>yr=[1988:1994]; sle=[8 12 20 22 18 24 27]; stairs(yr,sle)</pre>
Stem Plot Function Format <code>stem(x,y)</code>		<pre>yr=[1988:1994]; sle=[8 12 20 22 18 24 27]; stem(yr,sle)</pre>

Pie charts are useful for visualizing the relative sizes of different but related quantities. For example, the table below shows the grades that were assigned to a class. The data 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: <code>pie(x)</code>		<pre>grd=[11 18 26 9 5]; pie(grd) title('Class Grades')</pre> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> MATLAB draws the sections in different colors. The letters (grades) were added using the Plot Editor. </div>
--------------------------------------------------------------------	-------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

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.

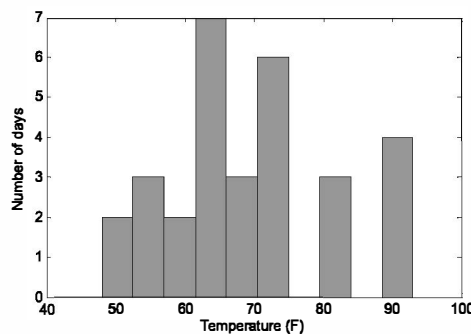


Figure 5-11: Histogram of temperature data.

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)`

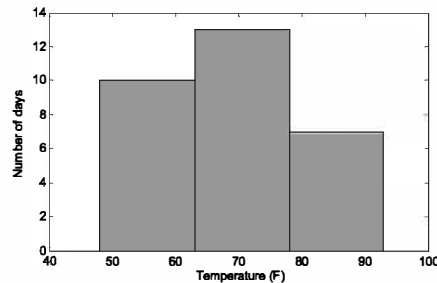
nbins is a scalar that defines the number of bins. MATLAB divides the range in equally spaced subranges.

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.

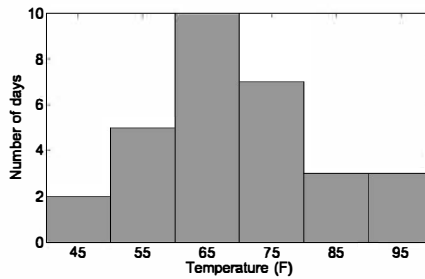
In the example above the user might prefer to divide the temperature range into three bins. This can be done with the command:

```
>> 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
```

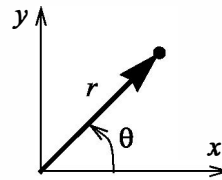
```
xout =
```

```
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

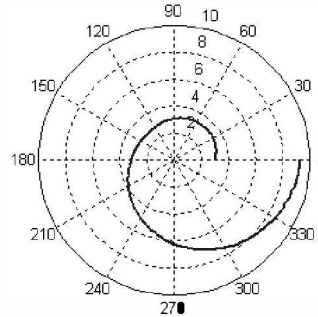
(Optional) Specifiers that define the type and color of the line and markers.

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 $0 \leq \theta \leq 2\pi$ is shown below.

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

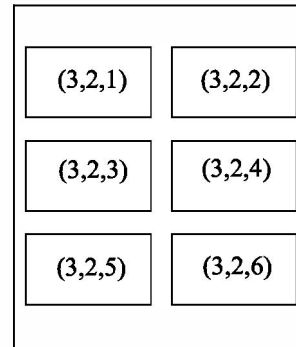


5.10 PUTTING MULTIPLE PLOTS ON THE SAME PAGE

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

```
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*cos(x)', [0,10])
>> figure
>> fplot('exp(-0.2*x)*cos(x)', [0,10])
```

Plot displayed in Figure 1 window.

Figure 2 window opens.

Plot displayed in Figure 2 window.

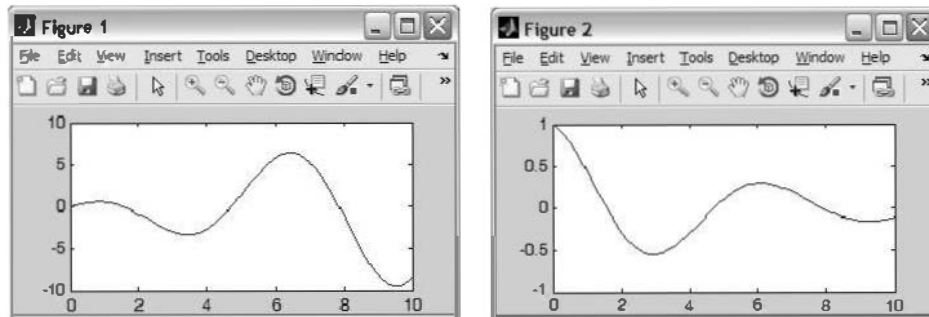


Figure 5-12: Two open Figure Windows.

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.

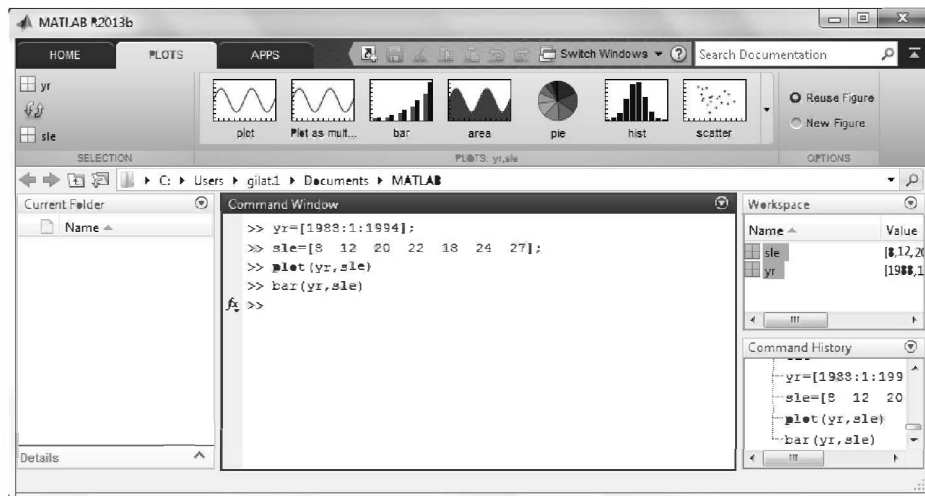


Figure 5-13: Using the PLOTS Toolstrip.

