```
b = a > 5;
a(b) = sqrt(a(b));
```

5.3.1 Creating the Equivalent of `if/else` Constructs with Logical Arrays

Logical arrays can also be used to implement the equivalent of an `if/else` construct inside a set of `for` loops. As we saw in the last section, it is possible to apply an operation to selected elements of an array using a logical array as a mask. It is also possible to apply a different set of operations to the *unselected* elements of the array by simply adding the not operator (~) to the logical mask. For example, suppose that we wanted to take the square root of any elements in a two-dimensional array whose value is greater than 5, and to square the remaining elements in the array. The code for this operation using loops and branches is

```
for ii = 1:size(a,1)
   for jj = 1:size(a,2)
      if a(ii,jj) > 5
         a(ii,jj) = sqrt(a(ii,jj));
      else
         a(ii,jj) = a(ii,jj)^2;
      end
   end
end
```

The vectorized code for this operation is

```
b = a > 5;
a(b) = sqrt(a(b));
a(~b) = a(~b).^2;
```

The vectorized code is significantly faster than the loops-and-branches version.

Quiz 5.1

This quiz provides a quick check to see if you have understood the concepts introduced in Sections 5.1 through 5.3. If you have trouble with the quiz, reread the section, ask your instructor, or discuss the material with a fellow student. The answers to this quiz are found in the back of the book.

Examine the following `for` loops and determine how many times each loop will be executed.

1. `for index = 7:10`
2. `for jj = 7:-1:10`
3. `for index = 1:10:10`
4. `for ii = -10:3:-7`
5. `for kk = [0 5 ; 3 3]`

Examine the following loops and determine the value in `ires` at the end of each of the loops.

6.
```
ires = 0;
for index = 1:10
   ires = ires + 1;
end
```

7.
```
ires = 0;
for index = 1:10
   ires = ires + index;
end
```

8.
```
ires = 0;
for index1 = 1:10
   for index2 = index1:10
      if index2 == 6
         break;
      end
      ires = ires + 1;
   end
end
```

9.
```
ires = 0;
for index1 = 1:10
   for index2 = index1:10
      if index2 == 6
         continue;
      end
      ires = ires + 1;
   end
end
```

10. Write the MATLAB statements to calculate the values of the function

$$f(t) = \begin{cases} \sin t & \text{for all } t \text{ where } \sin t > 0 \\ 0 & \text{elsewhere} \end{cases}$$

for $-6\pi \le t \le 6\pi$ at intervals of $\pi/10$. Do this twice, once using loops and branches and once using vectorized code.

5.4 The MATLAB Profiler

MATLAB includes a profiler, which can be used to identify the parts of a program that consume the most execution time. The profiler can identify "hot spots," where optimizing the code will result in major increases in speed.

Commands and Functions (*Continued*)

`textread`	Reads text data from a file into one or more input variables.
`toc`	Returns elapsed time since last call to `tic`.
`while` loop	Loops over a block of statements until a test condition becomes 0 (false).

5.8 Exercises

5.1 Write the MATLAB statements required to calculate $y(t)$ from the equation

$$y(t) = \begin{cases} -3t^2 + 5 & t \geq 0 \\ 3t^2 + 5 & t < 0 \end{cases}$$

for values of t between -9 and 9 in steps of 0.5. Use loops and branches to perform this calculation.

5.2 Rewrite the statements required to solve Exercise 5.1 using vectorization.

5.3 Write the MATLAB statements required to calculate and print out the squares of all the even integers between 0 and 50. Create a table consisting of each integer and its square, with appropriate labels over each column.

5.4 Write an M-file to evaluate the equation $y(x) = x^2 - 3x + 2$ for all values of x between –1 and 3, in steps of 0.1. Do this twice, once with a `for` loop and once with vectors. Plot the resulting function using a 3-point-thick dashed red line.

5.5 Write an M-file to calculate the factorial function N!, as defined in Example 5.2. Be sure to handle the special case of 0! Also, be sure to report an error if N is negative or not an integer.

5.6 Examine the following `for` statements and determine how many times each loop will be executed.

(a)
```
for ii = -32768:32767
```
(b)
```
for ii = 32768:32767
```
(c)
```
for kk = 2:4:3
```
(d)
```
for jj = ones(5,5)
```

5.7 Examine the following `for` loops and determine the value of `ires` at the end of each of the loops and also the number of times each loop executes.

(a)
```
ires = 0;
for index = -10:10
  ires = ires + 1;
end
```

(b)
```
ires = 0;
for index = 10:-2:4
  if index == 6
      continue;
   end
  ires = ires + index;
end
```

(c)
```
ires = 0;
for index = 10:-2:4
  if index == 6
    break;
  end
  ires = ires + index;
end
```

(d)
```
ires = 0;
for index1 = 10:-2:4
  for index2 = 2:2:index1
    if index2 == 6
      break
    end
    ires = ires + index2;
  end
end
```

5.8 Examine the following `while` loops and determine the value of `ires` at the end of each of the loops, and the number of times each loop executes.

(a)
```
ires = 1;
while mod(ires,10) ~= 0
  ires = ires + 1;
end
```

(b)
```
ires = 2;
while ires <= 200
  ires = ires^2;
end
```

(c)
```
ires = 2;
while ires > 200
  ires = ires^2;
end
```

5.9 What is contained in array `arr1` after each of the following sets of statements is executed?

(a)
```
arr1 = [1 2 3 4; 5 6 7 8; 9 10 11 12];
mask = mod(arr1,2) == 0;
arr1(mask) = -arr1(mask);
```

(b)
```
arr1 = [1 2 3 4; 5 6 7 8; 9 10 11 12];
arr2 = arr1 <= 5;
arr1(arr2) = 0;
arr1(~arr2) = arr1(~arr2).^2;
```

5.10 How can a logical array be made to behave as a logical mask for vector operations?

5.11 Modify program `ball` from Example 5.7 by replacing the inner `for` loops with vectorized calculations.

5.12 Modify program `ball` from Example 5.7 to read in the acceleration due to gravity at a particular location, and to calculate the maximum range of the ball for that acceleration. After modifying the program, run it with accelerations of –9.8 m/s^2, –9.7 m/s^2, and –9.6 m/s^2. What effect does the reduction in gravitational attraction have on the range of the ball? What effect does the reduction in gravitational attraction have on the best angle θ at which to throw the ball?

5.13 Modify program `ball` from Example 5.7 to read in the initial velocity with which the ball is thrown. After modifying the program, run it with initial velocities of 10 m/s, 20 m/s, and 30 m/s. What effect does changing the initial velocity v_0 have on the range of the ball? What effect does it have on the best angle θ at which to throw the ball?

5.14 Program `lsqfit` from Example 5.6 required the user to specify the number of input data points before entering the values. Modify the program so that it reads an arbitrary number of data values using a `while` loop and stops reading input values when the user presses the Enter key without typing any values. Test your program using the same two data sets that were used in Example 5.6. (*Hint:* The `input` function returns an empty array (`[]`) if a user presses Enter without supplying any data. You can use function `isempty` to test for an empty array, and stop reading data when one is detected.)

5.15 Modify program `lsqfit` from Example 5.6 to read its input values from an ASCII file named `input1.dat`. The data in the file will be organized in rows, with one pair of (x, y) values on each row, as shown below:

```
1.1   2.2
2.2   3.3
...
```

Use the `load` function to read the input data. Test your program using the same two data sets that were used in Example 5.6.

5.16 Modify program `lsqfit` from Example 5.6 to read its input values from a user-specified ASCII file named `input1.dat`. The data in the file will be organized in rows, with one pair of (x, y) values on each row, as shown below:

```
1.1   2.2
2.2   3.3
...
```

Use the `textread` function to read the input data. Test your program using the same two data sets that were used in Example 5.6.

5.17 **Factorial Function** MATLAB includes a standard function called `factorial` to calculate the factorial function. Use the MATLAB help system to look up this function, and then calculate 5!, 10!, and 15! using both the program in Example 5.2 and the `factorial` function. How do the results compare?

5.18 **Running Average Filter** Another way of smoothing a noisy data set is with a *running average filter*. For each data sample in a running average filter, the program examines a subset of n samples centered on the sample under test, and it replaces that sample with the average value from the n samples. (*Note:* For points near the beginning and the end of the data set, use a smaller number of samples

in the running average, but be sure to keep an equal number of samples on either side of the sample under test.)

Write a program that allows the user to specify the name of an input data set and the number of samples to average in the filter and then performs a running average filter on the data. The program should plot both the original data and the smoothed curve after the running average filter.

Test your program using the data in the file `input3.dat`, which is available from the book's website.

5.19 Median Filter Another way of smoothing a noisy data set is with a *median filter*. For each data sample in a median filter, the program examines a subset of n samples centered on the sample under test, and it replaces that sample with the median value from the n samples. (*Note:* For points near the beginning and the end of the data set, use a smaller number of samples in the median calculation, but be sure to keep an equal number of samples on either side of the sample under test.) This type of filter is very effective against data sets containing isolated "wild" points that are very far away from the other nearby points.

Write a program that allows the user to specify the name of an input data set and the number of samples to use in the filter and then performs a median filter on the data. The program should plot both the original data and the smoothed curve after the median filter.

Test your program using the data in the file `input3.dat`, which is available from the book's website. Is the median filter better or worse than the running average filter for smoothing this data set? Why?

5.20 Fourier Series A Fourier series is an infinite series representation of a periodic function in terms of sines and cosines at a fundamental frequency (matching the period of the waveform) and multiples of that frequency. For example, consider a square wave function of period L, whose amplitude is 1 for [0 L/2), −1 for [L/2 L), 1 for [L 3L/2), and so forth. This function is plotted in Figure 5.7. This function can be represented by the Fourier series

$$f(x) = \sum_{n=1,3,5,\ldots}^{\infty} \frac{1}{n} \sin\left(\frac{n\pi x}{L}\right) \tag{5.13}$$

Plot the original function assuming $L = 1$, and calculate and plot Fourier series approximations to that function containing 3, 5, and 10 terms.

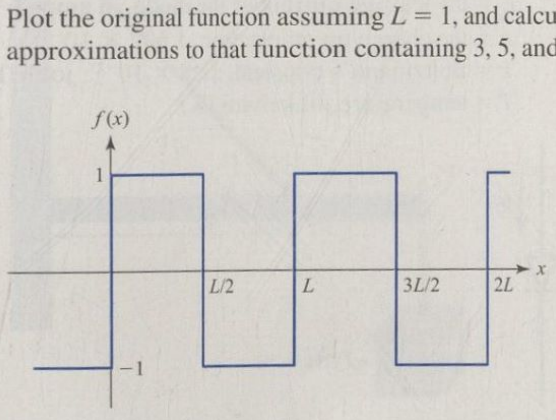

Figure 5.7 A square wave waveform.

5.21 Program `doy` in Example 5.3 calculates the day of year associated with any given month, day, and year. As written, this program does not check to see if the data entered by the user is valid. It will accept nonsense values for months and days and do calculations with them to produce meaningless results. Modify the program so that it checks the input values for validity before using them. If the inputs are invalid, the program should tell the user what is wrong and quit. The year should be a number greater than zero, the month should be a number between 1 and 12, and the day should be a number between 1 and a maximum that depends on the month. Use a `switch` construct to implement the bounds checking performed on the day.

5.22 Write a MATLAB program to evaluate the function

$$y(x) = \ln \frac{1}{1-x} \tag{5.14}$$

for any user-specified value of x, where ln is the natural logarithm (logarithm to the base e). Write the program with a `while` loop, so that the program repeats the calculation for each legal value of x entered into the program. When an illegal value of x is entered, terminate the program. (Any $x \geq 1$ is considered an illegal value.)

5.23 Fibonacci Numbers The nth Fibonacci number is defined by the following recursive equations:

$$f(1) = 1$$
$$f(2) = 2$$
$$f(n) = f(n-1) + f(n-2) \quad n > 2$$

Therefore, $f(3) = f(2) + f(1) = 2 + 1 = 3$, and so forth for higher numbers. Write an M-file to calculate and write out the nth Fibonacci number for $n > 2$, where n is input by the user. Use a `while` loop to perform the calculation.

5.24 Current through a Diode The current flowing through the semiconductor diode shown in Figure 5.8 is given by the equation

$$i_D = I_0\left(e^{\frac{qv_D}{kT}} - 1\right) \tag{5.15}$$

where i_D = the voltage across the diode, in volts
v_D = the current flow through the diode, in amps
I_0 = the leakage current of the diode, in amps
q = the charge on an electron, 1.602×10^{-19} coulombs
k = Boltzmann's constant, 1.38×10^{-23} joule/K
T = temperature, in kelvins (K)

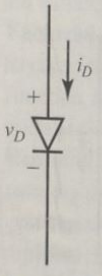

Figure 5.8 A semiconductor diode.

The leakage current I_0 of the diode is 2.0 μA. Write a program to calculate the current flowing through this diode for all voltages from -1.0 V to $+0.6$ V, in 0.1 V steps. Repeat this process for the following temperatures: 75 °F, 100 °F, and 125 °F. Create a plot of the current as a function of applied voltage, with the curves for the three different temperatures appearing as different colors.

5.25 Tension on a Cable A 100-kg object is to be hung from the end of a rigid 2-meter horizontal pole of negligible weight, as shown in Figure 5.9. The pole is attached to a wall by a pivot and is supported by a 2-meter cable that is attached to the wall at a higher point. The tension on this cable is given by the equation

$$T = \frac{W \cdot lc \cdot lp}{d\sqrt{lp^2 - d^2}} \tag{5.16}$$

where T is the tension on the cable, W is the weight of the object, lc is the length of the cable, lp is the length of the pole, and d is the distance along the pole at which the cable is attached. Write a program to determine the distance d at which to attach the cable to the pole in order to minimize the tension on the cable. To do this, the program should calculate the tension on the cable at regular 0.1 m intervals from $d = 0.3$ m to $d = 1.8$ m, and should locate the position d that produces the minimum tension. Also, the program should plot the tension on the cable as a function of d, with appropriate titles and axis labels.

5.26 Modify the program created in Exercise 5.25 to determine how sensitive the tension on the cable is to the precise location d at which the cable is attached. Specifically, determine the range of d values that will keep the tension on the cable within 10% of its minimum value.

5.27 Area of a Parallelogram The area of a parallelogram with two adjacent sides defined by vectors **A** and **B** can be found from Equation (5.17) (see Figure 5.10).

$$area = |\mathbf{A} \times \mathbf{B}| \tag{5.17}$$

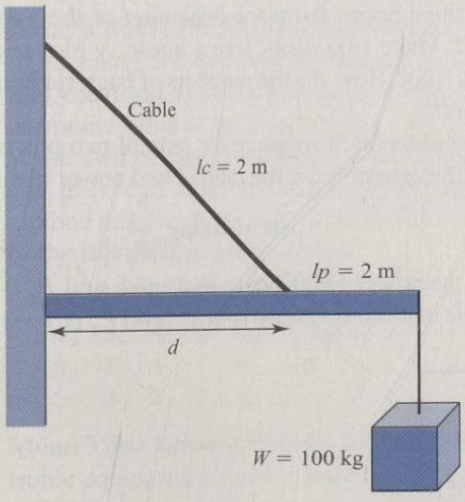

Figure 5.9 A 100 kg weight suspended from a rigid bar supported by a cable.

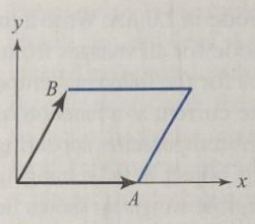

Figure 5.10 A parallelogram.

Write a program to read vectors **A** and **B** from the user, and calculate the resulting area of the parallelogram. Test your program by calculating the area of a parallelogram bordered by vectors $\mathbf{A} = 10\,\hat{\mathbf{i}}$ and $\mathbf{B} = 5\,\hat{\mathbf{i}} + 8.66\,\hat{\mathbf{j}}$.

5.28 Area of a Rectangle The area of the rectangle in Figure 5.11 is given by Equation (5.18) and the perimeter of the rectangle is given by Equation (5.19).

$$area = W \times H \tag{5.18}$$

$$perimeter = 2W + 2H \tag{5.19}$$

Assume that the total perimeter of a rectangle is limited to 10, and write a program that calculates and plots the area of the rectangle as its width is varied from the smallest possible value to the largest possible value. At what width is the area of the rectangle maximized?

5.29 Bacterial Growth Suppose that a biologist performs an experiment in which he or she measures the rate at which a specific type of bacterium reproduces asexually in different culture media. The experiment shows that in medium A the bacteria reproduce once every 60 minutes, and in medium B the bacteria reproduce once every 90 minutes. Assume that a single bacterium is placed on each culture medium at the beginning of the experiment. Write a program that calculates and plots the number of bacteria present in each culture at intervals of three hours from the beginning of the experiment until 24 hours have elapsed. Make two plots, one a linear *xy* plot and the other a linear-log (`semilogy`) plot. How do the numbers of bacteria compare on the two media after 24 hours?

5.30 Decibels Engineers often measure the ratio of two power measurements in *decibels*, or dB. The equation for the ratio of two power measurements in decibels is

$$\mathrm{dB} = 10 \log_{10} \frac{P_2}{P_1} \tag{5.20}$$

where P_2 is the power level being measured, and P_1 is some reference power level. Assume that the reference power level P_1 is 1 watt, and write a program

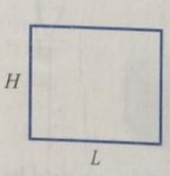

Figure 5.11 A rectangle.

that calculates the decibel level corresponding to power levels between 1 and 20 watts, in 0.5 W steps. Plot the dB-versus-power curve on a log-linear scale.

5.31 Geometric Mean The *geometric mean* of a set of positive numbers x_1 through x_n is defined as the nth root of the product of the numbers:

$$\text{geometric mean} = \sqrt[n]{x_1 x_2 x_3 \cdots x_n} \tag{5.21}$$

Write a MATLAB program that will accept an arbitrary number of positive input values and calculate both the arithmetic mean (*i.e.*, the average) and the geometric mean of the numbers. Use a `while` loop to get the input values, and terminate the inputs when a user enters a negative number. Test your program by calculating the average and geometric mean of the four numbers 10, 5, 2, and 5.

5.32 RMS Average The *root-mean-square* (rms) *average* is another way of calculating a mean for a set of numbers. The rms average of a series of numbers is the square root of the arithmetic mean of the squares of the numbers:

$$\text{rms average} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2} \tag{5.22}$$

Write a MATLAB program that will accept an arbitrary number of positive input values and calculate the rms average of the numbers. Prompt the user for the number of values to be entered, and use a `for` loop to read in the numbers. Test your program by calculating the rms average of the four numbers 10, 5, 2, and 5.

5.33 Harmonic Mean The *harmonic mean* is yet another way of calculating a mean for a set of numbers. The harmonic mean of a set of numbers is given by the equation:

$$\text{harmonic mean} = \frac{N}{\frac{1}{x_1} + \frac{1}{x_2} + \cdots + \frac{1}{x_n}} \tag{5.23}$$

Write a MATLAB program that will read in an arbitrary number of positive input values and calculate the harmonic mean of the numbers. Use any method that you desire to read in the input values. Test your program by calculating the harmonic mean of the four numbers 10, 5, 2, and 5.

5.34 Write a single program that calculates the arithmetic mean (average), rms average, geometric mean, and harmonic mean for a set of positive numbers. Use any method that you desire to read in the input values. Compare these values for each of the following sets of numbers:

(a) `4, 4, 4, 4, 4, 4, 4`
(b) `4, 3, 4, 5, 4, 3, 5`
(c) `4, 1, 4, 7, 4, 1, 7`
(d) `1, 2, 3, 4, 5, 6, 7`

5.35 Mean Time Between Failure Calculations The reliability of a piece of electronic equipment is usually measured in terms of Mean Time Between Failures (MTBF), where MTBF is the average time that the piece of equipment can operate before a failure occurs in it. For large systems containing many pieces