```
    public static void main(String[] args) {
        SimulationControl.createApp(new MonteCarloEstimatorApp());
    }
}
```

**Problem 11.7 Monte Carlo integration in one dimension**

(a) Use `MonteCarloEstimatorApp` to estimate $F_n$, the integral of $f(x) = 4\sqrt{1-x^2}$ in the interval $0 \le x \le 1$, using the hit or miss method. Choose $a = 0$, $b = 1$, and $h = 1$ and compute the mean value of the function $\sqrt{1-x^2}$. Multiply the estimate by 4 to determine $F_n$. Calculate the difference between $F_n$ and the exact result of $\pi$. This difference is a measure of the error associated with the Monte Carlo estimate. Make a log-log plot of the error as a function of $n$. What is the approximate dependence of the error on $n$ for large $n$, for example, $n \ge 10^6$?

(b) Estimate the same integral using the sample mean method (11.15) and compute the error as a function of the number of samples $n$ for $n \ge 10^6$. How many samples are needed to determine $F_n$ to two decimal places? What is the approximate dependence of the error on $n$ for large $n$?

(c) Determine the computational time per sample using the two Monte Carlo methods. Which Monte Carlo method is preferable? ■

## 11.3 ■ MULTIDIMENSIONAL INTEGRALS

Many problems in physics involve averaging over many variables. For example, suppose we know the position and velocity dependence of the total energy of ten interacting particles. In three dimensions each particle has three velocity components and three position components. Hence the total energy is a function of 60 variables, and a calculation of the average energy per particle involves computing a $d = 60$ dimensional integral. (More accurately, the total energy is a function of $60 - 6 = 54$ variables if we use center of mass and relative coordinates.) If we divide each coordinate into $p$ intervals, there would be $p^{60}$ points to sum. Clearly, standard numerical methods such as Simpson's rule would be impractical for this example.

A discussion of the $n$-dependence of the error associated with the standard numerical methods for $d$-dimensional integrals is given in Appendix 11A. We show that if the error decreases as $n^{-a}$ for $d = 1$, then the error decreases as $n^{-a/d}$ in $d$ dimensions. In contrast, we find (see Section 11.4) that the error for all Monte Carlo integration methods decreases as $n^{-1/2}$ *independently* of the dimension of the integral. Because the computational time is roughly proportional to $n$ in both the classical and Monte Carlo methods, we conclude that for low dimensions, the classical numerical methods such as Simpson's rule are preferable to Monte Carlo methods unless the domain of integration is very complicated. However, Monte Carlo methods are essential for higher-dimensional integrals.

To illustrate the evaluation of multidimensional integrals, we consider the two-dimensional integral



$$F = \int_R f(x, y)\, dx\, dy, \tag{11.16}$$

where $R$ denotes the region of integration. The extension to higher dimensions is straightforward but tedious. Form a rectangle that encloses the region $R$ and divide this rectangle into squares of length $h$. Assume that the rectangle runs from $x_a$ to $x_b$ in the $x$ direction and from $y_a$ to $y_b$ in the $y$ direction. The total number of squares is $n_x n_y$, where $n_x = (x_b - x_a)/h$ and $n_y = (y_b - y_a)/h$. If we use the midpoint approximation, the integral $F$ is estimated by

$$F \approx \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} f(x_i, y_j) H(x_i, y_j)\, h^2, \tag{11.17}$$

where $x_i = x_a + (i - \frac{1}{2})h$, $y_j = y_a + (j - \frac{1}{2})h$, and the (Heaviside) function $H(x, y) = 1$ if $(x, y)$ is in $R$ and equals 0 otherwise.

A simple Monte Carlo method for evaluating a two-dimensional integral uses the same rectangular region as in (11.17), but the $n$ points $(x_i, y_i)$ are chosen at random within the rectangle. The estimate for the integral is then

$$F_n = \frac{A}{n} \sum_{i=1}^{n} f(x_i, y_i) H(x_i, y_i), \tag{11.18}$$

where $A$ is the area of the rectangle. Note that if $f(x, y) = 1$ everywhere, then (11.18) is equivalent to the hit or miss method of calculating the area of the region $R$. In general, (11.18) represents the area of the region $R$ multiplied by the average value of $f(x, y)$ in $R$.

**Problem 11.8 Two-dimensional numerical integration**

(a) Write a program to implement the midpoint approximation in two dimensions and integrate the function $f(x, y) = x^2 + 6xy + y^2$ over the region defined by the condition $x^2 + y^2 \le 1$. Use $h = 0.1$, 0.05, 0.025, and 0.0125.

(b) Repeat part (a) using a Monte Carlo method and the same number of points $n$. For each value of $n$, repeat the calculation several times to obtain a crude estimate of the random error. ■

**Problem 11.9 Volume of a hypersphere**

(a) The interior of a $d$-dimensional hypersphere of unit radius is defined by the condition $x_1^2 + x_2^2 + \cdots + x_d^2 \le 1$. Write a program that finds the volume of a hypersphere using the midpoint approximation. If you are clever, you can write a program that applies to any dimension using a recursive method. Test your program for $d = 2$ and $d = 3$ and then find the volume for $d = 4$ and $d = 5$. Begin with $h = 0.2$ and decrease $h$ until your results do not change by more than 1%.

(b) Repeat part (a) using a Monte Carlo method. For each value of $n$, repeat the calculation several times to obtain a rough estimate of the random error. Is your program applicable for any $d$ easier to write than in part (a)? ■