$p(x, y)\,dx\,dy$ given by

$$p(x, y)\,dx\,dy = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}\,dx\,dy. \tag{11.38}$$

First, we make a change of variables to polar coordinates:

$$r = (x^2 + y^2)^{1/2}, \qquad \theta = \tan^{-1}\frac{y}{x}. \tag{11.39}$$

We let $\rho = r^2/2$ and write the two-dimensional probability as

$$p(\rho, \theta)\,d\rho\,d\theta = \frac{1}{2\pi} e^{-\rho} d\rho\,d\theta, \tag{11.40}$$

where we have set $\sigma = 1$. If we generate ρ according to the exponential distribution (11.34) and generate θ uniformly in the interval $0 \le \theta < 2\pi$, then the quantities

$$x = (2\rho)^{1/2}\cos\theta \quad \text{and} \quad y = (2\rho)^{1/2}\sin\theta \quad \text{(Box–Muller method)}, \tag{11.41}$$

will each be generated according to (11.37) with zero mean and $\sigma = 1$. (Note that the two-dimensional density (11.38) is the product of two independent one-dimensional Gaussian distributions.) This way of generating a Gaussian distribution is known as the *Box–Muller* method. We discuss other methods for generating the Gaussian distribution in Problems 11.14 and 11.17 and in Appendix 11C.

Problem 11.13 Nonuniform probability densities

(a) Write a program to simulate the simultaneous rolling of two dice. In this case the events are discrete and occur with nonuniform probability.

(b) Write a program to verify that the sequence of random numbers $\{x_i\}$ generated by (11.36) is distributed according to the exponential distribution (11.34).

(c) Generate random variables according to the probability density function

$$p(x) = \begin{cases} 2(1-x) & 0 \le x \le 1 \\ 0 & \text{otherwise.} \end{cases} \tag{11.42}$$

(d) Verify that the variables x and y in (11.41) are distributed according to the Gaussian distribution. What are the mean value and the standard deviation of x and y?

(e) How can you use the relations (11.41) to generate a Gaussian distribution with arbitrary mean and standard deviation? ■

Problem 11.14 Generating normal distributions

Fernández and Criado have suggested another method of generating normal distributions that is much faster than the Box–Muller method. We will just summarize the algorithm; the proof that the algorithm leads to a normal distribution is given in their paper.

(i) Begin with N numbers v_i in an array. Set all the $v_i = \sigma$, where σ is the desired standard deviation for the normal distribution.

(ii) Update the array by randomly choosing two different entries v_i and v_j from the array. Then let $v_i = (v_i + v_j)/\sqrt{2}$ and use the new v_i to set $v_j = -v_i + v_j\sqrt{2}$.

(iii) Repeat step (ii) many times to bring the array of numbers to "equilibrium."

(iv) After equilibration, the entries v_i will have a normal distribution with the desired standard deviation and zero mean.

Write a program to produce a series of random numbers according to this algorithm. Your program should allow the user to enter N and σ, and a button should be implemented to allow for equilibration before various averages are computed. The desired output is the probability distribution of the random numbers that are produced as well as their mean and standard deviation. First make sure that the standard deviation of the probability distribution approaches the desired input σ for sufficiently long times. What is the order of magnitude of the equilibration time? Does it depend on N? Plot the natural log of the probability distribution versus v^2 and check that you obtain a straight line with the appropriate slope. ■

11.6 ■ IMPORTANCE SAMPLING

In Section 11.4 we found that the error associated with a Monte Carlo estimate is proportional to the standard deviation σ of the integrand and inversely proportional to the square root of the number of samples. Hence, there are only two ways of reducing the error in a Monte Carlo estimate—either increase the number of samples or reduce the variance. Clearly the latter choice is desirable because it does not require much more computer time. In this section we introduce *importance sampling* techniques that reduce σ and improve the efficiency of each sample.

To do importance sampling in the context of numerical integration, we introduce a positive function $p(x)$ such that

$$\int_a^b p(x)\,dx = 1, \tag{11.43}$$

and rewrite the integral (11.1) as

$$F = \int_a^b \left[\frac{f(x)}{p(x)}\right] p(x)\,dx. \tag{11.44}$$

We can evaluate the integral (11.44) by sampling according to the probability distribution $p(x)$ and constructing the sum

$$F_n \doteq \frac{1}{n}\sum_{i=1}^{n} \frac{f(x_i)}{p(x_i)}. \tag{11.45}$$

The sum (11.45) reduces to (11.15) for the uniform case $p(x) = 1/(b-a)$.

The idea is to choose a form for $p(x)$ that minimizes the variance of the ratio $f(x)/p(x)$. To do so we choose a form of $p(x)$ that mimics $f(x)$ as much as possible, particularly where