could be the slope found from a least squares calculation. If we do this resampling n_r times, a measure of the error in the quantity G is given by

$$\sigma_G = \Big[\sum_{k=1}^{n_r}(G_k - \langle G_k\rangle)^2/(n_r-1)\Big]^{1/2}. \tag{11.26}$$

To see how this procure works, consider $n = 15$ pairs of points x_i randomly distributed between 0 and 1, and corresponding values of y given by $y_i = 2x_i + 3 + r_i$, where r_i is a uniform random number between -1 and $+1$. Compute the best slope, m_{ls}, and intercept, b_{ls} using the least squares method and their corresponding errors using (7.40). Now resample the data 200 times, computing a slope and intercept each time using the least squares method. From your results estimate the errors for the slope and intercept using (11.26). How well do the estimates from bootstrapping compare with the direct error estimates? Does the average of the bootstrap values for the slope and intercept equal m_{ls} and b_{ls}, respectively. If not why not? Do your conclusions change if you resample 1000 times?

11.6 Nonuniform Probability Distributions

In Sections 11.3 and 11.5 we learned how uniformly distributed random numbers can be used to estimate definite integrals. As we might expect, we will find that it is more efficient to sample the integrand $f(x)$ more often in regions of x where the magnitude of $f(x)$ is large or rapidly varying. Because such *importance sampling* methods require nonuniform probability distributions, we first consider several methods for generating random numbers that are not distributed uniformly before we consider importance sampling methods in Section 11.8. In the following, we will denote r as a member of a uniform random number sequence in the unit interval $0 \le r < 1$.

Suppose that two discrete events occur with probabilities p_1 and p_2 such that $p_1 + p_2 = 1$. How can we choose the two events with the correct probabilities using a uniform probability distribution? For this simple case, it is obvious that we choose event 1 if $r < p_1$; otherwise, we choose event 2. If there are three events with probabilities p_1, p_2, and p_3, then if $r < p_1$ we choose event 1; else if $r < p_1 + p_2$, we choose event 2; else we choose event 3. We can visualize these choices by dividing a line segment of unit length into three pieces whose lengths are as shown in Figure 11.4. A random point r on the line segment will land in the ith segment with a probability equal to p_i.

Now consider n discrete events. How do we determine which event, i, to choose given the value of r? The generalization of the procedure we have followed for $n = 2$ and 3 is to find the value of i that satisfies the condition

$$\sum_{j=0}^{i-1} p_j \le r \le \sum_{j=0}^{i} p_j, \tag{11.27}$$

where we have defined $p_0 \equiv 0$. Check that (11.27) reduces to the correct procedure for $n = 2$ and $n = 3$.

Now let us consider a continuous nonuniform probability distribution. One way to generate such a distribution is to take the limit of (11.27) and associate p_i with $p(x)\,dx$, where the *probability*

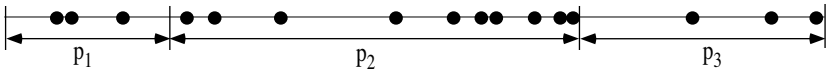

Figure 11.4: The unit interval is divided into three segments of lengths $p_1 = 0.2$, $p_2 = 0.5$, and $p_3 = 0.3$. Sixteen random numbers are represented by the filled circles uniformly distributed on the unit interval. The fraction of circles within each segment is approximately equal to the value of p_i for that segment.

density $p(x)$ is defined such that $p(x)\,dx$ is the probability that the event x is in the interval between x and $x + dx$. The probability density $p(x)$ is normalized such that

$$\int_{-\infty}^{+\infty} p(x)\,dx = 1. \tag{11.28}$$

In the continuum limit the two sums in (11.27) become the same integral and the inequalities become equalities. Hence we can write

$$P(x) \equiv \int_{-\infty}^{x} p(x')\,dx' = r. \tag{11.29}$$

From (11.29) we see that the uniform random number r corresponds to the *cumulative probability distribution* function $P(x)$, which is the probability of choosing a value less than or equal to x. The function $P(x)$ should not be confused with the probability density $p(x)$ or the probability $p(x)\,dx$. In many applications the meaningful range of values of x is positive. In that case, we have $p(x) = 0$ for $x < 0$.

The relation (11.29) leads to the *inverse transform* method for generating random numbers distributed according to the function $p(x)$. This method involves generating a random number r and solving (11.29) for the corresponding value of x. As an example of the method, we use (11.29) to generate a random number sequence according to the uniform probability distribution on the interval $a \le x \le b$. The desired probability density $p(x)$ is

$$p(x) = \begin{cases} (1/(b-a), & a \le x \le b \\ 0, & \text{otherwise.} \end{cases} \tag{11.30}$$

The cumulative probability distribution function $P(x)$ for $a \le x \le b$ can be found by substituting (11.30) into (11.29) and performing the integral. The result is

$$P(x) = \frac{x-a}{b-a}. \tag{11.31}$$

If we substitute the form (11.31) for $P(x)$ into (11.29) and solve for x, we find the desired relation

$$x = a + (b-a)r. \tag{11.32}$$

The variable x given by (11.32) is distributed according to the probability distribution $p(x)$ given by (11.30). Of course, the relation (11.32) is rather obvious, and we already have used (11.32) in our Monte Carlo programs.

We next apply the inverse transform method to the probability density function

$$p(x) = \begin{cases} (1/\lambda)\, e^{-x/\lambda}, & \text{if } 0 \le x \le \infty \\ 0, & x < 0. \end{cases} \tag{11.33}$$

In Section 11.7 we will use this probability density to find the distance between scattering events of a particle whose mean free path is λ. If we substitute (11.33) into (11.29) and integrate, we find the relation

$$r = P(x) = 1 - e^{-x/\lambda}. \tag{11.34}$$

The solution of (11.34) for x yields $x = -\lambda \ln(1-r)$. Because $1-r$ is distributed in the same way as r, we can write

$$x = -\lambda \ln r. \tag{11.35}$$

The variable x found from (11.35) is distributed according to the probability density $p(x)$ given by (11.33). On many computers the computation of the natural logarithm in (11.35) is relatively slow, and hence the inverse transform method might not necessarily be the most efficient method to use.

From the above examples, we see that two conditions must be satisfied in order to apply the inverse transform method. Specifically, the form of $p(x)$ must allow us to perform the integral in (11.29) analytically, and it must be practical to invert the relation $P(x) = r$ for x.

The Gaussian probability density,

$$p(x) = \frac{1}{(2\pi\sigma^2)^{1/2}} e^{-x^2/2\sigma^2}, \tag{11.36}$$

is an example of a probability density for which the cumulative distribution $P(x)$ cannot be obtained analytically. However, we can generate the two-dimensional probability $p(x,y)\,dx\,dy$ given by

$$p(x,y)\,dx\,dy = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}\,dx\,dy. \tag{11.37}$$

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

$$r = (x^2+y^2)^{1/2}, \qquad \theta = \tan^{-1}\frac{y}{x}. \tag{11.38}$$

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

$$p(\rho,\theta)\,d\rho d\theta = \frac{1}{2\pi} e^{-\rho} d\rho\, d\theta, \tag{11.39}$$

where we have set $\sigma = 1$. If we generate ρ according to the exponential distribution (11.33) and generate θ uniformly in the interval $0 \le \theta < 2\pi$, then the variables

$$x = (2\rho)^{1/2}\cos\theta \quad \text{and} \quad y = (2\rho)^{1/2}\sin\theta \qquad \text{(Box-Muller method)} \tag{11.40}$$

will each be generated according to (11.36) with zero mean and $\sigma = 1$. (Note that the two-dimensional density (11.37) 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 Problem 11.12 and 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. You might wish to revisit Problem 7.23 and simulate the game of craps.

b. Write a program to verify that the sequence of random numbers $\{x_i\}$ generated by (11.35) is distributed according to the exponential distribution (11.33).

c. Generate random variables according to the probability density function

$$p(x) = \begin{cases} 2(1-x), & \text{if } 0 \le x \le 1; \\ 0, & \text{otherwise.} \end{cases} \tag{11.41}$$

d. Verify that the variables x and y in (11.40) are distributed according to the Gaussian distribution. What is the mean value and the standard deviation of x and of y?

e. How can you use the relations (11.40) 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 which is much faster than the Box-Muller method. We will just discuss the algorithm; the proof that the algorithm leads to a normal distribution is given in their paper. The algorithm is as follows:

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 random numbers according to this algorithm. Your program should allow the user to enter N and σ and a button should be available to allow for equilibration. The output should be 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 equilibration 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.

Hence, the variance σ^2 of the nm individual trials is

$$\sigma^2 = \frac{1}{mn}\sum_{\alpha=1}^{m}\sum_{i=1}^{n} {d_{\alpha,i}}^2. \tag{11.76}$$

We write

$$e_\alpha = M_\alpha - \overline{M} = \tfrac{1}{n}\textstyle\sum_{i=1}^{n}\left(x_{\alpha,i} - \overline{M}\right) \tag{11.77}$$

$$= \tfrac{1}{n}\textstyle\sum_{i=1}^{n} d_{\alpha,i}. \tag{11.78}$$

If we substitute (11.78) into (11.74), we find

$${\sigma_m}^2 = \frac{1}{m}\sum_{\alpha=1}^{n}\left(\frac{1}{n}\sum_{i=1}^{n} d_{\alpha,i}\right)\left(\frac{1}{n}\sum_{j=1}^{n} d_{\alpha,j}\right). \tag{11.79}$$

The sum in (11.79) over trials i and j in set α contains two kinds of terms—those with $i = j$ and those with $i \neq j$. We expect that $d_{\alpha,i}$ and $d_{\alpha,j}$ are independent and equally positive or negative on the average. Hence in the limit of a large number of measurements, we expect that only the terms with $i = j$ in (11.79) will survive, and we write

$${\sigma_m}^2 = \frac{1}{mn^2}\sum_{\alpha=1}^{n}\sum_{i=1}^{n} {d_{\alpha,i}}^2. \tag{11.80}$$

If we combine (11.80) with (11.76), we arrive at the desired result

$${\sigma_m}^2 = \frac{\sigma^2}{n}. \tag{11.81}$$

Appendix 11C: The Acceptance-Rejection Method

Although the inverse transform method discussed in Section 11.6 can in principle be used to generate any desired probability distribution, in practice the method is limited to functions for which the equation, $r = P(x)$, can be solved analytically for x or by simple numerical approximation. Another method for generating nonuniform probability distributions is the *acceptance-rejection* method due to von Neumann. Suppose that $p(x)$ is a (normalized) probability density function that we wish to generate. For simplicity, we assume $p(x)$ is nonzero in the unit interval. Consider a positive definite *comparison function* $w(x)$ such that $w(x) > p(x)$ in the entire range of interest. A simple although not generally optimum choice of w is a constant greater than the maximum value of $p(x)$. Because the area under the curve $p(x)$ in the range x to $x + \Delta x$ is the probability of generating x in that range, we can follow a procedure similar to that used in the hit or miss method. Generate two numbers at random to define the location of a point in two dimensions which is distributed uniformly in the area under the comparison function $w(x)$. If this point is outside the area under $p(x)$, the point is rejected; if it lies inside the area, we accept it. This procedure implies that the accepted points are uniform in the area under the curve $p(x)$ and that their x values are distributed according to $p(x)$. One procedure for generating a uniform random point (x, y) under the comparison function $w(x)$ is as follows.

1. Choose a form of $w(x)$. One choice would be to choose $w(x)$ such that the values of x distributed according to $w(x)$ can be generated by the inverse transform method. Let the total area under the curve $w(x)$ be equal to A.

2. Generate a uniform random number in the interval $[0, A]$ and use it to obtain a corresponding value of x distributed according to $w(x)$.

3. For the value of x generated in step (2), generate a uniform random number y in the interval $[0, w(x)]$. The point (x, y) is uniformly distributed in the area under the comparison function $w(x)$. If $y \leq p(x)$, then accept x as a random number distributed according to $p(x)$.

Repeat steps (2) and (3) many times. Note that the acceptance-rejection method is efficient only if the comparison function $w(x)$ is close to $p(x)$ over the entire range of interest.

Appendix 11D: Polynomials

Interpolation is a technique that allows us to estimate a function within the range of a tabulated set of *sample points*.[1] For example, Fourier analysis (see Chapter 9) generates a trigonometric series that can be evaluated between the points that are used to calculate the coefficients. We now describe how polynomials are implemented and used including interpolation between sample points.

A polynomial is a function that is expressed as

$$p(x) = \sum_{i=0}^{n} a_i x^i, \tag{11.82}$$

where n is the *degree* of the polynomial and the n constants a_i are the *coefficients*. The evaluation of (11.82) as written is very inefficient because x is repeatedly multiplied by itself and the entire sum requires $\mathcal{O}(N^2)$ multiplications. A more efficient algorithm was published in 1819 by W. G. Horner.[2] It uses a factored polynomial and requires only n multiplications and n additions and is now known as *Horner's rule*. It is written as follows:

$$p(x) = a_0 + x\Big[a_1 + x\big[a_2 + x[a_3 + \cdots]\big]\Big]. \tag{11.83}$$

Using the correct algorithm for this simple task can dramatically reduce processor time if large polynomials are repeatedly evaluated.

Polynomials are important computationally because most analytic functions can be approximated as a polynomial using a Taylor series expansion. Polynomials can be added, multiplied, integrated, and differentiated analytically and the result is still a polynomial. This property makes them ideally suited for object-oriented programming. The `Polynomial` class in the numerics package implements many of these algebraic operations (see Table 11.6). Listing 11.3 shows how this class is used to calculate and display a polynomial's roots.

[1] Extrapolation estimates the function outside the range covered by the sample points.
[2] This method of evaluating polynomials by factoring was already known to Newton.