CHAPTER 8

Generating Random Variates

Recommended sections for a first reading: 8.1 and 8.2

8.1 INTRODUCTION

A simulation that has any random aspects at all must involve sampling, or *generating*, random variates from probability distributions. As in Chap. 7, we use the phrase "generating a random variate" to refer to the activity of obtaining an observation on (or a realization of) a random variable from the desired distribution. These distributions are often specified as a result of fitting some appropriate distributional form, e.g., exponential, gamma, or Poisson, to observed data, as discussed in Chap. 6. In this chapter we assume that a distribution has already been specified somehow (including the values of the parameters), and we address the issue of how we can generate random variates with this distribution in order to run the simulation model. For example, the queueing-type models discussed in Sec. 1.4 and Chap. 2 required generation of interarrival and service times to drive the simulation through time, and the inventory model of Sec. 1.5 needed randomly generated demand sizes at the times when a demand occurred.

As we shall see in this chapter, the basic ingredient needed for *every* method of generating random variates from *any* distribution or random process is a source of IID U(0, 1) random variates. For this reason it is essential that a statistically reliable U(0, 1) random-number generator be available. Most computer installations and simulation packages have a convenient random-number generator but some of them (especially the older ones) do not perform adequately (see Chap. 7). Without an acceptable random-number generator, it is impossible to generate random variates correctly from any distribution. In the rest of this chapter, we therefore assume that a good source of random numbers is available.

There are usually several alternative algorithms that can be used for generating random variates from a given distribution, and several factors should be considered when choosing which algorithm to use in a particular simulation study. Unfortunately, these different factors often conflict with each other, so the analyst's judgment of which algorithm to use must involve a number of tradeoffs. All we can do here is raise some of the pertinent questions.

The first issue is *exactness*. We feel that, if possible, one should use an algorithm that results in random variates with exactly the desired distribution, within the unavoidable external limitations of machine accuracy and exactness of the U(0, 1) random-number generator. Efficient and exact algorithms are now available for all of the commonly used distributions, obviating the need to consider any older, approximate methods. [Many of these approximations, e.g., the well-known technique of obtaining a "normal" random variate as 6 less than the sum of 12 U(0, 1) random variates, are based on the central limit theorem.] On the other hand, the practitioner may argue that a specified distribution is really only an approximation to reality anyway, so that an approximate generation method should suffice; since this depends on the situation and is often difficult to quantify, we still prefer to use an exact method.

Given that we have a choice, then, of alternative exact algorithms, we would clearly like to use one that is *efficient*, in terms of both storage space and execution time. Some algorithms require *storage* of a large number of constants or of large tables, which could prove troublesome or at least inconvenient. As for execution time, there are really two factors. Obviously, we hope that we can accomplish the generation of each random variate in a small amount of time; this is called the *marginal execution time*. Second, some algorithms have to do some initial computing to specify constants or tables that depend on the particular distribution and parameters; the time required to do this is called the *setup time*. In most simulations, we shall be generating a large number of random variates from a given distribution, so that marginal execution time is likely to be more important than setup time. If the parameters of a distribution change often or randomly during the course of the simulation, however, setup time could become an important consideration.

A somewhat subjective issue in choosing an algorithm is its overall *complexity*, including conceptual as well as implementational factors. One must ask whether the potential gain in efficiency that might be experienced by using a more complicated algorithm is worth the extra effort to understand and implement it. This issue should be considered relative to the purpose in implementing a method for random-variate generation; a more efficient but more complex algorithm might be appropriate for use as permanent software but not for a "one-time" simulation model.

Finally, there are a few issues of a more technical nature. Some algorithms rely on a source of random variates from distributions other than U(0, 1), which is undesirable, other things being equal. Another technical issue is that a given algorithm may be efficient for some parameter values but costly for others. We would like to have algorithms that are efficient for all parameter values (sometimes called *robustness* of the algorithm); see Devroye (1988). One last technical point is relevant if we want to use certain kinds of variance-reduction techniques in order to obtain better (less variable) estimates (see Chap. 11 and also Chaps. 10 and 12). Two commonly used variance-reduction techniques (common random numbers and antithetic variates) require synchronization of the basic U(0, 1) input random variates used in the simulation of the system(s) under study, and this synchronization is more easily accomplished for certain types of random-variate generation algorithms. In particular, the general inverse-transform approach can be very helpful in facilitating the desired synchronization and variance reduction; Sec. 8.2.1 treats this point more precisely.

There are a number of comprehensive references on random-variate generation, including Dagpunar (1988), Devroye (1986), Fishman (1996), Gentle (2003), Hörmann et al. (2004), and Johnson (1987). There are also several computer packages that provide good capabilities for generating random variates from a wide variety of distributions, such as the IMSL routines [Visual Numerics, Inc. (2004)] and the C codes in secs. 7.2 and 7.3 of Press et al. (1992).

The remainder of this chapter is organized as follows. In Sec. 8.2 we survey the most important general approaches for random-variate generation, including examples and general discussions of the relative merits of the various approaches. In Secs. 8.3 and 8.4 we present algorithms for generating random variates from particular continuous and discrete distributions that have been found useful in simulation. Finally, in Secs. 8.5 and 8.6 we discuss two more specialized topics: generating correlated random variates and generating realizations of both stationary and nonstationary arrival processes.

8.2 GENERAL APPROACHES TO GENERATING RANDOM VARIATES

There are many techniques for generating random variates, and the particular algorithm used must, of course, depend on the distribution from which we wish to generate; however, nearly all these techniques can be classified according to their theoretical basis. In this section we discuss these general approaches.

8.2.1 Inverse Transform

Suppose that we wish to generate a random variate X that is continuous (see Sec. 4.2) and has distribution function F that is continuous and strictly increasing when $0 < F(x) < 1$. [This means that if $x_1 < x_2$ and $0 < F(x_1) \le F(x_2) < 1$, then in fact $F(x_1) < F(x_2)$.] Let F^{-1} denote the inverse of the function F. Then an algorithm for generating a random variate X having distribution function F is as follows (recall that $\sim$ is read "is distributed as"):

1. Generate $U \sim U(0, 1)$.
2. Return $X = F^{-1}(U)$.

Note that $F^{-1}(U)$ will always be defined, since $0 \le U \le 1$ and the range of F is [0, 1]. Figure 8.1 illustrates the algorithm graphically, where the random variable

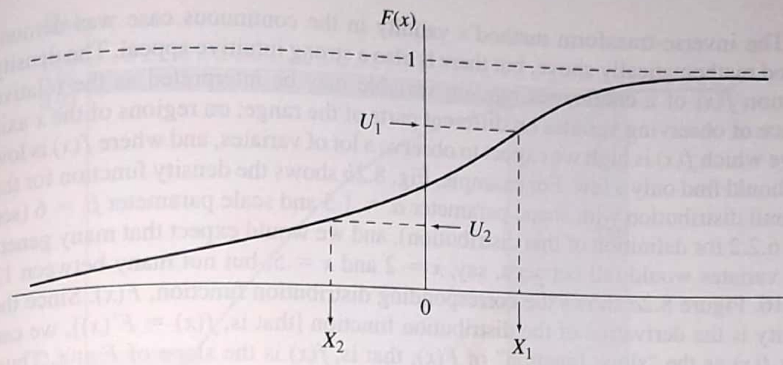

FIGURE 8.1
Inverse-transform method for continuous random variables.

corresponding to this distribution function can take on either positive or negative values; the particular value of U determines which will be the case. In the figure, the random number U_1 results in the positive random variate X_1, while the random number U_2 leads to the negative variate X_2.

To show that the value X returned by the above algorithm, called the general *inverse-transform method*, has the desired distribution F, we must show that for any real number x, $P(X \le x) = F(x)$. Since F is invertible, we have

$$P(X \le x) = P(F^{-1}(U) \le x) = P(U \le F(x)) = F(x)$$

where the last equality follows since $U \sim U(0, 1)$ and $0 \le F(x) \le 1$. (See the discussion of the uniform distribution in Sec. 6.2.2.)

EXAMPLE 8.1. Let X have the exponential distribution with mean β (see Sec. 6.2.2). The distribution function is

$$F(x) = \begin{cases} 1 - e^{-x/\beta} & \text{if } x \ge 0 \\ 0 & \text{otherwise} \end{cases}$$

so to find F^{-1}, we set $u = F(x)$ and solve for x to obtain

$$F^{-1}(u) = -\beta \ln (1 - u)$$

Thus, to generate the desired random variate, we first generate a $U \sim U(0, 1)$ and then let $X = -\beta \ln U$. [It is possible in this case to use U instead of $1 - U$, since $1 - U$ and U have the same U(0, 1) distribution. This saves a subtraction.] $U = 1-U$

In the above example, we replaced $1 - U$ by U for the sake of a perhaps minor gain in efficiency. However, replacing $1 - U$ by U in situations like this results in negative correlation of the X's with the U's, rather than positive correlation. Also, it is not true that wherever a "$1 - U$" appears in a variate-generation algorithm it can be replaced by a "U," as illustrated in Sec. 8.3.15.

The inverse-transform method's validity in the continuous case was demonstrated mathematically above, but there is also a strong intuitive appeal. The density function $f(x)$ of a continuous random variable may be interpreted as the relative chance of observing variates on different parts of the range; on regions of the x axis above which $f(x)$ is high we expect to observe a lot of variates, and where $f(x)$ is low we should find only a few. For example, Fig. 8.2*b* shows the density function for the Weibull distribution with shape parameter $\alpha = 1.5$ and scale parameter $\beta = 6$ (see Sec. 6.2.2 for definition of this distribution), and we would expect that many generated variates would fall between, say, $x = 2$ and $x = 5$, but not many between 13 and 16. Figure 8.2*a* shows the corresponding distribution function, $F(x)$. Since the density is the derivative of the distribution function [that is, $f(x) = F'(x)$], we can view $f(x)$ as the "slope function" of $F(x)$; that is, $f(x)$ is the slope of F at x. Thus, F rises most steeply for values of x where $f(x)$ is large (e.g., for x between 2 and 5), and, conversely, F is relatively flat in regions where $f(x)$ is small (e.g., for x between 13 and 16). Now, the inverse-transform method says to take the random number U, which should be evenly (uniformly) spread on the interval [0, 1] on the vertical axis of the plot for $F(x)$, and "read across and down." More U's will hit the steep parts of $F(x)$ than the flat parts, thus concentrating the X's under those regions where $F(x)$ is steep—which are precisely those regions where $f(x)$ is high. The interval [0.25, 0.30] on the vertical axis in Fig. 8.2*a* should contain about 5 percent of the U's, which lead to X's in the relatively narrow region [2.6, 3.0] on the x axis; thus, about 5 percent of the X's will be in this region. On the other hand, the interval [0.93, 0.98] on the vertical axis, which is the same size as [0.25, 0.30] and thus contains about 5 percent of the U's as well, leads to X's in the large interval [11.5, 14.9] on the x axis; here we will also find about 5 percent of the X's, but spread out sparsely over a much larger interval.

Figure 8.3 shows the algorithm in action, with 50 X's being generated (using the random-number generator from App. 7A with stream 1). The U's are plotted on the vertical axis of Fig. 8.3*a*, and the X's corresponding to them are obtained by following the dashed lines across and down. Note that the U's on the vertical axis are fairly evenly spread, but the X's on the horizontal axis are indeed more dense where the density function $f(x)$ is high, and become more spread out where $f(x)$ is low. Thus, the inverse-transform method essentially deforms the uniform distribution of the U's to result in a distribution of the X's in accordance with the desired density.

The inverse-transform method can also be used when X is discrete. Here the distribution function is

$$F(x) = P(X \le x) = \sum_{x_i \le x} p(x_i)$$

where $p(x_i)$ is the probability mass function

$$p(x_i) = P(X = x_i)$$

(We assume that X can take on only the values $x_1, x_2, \ldots$ where $x_1 < x_2 < \cdots$.) Then the algorithm is as follows:

1. Generate $U \sim \text{U}(0, 1)$.
2. Determine the smallest positive integer I such that $U \le F(x_I)$, and return $X = x_I$.

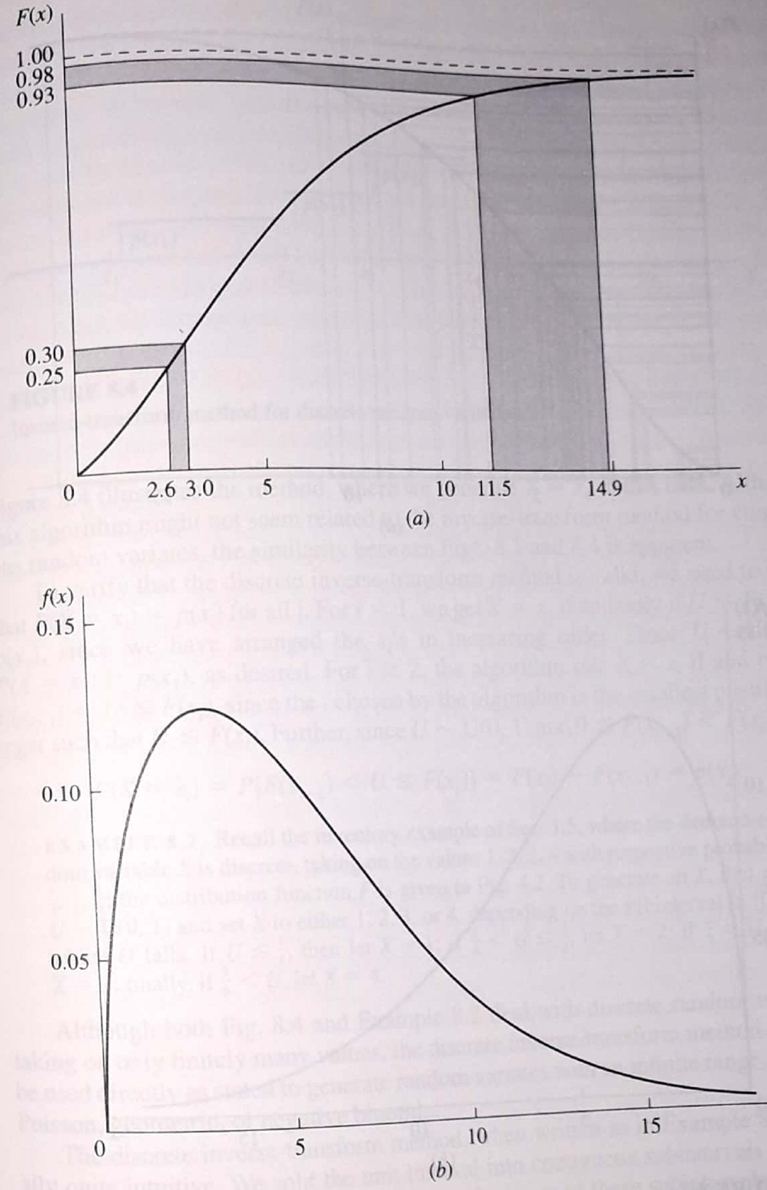

FIGURE 8.2
(*a*) Intervals for U and X, inverse transform for Weibull(1.5, 6) distribution; (*b*) density for Weibull(1.5, 6) distribution.

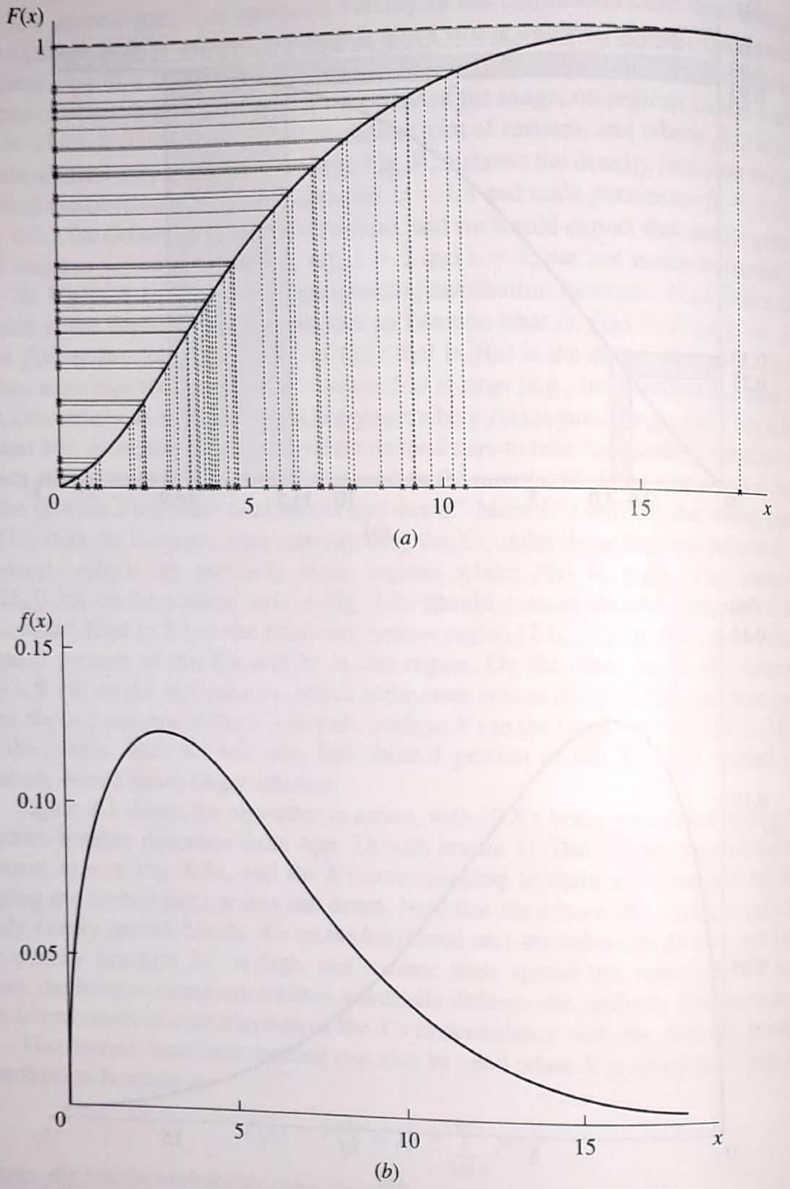

FIGURE 8.3
(*a*) Sample of 50 *U*'s and *X*'s, inverse transform for Weibull(1.5, 6) distribution; (*b*) density for Weibull(1.5, 6) distribution.

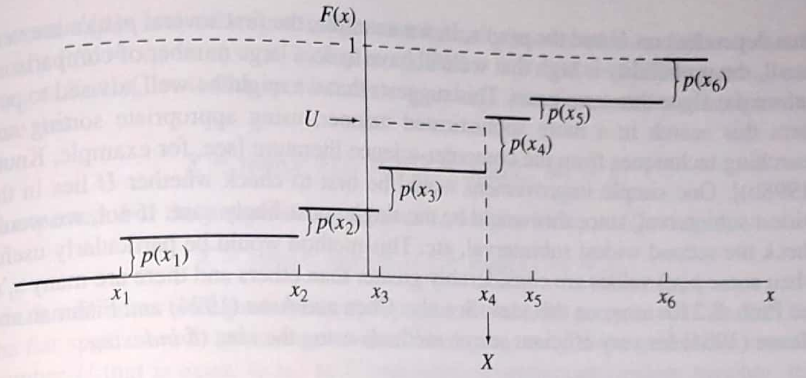

FIGURE 8.4
Inverse-transform method for discrete random variables.

Figure 8.4 illustrates the method, where we generate $X = x_4$ in this case. Although this algorithm might not seem related to the inverse-transform method for continuous random variates, the similarity between Figs. 8.1 and 8.4 is apparent.

To verify that the discrete inverse-transform method is valid, we need to show that $P(X = x_i) = p(x_i)$ for all i. For $i = 1$, we get $X = x_1$ if and only if $U \le F(x_1) = p(x_1)$, since we have arranged the x_i's in increasing order. Since $U \sim \mathrm{U}(0, 1)$, $P(X = x_1) = p(x_1)$, as desired. For $i \ge 2$, the algorithm sets $X = x_i$ if and only if $F(x_{i-1}) < U \le F(x_i)$, since the i chosen by the algorithm is the smallest positive integer such that $U \le F(x_i)$. Further, since $U \sim \mathrm{U}(0, 1)$ and $0 \le F(x_{i-1}) < F(x_i) \le 1$,

$$P(X = x_i) = P[F(x_{i-1}) < U \le F(x_i)] = F(x_i) - F(x_{i-1}) = p(x_i)$$

EXAMPLE 8.2. Recall the inventory example of Sec. 1.5, where the demand-size random variable X is discrete, taking on the values 1, 2, 3, 4 with respective probabilities $\frac{1}{6}$, $\frac{1}{3}$, $\frac{1}{3}$, $\frac{1}{6}$; the distribution function F is given in Fig. 4.2. To generate an X, first generate $U \sim \mathrm{U}(0, 1)$ and set X to either 1, 2, 3, or 4, depending on the subinterval in [0, 1] into which U falls. If $U \le \frac{1}{6}$, then let $X = 1$; if $\frac{1}{6} < U \le \frac{1}{2}$, let $X = 2$; if $\frac{1}{2} < U \le \frac{5}{6}$, let $X = 3$; finally, if $\frac{5}{6} < U$, let $X = 4$.

Although both Fig. 8.4 and Example 8.2 deal with discrete random variables taking on only finitely many values, the discrete inverse-transform method can also be used directly as stated to generate random variates with an infinite range, e.g., the Poisson, geometric, or negative binomial.

The discrete inverse-transform method, when written as in Example 8.2, is really quite intuitive. We split the unit interval into contiguous subintervals of width $p(x_1), p(x_2), \ldots$ and assign X according to whichever of these subintervals contains the generated U. For example, U will fall in the second subinterval with probability $p(x_2)$, in which case we let $X = x_2$. The efficiency of the algorithm will depend on how we look for the subinterval that contains a given U. The simplest approach would be to start at the left and move up; first check whether $U \le p(x_1)$, in which case we return $X = x_1$. If $U > p(x_1)$, check whether $U \le p(x_1) + p(x_2)$, in which case we return $X = x_2$, etc. The number of comparisons needed to determine a value for X is thus dependent on U and the $p(x_i)$'s. If, for example, the first several $p(x_i)$'s are very small, the probability is high that we will have to do a large number of comparisons before the algorithm terminates. This suggests that we might be well advised to perform this search in a more sophisticated manner, using appropriate sorting and searching techniques from the computer-science literature [see, for example, Knuth (1998b)]. One simple improvement would be first to check whether U lies in the widest subinterval, since this would be the single most-likely case. If not, we would check the second widest subinterval, etc. This method would be particularly useful when some $p(x_i)$ values are considerably greater than others and there are many x_i's; see Prob. 8.2 for more on this idea. See also Chen and Asau (1974) and Fishman and Moore (1984) for very efficient search methods using the idea of *indexing*.

Generalization, Advantages, and Disadvantages of the Inverse-Transform Method

Both the continuous and discrete versions of the inverse-transform method can be combined, at least formally, into the more general form

$$X = \min\{x\colon F(x) \geq U\}$$

which has the added advantage of being valid for distributions that are *mixed,* i.e., have both continuous and discrete components, as well as for continuous distribution functions with flat spots. To check that the above is valid in the continuous case, note in Fig. 8.1 that the set $\{x\colon F(x) \geq U_1\}$ is the interval $[X_1, \infty)$, which has minimum X_1. In the discrete case, we see in Fig. 8.4 that $\{x\colon F(x) \geq U\} = [x_4, \infty)$, which has minimum x_4. Figure 8.5 shows a mixed distribution with two jump

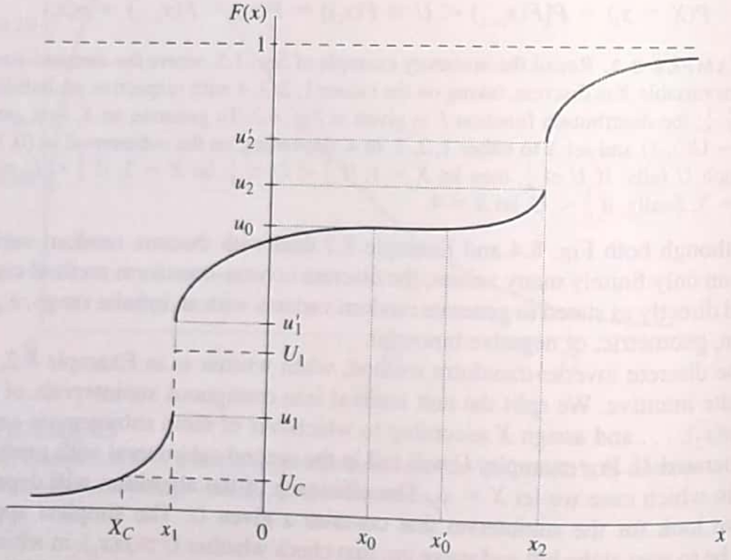

FIGURE 8.5
Inverse-transform method for a mixed distribution.

discontinuities and a flat spot; in this case the associated random variable X should satisfy $P(X = x_1) = u_1' - u_1$ (jump at x_1), $P(X = x_2) = u_2' - u_2$ (jump at x_2), and $P(x_0 \leq X \leq x_0') = 0$ (flat spot between x_0 and x_0'). For the continuous component, note that

$$X = \min\{x\colon F(x) \geq U_C\} = \min[X_C, \infty) = X_C$$

as expected. For the jump discontinuity at x_1, we get, for $u_1 \leq U_1 \leq u_1'$,

$$X = \min\{x\colon F(x) \geq U_1\} = \min[x_1, \infty) = x_1$$

which will occur with probability $u_1' - u_1$, as desired; the jump at x_2 is similar. For the flat spot, we will generate a variate X in (x_0, x_0') only if we generate a random number U that is *equal* to u_0; as U represents a continuous random variable, this occurs with probability 0, although in practice the finite accuracy of the generated random number U could result in $U = u_0$. Thus, this more general statement of the inverse-transform method handles any continuous, discrete, or mixed distribution. How it is actually implemented, though, will of course depend heavily on the distribution desired.

Let us now consider some general advantages and disadvantages of the inverse-transform method in both the continuous and discrete cases. One possible impediment to use of this method in the continuous case is the need to evaluate $F^{-1}(U)$. Since we might not be able to write a formula for F^{-1} in closed form for the desired distribution (e.g., the normal and gamma distributions), simple use of the method, as in Example 8.1, might not be possible. However, even if F^{-1} does not have a simple closed-form expression, we might be able to use numerical methods, e.g., a power-series expansion, to evaluate F^{-1}. (See, e.g., the discussion in Sec. 8.3 concerning the generation of gamma, normal, and beta random variates.) These numerical methods can yield arbitrary accuracy, so in particular can match the accuracy inherent in machine roundoff error; in this sense, they are exact for all practical purposes. However, Devroye (1986, pp. 31–35) points out that it may be difficult to specify an acceptable stopping rule for some distributions, especially those whose range is infinite. Kennedy and Gentle (1980, chap. 5) provide a comprehensive survey of numerical methods for computing distribution functions and their inverses; see also Abramowitz and Stegun (1964, chap. 26) and Press et al. (1992, chap. 6). The IMSL library [Visual Numerics, Inc. (2004)] includes routines to compute most of the common distribution functions and their inverses, using carefully chosen algorithms. As an alternative to approximating F^{-1} numerically, Marsaglia (1984) proposes that a function g be found that is "close" to F^{-1} and is easy to evaluate; then X is generated as $g(Y)$, where Y has a particular distribution that is "close" to U(0, 1). Marsaglia called this method the *exact-approximation method* [see also Fishman (1996, pp. 185–187)].

More recently, Hörmann and Leydold (2003) proposed a general adaptive method for constructing a highly accurate Hermite interpolation for F^{-1} in the case of continuous distributions. Based on a one-time setup, their method produces moderate-sized tables for the interpolation points and coefficients. Generating random variates using these tables is then very fast. The code for their method is available in a public-domain library called UNURAN.

A second potential disadvantage is that for a given distribution the inverse-transform method may not be the fastest way to generate the corresponding random variate; in Secs. 8.3 and 8.4 we discuss the efficiency of alternative algorithms for each distribution considered.

Despite these possible drawbacks, there are some important advantages in using the inverse-transform method. The first is to facilitate variance-reduction techniques (see Chap. 11) that rely on inducing correlation between random variates; examples of such techniques are common random numbers and antithetic variates. If F_1 and F_2 are two distribution functions, then $X_1 = F_1^{-1}(U_1)$ and $X_2 = F_2^{-1}(U_2)$ will be random variates with respective distribution functions F_1 and F_2, where U_1 and U_2 are random numbers. If U_1 and U_2 are independent, then of course X_1 and X_2 will be independent as well. However, if we let $U_2 = U_1$, then the correlation between X_1 and X_2 is made as positive as possible, and taking $U_2 = 1 - U_1$ (which, recall, is also distributed uniformly over [0, 1]) makes the correlation between X_1 and X_2 as negative as possible. Thus, the inverse-transform method induces the strongest correlation (of either sign) between the generated random variates, which we hope will propagate through the simulation model to induce the strongest possible correlation in the output, thereby contributing to the success of the variance-reduction technique. [It is possible, however, to induce correlation in random variates generated by methods other than the inverse-transform method; see Schmeiser and Kachitvichyanukul (1990)]. On a more pragmatic level, inverse transform eases application of variance-reduction techniques since we always need exactly one random number to produce one value of the desired X. (Other methods to be discussed later may require several random numbers to obtain a single value of X, or the number of random numbers might itself be random, as in the acceptance-rejection method.) This observation is important since proper implementation for many variance-reduction techniques requires some sort of synchronization of the input random numbers between different simulation runs. If the inverse-transform technique is used, synchronization is easier to achieve.

The second advantage concerns ease of generating from truncated distributions (see Sec. 6.8). In the continuous case, suppose that we have a density f with corresponding distribution function F. For $a < b$ (with the possibility that $a = -\infty$ or $b = +\infty$), we define the *truncated density*

$$f^*(x) = \begin{cases} \dfrac{f(x)}{F(b) - F(a)} & \text{if } a \le x \le b \\ 0 & \text{otherwise} \end{cases}$$

which has corresponding *truncated distribution function*

$$F^*(x) = \begin{cases} 0 & \text{if } x < a \\ \dfrac{F(x) - F(a)}{F(b) - F(a)} & \text{if } a \le x \le b \\ 1 & \text{if } b < x \end{cases}$$

(The discrete case is analogous.) Then an algorithm for generating an X having distribution function F^* is as follows:

1. Generate $U \sim U(0, 1)$.
2. Let $V = F(a) + [F(b) - F(a)]U$.
3. Return $X = F^{-1}(V)$.

We leave it as an exercise (Prob. 8.3) to show that the X defined by this algorithm indeed has distribution function F^*. Note that the inverse-transform idea is really used twice: first in step 2 to distribute V uniformly between $F(a)$ and $F(b)$ and then in step 3 to obtain X. (See Prob. 8.3 for another way to generate X and Prob. 8.4 for a different type of truncation, which results in a distribution function that is *not* the same as F^*.)

Finally, the inverse-transform method can be quite useful for generating order statistics. Suppose that $Y_1, Y_2, \ldots, Y_n$ are IID with common distribution function F and that for $i = 1, 2, \ldots, n$, $Y_{(i)}$ denotes the ith smallest of the Y_j's. Recall from Chap. 6 that $Y_{(i)}$ is called the ith order statistic from a sample of size n. [Order statistics have been useful in simulation when one is concerned with the reliability, or *lifetime*, of some system having components subject to failure. If Y_j is the lifetime of the jth component, then $Y_{(1)}$ is the lifetime of a system consisting of n such components connected in series and $Y_{(n)}$ is the lifetime of the system if the components are connected in parallel.] One direct way of generating $X = Y_{(i)}$ is first to generate n IID variates $Y_1, Y_2, \ldots, Y_n$ with distribution function F, then sort them into increasing order, and finally set X to the ith value of the Y_j's after sorting. This method, however, requires generating n separate variates with distribution function F and then sorting them, which can be slow if n is large. As an alternative, we can use the following algorithm to generate $X = Y_{(i)}$:

1. Generate $V \sim \text{beta}(i, n - i + 1)$.
2. Return $X = F^{-1}(V)$.

The validity of this algorithm is established in Prob. 8.5. Note that step 1 requires generating from a beta distribution, which we discuss below in Sec. 8.3.8. No sorting is required, and we need to evaluate F^{-1} only once; this is particularly advantageous if n is large or evaluating F^{-1} is slow. Two important special cases are generating either the minimum or maximum of the n Y_j's, where step 1 becomes particularly simple. For the minimum, $i = 1$ and V in step 1 can be defined by $V = 1 - U^{1/n}$, where $U \sim U(0, 1)$. For the maximum, $i = n$ and we can set $V = U^{1/n}$ in step 1. (See Prob. 8.5 for verification in these two special cases.) For more on generating order statistics, see Ramberg and Tadikamalla (1978), Schmeiser (1978a, 1978b), and Schucany (1972).

8.2.2 Composition

The *composition* technique applies when the distribution function F from which we wish to generate can be expressed as a convex combination of other distribution functions $F_1, F_2, \ldots$. We would hope to be able to sample from the F_j's more easily than from the original F.

Specifically, we assume that for all x, $F(x)$ can be written as

$$F(x) = \sum_{j=1}^{\infty} p_j F_j(x)$$

where $p_j \geq 0$, $\sum_{j=1}^{\infty} p_j = 1$, and each F_j is a distribution function. (Although we have written this combination as an infinite sum, there may be a k such that $p_k > 0$ but $p_j = 0$ for $j > k$, in which case the sum is actually finite.) Equivalently, if X has density f that can be written as

$$f(x) = \sum_{j=1}^{\infty} p_j f_j(x)$$

where the f_j's are other densities, the method of composition still applies; the discrete case is analogous. The general composition algorithm, then, is as follows:

1. Generate a positive random integer J such that

$$P(J = j) = p_j \qquad \text{for } j = 1, 2, \ldots$$

2. Return X with distribution function F_J.

Step 1 can be thought of as choosing the distribution function F_j with probability p_j and could be accomplished, for example, by the discrete inverse-transform method. Given that $J = j$, generating X in step 2 should be done, of course, independently of J. By conditioning on the value of J generated in step 1, we can easily see that the X returned by the algorithm will have distribution function F [see, for example, Ross (2003, chap. 3)]:

$$P(X \leq x) = \sum_{j=1}^{\infty} P(X \leq x \mid J = j)P(J = j) = \sum_{j=1}^{\infty} F_j(x)p_j = F(x)$$

Sometimes we can give a geometric interpretation to the composition method. For a continuous random variable X with density f, for example, we might be able to divide the area under f into regions of areas $p_1, p_2, \ldots$, corresponding to the decomposition of f into its convex-combination representation. Then we can think of step 1 as choosing a region and step 2 as generating from the distribution corresponding to the chosen region. The following two examples allow this kind of geometric interpretation.

EXAMPLE 8.3. The *double-exponential* (or *Laplace*) *distribution* has density $f(x) = 0.5e^{-|x|}$ for all real x; this density is plotted in Fig. 8.6. From the plot we see that except for the normalizing factor 0.5, $f(x)$ is two exponential densities placed back to back; this suggests the use of composition. Indeed, we can express the density as

$$f(x) = 0.5e^{x}I_{(-\infty,0)}(x) + 0.5e^{-x}I_{[0,\infty)}(x)$$

where I_A denotes the *indicator function* of the set A, defined by

$$I_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{otherwise} \end{cases}$$

Thus, $f(x)$ is a convex combination of $f_1(x) = e^{x}I_{(-\infty,0)}(x)$ and $f_2(x) = e^{-x}I_{[0,\infty)}(x)$, both of which are densities, and $p_1 = p_2 = 0.5$. Therefore, we can generate an X with density f by

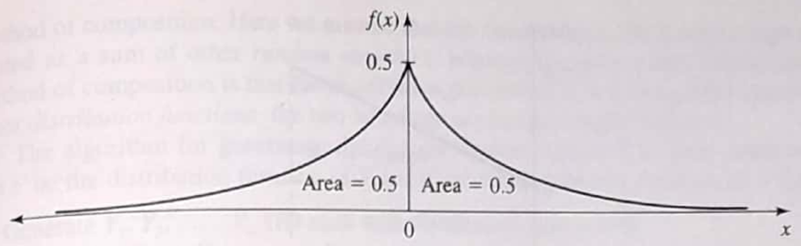

FIGURE 8.6
Double-exponential density.

composition. First generate U_1 and U_2 as IID U(0, 1). If $U_1 \leq 0.5$, return $X = \ln U_2$. On the other hand, if $U_1 > 0.5$, return $X = -\ln U_2$. Note that we are essentially generating an exponential random variate with mean 1 and then changing its sign with probability 0.5. Alternatively, we are generating from the left half of the density in Fig. 8.6 with probability equal to the corresponding area (0.5) and from the right half with probability 0.5.

Note that in Example 8.3, step 2 of the general composition algorithm was accomplished by means of the inverse-transform method for exponential random variates; this illustrates how different general approaches for generating random variates might be combined. Also, we see that *two* random numbers are required to generate a single X in this example; in general, we shall need *at least two* random numbers to use the composition method. (The reader may find it interesting to compare Example 8.3 with the inverse-transform method for generating a double-exponential random variate; see Prob. 8.6.)

In Example 8.3 we obtained the representation for f by dividing the area below the density with a vertical line, namely, the ordinate axis. In the following example, we make a horizontal division instead.

EXAMPLE 8.4. For $0 < a < 1$, the *right-trapezoidal distribution* has density

$$f(x) = \begin{cases} a + 2(1 - a)x & \text{if } 0 \leq x \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

(see Fig. 8.7). As suggested by the dashed lines, we can think of dividing the area under f into a rectangle having area a and a right triangle with area $1 - a$. Now $f(x)$ can be decomposed as

$$f(x) = aI_{[0,1]}(x) + (1 - a)2xI_{[0,1]}(x)$$

so that $f_1(x) = I_{[0,1]}(x)$, which is simply the U(0, 1) density, and $f_2(x) = 2xI_{[0,1]}(x)$ is a right-triangular density. Clearly, $p_1 = a$ and $p_2 = 1 - a$. The composition method thus calls for generating $U_1 \sim$ U(0, 1) and checking whether $U_1 \leq a$. If so, generate an independent $U_2 \sim$ U(0, 1), and return $X = U_2$. If $U_1 > a$, however, we must generate from the right-triangular distribution. This can be accomplished either by generating $U_2 \sim$ U(0, 1) and returning $X = \sqrt{U_2}$, or by generating U_2 and U_3 distributed as IID U(0, 1) and returning $X = \max\{U_2, U_3\}$ (see Prob. 8.7). Since the time to take a square root is probably greater than that required to generate an extra U(0, 1) random variate *and* to perform a comparison, the latter method would appear to be a faster way of generating an X with density f_2.

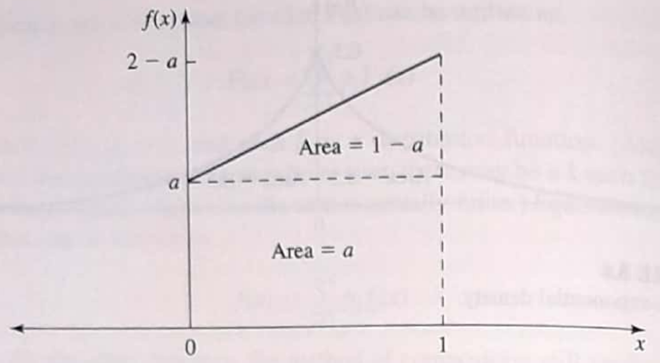

FIGURE 8.7
Right-trapezoidal density.

Again, the reader is encouraged to develop the inverse-transform method for generating a random variate from the right-trapezoidal distribution in Example 8.4. Note that especially if a is large, the composition method will be faster than the inverse transform, since the latter *always* requires that a square root be taken, while it is quite likely (with probability a) that the former will simply return $X = U_2 \sim U(0, 1)$. This increase in speed must be played off by the analyst against the possible disadvantage of having to generate two or three random numbers to obtain one value of X. Trapezoidal distributions like that in Example 8.4 play an important role in the efficient methods developed by Schmeiser and Lal (1980) for generating gamma random variates and for beta generation in Schmeiser and Babu (1980).

Composition methods (also called "mixture" methods) are further analyzed by Peterson and Kronmal (1982), who show as well that many specific variate-generation methods can actually be expressed as a composition of some sort. An interesting technique that is related closely to composition, the *acceptance-complement method,* was proposed by Kronmal and Peterson (1981, 1982); Devroye (1986, pp. 75–81) further discusses this and associated methods.

8.2.3 Convolution

For several important distributions, the desired random variable X can be expressed as a sum of other random variables that are IID and can be generated more readily than direct generation of X. We assume that there are IID random variables $Y_1, Y_2, \ldots, Y_m$ (for fixed m) such that $Y_1 + Y_2 + \cdots + Y_m$ has the same distribution as X; hence we write

$$X = Y_1 + Y_2 + \cdots + Y_m$$

The name of this method, *convolution,* comes from terminology in stochastic processes, where the distribution of X is called the *m-fold convolution* of the distribution of a Y_j. The reader should take care not to confuse this situation with the method of composition. Here we assume that the *random variable X* can be represented as a sum of other *random variables,* whereas the assumption behind the method of composition is that the *distribution function* of X is a (weighted) sum of other *distribution functions;* the two situations are fundamentally different.

The algorithm for generating the desired random variate X is quite intuitive (let F be the distribution function of X and G be the distribution function of a Y_j):

1. Generate $Y_1, Y_2, \ldots, Y_m$ IID each with distribution function G.
2. Return $X = Y_1 + Y_2 + \cdots + Y_m$.

To demonstrate the validity of this algorithm, recall that we assumed that X and $Y_1 + Y_2 + \cdots + Y_m$ have the same distribution function, namely, F. Thus,

$$P(X \le x) = P(Y_1 + Y_2 + \cdots + Y_m \le x) = F(x)$$

EXAMPLE 8.5. The m-Erlang random variable X with mean β can be defined as the sum of m IID exponential random variables with common mean β/m. Thus, to generate X, we can first generate $Y_1, Y_2, \ldots, Y_m$ as IID exponential with mean β/m (see Example 8.1), then return $X = Y_1 + Y_2 + \cdots + Y_m$. (See Sec. 8.3.3 for an improvement in efficiency of this algorithm.)

The convolution method, when it can be used, is very simple, provided that we can generate the required Y_j's easily. However, depending on the particular parameters of the distribution of X, it may not be the most efficient way. For example, to generate an m-Erlang random variate by the convolution method (as in Example 8.5) when m is large could be very slow. In this case it would be better to recall that the m-Erlang distribution is a special case of the gamma distribution (see Sec. 6.2.2) and to use a general method for generating gamma random variates (see Sec. 8.3.4). See also Devroye (1988).

Convolution is really an example of a more general idea, that of transforming some intermediate random variates into a final variate that has the desired distribution; the transformation with convolution is just adding, and the intermediate variates are IID. There are many other ways to transform intermediate variates, some of which are discussed in Sec. 8.2.6, as well as in Secs. 8.3 and 8.4.

8.2.4 Acceptance-Rejection

The three approaches for generating random variates discussed so far (inverse transform, composition, and convolution) might be called *direct* in the sense that they deal directly with the distribution or random variable desired. The *acceptance-rejection method* is less direct in its approach and can be useful when the direct methods fail or are inefficient. Our discussion is for the continuous case, where we want to generate X having distribution function F and density f; the discrete case is exactly analogous and is treated in Prob. 8.9. The underlying idea dates back to at least von Neumann (1951).

The acceptance-rejection method requires that we specify a function t, called the *majorizing function,* such that $t(x) \ge f(x)$ for all x. Now t will not, in general, be a density since

$$c = \int_{-\infty}^{\infty} t(x)\,dx \geq \int_{-\infty}^{\infty} f(x)\,dx = 1$$

but the function $r(x) = t(x)/c$ clearly *is* a density. (We assume that t is such that $c < \infty$.) We must be able to generate (easily and quickly, we hope) a random variate Y having density r. The general algorithm follows:

1. Generate Y having density r.
2. Generate $U \sim \mathrm{U}(0, 1)$, independent of Y.
3. If $U \leq f(Y)/t(Y)$, return $X = Y$. Otherwise, go back to step 1 and try again.

The algorithm continues looping back to step 1 until finally we generate a (Y, U) pair in steps 1 and 2 for which $U \leq f(Y)/t(Y)$, when we "accept" the value Y for X. Since demonstrating the validity of this algorithm is more complicated than for the three previous methods, we refer the reader to App. 8A for a proof.

EXAMPLE 8.6. The beta(4, 3) distribution (on the unit interval) has density

$$f(x) = \begin{cases} 60x^3(1-x)^2 & \text{if } 0 \leq x \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

[Since the distribution function $F(x)$ is a sixth-degree polynomial, the inverse-transform approach would not be simple, involving numerical methods to find polynomial roots.] By standard differential calculus, i.e., setting $df/dx = 0$, we see that the maximum value of $f(x)$ occurs at $x = 0.6$, where $f(0.6) = 2.0736$ (exactly). Thus, if we define

$$t(x) = \begin{cases} 2.0736 & \text{if } 0 \leq x \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

then t majorizes f. Next, $c = \int_0^1 2.0736\,dx = 2.0736$, so that $r(x)$ is just the U(0, 1) density. The functions f, t, and r are shown in Fig. 8.8. The algorithm first generates Y and U as IID U(0, 1) random variates in steps 1 and 2; then in step 3 we check whether

$$U \leq \frac{60\,Y^3(1-Y)^2}{2.0736}$$

If so, we return $X = Y$; otherwise, we reject Y and go back to step 1.

Note that in the preceding example, X is bounded on an interval (the unit interval in this case), and so we were able to choose t to be constant over this interval, which in turn led to r's being a uniform density. The acceptance-rejection method is often stated *only* for such bounded random variables X and *only* for this uniform choice of r; our treatment is more general.

The acceptance-rejection algorithm above seems curious to say the least, and the proof of its validity in App. 8A adds little insight. There is, however, a natural intuition to the method. Figure 8.9 presents again the $f(x)$ and $t(x)$ curves from Example 8.6, and in addition shows the algorithm in action. We generated 50 X's from the beta(4, 3) distribution by acceptance-rejection (using stream 2 of the random-number generator in App. 7A), which are marked by crosses on the x axis. On the $t(x)$ curve at the top of the graph we also mark the location of all the Y's generated in step 1 of the algorithm, regardless of whether they ended up being

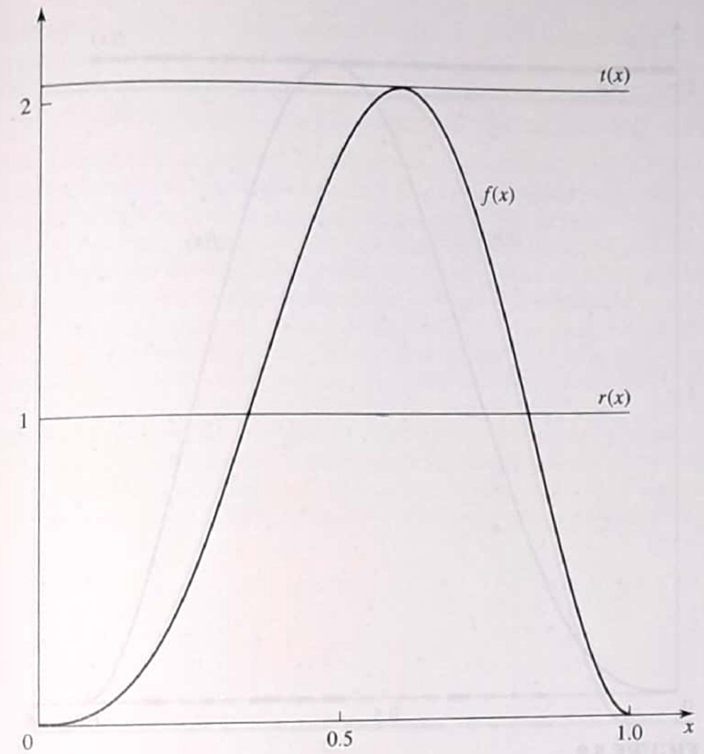

FIGURE 8.8
$f(x)$, $t(x)$, and $r(x)$ for the acceptance-rejection method, beta(4, 3) distribution.

accepted as X's; 50 of these Y's were accepted and made it down to the x axis. The uniformity of the Y's on the $t(x)$ curve is evident, and the higher concentration of the X's on the x axis where $f(x)$ is high is also clear. For those Y's falling in regions where $f(x)$ is low (e.g., x near 0 or 1), $f(Y)/t(Y)$ is small, and as this is the probability of accepting Y as an X, most of such Y's will be rejected. This can be seen in Fig. 8.9 for small (near 0) and large (near 1) values of Y where $f(x)$ is small. On the other hand, Y values where $f(x)$ is high (e.g., near $x = 0.6$) will probably be kept, since $f(Y)/t(Y)$ is nearly 1; thus, most of the Y's around $x = 0.6$ are accepted as X's and make it down to the x axis. In this way, the algorithm "thins out" the Y's from the $r(x)$ density where $t(x)$ is much larger than $f(x)$, but retains most of the Y's where $t(x)$ is only a little higher than $f(x)$. The result is that the concentration of the Y's from $r(x)$ is altered to agree with the desired density $f(x)$.

The principle of acceptance-rejection is quite general, and looking at the above algorithm in a slightly different way clarifies how it can be extended to generation of random points in higher-dimensional spaces; this is important, for example, in Monte Carlo estimation of multiple integrals (see Sec. 1.8.3). The acceptance

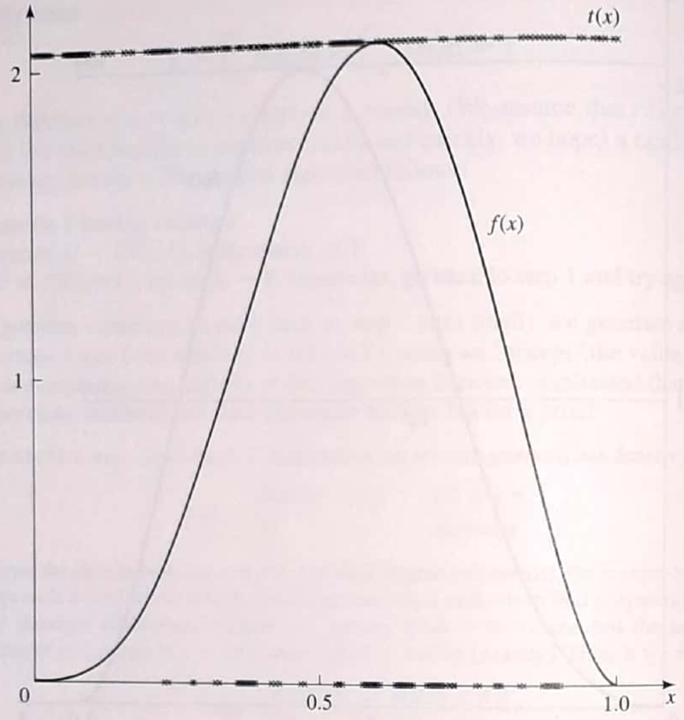

FIGURE 8.9
Sample of 50 X's (on horizontal axis) and the required Y's [on the $t(x)$ line], acceptance-rejection method for beta(4, 3) distribution.

condition in step 3 of the algorithm can obviously be restated as $Ut(Y) \le f(Y)$, which means geometrically that Y will be accepted as an X if the point $(Y, Ut(Y))$ falls under the curve for the density f. Figure 8.10 shows this for the same Y values as in Fig. 8.9, with the dots being the points $(Y, Ut(Y))$ and the 50 accepted values of X again being marked by crosses on the x axis. By accepting the Y values for those $(Y, Ut(Y))$ points falling under the $f(x)$ curve, it is intuitive that the accepted X's will be more dense on the x axis where $f(x)$ is high, since it is more likely that the uniformly distributed dots will be under $f(x)$ there. While in this particular example the rectangular nature of the region under $t(x)$ makes the uniformity of the points $(Y, Ut(Y))$ clear, the same is true for regions of any shape, and in any dimension. The challenge is to find a way of efficiently generating points uniformly in an arbitrary non-rectangular region; Smith (1984) discusses this, and proposes a more efficient alternative to acceptance-rejection in high-dimensional spaces.

Although acceptance-rejection generates a value of X with the desired distribution regardless of the choice of the majorizing function t, this choice will play an important role in its efficiency in two ways. First, since step 1 requires generating Y with density $t(x)/c$, we want to choose t so that this can be accomplished rapidly.

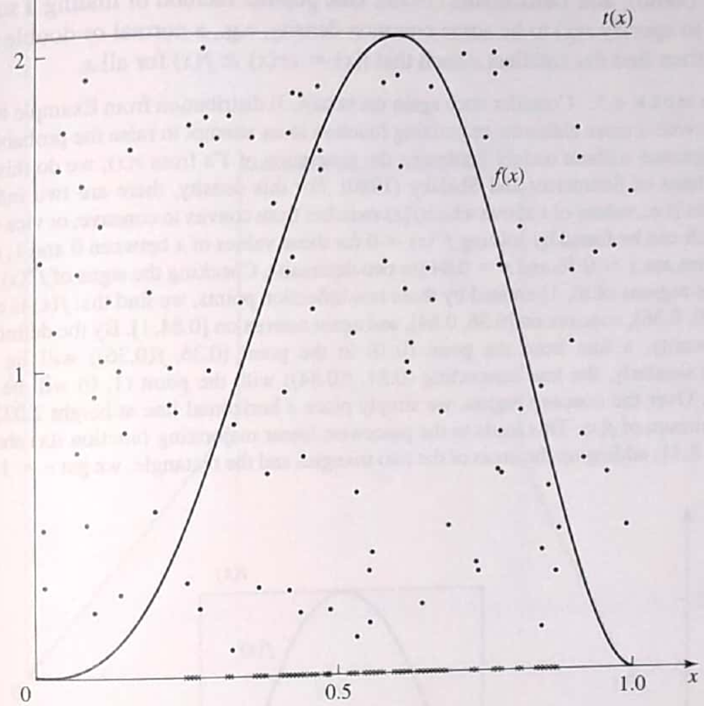

FIGURE 8.10
Sample of 50 X's (on horizontal axis) and the required $(Y, Ut(Y))$ pairs, acceptance-rejection method for beta(4, 3) distribution.

(The uniform t chosen in Example 8.6 certainly satisfies this wish.) Second, we hope that the probability of rejection in step 3 can be made small, since we have to start all over if this rejection occurs. In App. 8A we show that on any given iteration through the algorithm, the probability of acceptance in step 3 is $1/c$; we therefore would like to choose t so that c is small. Thus, we want to find a t that fits closely above f, bringing c closer to 1, its lower bound. Intuitively, a t that is only a little above f leads to a density r that will be close to f, so that the Y values generated from r in step 1 are from a distribution that is almost correct, and so we should accept most of them. (From this standpoint, then, we see that the uniform choice of t in Example 8.6 might not be so wise after all, since it does not fit down on top of f very snugly. Since $c = 2.0736$, the probability of acceptance is only about 0.48, lower than we might like.) These two goals, ease of generation from $t(x)/c$ and a small value of c, may well conflict with each other, so the choice of t is by no means obvious and deserves care. Considerable research has been aimed at identifying good choices for t for a given distribution; see, for example, Ahrens and Dieter (1972, 1974), Atkinson (1979b), Atkinson and Whittaker (1976), Schmeiser (1980a, 1980b), Schmeiser and Babu (1980), Schmeiser and Lal (1980), Schmeiser and Shalaby (1980), and Tadikamalla (1978). One popular method of finding a suitable t is *first* to specify $r(x)$ to be some common density, e.g., a normal or double exponential, then find the smallest c such that $t(x) = cr(x) \geq f(x)$ for all x.

EXAMPLE 8.7. Consider once again the beta(4, 3) distribution from Example 8.6, but now with a more elaborate majorizing function in an attempt to raise the probability of acceptance without unduly burdening the generation of Y's from $r(x)$; we do this along the lines of Schmeiser and Shalaby (1980). For this density, there are two *inflection points* [i.e., values of x above which $f(x)$ switches from convex to concave, or vice versa], which can be found by solving $f''(x) = 0$ for those values of x between 0 and 1; the solutions are $x = 0.36$ and $x = 0.84$ (to two decimals). Checking the signs of $f''(x)$ on the three regions of [0, 1] created by these two inflection points, we find that $f(x)$ is convex on [0, 0.36], concave on [0.36, 0.84], and again convex on [0.84, 1]. By the definition of convexity, a line from the point (0, 0) to the point (0.36, $f(0.36)$) will lie above $f(x)$; similarly, the line connecting (0.84, $f(0.84)$) with the point (1, 0) will be above $f(x)$. Over the concave region, we simply place a horizontal line at height 2.0736, the maximum of $f(x)$. This leads to the piecewise-linear majorizing function $t(x)$ shown in Fig. 8.11; adding up the areas of the two triangles and the rectangle, we get $c = 1.28$, so

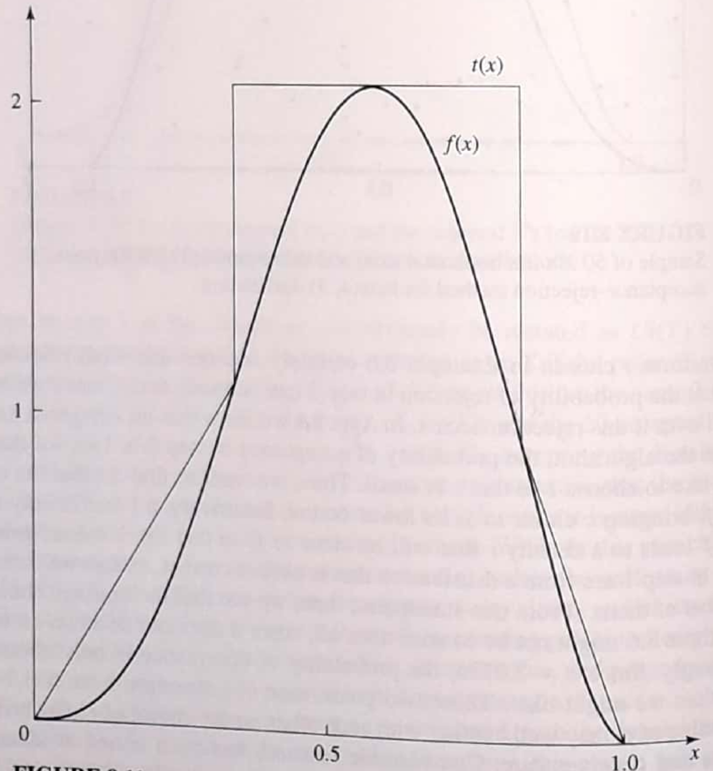

FIGURE 8.11
$f(x)$ and piecewise-linear majorizing function $t(x)$, acceptance-rejection method for beta(4, 3) distribution.

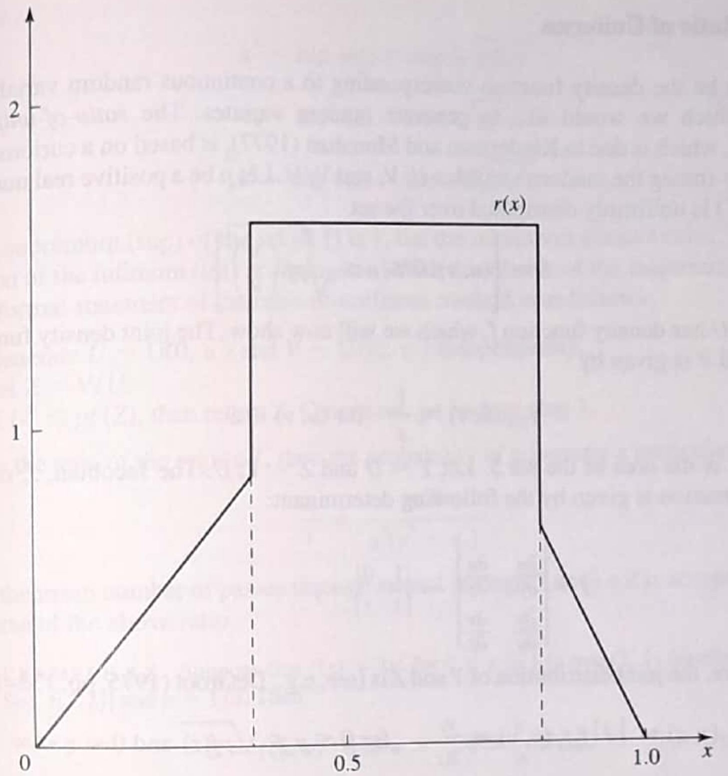

FIGURE 8.12
$r(x)$ corresponding to the piecewise-linear majorizing function $t(x)$ in Fig. 8.11.

that the probability of acceptance on a given pass through the algorithm is 0.78, being considerably better than the 0.48 acceptance probability when using the simple uniform majorizing function in Example 8.6. However, it now becomes more difficult to generate values from the density $r(x)$, which is plotted in Fig. 8.12; this is a typical tradeoff in specifying a majorizing function. As suggested in Fig. 8.12, generating from $r(x)$ can be done using composition, dividing the area under $r(x)$ into three regions, each corresponding to a density from which generation is easy; see Prob. 8.16. Thus, we would really be combining three different generation techniques here: inverse transform [for the component densities of $r(x)$], composition [for $r(x)$], and finally acceptance-rejection [for $f(x)$]. In comparison with Example 8.6, whether the higher acceptance probability justifies the increased work to generate a Y is not clear, and may depend on several factors, such as the particular parameters, code efficiency, as well as the programming language, compiler, and hardware.

There have been many variations of and modifications to the general acceptance-rejection method, mostly to improve speed. For example, Hörmann and Derflinger (1996) give a version for generating from discrete distributions (including those with infinite tails), which uses a continuous majorizing function and avoids generating a separate random number to decide between acceptance and rejection.

8.2.5 Ratio of Uniforms

Let $f(x)$ be the density function corresponding to a continuous random variable X, from which we would like to generate random variates. The *ratio-of-uniforms method*, which is due to Kinderman and Monahan (1977), is based on a curious relationship among the random variables U, V, and V/U. Let p be a positive real number. If (U, V) is uniformly distributed over the set

$$S = \left\{(u, v) : 0 \le u \le \sqrt{pf\left(\frac{v}{u}\right)}\right\}$$

then V/U has density function f, which we will now show. The joint density function of U and V is given by

$$f_{U,V}(u, v) = \frac{1}{s} \quad \text{for } (u, v) \in S$$

where s is the area of the set S. Let $Y = U$ and $Z = V/U$. The Jacobian, J, of this transformation is given by the following determinant:

$$J = \begin{vmatrix} \frac{\partial u}{\partial y} & \frac{\partial u}{\partial z} \\ \frac{\partial v}{\partial y} & \frac{\partial v}{\partial z} \end{vmatrix} = \begin{vmatrix} 1 & 0 \\ z & y \end{vmatrix} = y$$

Therefore, the joint distribution of Y and Z is [see, e.g., DeGroot (1975, pp. 133–136)]

$$f_{Y,Z}(y, z) = |J| f_{U,V}(u, v) = \frac{y}{s} \qquad \text{for } 0 \le y \le \sqrt{pf(z)} \text{ and } 0 < z < \infty$$

so that the density function of Z is

$$f_Z(z) = \int_0^{\sqrt{pf(z)}} f_{Y,Z}(y, z)\, dy = \int_0^{\sqrt{pf(z)}} \frac{y}{s}\, dy = \frac{p}{2s} f(z)$$

Since $f_Z(z)$ and $f(z)$ must both integrate to 1, it follows that $s = p/2$ and that $Z = V/U$ has density $f(x)$ as desired.

To generate (u, v) uniformly in S, we may choose a majorizing region T that contains S, generate a point (u, v) uniformly in T, and accept this point if

$$u^2 \le pf\left(\frac{v}{u}\right)$$

Otherwise, we generate a new point in T and try again, etc. Hopefully, it should be easy to generate a point uniformly in the region T.

The boundary of the acceptance region S is defined parametrically by the following equations (see Prob. 8.19):

$$u(z) = \sqrt{pf(z)} \quad \text{and} \quad v(z) = z\sqrt{pf(z)} \tag{8.1}$$

If $f(x)$ and $x^2 f(x)$ are bounded, then a good choice of T is the rectangle

$$T = \{(u, v) : 0 \le u \le u^* \text{ and } v_* \le v \le v^*\}$$

where

$$u^* = \sup_z u(z) = \sup_z \sqrt{pf(z)}$$

$$v_* = \inf_z v(z) = \inf_z z\sqrt{pf(z)}$$

$$v^* = \sup_z v(z) = \sup_z z\sqrt{pf(z)}$$

[The supremum (sup) of the set (0,1) is 1, but the maximum doesn't exist. The definition of the infimum (inf) is analogous.] With this choice of the majorizing region T, a formal statement of the ratio-of-uniforms method is as follows:

1. Generate $U \sim U(0, u^*)$ and $V \sim U(v_*, v^*)$ independently.
2. Set $Z = V/U$.
3. If $U^2 \le pf(Z)$, then return Z. Otherwise, go back to step 1.

If t is the area of the region T, then the probability of accepting a particular Z is

$$\frac{s}{t} = \frac{p/2}{u^*(v^* - v_*)}$$

and the mean number of passes through steps 1 through 3 until a Z is accepted is the inverse of the above ratio.

EXAMPLE 8.8. Suppose that $f(x) = 3x^2$ for $0 \le x \le 1$ [a beta(3, 1) distribution (see Sec. 6.2.2)] and $p = 1/3$. Then

$$S = \{(u, v) : 0 \le u \le \frac{v}{u}, 0 \le \frac{v}{u} \le 1\}$$

$$s = \frac{1}{6}$$

$$u^* = 1, \quad v_* = 0, \quad v^* = 1$$

We generated 2000 points (u, v) uniformly in the square T (with area $t = 1$), and 330 of these points satisfied the stopping rule stated in the definition of S. Note that $330/2000 = 0.165$, which is approximately equal to $s/t = \frac{1}{6}/1 = 0.167$. A plot of these 330 accepted points is given in Fig. 8.13. Note that the acceptance region S is bounded above and below by the curves $v = u$ and $v = u^2$, respectively (see Prob. 8.20).

The rectangle T had a probability of acceptance of 1/6 in Example 8.8. We could choose a majorizing region T that is closer in shape to the acceptance region S so that the ratio s/t is brought closer to 1 (see Prob. 8.21). However, this gain in efficiency has to be traded off with the potentially greater difficulty of generating a point uniformly in a more complicated majorizing region. Cheng and Feast (1979) give a fast algorithm for generating from a gamma distribution, where T is a parallelogram. Leydold (2000) develops fast ratio-of-uniforms algorithms that use polygonal majorizing regions and are applicable to a large class of distributions.

Stadlober (1990) shows that the ratio-of-uniforms method is, in fact, an acceptance-rejection method. He also extends the ratio-of-uniforms method to

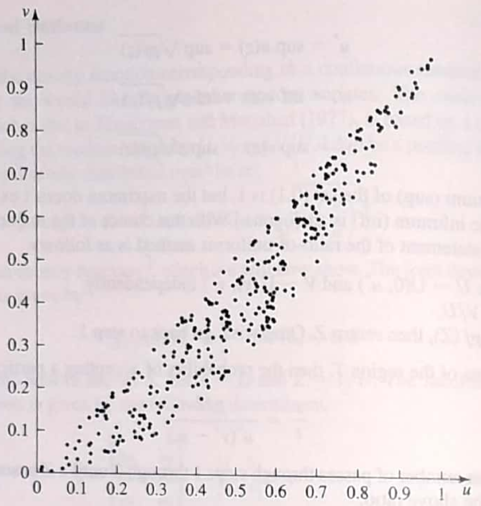

FIGURE 8.13
Accepted points for the ratio-of-uniforms method.

discrete distributions, while Wakefield et al. (1991) and Stefănescu and Văduva (1987) consider multivariate distributions.

8.2.6 Special Properties

Although most methods for generating random variates can be classified into one of the five approaches discussed so far in Sec. 8.2, some techniques simply rely on some *special property* of the desired distribution function F or the random variable X. Frequently, the special property will take the form of representing X in terms of other random variables that are more easily generated; in this sense the method of convolution is a "special" special property. The following four examples are based on normal-theory random variables (see Secs. 8.3 and 8.4 for other examples that have nothing to do with the normal distribution).

EXAMPLE 8.9. If $Y \sim N(0, 1)$ (the standard normal distribution), then Y^2 has a chi-square distribution with 1 df. (We write $X \sim \chi_k^2$ to mean that X has a chi-square distribution with k df.) Thus, to generate $X \sim \chi_1^2$, generate $Y \sim N(0, 1)$ (see Sec. 8.3.6), and return $X = Y^2$.

EXAMPLE 8.10. If $Z_1, Z_2, \ldots, Z_k$ are IID χ_1^2 random variables, then $X = Z_1 + Z_2 + \cdots + Z_k \sim \chi_k^2$. Thus, to generate $X \sim \chi_k^2$, first generate $Y_1, Y_2, \ldots, Y_k$ as IID N(0, 1) random variates, and then return $X = Y_1^2 + Y_2^2 + \cdots + Y_k^2$ (see Example 8.9). Since for large k this may be quite slow, we might want to exploit the fact that the χ_k^2 distribution is a gamma distribution with shape parameter $\alpha = k/2$ and scale parameter $\beta = 2$. Then X can be obtained directly from the gamma-generation methods discussed in Sec. 8.3.4.

EXAMPLE 8.11. If $Y \sim N(0, 1)$, $Z \sim \chi_k^2$, and Y and Z are independent, then $X = Y/\sqrt{Z/k}$ is said to have *Student's t distribution* with k df, which we denote $X \sim t_k$. Thus, to generate $X \sim t_k$, we generate $Y \sim N(0, 1)$ and $Z \sim \chi_k^2$ independently of Y (see Example 8.10), and return $X = Y/\sqrt{Z/k}$.

EXAMPLE 8.12. If $Z_1 \sim \chi_{k_1}^2$, $Z_2 \sim \chi_{k_2}^2$, and Z_1 and Z_2 are independent, then

$$X = \frac{Z_1/k_1}{Z_2/k_2}$$

is said to have an *F distribution* with (k_1, k_2) df, denoted $X \sim F_{k_1,k_2}$. We thus generate $Z_1 \sim \chi_{k_1}^2$ and $Z_2 \sim \chi_{k_2}^2$ independently, and return $X = (Z_1/k_1)/(Z_2/k_2)$.

For some continuous distributions, it is possible to transform the density function so that it is easy to construct majorizing functions for use with the acceptance-rejection method. In particular, Hörmann (1995) suggested the *transformed density rejection method* for generating random variates from a continuous distribution with density f. The idea is to transform f by a strictly increasing function T so that $T(f(x))$ is concave, in which case we say that f is *T-concave*. [A function g is said to be *concave* if

$$\frac{g(x_1) + g(x_2)}{2} < g\left(\frac{x_1 + x_2}{2}\right)$$

for $x_1 < x_2$.] Since $T(f(x))$ is a concave function, a majorizing function for $T(f(x))$ can be constructed easily as the minimum of several tangents. Then T^{-1} is used to transform the majorizing function back to the original scale. This results in a majorizing function for the density f, and random variates can be generated from f by the acceptance-rejection method. If $T(x) = -1/\sqrt{x}$, then a large number of distributions are T-concave, including the beta, exponential, gamma, lognormal, normal, and Weibull distributions. (For some distributions, there are restrictions on the values of the parameters.) Since transformed density rejection is applicable to a large class of distributions, it is sometimes called a *universal* method [see Hörmann et al. (2004)].

8.3 GENERATING CONTINUOUS RANDOM VARIATES

In this section we discuss particular algorithms for generating random variates from several commonly occurring continuous distributions; Sec. 8.4 contains a similar treatment for discrete random variates. Although there may be several different algorithms for generating from a given distribution, we explicitly present