CHAPTER 5

Random Variate Generation

RUSSELL C. H. CHENG
University of Kent at Canterbury

5.1 INTRODUCTION

In Chapter 4 the generation of (pseudo)*random numbers* was discussed. In this chapter, by *random number* is always meant a uniform random variable, denoted by RN(0,1), whose distribution function is

$$F_U(u) = \begin{cases} 0 & u \le 0 \\ u & 0 < u < 1 \\ 1 & u \ge 1 \end{cases} \tag{1}$$

Thus random numbers are always uniformly distributed on the unit interval (0,1). Random number generation is an important topic in its own right (see Chapter 4). In contrast, random variate generation always refers to the generation of variates whose probability distribution is different from that of the uniform on the interval (0,1). The basic problem is therefore to generate a random variable, X, whose distribution function

$$F(x) = \Pr(X \le x) \qquad -\infty < x < \infty \tag{2}$$

is assumed to be completely known, and which is different from that of (1).

Most mainframe computing systems contain libraries with implementations of generators for the more commonly occurring distributions. Generators are also available in many of the existing statistical and simulation packages for personal computers. Choice of a generator often has as much to do with properties of the distribution to be used, as with the properties of the generator itself. In the description of specific generators given below, emphasis is therefore given to the underlying characteristics of the distributions being considered and in the relations between different distributions. This information should be useful in aiding the understanding of properties of generators. The intention is to allow an informed choice to be made, and the algorithms will then not need to be

Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice. Edited by J. Banks

treated so much as black box implementations. Additionally, general principles and general methods of random variate generation are described which will enable the reader to construct generators for nonstandard distributions.

Random variable generators invariably use as their starting point a random number generator, which yields RN(0,1), with distribution given by (1). The technique is to transform or manipulate one or more such uniform random variables in some efficient way to obtain a variable with the desired distribution (2). In this chapter it will be assumed that a source of random numbers is available. In addition to being uniform, the random numbers must also be mutually independent. A significant problem, discussed in Chapter 4, is to construct generators which produce random numbers that can safely be treated as being independent. It will be assumed in what follows that such a random number generator is available. In what follows it is assumed that a sequence of numbers produced by such a generator: $\{U_1, U_2, \ldots\}$ behaves indistinguishably from a sequence of independently distributed RN(0,1) variates.

As with all numerical techniques, there will usually be more than one method available to generate variates from some prescribed distribution. Four factors should be considered in selecting an appropriate generator:

1. *Exactness.* This refers to the distribution of the variates produced by the generator. The generator is said to be *exact* if the distribution of variates generated has the exact form desired. In certain applications where the precise distribution is not critical, a method may be acceptable which may be good as far as the other factors are concerned, but which produces variates whose distribution only approximates that of the desired one.

2. *Speed.* This refers to the computing time required to generate a variate. There are two contributions to the overall time. Usually, an initial *setup time* is needed to construct constants or tables used in calculating the variate. Calculation of the variate itself then incurs a further *variable generation time.* The relative importance of these two contributions depends on the application. There are two cases. The more usual one is where a sequence of random variates is needed, all with exactly the same distribution. Then the constants or tables need be setup only once, as the same values can be used in generating each variate of the sequence. In this case the setup time is negligible and can be ignored. However, if each variate has a different distribution, this setup time will be just as important, since in this case it has to be added to every marginal generation time.

3. *Space.* This is simply the computer memory requirement of the generator. Most algorithms for random variate generation are short. However, some make use of extensive tables, and this can become significant if different tables need to be held simultaneously in memory.

4. *Simplicity.* This refers to both the algorithmic simplicity and the implementational simplicity. Its importance depends on the context of the application. If, for example, the generator is needed in a single study, a simple generator that is easy to understand and to code will usually be more attractive than a complicated alternative even if the latter has better characteristics as far as other factors are concerned. If variates from a particular distribution are needed often and there is not already a library implementation, it will be worthwhile selecting a generator that is exact and fast and to write an efficiently coded version for permanent use.

In general, random variate generation will be only an incidental part of a computer

simulation program. The users who need to make their own generator will usually therefore not wish to spend an inordinate time writing and testing such generators. Preference has thus been given below to specific methods which are simple and reasonably fast without necessarily being the fastest.

Section 5.2 covers general principles commonly used in variate generation. Section 5.3 then discusses how these ideas are specialized for continuous random variables, together with specific algorithms for the more commonly occurring continuous distributions. Section 5.4 repeats this discussion for the case of discrete random variables. Section 5.5 covers extensions to multivariate distributions. In Section 5.6 we discuss the generation of stochastic processes.

Good references to the subject are refs. 1 to 3. A good elementary introduction to the subject is given in ref. 4, Chapter 9. A more advanced discussion is given in ref. 5, Chapter 3.

5.2 GENERAL PRINCIPLES

As mentioned in Section 5.1, from now on it is assumed that a (pseudo) random number generator is available that produces a sequence of independent RN(0,1) variates, with the understanding that each time this function is called, a new RN(0,1) variate is returned that is independent of all variates generated by previous calls to this function.

5.2.1 Inverse Transform Method

When X is a continuously distributed random variable, the distribution can be defined by its *density function* $f(x)$. The density function allows the probability that X lies between two given values a and $b(> a)$ to be evaluated as

$$\Pr\{a \le X \le b\} = \int_a^b f(x)\,dx$$

From its definition, (2), it will be seen that the distribution function in this case can be evaluated from the density function as the integral

$$F(x) = \int_{-\infty}^{x} f(z)\,dz$$

The distribution function is thus strictly increasing and continuous (Figure 5.1). In this case, for any $0 < u < 1$, the equation $F(x) = u$ can be solved to give a unique x. If (and this is the critical proviso) the inverse of a distribution function can be expressed in a simple form, this process can be denoted by rewriting the equation in the equivalent form $x = F^{-1}(u)$; so that x is expressed as an explicit function of u. This function F^{-1} is called the *inverse of* F. This yields the following convenient method for generating variables with the given distribution function F.

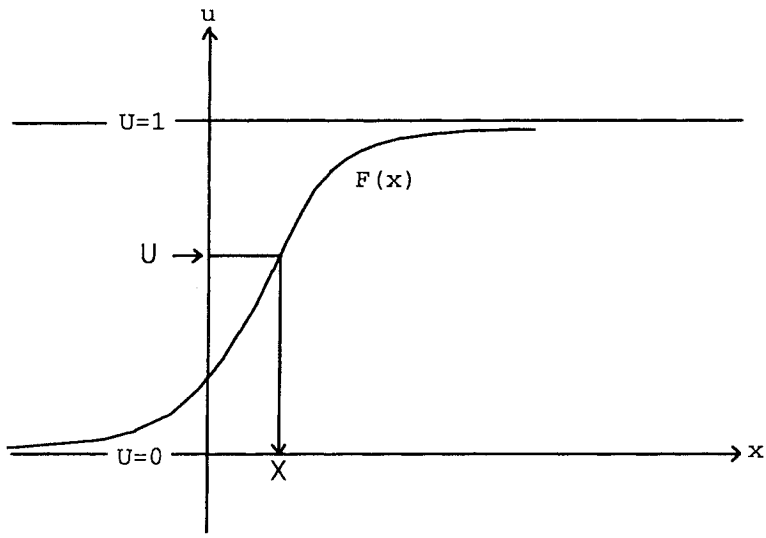

Figure 5.1 Inverse transform method $X = F^{-1}(U)$, continuous distribution.

Inverse Transform Method (Continuous Case)

```
Let U = RN(0,1)
Return X = F^-1(U)
```

To show that it works, all that is needed is to verify that the distribution function of X is indeed F [i.e., that $\Pr(X \le x) = F(x)$]. Now

$$\Pr(X \le x) = \Pr[F^{-1}(U) \le x] = \Pr[U \le F(x)]$$

But the right-hand probability is the distribution function of a uniform random variable evaluated at the value $F(x)$, and from (1) this is equal to $F(x)$, as required.

The best known and most useful example is when X is an exponentially distributed random variable with mean $a > 0$. Its distribution function is

$$F(x) = \begin{cases} 1 - \exp\left(\dfrac{-x}{a}\right) & x > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Solving $u = F(x)$ for x in this case yields

$$x = F^{-1}(u) = -a \ln(1 - u) \tag{4}$$

A random variate with distribution function (3) is therefore obtained by using this formula (4) to calculate X, with u generated as a $U(0, 1)$ variate.

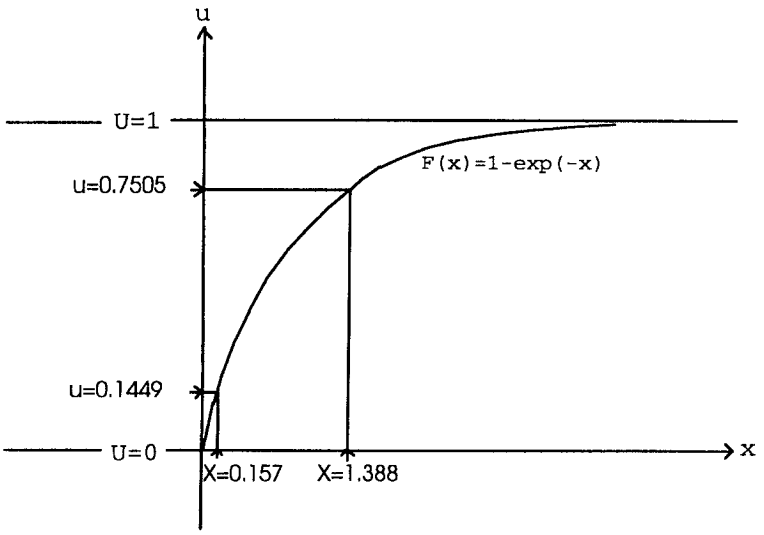

Figure 5.2 Inverse transform method $X = F^{-1}(U)$, exponential distribution.

Example 1 Figure 5.2 illustrates the use of the formula in the case when $a = 1$. The curve is the distribution function $F(x) = 1 - e^{-x}$. Two random variates have been generated using (4). For the first variate the random number is $u_1 = 0.7505$ and this is depicted on the vertical (y) scale. The horizontal distance at this vertical height from the *y-axis* to the graph of $F(x)$ is $x_1 = -\ln(1 - 0.7505) = 1.388$. This is the corresponding exponential variate that is generated. Similarly, the random number $u_2 = 0.1449$ generates the exponential variate value $x_2 = 0.1565$.

Random variates from many distributions can be generated in this way. The most commonly occurring are listed in Section 5.5.2. Johnson [6] lists the following additional cases.

1. Burr:

$$f(x) = \frac{ckx^{c-1}}{(1+x^c)^{k+1}} \qquad x > 0$$

$$F^{-1}(u) = [(1-u)^{-1/k} - 1]^{1/c}$$

2. Laplace:

$$f(x) = \tfrac{1}{2}\exp(-|x|) \qquad -\infty < x < \infty$$

$$F^{-1}(u) = \begin{cases} \ln(2u) & u \le \frac{1}{2} \\ -\ln[2(1-u)] & u > \frac{1}{2} \end{cases}$$

3. Logistic:

$$f(x) = \frac{\exp(-x)}{[1+\exp(-x)]^2} \qquad -\infty < x < \infty$$

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

4. Pareto:

$$f(x) = cx^{-c-1} \qquad x \geq 1$$
$$F^{-1}(u) = (1-u)^{-1/c}$$

In all these cases further flexibility is afforded by rescaling (a) and relocation (b) using the linear transformation

$$Y = aX + b$$

although some care may be needed in interpreting the meaning of a and b in particular contexts.

The inverse transform method extends to *discrete* random variables. Suppose that X takes only the values $x_1, x_2, \ldots, x_n$ with probabilities $p_i = \Pr(X = x_i)$ such that $\sum_{i=1}^{n} p_i = 1$. The distribution function for such a variable is

$$F(x) = \Pr(X \leq x) = \sum_{i:x_i \leq x} p_i$$

It can be verified (see, e.g., ref. 2) that if the inverse is defined by

$$F^{-1}(u) = \min\{x \mid u \leq F(x)\}$$

the inverse transform method will still work despite the discontinuities in $F(x)$ (see Figure 5.3). Moreover, we still have $\Pr(X \leq x) = F(x)$, so that X generated in this way has the required discrete distribution. The method reduces to:

Inverse Transform Method (Discrete Case)

```
Let U = RN(0,1)
Let i = 1
While (F(x_i) < U) {i = i + 1}
Return X = x_i
```

Because the method uses a linear search, it can be inefficient if n is large. More efficient methods are described later.

If a table of x_i values with the corresponding $F(x_i)$ values is stored, the method is also called the *table look-up method*. The method runs through the table comparing U with each $F(x_i)$, and returning, as X, the first x_i encountered for which $F(x_i) \geq U$.

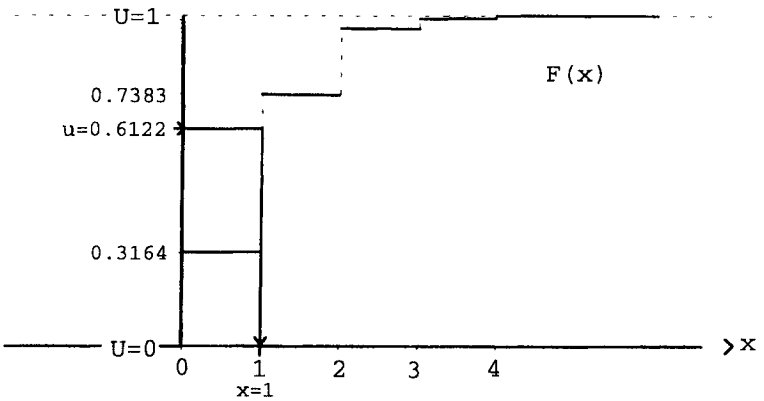

Figure 5.3 Inverse transform method $X = F^{-1}(U)$, Bin(4, 0.25) distribution.

Example 2 The well-known binomial distribution Bin(n,p), described later, is an example of a discrete distribution. The possible values that X can take in this case are 0, 1, ... n, with probabilities that depend on the parameters p and n (the general formula is given below). For the case $n = 4$ and $p = 0.25$, the possible values are $x_i = i$ for $i = 0, 1, 2, 3, 4$; and the distribution function is given in Table 5.1.

The table look-up method works as follows. Suppose that $u = 0.6122$ is a given random number. Looking along the row of $F(x_i)$ values, it will be seen that $F(x_0) = 0.3164 < u = 0.6122 < F(x_1) = 0.7383$. Thus $x_1 = 1$ is the first x_i encountered for which $u \le F(x_i)$. The variate generated in this case is therefore $X = 1$. The generation of this particular variate is depicted geometrically in Figure 5.3.

Sometimes the test of whether $F(x_i) < u$ takes a simple and explicit algebraic form. An example is the geometric distribution described below.

5.2.2 Acceptance–Rejection Method

The acceptance–rejection method is most easily explained for a continuous distribution. Suppose that we wish to sample from the distribution with density $f(x)$ but that it is difficult to do so by the inverse transform method. Suppose now that the following three assumptions hold (see also Figure 5.4):

1. There is another function $e(x)$ that dominates $f(x)$ in the sense that $e(x) \ge f(x)$ for all x.

TABLE 5.1 Distribution of Bin(4,0.25)

i	0	1	2	3	4
p_i	0.3164	0.4219	0.2109	0.0469	0.0039
$F(x_i)$	0.3164	0.7383	0.9492	0.9961	1.0000

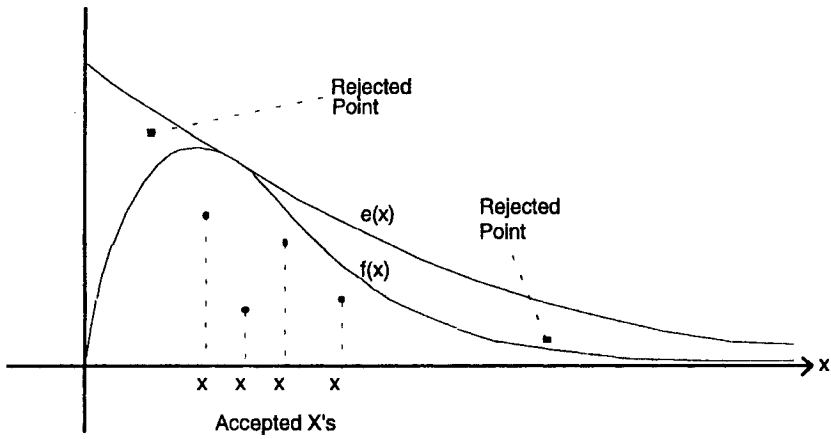

Figure 5.4 Acceptance–rejection method, points uniformly scattered under $e(x)$, gamma distribution with $b = 2$.

2. It is possible to generate points uniformly scattered under the graph of $e(x)$ (above the x-axis). Denote the coordinates of such a typical point by (X, Y).
3. If the graph of $f(x)$ is drawn on the same diagram, the point (X, Y) will be above or below it according as $Y > f(X)$ or $Y \leq f(X)$.

The acceptance–rejection method works by generating such points (X, Y) and returning the X coordinate as the generated X value, but only if the point lies under the graph of $f(x)$ [i.e., only if $Y \leq f(X)$]. Intuitively it is clear that the density of X is proportional to the height of f, so that X has the correct density. A formal proof can be found in ref. 3.

There are many ways of constructing $e(x)$. One requirement is that the area between the graphs of f and e be small, to keep the proportion of points rejected small, as such points represent wasted computing effort. A second requirement is that it should be easy to generate points uniformly distributed under $e(x)$. The average number of points (X, Y) needed to produce one acceptable X will be called the *trials ratio*. Clearly, the trials ratio is always greater than or equal to unity. The closer the trials ratio is to unity, the more efficient is the resulting generator.

A neat way of constructing a suitable $e(x)$ is to take $e(x) = Kg(x)$, where $g(x)$ is the density of a distribution for which an easy way of generating variates already exists. It can be shown that if X is a variate from this density, then points of the form $(X, Y) = (X, KUg(X))$, where U is a RN(0,1) variable that is generated independently of X, will be uniformly distributed under the graph of $e(x)$. For a proof of this result see Devroye [1]. Usually, K is taken just large enough to ensure that $e(x) \geq f(x)$ (see assumption 1 above).

The trials ratio is precisely K. The method is thus dependent on being able to find a density $g(x)$ for which K can be kept small.

A nice example of this last technique is the third method given below for generating from the gamma distribution GAM$(1, b)$, due to Fishman [7]. The method is valid for the case $b > 1$. The distribution has density $f(x) = x^{b-1} \exp(-x)/\Gamma(b)$, $x > 0$. There is no general closed-form expression for the distribution function, so the inverse trans-

form method is not easy to implement for this distribution. Fishman's method takes an exponential envelope $e(x) = K\exp(-x/b)/b$. A little calculation shows that if we set

$$K = \frac{b^b \exp(1-b)}{\Gamma(b)}$$

then $e(x) \geq f(x)$ for all $x \geq 0$. For values of b close to unity, K also remains close to unity and K does not increase all that fast as b increases ($K = 1$ when $b = 1$, $K = 1.83$ when $b = 3$, and $K = 4.18$ when $b = 15$). The method is thus convenient when gamma variates are needed for b not too large, which is often the case. Figure 5.4 illustrates the envelope and the gamma distribution for the case $b = 2$, showing how the envelope fits quite neatly over the gamma density function, even though its shape is somewhat different.

5.2.3 Composition Method

Suppose that a given density f can be written down as the weighted sum of r other densities:

$$f(x) = \sum_{i=1}^{r} p_i f_i(x)$$

where the weights, p_i, satisfy $p_i > 0$ and $\sum_{i=1}^{r} p_i = 1$. The density f is then said to be a *mixture* or a *compound* density. An example of where such a mixture of distributions occurs is a queueing simulation where customer arrivals are composed of a mixture of different customer types, each with its own arrival pattern. If methods exist for the generation of variates from the component densities f_i, the following method can be used to generate from the mixture density.

Composition Method

```
Setup: Let F_j = Σ_{i=1}^{j} p_i for j = 1, 2, . . . , r
Let U = RN(0,1)
Set i = 1
While (F_i < U) {i = i + 1}
Return X = X_i, a variate drawn from the density f_i
```

The method simply selects the ith component distribution with probability p_i and then returns a variate from this distribution.

One use of the method is to split up the range of X into different intervals in such a way that it is easy to sample from each interval. Another common use is where the first component is made easy to generate from and also has a high probability of being chosen. Then the generator will probably generate from this easy distribution and will only occasionally have to resort to generation from one of the other distributions. Several fast normal variate generators have been constructed in this way [1].

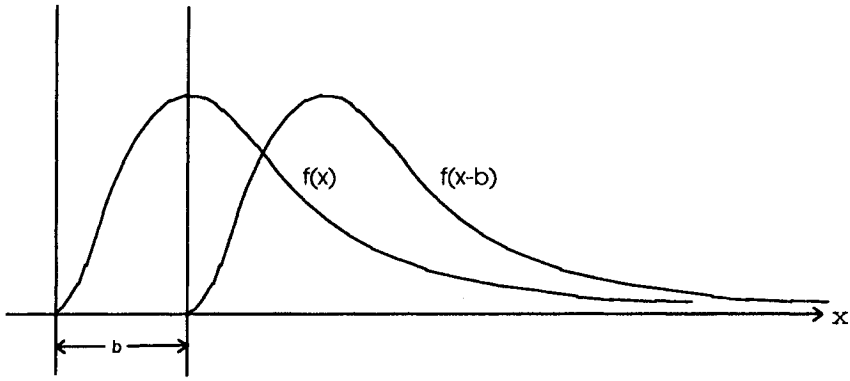

Figure 5.5 Translation of a distribution using a location parameter.

5.2.4 Translations and Other Simple Transforms

Although not strictly a method in its own right, often a random variable can be obtained by some elementary transformation of another. Many explicit generators fall into this category. For example, a *lognormal variable* is simply an exponentiated normal variable; a *chi-squared variable* with one degree of freedom (χ_1^2) is simply a standard normal variable that has been squared. Even more elementary than such transformations, but forming a broad class, are the *location-scale models*. If X is a continuously distributed random variable with density function $f(x)$, say, we can rescale and reposition the distribution by the linear transform

$$Y = aX + b$$

where a is a given *scale* constant (typically, $a > 0$) and b is a given *location* constant. The variable Y is thus a location-scale version of X and has density $g(y)$ given by

$$g(y) = a^{-1} f\left(\frac{y - b}{a}\right)$$

Note that if X has a restricted range, Y will be restricted, too. For example, if $X > 0$, then $Y > b$ (if $a > 0$). Figure 5.5 illustrates a random variate X that has been translated to a new location that is b units away from its original position.

Below we shall give distributions in their most commonly used form. Sometimes this form incorporates the location-scale formulation; an example is the normal distribution. Sometimes this form incorporates the scale parameter but not the location parameter; an example is the Weibull distribution. It is left to the reader to extend these forms to their full version if needed.

5.3 CONTINUOUS DISTRIBUTIONS

5.3.1 Inverse Transform by Numerical Solution of $F(X) = U$

In this section we list generators for a number of commonly occurring continuous distributions. If a generator is required for a distribution not listed and the inverse distribution is not expressible in simple terms, the inverse transform method can still be employed by solving $F(X) = U$ numerically for X. The following bisection method is one way of doing this.

Inverse Transform Method by Bisection

```
Setup: Let δ = 0.001, say
Let U = RN(0,1)
Let a = -1.0 While(F(a) > U) {a = 2a}
Let b = +1.0 While(F(b) < U) {b = 2b}
While(b - a > δ){
  X = (a + b)/2
  If(F(X) ≤ U) a = X
  Else b = X
  }
```

Notes:

1. The generator assumes that $F(x)$ can be calculated for all x.
2. The method first finds an interval (a, b) in which X lies and then checks the midpoint of the interval to see which side of this X lies on. This reduces the length of the interval of uncertainty for X by half. The process is repeated by considering the midpoint of this reduced interval, and so on, until the width of the interval of uncertainty for X is reduced to a prescribed, sufficiently small, value δ.
3. The initial interval (a, b) in which X lies is obtained by setting arbitrary initial values for a and b (i.e., $a = -1$ and $b = 1$). These values are then doubled repeatedly until $a \le X \le b$. This procedure can be replaced by a more efficient one if a convenient one exists. For example, if X is positive, the setup for a can become

```
Set a = 1.0 While(F(a) > U) {a = a/2}
```

For unimodal densities with known mode X_m, the following alternative is quicker. (See ref. 1 for details on how the method works.)

Inverse Transform Method by Newton–Raphson Iteration

```
Setup: δ = 0.001, Y_m = F(X_m)
Let U = RN(0,1) Set X = X_m, Y = Y_m, h = Y - U
While(|h| > δ){
  X = X - h/f(X)
  h = F(X) - U
  }
Return X
```

Notes:

1. Convergence is guaranteed for unimodal densities because $F(x)$ is convex for $x \in (-\infty, X_m)$, and concave for $x \in (X_m, \infty)$.
2. The tolerance criterion guarantees that the value of $F(X)$ for the X value returned will be within δ of U. However, it does not guarantee that X will be close to the exact solution of $F(X) = U$.

With both previous generators, if $F(x)$ is not known, it can be evaluated from the integral of the density $f(x)$:

$$F(x) = \int_{-\infty}^{x} f(u)\, du$$

using numerical quadrature. This is likely to be a slow procedure.

5.3.2 Specific Continuous Distributions

Uniform U(a, b), a < b

Density:

$$f(x) = \begin{cases} \dfrac{1}{b-a} & a < x < b \\ 0 & \text{otherwise} \end{cases}$$

Distribution Function:

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

Generator: Inverse transform method:

```
Let U = RN(0,1)
Return X = a + (b - a)U
```

Exponential EXP(a), a > 0

Density:

$$f(x) = \begin{cases} a^{-1} \exp\left(-\dfrac{x}{a}\right) & x > 0 \\ 0 & \text{otherwise} \end{cases}$$

Distribution Function:

$$F(x) = \begin{cases} 1 - \exp\left(-\dfrac{x}{a}\right) & x > 0 \\ 0 & \text{otherwise} \end{cases}$$

Generator: Inverse transform method:

```
Generate U = RN(0,1)
Return X = -a ln(1 - U)
```

Note: The parameter $a > 0$ is the mean of the distribution. The rate parameter is a^{-1}.

Weibull WEIB*(a, b), a, b > 0*

Density:

$$f(x) = \begin{cases} ba^{-b}x^{b-1} \exp\left[-\left(\dfrac{x}{a}\right)^b\right] & x > 0 \\ 0 & \text{otherwise} \end{cases}$$

Distribution Function:

$$F(x) = \begin{cases} 1 - \exp\left[-\left(\dfrac{x}{a}\right)^b\right] & x > 0 \\ 0 & \text{otherwise} \end{cases}$$

Generator: Inverse transform method:

```
Generate U = RN(0,1)
Return X = a[-ln(1 - U)]^1/b
```

Notes:

1. Some references replace $1 - U$ with U in the final formula for X, as this also has the $U(0, 1)$ distribution. However, this is not usually recommended; see ref. 3.
2. The parameters $a, b > 0$ are scale and shape parameters, respectively.
3. The special case $b = 1$ is the exponential distribution.
4. If X is a Weibull variate, X^b is an exponential variate with mean a^b. Conversely, if E has exponential distribution with mean a^b, then $E^{1/b}$ is Weibull with scale and shape parameters a and b, respectively.

Extreme Value EXTREME(μ, σ), $\sigma > 0$

Density:

$$f(x) = \sigma^{-1} \exp\left(-\frac{x-\mu}{\sigma}\right) \exp\left[-\exp\left(-\frac{x-\mu}{\sigma}\right)\right] \qquad -\infty < x < \infty$$

Distribution Function:

$$F(x) = \exp\left\{-\exp\left[\left(-\frac{x-\mu}{\sigma}\right)\right]\right\} \qquad -\infty < x < \infty$$

Generator: Inverse transform method:

```
Let U = RN(0,1)
Return X = -σ ln[-ln(U)] + μ
```

Note: The extreme value distribution is sometimes given in the form where X is the negative of that given here.

Gamma GAM(*a*,*b*), *a*,*b* > 0

Density:

$$f(x) = \begin{cases} \dfrac{(x/a)^{b-1}}{a\Gamma(b)} \exp\left(-\dfrac{x}{a}\right) & x > 0 \\ 0 & \text{otherwise} \end{cases}$$

where $\Gamma(b)$ is the *gamma function*:

$$\Gamma(b) = \int_0^\infty u^{b-1} e^{-u}\, du$$

Distribution Function: No simple closed form.

Here $a > 0$ and $b > 0$ are scale and shape parameters, respectively. The key parameter is the shape parameter, and no single method of generation is satisfactory for all values of b. The following methods cover different ranges of b.

Generator 1. Acceptance–rejection given by Ahrens and Dieter [8]; requires that $0 < b < 1$:

```
Setup: β = (e + b)/e where e = 2.71828 . . . is the base of the natural
logarithm.
While(True){
  Let U = RN(0,1), W = β U
  If (W < 1){
    Let Y = W^(1/b), V = RN(0,1)
    If (V ≤ e^(-Y)) Return X = aY
```

```
    }
  Else{
    Let Y = -ln[(β - W)/b], V = RN(0,1)
    If (V ≤ Y^(b-1)) Return X = aY
    }
}
```

Generator 2. Acceptance–rejection Cheng [9]; requires that $b > 1$:

```
Setup: α = (2b - 1)^(-1/2), β = b - ln 4, γ = b + α^(-1), δ = 1 + ln 4.5.
While(True){
  Let U1 = RN(0,1), U2 = RN(0,1)
  Let V = α ln[U1/(1 - U1)], Y = be^V, Z = U1^2 U2, W = β + γ V - Y
  If (W + δ - 4.5Z ≥ 0){
    Return X = aY
    }
  Else{
    If (W ≥ ln Z) Return X = aY
    }
}
```

Note: The trials ratio improves from $4/e \simeq 1.47$ to $(4/\pi)^{1/2} \simeq 1.13$ as b increases from 1 to ∞.

Generator 3. Acceptance–rejection Fishman [7]; requires that $b > 1$. The method is simple and is efficient for values of $b < 5$, say:

```
While(True){
  Let U1 = RN(0,1), U2 = RN(0,1), V1 = -ln U1, V2 = -ln U2
  If (V2 > (b - 1)(V1 - ln V1 - 1)){
    Return X = aV1
    }
}
```

Note: The trials ratio degrades from unity at $b = 1$, to 2.38 at $b = 5$, to 2.84 at $b = 7$.

k-Erlang ERL(*m*, *k*), $m > 0$, *k* a Positive Integer

Density: Same as that of GAM($m/k, k$) [i.e., GAM(a, b) with $a = m/k$ and $b = k$ (an integer)]

Distribution Function: No closed form except for the case $k = 1$.

Generator 1. If X is a k-Erlang variate with mean m, it is the same as the sum of k independent exponential variates each with mean m/k:

```
Let U1 = RN(0,1), U2 = RN(0,1), . . . , Uk = RN(0,1)
Return X = -(m/k) ln[(1 - U1)(1 - U2) . . . (1 - Uk)]
```

Generator 2. Generate X as a gamma variate with $a = m/k$ and $b = k$:

```
Return X = GAM(m/k, k)
```

Note: The first method is more efficient for small values of k ($k < 10$, say). For larger values of k, the second method is faster and not to subject to finite arithmetic error through repeated multiplication of quantities all less than unity as might occur with the first method.

Normal N(μ, σ^2), $\sigma > 0$

Density:

$$f(x) = (2\pi\sigma^2)^{-1/2} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right] \qquad -\infty < x < \infty$$

Here μ is the mean and σ^2 is the variance of the distribution.

Distribution Function: No closed-form expression.

Generator: Polar version of the Box–Muller [10]; special transform:

```
While(True){
  Generate U1, U2, RN(0,1) variates.
  Let V1 = 2U1 - 1, V2 = 2U2 - 1, W = V1^2 + V2^2
  If(W < 1){
    Let Y = [(-2 ln W)/ W]^1/2
    Return X1 = μ + σ V1Y and X2 = μ + σ V2Y
    }
}
```

Notes:

1. The method returns pairs of independent normal variates, each with mean μ and variance σ^2. If only one variate is needed each time, just X_1 can be returned; the method is then not as efficient, as X_2 is not used even though most of the work to get it is done.
2. There are many alternative methods. The one shown above is one of the simplest and is reasonably fast. The *original* Box–Muller method simply returns $X_1 = R\cos T$ and $X_2 = R\sin T$, where $R = [-2\ln(U_1)]$ and $T = 2\pi U_2$. This is elegantly simple but does require the relatively slow calculation of a sine and a cosine.
3. The standard normal distribution is where $\mu = 0$ and $\sigma^2 = 1$.

Chi-Squared with k Degrees of Freedom $\chi^2(k)$, k a Positive Integer

Density and Distribution Function: These are the same as those of GAM(2, $k/2$).

Generator 1: Use the relationship with the gamma distribution:

```
Return X = GAM(2, k/2)
```

Generator 2: When k is even, use the relationship with the k-Erlang distribution:

```
Return X = ERL(k, k/2)
```

Generator 3: When k is small and a fast normal variate generator is available, use the fact that $\chi^2(k)$ is the sum of k independent squared standard normal variates:

```
X = 0
For(i = 1 to k){X = X + [N(0,1)]^2}
Return X
```

t-Distribution *t*(ν), ν a Positive Integer

Density:

$$f(x) = \frac{\Gamma[(\nu+1)/2]}{\sqrt{\pi\nu}\,\Gamma(\nu/2)} \left(1 + \frac{x^2}{\nu}\right)^{-(\nu+1)/2} \qquad -\infty < x < \infty$$

This is called the *t-distribution with ν degrees of freedom.*

Generator: Use the fact that $X = t(\nu)$ has the same distribution as the ratio $Z/\sqrt{Y/\nu}$, where Z is a standard normal variate and Y is a $\chi^2(\nu)$ variate independent of Z:

```
Let Z = N(0,1), W = sqrt(χ^2(ν)/ν)
Return X = Z/W
```

Lognormal LN(μ, σ^2), $\sigma > 0$

Density:

$$f(x) = \frac{1}{x\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(\ln x - \mu)^2}{2\sigma^2}\right] \qquad x > 0$$

Distribution Function: No simple closed form.

Generator: Use the property that if Y is $N(\mu, \sigma^2)$, then $X = e^Y$ is LN(μ, σ^2):

```
Let Y = N(μ, σ^2)
Return X = e^Y
```

Note: If θ and τ^2 are the mean and variance of the lognormal, respectively, they are related to μ and σ^2 by the formulas

$$\mu = \ln \frac{\theta^2}{\sqrt{\theta^2 + \tau^2}} \quad \text{and} \quad \sigma^2 = \ln \frac{\theta^2 + \tau^2}{\theta^2}$$

So to generate lognormals with given mean and variance, these formulas should be used in computing the correct μ and σ^2 to use in the generator.

Beta BETA(*p*, *q*), *p*, *q* > 0

Density:

$$f(x) = \begin{cases} \dfrac{x^{p-1}(1-x)^{q-1}}{B(p,q)} & \text{if } 0 < x < 1 \\ 0 & \text{otherwise} \end{cases}$$

where $B(p,q)$ is the beta function:

$$B(p,q) = \int_0^1 z^{p-1}(1-z)^{q-1}\,dz$$

and $p, q > 0$ are shape parameters.

Distribution Function: No closed form in general.

Generator 1: If G_1 and G_2 are independent GAM(a,p) and GAM(a,q) gamma variates, then $X = G_1/(G_1 + G_2)$ is a BETA(p,q) variate:

```
Let G1 = GAM(a,p)
Let G2 = GAM(a,q)
Return X = G1/(G1 + G2)
```

Generator 2: For $p, q > 1$, Cheng [11] gives the following acceptance–rejection method with bounded trials ratio ($< 4/e \simeq 1.47$):

```
Setup: α = p + q, β = √((α − 2)/(2pq − α)), γ = p + β⁻¹.
Do{
  Let U1 = RN(0,1), U2 = RN(0,1)
  V = β ln[U1/(1− U1)], W = peV
}While(α ln[α/(q + W)] + γV − ln 4 < ln[U1²U2])
Return X = W/(q + W)
```

Generator 3: For $p, q < 1$, Jöhnk [12] gives the following acceptance–rejection method:

```
Do{
  Let U = RN(0,1), V = RN(0,1)
  Y = U^1/p, Z = V^1/q
}While(Y + Z > 1)
Return X = Y/(Y + Z)
```

Inverse Gaussian IG(μ, λ), λ > 0, μ > 0

Density:

$$f(x) = \sqrt{\frac{\lambda}{2\pi x^3}} \exp\left[-\frac{\lambda(x-\mu)^2}{2\mu^2 x}\right] \qquad x > 0$$

Distribution Function:

$$F(x) = \Phi\left[\sqrt{\frac{\lambda}{x}}\left(\frac{x}{\mu} - 1\right)\right] + e^{2\lambda/\mu}\Phi\left[-\sqrt{\frac{\lambda}{x}}\left(\frac{x}{\mu} + 1\right)\right]$$

where Φ is the distribution function of $N(0, 1)$, the standard normal distribution.

Generator: Use the many-to-one transformation of Michael et al. [13]:

```
Setup: Let φ = λ/μ
Let Z = N(0,1), Y = Z²
Let T = 1 + (Y − √(4φY + Y²))/(2φ)
Let U = RN(0,1)
If(U ≤ 1/(1 + T)) Return X = μT
Else Return X = μ/T
```

Notes:

1. The method is based on the fact that

$$Y = \frac{\lambda(X - \mu)^2}{\mu^2 X} \tag{5}$$

   is a chi-squared variable with one degree of freedom when X is an $IG(\mu, \lambda)$ variable. If Y is generated as a chi-squared variate and equation (5) solved for X, this gives two possible values for X: $X = \mu T$ or $X = \mu/T$. The randomized choice ensures that the resulting overall distribution of X is IG(μ, λ).

2. The parameterization using $\phi = \lambda/\mu$ as shape parameter rather than λ is often preferable, as the shape, when measured by ϕ, is invariant as μ varies.

Pearson Type V PT*5(a, b), a, b* > *0*

Density:

$$f(x) = \begin{cases} \dfrac{a^b x^{-b-1}}{\Gamma(b)} \exp\left(-\dfrac{a}{x}\right) & x > 0 \\ 0 & \text{otherwise} \end{cases}$$

where a is a scale parameter and b a shape parameter.

Generator: X = PT5(a, b) is precisely the same as the reciprocal of a gamma variate with scale $1/a$ and shape b:

```
Return X = 1/GAM(1/a, b)
```

Note: The mean is $a/(b - 1)$, which exists only when $b > 1$. So b must be set greater than unity if the variates generated are to have finite mean.

Pearson Type VI PT6(*a, p, q*), *a, p, q* > 0

Density:

$$f(x) = \begin{cases} \dfrac{(x/a)^{p-1}}{aB(p,q)[1+(x/a)]^{p+q}} & x > 0 \\ 0 & \text{otherwise} \end{cases}$$

Scale parameter $a > 0$ and shape parameters $p, q > 0$.

Generator 1: Use the fact that $X = \text{PT6}(1, p, q)$ has precisely the same distribution as $Y/(1-Y)$, where $Y = \text{BETA}(p, q)$. This is known as a *beta variate of the second kind* [14]:

```
Let Y = BETA (p, q)
Return X = aY/(1 - Y)
```

Generator 2: Use the fact that if $Y_1 = \text{GAM}(a, p)$ and $y_2 = \text{GAM}(a, q)$, with Y_1, Y_2 independent, then Y_1/Y_2 has the same distribution as $\text{PT6}(a, p, q)$:

```
Let Y1 = GAM (a, p), Y2 = GAM (a, q)
Return X = Y1/Y2
```

Note: The mean is $ap/(q-1)$, which exists only when $q > 1$.

F-Distribution F(ν_1, ν_2), ν_1, ν_2 *Positive Integers*

Density and Distribution Function: This is a distribution frequently used in statistical tests. It is known as the *F-distribution with* ν_1 *and* ν_2 *degrees of freedom.* It is a special case of PT6, being the same distribution as $\text{PT6}(a, p, q)$ with $a = \nu_2/\nu_1$, $p = \nu_1/2$, $q = \nu_2/2$.

Generator: Use the fact that $X = F(\nu_1, \nu_2)$ has the same distribution as the ratio of two independent χ^2 variates each scaled by its own mean:

```
Let Y1 = χ²(ν1)/ν1, Y2 = χ²(ν2)/ν2
Return X = Y1/Y2
```

Triangular TRI(*a, b, c*), *a* < *b* < *c*

Density:

$$f(x) = \begin{cases} \dfrac{2(x-a)}{(b-a)(c-a)} & a \le x \le b \\ \dfrac{2(c-x)}{(c-a)(c-b)} & b \le x \le c \\ 0 & \text{otherwise} \end{cases}$$

The shape of the density gives the distribution its name (see Figure 5.6).

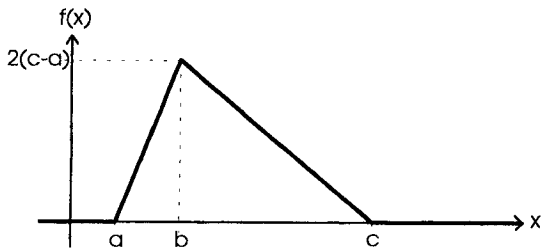

Figure 5.6 Density function for the triangular distribution.

Distribution Function:

$$F(x) = \begin{cases} 0 & x < a \\ \dfrac{(x-a)^2}{(b-a)(c-a)} & a \le x \le b \\ 1 - \dfrac{(c-x)^2}{(c-a)(c-b)} & b \le x \le c \\ 1 & x > c \end{cases}$$

Generator: $F(x)$ can be inverted. The inverse transform method gives

```
Setup: β = (b − a)/(c − a)
Let U = RN(0,1)
If (U < β) T = √(βU)
Else T = 1 − √((1 − β)(1 − U))
Return X = a + (c − a)T
```

Cauchy CAUCHY*(a, b), b > 0*

Density:

$$f(x) = \frac{b}{\pi[b^2 + (x-a)^2]} \qquad -\infty < x < \infty$$

Distribution Function:

$$F(x) = \frac{1}{2} + \frac{1}{\pi} \arctan \frac{x-a}{b} \qquad -\infty < x < \infty$$

Generator: The distribution function can be inverted. The inverse transform method gives

```
Let U = RN(0,1)
Return X = a + b tan πU
```

Notes:

1. The distribution does not have a finite mean, even though it is symmetric about a. The distribution has long tails.
2. The ratio of two independent standard normal variates has the CAUCHY(0,1) distribution.

von Mises VM(κ), $\kappa > 0$

Density:

$$f(x) = \frac{\exp(\kappa \cos x)}{2\pi I_0(\kappa)} \qquad -\pi < x < \pi$$

where I_0 is the modified Bessel function of the first kind of order zero (see Abramowitz and Stegun [15] for details of the Bessel function).

Generator: Use the Best and Fisher [16] rejection method:

```
Setup r = 1 + √(1 + 4κ²) ρ = (r − √(2r))/(2κ) s = (1 + ρ²)/(2ρ)
Do{
  U = U(−1, 1), V = U(−1, 1)
  Z = cos πU
  W = (1 + sZ)/(s + Z)
  Y = κ(s − W)
  }While (W(2 − W) < V AND ln(W/V) + 1 < W)
Return X = sgn U/ cos W
```

Note: X is the random angle of the direction on a circle.

Empirical Distribution EDF($x_1, x_2, \ldots, x_n$)

Distribution: Suppose that $x_1, x_2, \ldots, x_n$ is a random sample of size n. A piecewise linear distribution function can be constructed corresponding to this sample as follows. First sort the x's into ascending order: $x_{(1)} \le x_{(2)} \le \cdots \le x_{(n)}$. The smoothed empirical distribution function is then

$$F(x) = \begin{cases} 0 & x < x_{(1)} \\ \dfrac{i-1}{n-1} + \dfrac{x - x_{(i)}}{(n-1)(x_{(i+1)} - x_{(i)})} & x_{(i)} \le x \le x_{(i+1)}, \quad i = 1, \ldots, n-1 \\ 1 & x > x_{(n)} \end{cases}$$

Generator: The smoothed distribution function can be inverted. Using the inverse transform method gives

```
Let U = RN(0,1), A = (n − 1)U, i = trunc (A) + 1
Return X = x(i) + (A − i + 1)(x(i+1) − x(i))
```

Here trunc(A) denotes integer part of A.

5.4 DISCRETE DISTRIBUTIONS

The general methods of Section 5.2 are in principle available for constructing discrete variate generators. However, the special characteristics of discrete variables imply that modifications of the general techniques are usually necessary. Two general methods are especially useful: the *look-up table method* and the *alias method.* In what follows trunc(x) and frac(x) mean the integer and fractional part of x.

5.4.1 Look-up Tables

The basic technique was described in Section 5.2.1. If the table is large, the look-up procedure can be slow, with the ith value requiring i steps to find. A simple alternative is to use a binary search to home in more rapidly on the value to be returned. We assume that the distribution has the form

$$p_i = \Pr(X = x_i) \qquad i = 1, 2, \ldots, n$$

$$P_i = \sum_{j=1}^{i} p_j = \Pr(X \le x_i) \qquad i = 1, 2, \ldots, n \tag{6}$$

For distributions where the number of points extends to infinity, an appropriate cutoff for the distribution must be made. For example, set n so that

$$P_n > 1 - \delta = 0.99999, \text{say}$$

The precise value of the cutoff has to be selected carefully. For example, if the simulation involves tail probabilities that are important, δ must be chosen small enough to enable these probabilities to be estimated sufficiently accurately.

Look-up by Binary Search

```
Let U = RN(0,1), A = 0, B = n
While(A < B - 1){
  i = trunc[(A + B)/2]
  If(U > P_i) A = i
  Else B = i
  }
Return X = X_i
```

An alternative is to make a table of the starting points approximately every (n/m)th entry, in the same way that the letters of the alphabet form convenient starting points for search in a dictionary.

Look-up by Indexed Search. Set up the index table:

```
i = 0
For (j = 0 to m - 1){
  While(P_i < j/m){i = i + 1}
  Q_j = i
  }
```

```
Let U = RN(0,1), j = trunc(mU), i = Q_j
While(U ≥ P_i){i = i + 1}
Return X = X_i
```

5.4.2 Alias Method

The alias method, proposed by Walker [17], gives a way of returning a value from a table without searching. For a more recent reference, clear explanation is given in ref. 3. Assume that the distribution still has the form (6). We select a value of j in the range(1, 2, ..., n) all with equal probability. We then return one of two values: either the selected j which we return with probability q_j, or a precomputed alternative called the *alias* of j: $a(j)$, which is returned with probability $1 - q_j$. The $a(j)$ and q_j are selected to ensure that overall, the probability of returning each j is p_j, as required. A setup that does not need linked lists is given below. For a discussion of how it works and faster versions of the setup using linked lists, see ref. 2. (The definition and implementation of linked lists is discussed in ref. 3.)

Setup for Alias Method

```
For( j = 1 to n){
  Let q_j = 1, r_j = p_j, t_j = True
  }
For(k = 1 to n - 1){
  r_min = 2.0
  For(l = 1 to n){
    If(r_l < r_min AND t_l = True){r_min = r_l, i = l}
    }
  r_max = -1.0
  For(l = 1 to n){
    If(r_l > r_max AND t_l = True){r_max = r_l, j = l}
    }
If(r_i = r_j) Exit
t_i = False, a(i) = j, q_i = nr_i, r_j = r_j - (1 - q_i)/n
}
```

Alias Method

```
Let U = RN(0,1), j = 1 + trunc(nU), p = frac(nU)
If(p ≤ q_j) Return X = x_j
Else Return X = x_a(j)
```

5.4.3 Empirical Distribution

Empirical distributions can be handled as a special case of a discrete distribution. Suppose that $x_1, x_2, \ldots, x_n$ is a random sample of size n. Assume that each value has equal probability of occurring:

$$\Pr(X = x_i) = \frac{1}{n} \qquad i = 1, 2, \ldots, n$$

Then variates can be generated from this discrete distribution using

```
Let U = RN(0,1), i = trunc(nU) + 1
Return X = x_i
```

Sampling Without Replacement; Permutations

The following modification (see ref. 18) allows sampling of $m \leq n$ items from the random sample $x_1, x_2, \ldots, x_n$ of size n, *without replacement*. To avoid losing the original order, assume that the x's are held in the array a_i, $i = 1, 2, \ldots, n$:

```
For( j = 1 to m){
  Let U = RN(0,1), i = trunc[(n - j + 1)U] + j
  a = a_j, a_j = a_i, a_i = a
  }
Return a_1, a_2, . . . , a_m
```

The routine progressively swaps each entry with one drawn from the remaining list. At the end of the call the entries in the first m positions (i.e., $a_1, a_2, \ldots, a_m$) contain the elements sampled without replacement. The advantage of this algorithm is that repeated calls to it give further samples. The special case $m = n$ generates a random permutation of the initial sample.

5.4.4 Specific Discrete Distributions

The general look-up table or alias method is available for all the distributions listed below. So these two general methods will not be stated explicitly in individual cases unless there is a special interest in doing so.

Bernoulli BER(*p*), $0 < p < 1$

Probability Mass Function:

$$X = \begin{cases} 1 & \text{with probability } p \\ 0 & \text{with probability } 1 - p \end{cases}$$

This represents a trial with probability p of success and probability $(1 - p)$ of failure.

Generator: Elementary look-up table:

```
Let U = RN(0,1)
If (U ≤ p) Return X = 1
Else Return X = 0
```

Binomial BIN(*n*, *p*), *n a Positive Integer*, $0 < p < 1$

Probability Mass Function:

$$p(x) = \begin{cases} \binom{n}{x} p^x (1-p)^{n-x} & x = 0, 1, \ldots, n \\ 0 & \text{otherwise} \end{cases}$$

where

$$\binom{n}{x} = \frac{n!}{x!\,(n-x)!}$$

is the *binomial coefficient.*

Generator: Use the special property that $X = \text{BIN}(n, p)$ if it is the sum of n independent BER(p) variables:

```
X = 0
For(i = 1 to n){
  Let B = BER(p), X = X + B
  }
Return X
```

Note: The generation time increases linearly with n. One of the general methods will be preferable for n large (> 20, say).

Geometric GEOM(*p*), $0 < p < 1$

Probability Mass Function:

$$p(x) = \begin{cases} p(1-p)^x & x = 0, 1, \ldots \\ 0 & \text{otherwise} \end{cases}$$

This gives the number of failures before the first success is encountered in a sequence of independent Bernoulli trials.

Generator: The distribution function is invertible:

```
Setup: a = 1/ln(1 - p)
Let U = RN(0,1)
Return X = trunc(a ln U)
```

Negative Binomial NEGBIN(*n*, *p*), *n an Integer*, $0 < p < 1$

Probability Mass Function:

$$p(x) = \begin{cases} \binom{n+x-1}{x} p^n (1-p)^x & x = 0, 1, \ldots \\ 0 & \text{otherwise} \end{cases}$$

This gives the number of failures before the nth success is encountered in a sequence of independent Bernoulli trials.

Generator: By definition this variable is the sum of n independent GEOM(p) variables:

```
X = 0
For(i = 1 to n){
  Let Y = GEOM(p), X = X + Y
  }
Return X
```

Note: The generation time increases linearly with n. One of the general methods will be preferable for n large (> 10, say).

Hypergeometric HYP(*a, b*), *a, b Positive Integers*

Probability Mass Function:

$$p(x) = \begin{cases} \binom{a}{x}\binom{b}{n-x} \Big/ \binom{a+b}{n} & x = 0, 1, \ldots, n \\ 0 & \text{otherwise} \end{cases}$$

Consider a population where a individuals have a particular characteristic and b do not. If $n \le a+b$ individuals are selected without replacement, the number X with the characteristic has the given hypergeometric distribution.

Generator: Fishman [19] gives the following inverse transform method:

```
Setup: α = p0 = [b!(a + b − n)!]/[(b − n)!(a + b)!]
Let A = α, B = α, X = 0
Let U = RN(0,1)
While(U > A){
  X = X + 1, B = B(a − X)(n − X)/[(X + 1)(b − n + X + 1), A = A + B
  }
Return X
```

Poisson POIS(λ), λ > 0

Probability Mass Function:

$$p(x) = \begin{cases} \dfrac{e^{-\lambda}\lambda^x}{x!} & x = 0, 1, \ldots \\ 0 & \text{otherwise} \end{cases}$$

If events occur randomly in time at rate r and X is the number of events that occurs in a time period t, then X = POIS(rt).

Generator 1: The direct method is to count the number of events in an appropriate time period, as indicated above:

```
Setup: a = e^{-λ}
Let p = 1, X = -1
While(p > a){
  Let U = RN(0,1), p = pU, X = X + 1
  }
Return X
```

Generator 2: The time taken by generator 1 to return a variate increases approximately as λ. So for large λ (> 30, say) it is slow. The following acceptance–rejection technique given by Atkinson [20] is then preferable:

```
Setup: a = π√(λ/3), b = a/λ, c = 0.767 - 3.36/λ, d = ln c - ln b - λ
Do{
  Do{
    U = RN(0,1), Y = [a - ln((1 - U)/U)]/b
    } While(Y ≤ -1/2)
  Let X = trunc(Y + 1/2), V = RN(0,1)
  } While(a - bY + ln[V/(1 + e^{a-bY})^2] > d + X ln λ - ln X!)
Return X
```

Generator 3: An alternative to generator 2 uses the fact that for large λ, the distribution of $\lambda^{-1/2}(X - \lambda)$ tends to that of the standard normal. For large λ (> 20, say) we then have the following:

```
Setup: a = λ^{1/2}
Let Z = N(0,1)
Let X = max[0, trunc(0.5 + λ + aZ)]
Return X
```

5.5 MULTIVARIATE DISTRIBUTIONS

5.5.1 General Methods

The generation of multivariate distributions is not nearly as well developed as that of the univariate case. The key requirement in the generation of multivariate samples is the need to ensure an appropriate correlation structure among the components of the multivariate vector. Often the correlation arises because the variates are the output of a stochastic model with a certain structure; for example, the variates may describe the state of some stochastic process at given points in time. Correct sampling from the underlying form of the multivariate distribution can then be obtained simply by ensuring that the variates are generated according to the definition of the process. It is often not only more natural but considerably easier to generate multivariate samples in this way rather than to attempt to derive the distribution itself and then generate variates directly from it. There is one general approach that is sometimes useful, however [6].

Conditional Sampling

Let $\mathbf{X} = (X_1, X_2, \ldots, X_n)^T$ be a random vector with joint distribution function $F(x_1, x_2, \ldots, x_n)$. Suppose that the conditional distribution of X_j given that $X_i = x_i$, for $i =$

1, 2, ..., $j-1$, is known, for each j. Then the vector $\mathbf{X}$ can be built up one component at a time, with each component obtained by sampling from a *univariate* distribution:

```
Generate x_1 from the distribution F_1(x)
Generate x_2 from the distribution F_2(x|X_1 = x_1)
Generate x_3 from the distribution F_3(x|X_1 = x_1, X_2 = x_2)
...
Generate x_n from the distribution F_n(x|X_1 = x_1, X_2 = x_2, ..., X_{n-1} = x_{n-1})
Return X = (x_1, x_2, ..., x_n)^T
```

The usefulness of this method is dependent on the availability of the conditional distributions, and on the ease of sampling from them.

5.5.2 Special Distributions

Multivariate Normal MVN($\boldsymbol{\mu}, \boldsymbol{\Sigma}$)

Density:

$$f(x) = (2\pi|\boldsymbol{\Sigma}|)^{-n/2} \exp[-\tfrac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})]$$
$$-\infty < x_i < \infty, \quad i = 1, 2, \ldots, n$$

where $\boldsymbol{\mu}$ is a $n \times 1$ vector and $\boldsymbol{\Sigma}$ is a $n \times n$ positive-definite symmetric matrix.

Generator: The simplest method uses a linear transformation of a set of independent $N(0, 1)$ variates: $\mathbf{Z} = (Z_1, Z_2, \ldots, Z_n)^{\mathrm{T}}$ is transformed into $\mathbf{X} = \mathbf{LZ} + \boldsymbol{\mu}$, where $\mathbf{L}$ is a lower triangular matrix satisfying $\boldsymbol{\Sigma} = \mathbf{LL}^{\mathrm{T}}$. $\mathbf{L}$ can be obtained by the standard Choleski decomposition [19].

```
Setup: L (see below)
Let a = √σ_11
For(i = 1 to n){L_i1 = σ_i1/ a}
Let i = 2
While(True){
  S = 0
  For( j = 1 to i - 1) S = S + L_ij^2
  L_ij = (σ_ij - S)^1/2
  If(i = n) Return L
  i = i + 1
  For( j = 2 to i - 1){
    S = 0
    For(k = 1 to j - 1) S = S + L_ik L_jk
    L_ij = (σ_ij - S)/ L_jj
    }
}
```

MVN Generator:

```
For(i = 1 to n) Z_i = N(0,1)
For(i = 1 to n){
  X_i = μ_i
  For( j = 1 to i) X_i = X_i + L_ij Z_j
  }
Return X = (X_1, X_2, . . . , X_n)
```

Uniform Distribution on the n-Dimensional Sphere. If the components of MVN($\mathbf{0}, \mathbf{I}$) are treated as a direction vector in n-dimensional Euclidean space, all directions are equally likely. Rescaling the vector to unit length therefore gives a point uniformly distributed on the unit n-dimensional sphere.

Generator:

```
S = 0
For(i = 1 to n){
  Z_i = N(0,1), S = S + Z_i^2
  }
S = √S
For(i = 1 to n){
  X_i = Z_i/S
  }
Return X = (X_1, X_2, . . . , X_n)
```

Order Statistics. The order statistics of a random sample $X_1, X_2, \ldots, X_n$ of size n are just the individual variates arranged by value in ascending order:

$$X_{(1)} \le X_{(2)} \le \cdots \le X_{(n)}$$

This can be done by generating the sample and then reordering. The fastest sorting routines are $O(n \ln n)$, and the sorting dominates once n is large.

If the X's can be generated by the inverse transform $X = F^{-1}(U)$, the sample can be generated in order from the order statistics of a uniform sample

$$U_{(1)} \le U_{(2)} \le \cdots \le U_{(n)} \tag{7}$$

This uses the fact that (1) the largest uniform order statistic $U_{(n)}$ has an invertible distribution function, and (2) $U_{(1)}, U_{(2)}, \ldots, U_{(i)}$ are the order statistics of a sample of size i drawn from the uniform distribution $U(0, U_{(i+1)})$.

```
Let U = RN(0,1), U_(n) = U^(1/n)
For(i = n − 1 down to 1){
  Let U = RN(0,1)
  U_(i) = U_(i+1) U^(1/i)
  }
```

An alternative way of generating (7) is given in (2):

```
Let E_1 = EXP(1), S_1 = E_1
For(i = 2 to n + 1){
  Let E_i = EXP(1), S_i = S_{i-1} + E_i
  }
For(i = 1 to n) Let U_(i) = S_i/S_{n+1}
```

Dirichlet

Density:

$$f(x) = Cx_1^{p_1-1}x_2^{p_2-1}\cdots x_k^{p_k-1}(1-x_1-x_2-\cdots-x_k)^{p_{k+1}-1}$$

$$\text{all } p_i > 0, \quad x_1, x_2, \ldots, x_k > 0, \quad \sum_{i=1}^{k} x_i \le 1$$

where

$$\frac{1}{C} = \frac{\Gamma(p_1)\Gamma(p_2)\cdots\Gamma(p_k)}{\Gamma(p_1+p_2+\cdots+p_k)} \int_0^1 u^{p_1+p_2+\cdots+p_k-1}(1-u)^{p_{k+1}-1}\,du$$

Generator: Use the fact that a set of gamma variates scaled by their sum, so that they sum to unity, has the Dirichlet distribution:

```
Y = 0
For(i = 1 to k + 1){
Let G_i = GAM(1, p_i), Y = Y + G_i
}
For(i = 1 to k + 1){
Let X_i = G_i/Y
}
Return (X_1, X_2, . . . , X_{k+1})
```

Note: The $X_1, X_2, \ldots, X_k$ have the distribution given above. Including X_{k+1} gives the sum $\Sigma_{i=1}^{k+1} X_i = 1$. The distribution is therefore useful in representing random proportions that have to sum to unity.

5.6 STOCHASTIC PROCESSES

5.6.1 Point Processes

A sequence of points $t_0 = 0, t_1, t_2, \ldots$ in time is known as a *point process*. The times between occurrences $x_i = t_i - t_{i-1}$ are usually random. Examples are where the t_i are arrival times of customers and the x_i are interarrival times, or where the t_i are moments of breakdowns and the x_i are lifetimes.

Poisson Process. When the x_i are independent EXP(1/λ) variables, the t_i sequence is known as a *Poisson process with rate* λ. To generate the next time point, assuming that t_{i-1} has already been generated:

```
Let U = RN(0,1)
Return t_i = t_{i-1} - λ^{-1} ln U
```

Nonstationary Poisson Process. Suppose that the Poisson process has $\lambda = \lambda(t)$; that is, the rate varies with time. One way to generate such a nonstationary process is via an analog of the inverse transform technique. Define the cumulative rate

$$\Lambda(t) = \int_0^t \lambda(u)\, du$$

and suppose this is invertible with inverse $\Lambda^{-1}(\cdot)$.

To generate the next time point, assume that s_{i-1}, the previous point of a unit rate Poisson process, has already been generated. Then the next point of the nonstationary process is given by

```
Let U = RN(0,1), s_i = s_{i-1} - ln U
Return t_i = Λ^{-1}(s_i)
```

An alternative is to use an analog of the acceptance–rejection method proposed by Lewis and Shedler [21], called *thinning*. Suppose that $\lambda_M = \max_t \lambda(t)$. Then, assuming that t_{i-1} has already been generated, the next point of the nonstationary process is given by

```
Let t = t_{i-1}
Do{
  U = RN(0,1)  t = t - λ_M^{-1} ln U   V = RN(0,1)
  }While(V > λ(t)/λ_M)
Return t_i = t
```

Markov Process. The simplest Markov process is the *discrete-time* Markov chain. Here time is advanced one unit at a time: $t = 0, 1, 2, \ldots$. At each time point the system is assumed to be in one of n states: $X = 1, 2, \ldots, n$, say. Given that $X_t = i$, the next state X_{t+1} is selected according to the discrete probability distribution

$$\Pr(X_{t+1} = j | X_t = i) = p_{ij} \qquad j = 1, 2, \ldots, n$$

Continuous-time Markov chains are best simulated slightly differently. Assume that the system has just entered state i at time t_k. Then the next change of state occurs at $t_{k+1} = t_k + \text{EXP}(1/\lambda_i)$. The state entered is j with probability p_{ij} ($j = 1, 2, \ldots, n$).

5.6.2 Time-Series Models and Gaussian Processes

A stochastic process $X(t)$ all of whose joint distributions are multivariate normal (i.e., $X_{t_1}, X_{t_2}, \ldots, X_{t_r}$ is multivariate normal for any given set of times $t_1, t_2, \ldots, t_r$) is said to be a *Gaussian process*.

Many time-series models use normal perturbations (usually called *innovations*) and are Gaussian.

Moving Average. A moving-average process X_t is defined by

$$X_t = Z_t + \beta_1 Z_{t-1} + \cdots + \beta_q Z_{t-q} \qquad t = 1, 2, 3, \ldots$$

where the Z's are all independent $N(0, \sigma^2)$ normal variates and the β's are user-prescribed coefficients. The X's can be generated directly from this definition.

Autoregressive Process. An autoregressive process X_t is defined by

$$X_t = \alpha_1 X_{t-1} + \cdots + \alpha_p X_{t-p} + Z_t \qquad t = 1, 2, 3, \ldots$$

where the Z's are all independent $N(0, \sigma^2)$ normal variates and the α's are user prescribed coefficients. The X's can again be generated directly from this definition, but in this case the initial values $X_0, X_{-1}, \ldots, X_{1-p}$ need to be obtained. Now

$$(X_0, X_{-1}, \ldots, X_{1-p}) = \text{MVN}(\mathbf{0}, \boldsymbol{\Sigma})$$

where $\boldsymbol{\Sigma}$ satisfies

$$\boldsymbol{\Sigma} = \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^{\mathrm{T}} + \mathbf{B} \tag{8}$$

with

$$\mathbf{A} = \begin{bmatrix} \alpha_1 & \alpha_2 \cdots \alpha_{p-1} & \alpha_p \\ 1 & 0 \cdots 0 & 0 \\ 0 & 1 \cdots 0 & 0 \\ \cdot & \cdots & \cdot \\ 0 & 0 \cdots 1 & 0 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} \sigma^2 & 0 \cdots 0 \\ 0 & 0 \cdots 0 \\ 0 & \cdots 0 \\ \cdots & \cdots \\ 0 & 0000 \end{bmatrix}$$

Once $\boldsymbol{\Sigma}$ has been found from (8), $(X_0, X_{-1}, \ldots, X_{1-p})$ can be generated using $\text{MVN}(0, \boldsymbol{\Sigma})$; see ref. 22. An alternative is to set $X_0 = X_{-1} = \cdots = X_{1-p} = 0$ and run the sequence for a settling in period before collecting the results. For more complicated Gaussian models, see ref. 2.

REFERENCES

1. Devroye, L. (1986). *Non-uniform Random Variate Generation*, Springer-Verlag, New York.
2. Ripley, B. D. (1987). *Stochastic Simulation*, Wiley, New York.
3. Law, A. M., and W. D. Kelton (1991). *Simulation Modeling and Analysis*, 2nd ed., McGraw-Hill, New York.
4. Banks, J., J. S. Carson II, and B. L. Nelson (1996). *Discrete-Event Simulation*, 2nd ed., Prentice Hall, Upper Saddle River, N.J.
5. Fishman, G. S. (1996). *Monte Carlo: Concepts, Algorithms, and Applications*, Springer-Verlag, New York.
6. Johnson, M. E. (1987). *Multivariate Statistical Simulation*, Wiley, New York.
7. Fishman, G. S. (1976). Sampling from the gamma distribution on a computer, *Communications of the ACM*, Vol. 19, pp. 407–409.
8. Ahrens, J. H., and U. Dieter (1974). Computer methods for sampling from gamma, beta, Poisson and binomial distributions, *Computing*, Vol. 12, pp. 223–246.
9. Cheng, R. C. H. (1977). The generation of gamma variables with non-integral shape parameter, *Applied Statistics*, Vol. 26, pp. 71–75.

10. Box, G. E. P., and M. E. Muller (1958). A note on the generation of random normal deviates, *Annals of Mathematical Statistics*, Vol. 29, pp. 610–611.
11. Cheng, R. C. H. (1978). Generating beta variables with nonintegral shape parameters, *Communications of the ACM*, Vol. 21, pp. 317–322.
12. Jöhnk, M. D. (1964). Generation of beta distribution and gamma distribution random variates (in German), *Metrika*, Vol. 8, pp. 5–15.
13. Michael, J. R., W. R. Schucany, and R. W. Haas (1976). Generating random variates using transformations with multiple roots, *The American Statistician*, Vol. 30, pp. 88–90.
14. Stuart, A., and J. K. Ord (1987). *Kendall's Advanced Theory of Statistics*, 5th ed., Vol. 1, Griffin, London.
15. Abramowitz, M., and I. A. Stegun (1965). *Handbook of Mathematical Functions*, Dover, New York.
16. Best, D. J., and N. I. Fisher (1979). Efficient simulation of the von Mises distribution, *Applied Statistics*, Vol. 28, pp. 152–157.
17. Walker, A. J. (1977). An efficient method for generating discrete random variables with general distributions, *ACM Transactions on Mathematical Software*, Vol. 3, pp. 253–256.
18. Moses, L. E., and R. V. Oakford (1963). *Tables of Random Permutations*, Stanford University Press, Stanford, Calif.
19. Fishman, G. S. (1978). *Principles of Discrete Event Simulation*, Wiley, New York.
20. Atkinson, A. C. (1979). The computer generation of Poisson random variables, *Applied Statistics*, Vol. 28, pp. 29–35.
21. Lewis, P. A. W., and G. S. Shedler (1979). Simulation of non-homogeneous Poisson processes by thinning, *Naval Research Logistics Quarterly*, Vol. 26, pp. 403–413.
22. Gardner, G., A. C. Harvey, and G. D. A. Phillips (1979). Algorithm AS-154: an algorithm for the exact maximum likelihood estimation of autoregressive–moving average models by means of Kalman filtering, *Applied Statistics*, Vol. 29, pp. 311–322.