ATIN

11

Simulation

11.1 INTRODUCTION

Simulation in probability theory means numerical construction of samples of a random process. It may be difficult to understand the general behaviour of a system involving stochastic variables in certain situations, because of system complexities. Formulation of analytic models often involve a large number of assumptions. In such a situation, numerical construction of samples of the system is often helpful. One would be familiar with methods of numerical differentiation and integration and of numerical solution of differential equation. Simulation allows numerical solution of problems in probability theory (or in stochastic experiments) and also in other areas where there are significant uncertainties.

Monte Carlo methods are used for simulating the behaviour of physical and or mathematical systems, specially when analytical solutions are difficult to obtain. These methods are non-deterministic (or stochastic). A notable early use of such a method was by Enrico Fermi (1901–1954) in 1930. However, after the world War II, when electronic computers came to be built these methods have been widely used. Random numbers are required, while in practice, pseudorandom numbers are used. Because of the randomness and repetitive nature involved in the process it is so called (linking it to the famous casinos-gambling houses in Monaco, randomness and the repetitive nature of a process being similar to the activities in a casino).

Applications of Monte Carlo methods are quite varied: these include physics, computer science, engineering, environmental sciences, finance etc., and systems with uncertainties in addition to pure mathematical systems not involving any uncertainty.

Consider that X has unifrom distribution on (0, 1). A random value of X is known as a *random number*. In practice, values of X can be *deterministically* generated and the random numbers so generated are known as *pseudorandom numbers:* such a pseudorandom number contains a bounded (fixed) number of digits, implying that continuous uniform distribution is approximated by a discrete one. Pseudorandom numbers are made use of in simulation studies. Monte Carlo methods need large number of pseudorandom numbers and computers are used in their generation. We consider here how such numbers are generated.

11.1.1 Generation of Pseudorandom Numbers

The simplest procedure for generation of pseudorandom numbers is through a *recursive* formula (involving x_{n-1} and x_n) of the form

$$x_n = ax_{n-1} \pmod m$$
$$n = 1, 2, \ldots, \tag{1.1}$$

where a and m are given positive integers. The above implies that when ax_{n-1} is divided by m, the remainder is x_n. Thus x_n can take values 0, 1, 2,..., $m-1$ and the pseudorandom number x_n/m is taken as an approximate value of the r.v X. distributed uniformly on (0, 1). Thus these are generated through a deterministic method.

The above method is known as *multiplicative congruence method.* The *factor* a, the *module* m and (the initial integral value) the *seed* x_0 have to be suitably chosen. As x_n assumes values 0, 1, 2,..., $m-1$, it is clear that after some finite number n (which could be at most m) there will be repetitions with a period of length $m-1$. Thus these values of a and m are to be suitably chosen, so as to see that repetitions do not occur soon enough; also a and m are to be relatively prime, also m is to be sufficiently large.

In practice, $m = 2^{31} - 1$ and $a = 7^5$ are used (m being a prime number, the period then will be $m-1$), as these are found to be satisfactory.

A more general form of (1.1) considered is

$$x_n = (ax_{n-1} + c) \bmod m \tag{1.2}$$

This is called *mixed congruence method.*

Here we shall limit ourselves to multiplicative congruence method (given in (1.1)) which is mostly used. Knuth (1981), Marsasglia (1972, 1983) and so on discuss this method.

In addition to the congruence (method) generators, two other random generators namely, *shift-register generator* and *lagged-Fibonacci generator* are the most common classes of generators of random numbers. For example, see Marsaglia and Zaman (1994).

Several random number tables are available (e.g. The Rand corporation, 1955). We shall here assume that pseudorandom numbers are available; without going into the theoretical considerations such as test of randomness and other statistical tests (chisquare test, serial test, run test etc.-see, Robert & Casella, 2004, Chang *et al.*, 2007). We shall assume that the pseudorandom numbers generated correspond to the values of a sequence of *i.i.d.* uniform random variable (0, 1).

Before proceeding to the use of random numbers to simulate stochastic experiments we consider here, how the random numbers could be used in computation of definite integrals.

For a detailed account of simulation, see for example, Rubenstein (1981), Ripley (1986), Fishman (1995) and Ross (2002, 2003).

11.2 EVALUATION OF INTEGRALS USING RANDOM NUMBERS

Monte Carlo Approach

First, we consider a purely mathematical problem without involving any uncertainty. Let us consider evaluation of an integral in (0, 1). Let $f(x)$ be a function of x and suppose that we are interested in the computation of the integral

$$I = \int_0^1 f(x)\,dx. \tag{2.1}$$

Suppose that the random variable U is uniformly distributed over (0, 1), then the above integral is equivalent to the expectation

$$I = E[f(U)] \tag{2.2}$$

Now, if $u_1, u_2, \ldots$, represent a random sample of U, then, for each u_i, we get $f(u_i)$. Then, using the law of large numbers, one gets

$$\lim_{n\to\infty} \sum_{k=1}^{n} \frac{f(u_i)}{n} = E[f(U)] \tag{2.3}$$

By taking a large number of random numbers u_i, $i = 1, 2, \ldots$, one can thus approximate the value of $I = E[f(U)]$

For evaluating an integral of the type

$$I = \int_a^b f(x)\,dx$$

one can make the substitution $y = \dfrac{(x-a)}{b-a}$

then
$$I = \int_0^1 f([b-a]y + a)(b-a)\,dy$$

$$= \int_0^1 h(y)\,dy \tag{2.4}$$

where $h(y) = (b-a)[f(a + (b-a)y)]$, and proceed as discussed above for an integral in (0, 1).

For evaluating an infinite integral

$$I = \int_0^\infty f(x)\,dx$$

one can make the substitution

$$y = \frac{1}{x+1}, \quad dx = -\left(\frac{1}{y^2}\right)dy$$

and evaluate

$$I = \int_1^0 f\left(\frac{1}{y} - 1\right)\left(-\frac{dy}{y^2}\right) \tag{2.5}$$

$$= \int_0^1 \frac{f\left(\frac{1}{y} - 1\right)}{y^2}\,dy$$

by proceeding as described above.

11.2.1 Evaluation of Multiple Integrals

Suppose that one has to evaluate the multiple integral

$$I = \int_0^1 \int_0^1 \cdots \int_0^1 f(x_1, x_2, \ldots, x_k)\,dx_1\,dx_2 \ldots dx_k$$

then the same Monte Carlo approach can be used to find I which can be approximated by

$$E[f(U_1, \ldots, U_k)]$$

where $U_1, \ldots, U_k$ are *i.i.d.* uniform random variables over (0, 1). If $(u_1^i, u_2^i, \ldots, u_k^i)\, i = 1, 2, \ldots, n$, are the n samples of $U_1, \ldots, U_k$ then I can be approximated, by

$$\sum_{k=1}^{n} \frac{f(u_1^i, u_2^i, \ldots, u_k^i)}{n} \tag{2.6}$$

Utility of Monte Carlo method in evaluation of multiple integrals is quite pronounced.

Note. Proceeding as indicated and using random numbers from any source, integrals can be evaluated.

11.3 GENERATION OF CONTINUOUS RANDOM VARIABLES

Introduction

Simulation of a random variable implies generation or construction of samples of the random variable. The method or procedure described is *algorithmic*. Now an *algorithm* is a precise set of rules which enable one to transform specified inputs into specified outputs in a *finite number of steps. Euclid's algorithm,* discovered over 2250 years ago, (which has stood the test of time) refers to the set of rules for finding the greatest common factor (output) of two positive numbers (input)–without factoring them: it has been called, by Knuth (1981), as *'the granddaddy of all algorithms'*.

The basic ingredient in simulation of a *r.v.* is a source of (pseudo) random numbers (a sample of i.i.d. uniform distribution on (0, 1)). It is necessary to have a statistically reliable set of random numbers. Now most computers and simulation languages have suitable random number generator (see, for example, Law & Kelton 1991).

Programming environments, in general, also include a function called *rand*, which generates random numbers (samples of uniform distribution on (0, 1)). A random number appears each time it is invoked. This function could be used, whenever a random number is needed (see, for example, D. Medhi & Ramasamy, 2007, p. 711). Now we come to specific method or algorithm for simulation of a *r.v.*

11.3.1 Inverse Transform Method (Algorithm)

This method is based on the following theorem.

Theorem 3.1. Let $F(x)$ be the *d.f.* of the continuous random variable X, *i.e.* $F(x) = P\{X \le x\}$, and let U have uniform distribution on (0, 1).

Then
$$X = F^{-1}(U) \tag{3.1}$$

has the *d.f.* $F(x)$.

The relation (3.1) above implies that $F(x) = u$.

Proof: Assuming $X = F^{-1}(U)$, one gets as *d.f.* of X,

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

$$= \Pr\{U \le F(x)\} = F(x),$$

Since U is uniformly distributed in (0, 1).

Thus X has the *d.f.* $F(x)$.

Example 3(a). Suppose that we are to generate a sample of the r.v. X uniformly distributed in (a, b). The *pdf* of X is given by

$$f(x) = \frac{1}{b-a}, \quad a \le x \le b$$

$$= 0, \quad \text{otherwise}$$

$$F(x) = P\{X = x\} = \int_a^x \frac{dx}{b-a}$$

$$= \frac{x-a}{b-a}, \quad a \le x \le b$$

$$= 1, \quad x > b.$$

Now, let $x = F^{-1}(u)$, then

$$u = F(x) = \frac{x-a}{(b-a)},$$

or, $$x = a + (b-a)\,u. \tag{3.2}$$

By getting a set of random numbers $u_1, \ldots, u_n$ one gets a sample

$$x_i = a + (b-a)u_i, \quad i = 1, 2, \ldots, n.$$

of the random variable X uniformly distributed on (a, b).

Example 3(b). Suppose that we are to generate a sample of the r.v. X whose *d.f.* is given by

$$F(x) = x^m, \quad 0 < x < 1.$$

Let $x = F^{-1}(u)$, then

$$u = F(x) = x^m$$

or, $$x = u^{\frac{1}{m}}$$

By using a set of random numbers $u_1, \ldots, u_n$ one gets a sample

$$x_i = (u_i)^{1/m}, \quad i = 1, 2, \ldots, n$$

of the *r.v.* X.

Example 3(c). Sample of exponential distribution X with mean 1.

The *d.f.* of X is given by

$$F(x) = 1 - e^{-x}$$

Letting $x = F^{-1}(u)$, we get

$$u = F(x) = 1 - e^{-x}$$

or $$e^{-x} = 1 - u$$

or $$x = -\log(1-u). \tag{3.3}$$

From a set of random numbers $u_1, \ldots, u_n$ one gets a sample of X as

$$x_i = -\log(1-u_i), \quad i = 1, 2, \ldots, n$$

Now, if U is uniformly distributed on (0, 1) then it follows that $V = 1 - U$ is also uniformly distributed on (0, 1). Thus instead of taking $1 - u_i$ one can use u_i, that is

$$x_i = -\log u_i, \quad i = 1, 2, \ldots, n$$

which is a sample of X.

Note: Here log has base *e, i.e.*, $\log \equiv ln$.

Example 3(d). Sample of exponential distribution Y with mean λ.

The *d.f.* of Y is given by

$$F(y) = 1 - e^{-\lambda y}, \quad 0 < y < \infty \tag{3.4}$$

Letting $y = F^{-1}(u)$ or

$$u = F(y) = 1 - e^{-\lambda y}$$

we get $$y = -\frac{1}{\lambda}\log(1-u)$$

As before, from a set of random numbers $u_i, \ldots, x_n$, we get

$$y_i = -\frac{1}{\lambda}\log u_i, \quad i = 1, \ldots, n \tag{3.5}$$

as a sample of the exponential distribution Y with mean λ.

Example 3(e). Suppose we are to generate a sample of gamma distribution X ($g(k, \lambda)$) with parameters λ, k ($k > 0$) having *p.d.f.*

$$f(x) = \frac{\lambda^k x^{k-1} \exp(-\lambda x)}{\Gamma(k)}, \quad x > 0, \tag{3.6}$$

where k is an integer.

Its *d.f.*

$$F(x) = \int_0^x f(y)\,dy \tag{3.7}$$

cannot be expressed in a convenient form; thus using $x = F^{-1}(u)$ is not possible in this case.

In this case (when k is an integer), we can use the fact that the sum of k *i.i.d.* exponential distributions, each with mean λ has a gamma distribution (**Theorem 1.10**).

Thus we can generate samples of k number of *i.i.d.* exponential variables (as described above)

$$u_i^1, u_i^2, \ldots, u_i^k \text{ ; then}$$

$$y_i^j = \frac{1}{-\lambda}\log u_i^j,\ i = 1, \ldots, n,\ j = 1, \ldots, k$$

u's being k sets of random numbers. Thus

$$x_i = \sum_{j=1}^{k} y_i^j = \left[-\frac{1}{\lambda}\log u_i^1\right] + \cdots + \left[-\frac{1}{\lambda}\log u_i^k\right]$$

$$= -\frac{1}{\lambda}\log\left(u_i^1 u_i^2 \ldots u_i^k\right)$$

$$= -\frac{1}{\lambda}\log\left(\prod_{j=1}^{k} u_i^j\right),\ i = 1, \ldots, n \tag{3.8}$$

is a sample of the gamma variable X.

The case when k (> 0) is not an integer would be considered later.

Example 3(f). Suppose that we are to generate a sample of normal distribution $N(0.1)$, *i.e.* with mean 0 and s.d. 1. Special methods are available.

One such method uses central limit theorem (CLT). Let X_i have *i.i.d.* uniform distribution in (0, 1). Then by CLT

$$\frac{\sum_{i=1}^{n} X_i - \frac{1}{2}\cdot n}{\sqrt{\left(n\cdot\frac{1}{12}\right)}} \tag{3.9}$$

tends to $N(0,1)$ for large n (X_i having mean $\frac{1}{2}$ and variance $\frac{1}{12}$). If we take $n = 12$, and x_i are random numbers, then

$$y_i = \sum_{i=1}^{12} x_i - 6$$

gives a sample of $N(0,1)$. By taking k different sets of 12 random numbers, one can get k samples of $N(0,1)$. The idea of taking $n = 12$ here is not only to make the above expression simple, but $n = 12$ is considered as good enough for most purposes. Here, since $0 \le \sum x_i \le 12$, we have $-6 \le y_i \le 6$ and this implies that extreme values of $N(0,1)$ are not that well represented. We shall discuss another method below.

Another special method is described below.

Let X and Y be two independent $N(0, 1)$ random variables. Then the joint probability density function of (X, Y) is given by

$$f(x, y) = \left[\frac{1}{\sqrt{(2\pi)}}e^{-\frac{x^2}{2}}\right]\left[\frac{1}{\sqrt{(2\pi)}}e^{-\frac{y^2}{2}}\right]$$

$$= \frac{1}{2\pi}e^{-(x^2+y^2)/2} \quad -\infty \le x, y \le \infty \tag{3.10}$$

since X and Y are independent

If the vector (X, Y) is denoted by polar coordinates R and θ, then

$$R^2 = X^2 + Y^2$$

$$\tan\theta = Y/X.$$

To obtain the joint *p.d.f.* (say $f(t, \theta)$) of R^2 and θ, we make the transformation

$$t = x^2 + y^2,\ \tan\theta = y/x$$

or

$$x = \sqrt{t}\cos\theta,\quad y = \sqrt{t}\sin\theta \tag{3.11}$$

$$0 < t < \infty,\quad 0 < \theta < 2\pi$$

The variables x, y in $f(x, y)$ can be changed to (t, θ) using the change of variable formula in double integral, that is,

$$\iint f(x, y)\,dx\,dy = \iint f\left[\sqrt{t}\cos\theta, \sqrt{t}\sin\theta\right]\left[\frac{\partial(x, y)}{\partial(t,\theta)}\right]dt\,d\theta \tag{3.12}$$

where

$$\frac{\partial(x, y)}{\partial(t, \theta)} = \begin{bmatrix} \frac{\partial x}{\partial t} & \frac{\partial x}{\partial \theta} \\ \frac{\partial y}{\partial t} & \frac{\partial y}{\partial \theta} \end{bmatrix} = \frac{1}{2}$$

The *p.d.f.* of (R^2, θ) which is the integrand on the right hand side of (3.12) becomes

$$f(t, \theta) = \frac{1}{2\pi}e^{-\frac{t}{2}}\cdot\frac{1}{2}$$

$$= \left[\frac{1}{2}e^{-\frac{t}{2}}\right]\left[\frac{1}{2\pi}\right] \tag{3.13}$$

Now $\frac{1}{2}e^{-\frac{t}{2}}$ is the p.d.f. of exponential *r.v.* with mean 2 and $\frac{1}{2\pi}$ is the *p.d.f.* of uniform r.v. in $(0, 2\pi)$. That is, (the *p.d.f.* of) R^2 is exponential with mean 2 and (the *p.d.f.* of) θ is uniform in $(0, 2\pi)$. Moreover, R^2 and θ are independent. If we take a sample u_i, from exponential *r.v.* with mean 2 and another sample v_i, from uniform random variable in $(0, 2\pi)$, then a sample of x_i, y_i of X, Y ($X = R\cos\theta$, $y = R\sin\theta$) is given by

$$x_i = \left[\sqrt{-2\log u_i}\right]\left[\cos(2\pi v_i)\right]$$

$$y_i = \left[\sqrt{-2\log u_i}\right]\left[\sin(2\pi v_i)\right] \tag{3.14}$$

Similarly, by taking samples u_i, v_i, $i = 1, 2, \ldots, n$, we can get n samples of the *r.v*'s X and Y.

Remark:

The relations (3.14) are known as *Box - Muller transformations*. One difficulty here is that one has to obtain sines and cosines of angles of *random magnitudes*.

Note 1. For a sample of the normal variable Z with mean μ and *s.d.* σ, one may consider $U = (Z - \mu)/\sigma$ which is $N(0, 1)$. If u_i is a sample of the *r.v.* U (which one can get as above) then

$$z_i = \sigma u_i + \mu \tag{3.15}$$

is a sample of z.

Note 2. In Examples 3(a) - 3(d), explicit expressions of *d.f* are available from which the inverses could be obtained. In **Examples 3(e), 3(f)**, where such explicit expressions cannot be had, specialised methods, as indicated above could be used. In general, when explicit inverse function cannot be found, a method is to use the graph of *d.f.* of the random variable to find the sample value of the variable from a given random number.

Note 3. Apart from using *inverse transform method* (which can also be used to simulate a discrete random variable—as is shown in § 11.4), there are other methods such as *Acceptance-Rejection Method and Composite Method* (which uses both Inverse transform method and acceptance-rejection methods) is simulate a random variable.

We discuss this method below.

11.3.2 Rejection Method (Acceptance-Rejection Method or Algorithm)

This method involves simulation of a *r.v.* Y having *p.d.f.* $g(x)$ through simulation of another continuous *r.v.* X having *p.d.f.* $f(x)$ (which could be done with some ease by inverse-transform method). If V is the simulated value of X, then this generated value in accepted as a value of Y with probability proportional to $g(V)/f(V)$.

Let c be a constant such that

$$\frac{g(x)}{f(x)} \le c \quad \text{for all } x \tag{3.16}$$

The algorithm is as given below:

1. Generate a simulated value of the *r.v.* X having *p.d.f.* $f(x)$ - let it be V.
2. Generate a random number U.
3. If

$$U \le \frac{g(V)}{cf(V)}, \tag{3.17}$$

where c is such that $\frac{g(x)}{cf(x)} \le 1$ for all x, then accept V as the simulated value of Y; otherwise go back to step **1** and proceed as before

When we choose c as the maximum of $g(x)/f(x)$ then $g(x)/cf(x)$ will be less than or equal to 1 for all x.

This will, incidentally, eliminate all avoidable (unnecessary) step as U (≤ 1) is a random number. We shall call X an *auxiliary variable*.

By this procedure, we go back to step **1** until we generate a pair (V, U) such that $U \le g(V)/cf(V)$ when we finally accept the value V for Y. Hence the algorithm is named as such.

The justification is given in the following Theorem.

Theorem 3.2. The *r.v.* Y, whose simulated value is obtained as above has the *p.d.f.* $g(x)$.

Proof: We are to prove that

$$P\{Y \le x\} = \int_{-\infty}^{x} g(y)\,dy \tag{3.18}$$

Let A be the event that the acceptance of simulated value of Y occurs at step 3 (at, say, n^{th} iteration or time). Thus $A \equiv U \le g(V)/cf(V)$. Denote $K = P(A)$. We have

$$P\{Y \le x\} = P\{Y \le x \mid A\}$$

$$= \frac{P\{Y \le x, A\}}{P(A)}$$

$$= \frac{1}{K} P\{Y \le x, A\}$$

$$= \frac{1}{K} P\{Y \le x, U \le g(V)/cf(V)\}$$

$$= \frac{1}{K} P\{Y \le x, U \le g(V)/cf(V) \mid V = y\} f(y)\,dy$$

$$= \frac{1}{cK} \int_{-\infty}^{x} \{g(y)/f(y)\} f(y)\,dy$$

$$= \frac{1}{cK} \int_{-\infty}^{x} g(y)\,dy \tag{3.18a}$$

Now as $x \to \infty$, L.H.S. of (3.18a) = 1 and the R.H.S. $= \frac{1}{cK}$,

which implies that

$$K = \frac{1}{c}$$

Thus $P\{Y \le x\} = \int_{-\infty}^{x} g(y)\,dy$ (which is (3.18)).

We also have the following:

Theorem 3.3. Each iteration process to arrive at an accepted value is independent of other iteration processes (to arrive at another accepted value, and the total number N of iterations needed to arrive at an accepted value is a geometric *r.v.* with mean c.

In other words, an iteration may result in acceptance of a value with probability $\frac{1}{c}$ and

$$P\{N=n\}=\left(1-\frac{1}{c}\right)^{n-1}\cdot\frac{1}{c},\quad n=1,2,\ldots,\text{ and } E(N)=c.$$

For proof, see Ross, Kohlas, Law & Kelton.

In **Example 3(e)** we considered generation of a sample of gamma random variable $g(k, \lambda)$ with parameters λ, k ($\lambda > 0$, $k > 0$) having *p.d.f.*

$$f(x)=\frac{\lambda^k x^{k-1}\exp(-\lambda x)}{\Gamma(k)},\quad x>0 \tag{3.19}$$

where k is an integer. Below we can consider a case where k is not an integer. For simplicity, we take $k=n+\frac{1}{2}$, where n is an integer (here $n = 2$), and $\Gamma(k)$ is easily calculable without referring to any Table.

Example 3(g). To generate a sample of gamma *r.v.* with parameters $\lambda=1$, $k=\frac{5}{2}$, that is, $g\left(1,\frac{5}{2}\right)$, having *p.d.f.*

$$g(x)=\frac{x^{\frac{3}{2}}\exp(-x)}{\Gamma\left(\frac{5}{2}\right)},\quad x>0 \tag{3.20}$$

$$\left(\Gamma\left(\frac{5}{2}\right)=\frac{3}{2}\cdot\frac{1}{2}\cdot\Gamma\left(\frac{1}{2}\right)=\frac{3}{4}\sqrt{\pi}\right).$$

The mean of the *r.v.* is $\frac{5}{2}$.

As an auxiliary variable, consider an exponential *r.v.* having the same mean $\left(\frac{5}{2}\right)$, that is, having *p.d.f.*

$$f(x)=\frac{2}{5}\exp\left(-\frac{2x}{5}\right),\quad x>0$$

In order to find the smallest constant c such that $\frac{g(x)}{f(x)}\le c$ we are first to obtain the maximum c of

$$\frac{g(x)}{f(x)}=A\cdot x^{\frac{3}{2}}e^{-\frac{3x}{5}},$$

where $A=\left[\frac{5}{2}\right]\Big/\Gamma\left(\frac{5}{2}\right)=\frac{10}{3\sqrt{\pi}}=1.880$

The maximum, if any, will occur for the value of x for which $\frac{d}{dx}\left(\frac{g(x)}{f(x)}\right)=0$.

This happens, when

$$A\cdot x^{\frac{1}{2}}\cdot e^{-\frac{3x}{5}}\left[\frac{3}{2}-\frac{3}{5}x\right]=0$$

that is, $x=\frac{5}{2}$. (It coincides with the mean of both the distributions).

Thus, the average number of iterations needed is

$$c=A\cdot\left(\frac{5}{2}\right)^{\frac{3}{2}}\cdot e^{-\frac{3}{2}}=\frac{10}{3\sqrt{\pi}}\cdot\left(\frac{5}{2}\right)^{\frac{3}{2}}e^{-\frac{3}{2}}=1.66 \tag{3.20a}$$

and the probability of acceptance at an iteration is $\frac{1}{c}=0.60$. Thus the algorithm will be:

1. Generate a simulated value of the exponential *r.v.* X having mean $\lambda=\frac{5}{2}$. (See **Example 3(d)**).

   If U is a random number then a simulated value of X is

   $$U_1=-\frac{1}{\lambda}\log U=-\frac{2}{5}\log U.$$

2. Generate another random number U_2
3. Test whether

$$U_2\le\frac{g(U_1)}{cf(U_1)}=\frac{U_1^{\frac{3}{2}}\exp(-3U_1/5)}{\left[\left(\frac{5}{2}\right)^{\frac{3}{2}}\exp(-3/2)\right]} \tag{3.21}$$

If this holds, then accept U_1 as the generated value of the distribution with *p.d.f.* $g(x)$. Otherwise start again from step **1.**

Remark:

1. The above method may be generalised for any gamma distribution with $\lambda = 1$ and $k=n+\frac{1}{2}$ where n is a positive integer. It can be seen that the maximum of the ratio $\frac{g(x)}{f(x)}$ will be at $x=n+\frac{1}{2}$, that is,

at the mean of the gamma distribution having *p.d.f.* $f(x)$ and at the mean of the auxiliary exponential random variable having *p.d.f.* $g(x)$.

In this case, It may be verified that

$$A = \frac{1}{\Gamma\left(\frac{2n+1}{2}\right)} \cdot \frac{2n+1}{2} = \frac{2n+1}{2} \cdot \frac{2^n}{\sqrt{\pi} \cdot 1 \cdot 3 \cdot 5 \ldots (2n-1)} \tag{3.22}$$

and
$$c = A, B \tag{3.23}$$

where
$$B = \left(\frac{2n+1}{2}\right)^{\left(\frac{2n-1}{2}\right)} \exp\left(-\frac{2n-1}{2}\right). \tag{3.24}$$

and
$$\frac{g(x)}{cf(x)} = \frac{x^{\frac{(2n-1)}{2}} \exp\left(-\frac{2n-1}{2n+1}x\right)}{B} \tag{3.25}$$

For given integral n, the value of A, B, c can be calculated.

For simulation of gamma *r.v.* with parameters $\left(1, \frac{3}{2}\right)$ with mean $\frac{3}{2}$ (as Y) and exponential *r.v.* with the same mean as the auxiliary variable X, putting $n = 1$ in (3.22) - (3.24), one gets

$$c = \frac{3}{\sqrt{\pi}}\left(\frac{3}{2}\right)^{\frac{1}{2}} e^{-\frac{1}{2}} = 1.26 \tag{3.26}$$

and also the expression corresponding to (3.25)

For simulation of gamma *r.v.* with parameters $\left(1, \frac{5}{2}\right)$ (as Y) and exponential *r.v.* with the same mean as X, putting $n = 2$, in (3.22)-(3.24), one gets

$$c = \frac{10}{3\sqrt{\pi}}\left(\frac{5}{2}\right)^{\frac{3}{2}} e^{-\frac{3}{2}} = 1.66$$

(as obtained in (3.20a)), and also the expression corresponding to (3.25).

It can be seen that as to why one should take, as the auxiliary variable, an exponential *r.v.* having with the same mean.

2. We have discussed the case for gamma *r.v.* when $k = n + \frac{1}{2}$. In such a case the computation can be handled even with a simple hand calculator. For the general case, $k > 0$ if the same method is tried–one may have to refer to Tables for values of $\Gamma(k)$ and handle more numerical computations (see Remark: 3 below).

3. Now consider the general case–generation of gamma variable $g(k, \lambda)$ having *p.d.f.* as given in (3.19) where λ, k are any positive numbers. If $X = g(k, 1)$, then $X' = \lambda X$ would enable us to generate $g(k, \lambda)$.

Now, $g(1, \lambda)$ is exponential with parameter λ. Two cases $0 < k < 1$ and $k > 1$ are to be considered. Various algorithms for these cases have been given (For details, see, for example, Law & Kelton, 1991).

4. Both the inversion and acceptance-rejection methods may be combined; the inter-connected method is called *composition method.* This method is particularly of interest when the density $g(x)$ to be simulated can be represented in the form $g(x) = p_1 f_1(x) + p_2 f_2(x)$, $p_1 + p_2 = 1$, p_1 is much smaller than p_2 and $f_i(x)$ can be easily simulated.

5. As we have seen that for generating sample of $g(k, \lambda)$ where k is a positive *integer*, we can use the property that $g(k, \lambda)$ is the sum of k independent exponential distributions each with parameter λ.

There are many relationships between random variables that could be used for generation of some particular random variables.

11.4 SIMULATION OF DISCRETE RANDOM VARIATES: INVERSE-TRANSFORM METHOD

The inverse-transform method could be used also in the case when the *r.v.* X is discrete.

Suppose that the distribution of X has the *p.m.f.*

$$P\{X = x_i\} = p_i, \quad i = 0, 1, 2, \ldots, \sum_i p_i = 1. \tag{4.1}$$

($x_0, x_1, \ldots$ are so ordered that $x_0 < x_1 < x_2 < \ldots$) and *d.f.*

$$F(x_k) = P(X \le x_k) = \sum_{i=0}^{k} p_i \tag{4.2}$$

Then the algorithm for generating X is as follows:

1. Generate a sample of uniform distribution in (0, 1), *i.e.* take a random number-call it U.
2. If $U < p_0$, set $X = x_0$ and stop.
   If $U \ge p_0$ but less than $p_0 + p_1$ set $X = x_1$ and stop. (4.3)
   If $U \ge p_0 + p_1$ but less than $p_0 + p_1 + p_2$ set $X = x_2$ and stop.
   ...
   and so on
   In other words, we shall have $X = x_i$ if U, the random number taken is such that

$$F(x_{i-1}) \le U < F(x_i), \quad i = 1, 2, \ldots \tag{4.4}$$

That is, to find the smallest nonnegative integer I such that $U \le F(x_i)$, then $X = I$.

Example 4. Suppose that we wish to simulate a discrete *r.v.* X such that

$$p_1 = 0.25, \; p_2 = 0.35, \; p_3 = 0.30, \; p_4 = 0.10$$

(where $p_i = P\{X = i\}$, $i = 1, 2, 3, 4$).

Its *d.f.* is given by

$$F(1) = P\{X \le 1\} = 0.25$$

$$F(2) = P\{X \le 2\} = 0.25 + 0.35 = 0.60$$

$$F(3) = P\{X \le 3\} = 0.25 + 0.35 + 0.30 = 0.90$$

$$F(4) = P\{X \le 4\} = 0.25 + 0.35 + 0.30 + 0.10 = 1$$

Now take a random number, say, U and if $U = 0.35$, then set $X = 2$ (because $U > 0.25$ but < 0.60). $I = 2$ is the smallest nonnegative integer such that $U \le F(2)$.

Note. 1. The p_i's may be from a specific discrete distribution or from an *empirical data set.*

2. The p_i's may be put into decreasing order to achieve the result with a smaller number of comparisons.

3. The above transform method is also applicable when the range of values of X is infinite. Special alternative methods are advanced when the range is finite (Law & Kelton, § 8.4.3). (Finite range implies $p_i = 0$ for all $i \ge n+1$).

11.4.1 Generation of a Bernoulli Random Variable

Suppose that X is a Bernoulli *r.v.* with parameter p, having *p.m.f.*

$$P\{X=1\} = p = 1 - P\{X=0\}$$

The algorithm is as follows:

1. Generate a random number U
2. If $U < 1 - p$, set $X = 0$ (4.5)

Now $U < 1 - p$ implies that $p < 1 - U$; since $1 - U$ is also a random number, it suffices if $1 - U$ is taken as U, that is, if $p < U$, set $X = 0$, otherwise set $X = 1$.

11.4.2 Generation of a Binomial Random Variable

A binomial *r.v.* $b(n, p)$ with parameters n, p, is the sum of n independent Bernoulli *r.v.s.* and thus can be generated by generating n Bernoulli *r.v.s.*, this is, n random numbers would be needed.

For large n, alternative methods have been put forward [Ahrens & Dieter (1974), Kachitvichyanukul & Schmeiser (1988), Ross (2003)].

11.4.3 Generation of a Geometric Random Variable

Suppose that X is a geometric *r.v.* with parameter p having *p.m.f.*

$$P\{X=k\} = pq^{k-1}, \quad k = 1, 2, \ldots, \; q = 1 - p.$$

We have

$$\sum_{k=1}^{j-1} P\{X=k\} = 1 - P\{X > j-1\}$$

$$= 1 - q^{j-1}, \quad j \ge 1$$

Accept $X = j$, using (4.4), that is if U' is a random number and if

$$1 - q^{j-1} \le U' < 1 - q^{j}$$

or $$q^{j} < U \le q^{j-1} \tag{4.6}$$

since $U = 1 - U'$ is also a random number when U' is a random number. The algorithm is as follows:

1. Generate a random number U
2. If $U < q$, accept $X = 0$
3. If $q^j < U \le q^{j-1}$, accept $X = j$, $\quad j = 1, 2, \ldots$

Alternative method using the following relationship between geometric *r.v.* and Bernoulli *r.v.s.* is as follows:

Suppose that $Y_1, Y_2, \ldots$ is a sequence of independent Bernoulli *r.v.s* with parameter p (probability of getting '1'). Let

$$P\{X = j\} = \min\{j, Y_j = 1\}, \quad j = 1, 2, \ldots \tag{4.7}$$

Then $P\{X = j\} = (1 - p)^{j-1} p$. Thus X is a geometric *r.v.* with mean $\frac{1}{p}$.

Note. An alternative method of generating a geometric *r.v.* is by using the relationship between geometric and exponential distributions (See § 1.3.6)

11.4.4 Generation of a Poisson Random Variable

Suppose that X is a Poisson *r.v.* with parameter (mean) λ having *p.m.f.*

$$p_j = P\{X = j\} = \frac{e^{-\lambda}\lambda^j}{j!}, \quad j = 0, 1, 2, \ldots$$

and *d.f.* $F(j) = P\{X \le j\}$. We have the recurrence relation

$$p_{j+1} = \frac{\lambda}{j+1} p_j, \quad j = 0, 1, \ldots \tag{4.8}$$

and $$F(j+1) = F(j) + \frac{\lambda}{j+1} p_j, \quad j = 0, 1, 2, \ldots \tag{4.9}$$

$$\left(p_0 = e^{-\lambda}\right).$$

The algorithm is as follows:

1. Generate a random number U
2. If $U < e^{-\lambda}$ ($\equiv p_0 = F(0)$), accept $X = 0$
3. If $U \ge e^{-\lambda}$ but less than $F(1) \equiv p_0 + p_1 = p_0(1 + \lambda)$, accept $X = 1$, that is, if $F(0) \le U < F(1)$, accept $X = 1$.
   If U is such that

$$F(j-1) \le U \le F(j),$$

accept $X = j$, $j = 1, 2, 3, \ldots$
[$F(j)$ can be calculated from $F(j-1)$, $j = 1, 2, \ldots$ using (4.9)]

Note 1. An alternative method to generate a Poisson *r.v.* with mean λ is to used the following relationship between Poisson and exponential distributions.

Let $Y_1, Y_2, \ldots$ be a sequence of *i.i.d.* exponential *r.v.s.* with mean $\frac{1}{\lambda}$, then

$$X = \max\left\{j : \sum_{i=1}^{j} Y_i \le 1\right\} \tag{4.10}$$

is a Poisson *r.v.* with mean λ.

Here $\sum_{i=1}^{j} Y_i$ can be generated using j random numbers

$$U_1, U_2, ..., U_j$$

and

$$\sum_{i=1}^{j} Y_i = \sum_{i=1}^{j} -\frac{1}{\lambda} \log U_i.$$

2. There are various relationships between random variables; when it exists, such a relationship between two *r.v.s* could be utilised to generate a *r.v.* which can be generated easily using the inverse-transform method or otherwise.

11.4.5 Generation of a Discrete Random Variable; Acceptance-Rejection Method

As in the case of continuous random variables, acceptance-rejection technique could be used to generate a discrete *r.v.*

Suppose that a method is available for generating a *r.v.* X having *p.m.f.*

$$P\{X = j\} = p_j, \quad j = 0, 1, 2, ...$$

Suppose that we are to generate another discrete *r.v.* Y having *p.m.f.*

$$P\{Y = j\} = q_j, \quad j = 0, 1, 2, ...$$

Let c be a constant such that

$$\frac{q_j}{p_j} \le c \text{ for all } j \tag{4.11}$$

The algorithm for generating Y is as follows:

1. Generate a simulated value of the *r.v.* X — let it k
2. Generate a random number U
3. If $U \le \frac{q_k}{cp_k}$

then accept k as the simulated value of Y. Otherwise, go to step **1** (generate a new random number U) and proceed as before.

When c is chosen as the maximum of $\frac{q_j}{p_j}$, then $\frac{q_j}{cp_j}$ will be less than or equal to 1 for all j.

It can shown, as in **Theorems 3.2 and 3.3**, that Y will have the p.m.f. $\{q_j, j = 0, 1,...\}$ and further that the number of iterations needed to obtain a simulated value of Y is a geometric random variable with mean c $\left[\text{or that } P(\text{acceptance}) = \frac{1}{c}\right]$.

11.5 SIMULATION OF STOCHASTIC PROCESSES

A stochastic process is a family or collection of random variables. Thus a stochastic process can be simulated through simulation of the family of random variables.

11.5.1 Simulation of Discrete Parameter Stochastic Process: Markov Chains

Suppose that $\{X_n, n = 0, 1,...\}$ is discrete (time) parameter stochastic process. One is concerned with generation of samples of the n-dimensional vector $\{X_0, X_1,..., X_{n-1}\}$, $n = 1, 2,...$ Once such a sample $\{x_0, x_1,..., x_{n-1}\}$ is available and if the conditional distribution $P\{X_n = x_n | X_0 = x_0, X_1 = x_1,..., X_{n-1} = x_{n-1}\}$, $n = 1, 2,...$, is given then a sample $(x_0, x_1,..., x_{n-1}, x_n)$ can be obtained recursively.

The general recursive method could then be applied by first generating the value x_0 of X_0 according to the initial distribution $F_0(x) = P\{X_0 \le x\}$.

Consider simulation of a Markov chain having transition probabilities $\{p_{i,j}\}$, $i, j = 0, 1, 2,...$ The algorithm is as follows:

First, generate a value of the initial state x_0 from the initial distribution of X_0.

Suppose that $x_0 = i$, then X_1 can be simulated according to the distribution $p_{i,j}$ and so on for $X_2, X_3,...$

Note 1. Even in cases of certain Markov chains $p_{i,j}$'s may not be explicitly known. Consider, for example, a Galton-Watson process $\{X_n, n = 0, 1, 2,\}$ such that

$$X_{n+1} = \sum_{r=1}^{X_n} Y_r$$

where Y_r's are *i.i.d.* discrete random variables having a common distribution $\{p_k, k = 0, 1, 2,...\}$ (offspring distribution). This process is a discrete parameter Markov chain. If we assume that $X_0 = 1$, then $X_1 = Y_1$, and X_1 can be simulated from the distribution of Y_1 and so on for $X_2, X_3,...$

Since the initial values of a Markov chain can have a significant influence on subsequent values of the chain, usually the first k state values are generally discarded.

11.5.2 Markov Chain Monte Carlo (MCMC)

An important question arises as how to generate (simulate) a Markov chain $P = (p_{ij})$ with a given stationary (equilibrium) distribution $\{v_k\}$ where

$$v_k = \lim_{n \to \infty} p_{ij}^{(n)}, \quad \sum v_k = 1$$

A method for generating a Markov chain with a given stationary (equilibrium) distribution was first put forward by a Metropolis et al. (in a Journal of Chemical Physics in 1953). The paper lays down the basic structure leading to what is now known as Markov Chain Monte Carlo (MCMC). The method is now known as *Metropolis-Hastings algorithm* (Hastings, 1970). An important special case of the algorithm is known as Gibbs sampler, which is the commonly used version of the algorithm. Several applications of MCMC used in practice are based on Gibbs sampler.

It is not very difficult to generate such a Markov chain. It is much more difficult to determine how many steps are needed to converge to the stationary distribution within a given error.

MCMC is a rapidly growing area and has applications in Bayesian statistical inference. The method has been used in many other areas, such as statistical computing, computer science etc. and this is still growing. Further details are beyond the scope of this book. Those interested may refer to Gilks *et al.* (1996); also Geyer (1992), Diaconis *et al.* (1995), Roberts and Casella (2001), Ross (2003), Tijms (2003), Berg (2004), and so on.

11.5.3 Simulation of a Poisson Process

Let us consider a counting process such as a Poisson process $\{N(t), t \ge 0\}$ with parameter λ (mean rate per unit time). Such a process has the property that the interarrival time $A_i = t_i - t_{i-1}$, $i = 1, 2,...$, $t_0 = 0$

(t_i is the i^{th} *event time*) are *i.i.d.* exponential *r.v.s.* with mean $\frac{1}{\lambda}$ (**Theorem 3.3**).

The procedure to generate the first n event times t_i, $i = 1, 2,...n$, of the Poisson process is as follows:

1. Generate a random number U
2. The first event time t_1 can be generated as

$$t_1 = -\frac{1}{\lambda}\log U$$

3. Then the subsequent event times $t_2, t_3,..., t_n$ can be generated as

$$t_i = t_{i-1} - \frac{1}{\lambda}\log U_i, \quad i = 2, 3, ..., n$$

taking a separate random number U_i in each case.

11.5.4 Simulation of a Renewal Process

A renewal process is a generalisation of a Poisson process such that the interarrival times $A_i = t_i - t_{i-1}$, $i = 1, 2,...$, $t_0 = 0$ are *i.i.d.* random variable but not necessarily is an exponential random variable, that, is A_i's are *i.i.d.*random variables.

Thus a renewal process can be generated by replacing step **2** of the above (of § 11.5.2) where A_i's are *i.i.d.* random variables.

Suppose that V_i is the generated value of A_i. Thus $t_1 = V_1$ and $t_2, t_3,...$ can be generated using

$$t_i = t_{i-1} + V_i, \quad i = 2, 3, ...$$

11.5.5 Simulation of a Non-homogeneous Poisson Process

For this, a method, known as *thinning* has been advanced by Lewis & Shedler (*Nav. Res. Log. Orly* **26**, 403-13, 1979). This is as follows:

Suppose that $N(t)$ is a non-homogeneous Poisson process with intensity $\lambda(t)$. Suppose further that $\lambda^* = \max_t\{N(t)\}$ is finite.

We can generate a Poisson process with parameter λ^* by using the method described in § 11.5.3. Suppose that t_i^* denote the event times for this process.—suppose that each event of this process is recorded (or counted) with probability $p(t) = \lambda(t)/\lambda^*$, then the randomly recorded (counted) events of the Poisson process constitute a Poisson process with rate $\lambda^* \cdot p(t) = \lambda(t)$ (see § 4.1.3 # 3). Thus we get a non-homogeneous Poisson process with intensity $\lambda(t)$.

By this process, one considers event time t_i^* only when $\lambda(t_i^*)$ is high with probability $\lambda(t_i^*)/\lambda^*$; where this does not happen t_i is thrown out (or not considered) the probability of this happening is $1 - \lambda(t_i^*)/\lambda^*$. This is why the method is called *thinning process* or *random selection process.*

An equivalent algorithm is as follows. This starts with generation of the first event time. Generate random numbers $U_1, U_2,...$, $t_1 = -\frac{1}{\lambda}\log U_1$ and generating the subsequent t_i, $i = 2.3....$ Suppose that t_{i-1} is generated, then $t_i = t_{i-1} - \frac{1}{\lambda^*}\log U_i$.

Generate another random number V_i.

If $V_i \le \lambda(t)/\lambda^*$, then set $t = t_i$. Otherwise, repeat this process by taking another set of rando numbers u_i, V_i and so on.

11.6 SIMULATION OF STOCHASTIC SYSTEM: QUEUEING SYSTEM

Consider an *M/M/1* queueing system [with a Poisson arrival process (having parameter say, λ) and an independent exponential service time (having parameter say μ) and having a single server]. First, one can simulate the arrival process (a Poisson process as in § 11.5.3) and also the service time distribution (an exponential distribution as in Example 3 (d)). Then with these inputs, it is possible to find the waiting time (in the queue as well as in the system) and other characteristics (like busy period etc.) of the system. With a large number of readings, one can estimate some of the performance measures of the system. In the same manner it is possible to work with more general queueing systems. (see, Law & Kelton 1997, J.L. Jain *et al.*, 2007, and so on.)

Simulation softwares have now been advanced for making simulating such system simpler and more convenient. Many special purpose simulation languages have been developed during the last few decades and are available.

The popularity of these languages have been on the increase due to many efforts made towards improving these, as also their standardisation. These have been made easily available also. The simulation language GPSS (General Purpose Simulation System) has been found useful for simulation of queueing systems. For other stochastic systems there are special simulation languages (such as SIMSCRIPT).

These are beyond the scope of this book. Readers interested may look into books for these languages and the methods to be adopted (for example, Law & Kelton, 1997 and so on).

11.7 STATISTICAL ANALYSIS OF SIMULATED DATA

As we have seen Monte Carlo methods are used for generation of *random samples* of *r.v.s*, stochastic processes, and for stochastic systems. Thus such methods become problems of mathematical statistics. Statistical analysis of simulated data has to be taken recourse to. Amongst the important questions that arise are:

1. when to stop the simulation process, and
2. the accuracy of the method used in obtaining the desired output.

Various methods have been put forward to meet these. More complicated estimators like "*bootstrap estimator*" are used in place of the simple sample mean. In order to obtain high accuracy, it is necessary to look into a solution with small variance. Attempts in this regard have led to *variance reduction techniques.*

In this Chapter, an attempt has only been made to indicate the very basics of Monte Carlo methods and Simulation of Stochastic Systems. For more elaborate account of the statistical techniques used reference may be made books on Simulation, for example, Kleijnen, 1974/75, Rubenstein, 1981, Ripley, 1986, Law and Kelton, 1997, Ross, 2002 and Gilks *et al.* (1996)); and also Research publications in this area.

EXERCISES

11.1 With $a = 7^3 = 343$, $m = 2^{17} - 1 = 131071$, $x_0 = 693153$ and using the multiplicative congruence method, obtain 10 random numbers

11.2 Using Monte Carlo method, evaluate the integral

(a) $I = \int_0^1 \frac{dx}{1+x^2}$

and hence estimate the value of π.

(b) $I = \int_1^2 \frac{dx}{x}$

and compare your result with the exact value.

(c) $\int_1^4 \frac{dx}{1+\sqrt{x}}$

and compare with the exact result.

(d) $I = \int_1^2 \int_1^2 \frac{dx\,dy}{x+y}$

and compare your result with the exact solution $ln(1024/729)$. [Use 20-40 random numbers]

11.3 How would you simulate a truncated *r.v.* having *p.d.f.* f and *d.f.* F

$$f(x) = \frac{f(x)}{F(b)-F(a)}, \quad a \le x \le b$$
$$= 0, \qquad \text{otherwise} \quad ?$$

11.4 Simulate a *r.v.* having *p.d.f.*

$$f(x) = ax(1-x)^2, \quad 0 < x < 1$$
$$= 0, \qquad \text{otherwise}$$

11.5 Write down the algorithm you would adopt to simulate the *r.v.* having *p.d.f.*

$$f(x) = \frac{4}{\Gamma(2)} x e^{-2x}, \quad x > 0$$
$$= 0, \qquad \text{otherwise}$$

11.6 Write down an algorithm for Simulating the *r.v.* having *p.d.f.*

$$f(x) = \frac{2}{\sqrt{\pi}} x^{1/2} e^{-x}, \quad x > 0$$
$$= 0, \qquad \text{otherwise}$$

by taking the auxiliary variable Y as exponential having the same mean.

11.7 How would you generate a sample of the Weibull distribution having *d.f.*

$$F(x) = 1 - \exp(-\alpha x^{\beta}), \quad 0 < x < \infty \quad (\alpha > 0, \beta > 0)\,?$$

11.8 Describe an efficient algorithm to simulate the random variable X such that

$$P\{X=0\} = 0.25, \quad P\{X=1\} = 0.30, \quad P\{X=2\} = 0.15$$
$$P\{X=3\} = 0.20, \quad P\{X=4\} = 0.10.$$

11.9 A pair of fair and similar coins are tossed together till *two* heads occur for the *first time*. Develop a simulation method to estimate the average number of tosses required.

11.10 Consider a Poisson arrival process with mean of $\lambda = 10$ arrivals per sec. Simulate the above process to find the event times t_i of i^{th} arrival, $i = 1, 2, \ldots, 50$. Find the average interarrival time and compare with the expected value.

REFERENCES

Berg, B.A. *Markov Chain Monte Carlo Simulations and their Statistical Analysis*, World Scientific (2004).

Casella, G. and R.L. Berger. *Statistical Inference*, 2nd ed., Pacific Grove, CA, (2002).

Chang, H.S., Fu, M.C., Hu, J. and S.J. Marcus. *Simulation-based Algorithms for Markov Decision Processes*, Springer, New York (2007).

Diaconis, P. and S. Holmes. *Three Examples of Monte Carlo Markov Chains: At the Interface between Statistical Computing*, Computer Science and Statistical Mechanics IN *Discrete Probability and Algorithms* (Eds. Aldous, D. Diaconis, P., Spence, J. and Steele, M.) pp. 43–56 Springer-Verlag (1995).

Dudewicz, E.J. and S.N. Mishra. *Modern Mathematical Statistics*, Wiley, New York (1988).

Ecuyer, P. L. *Random Numbers for Simulation, Comm. Ass. Comp. Mach.*, **33** (1990).

Fishman, G. *Discrete Event Simulation*. Wiley, New York (1978).

Gamerman, D. *Markov Chain Monte Carlo for Bayesian Inference*, Chapman & Hall/CRC (1997).

Geyer, C.J. *Practical Markov Chain Monte Carlo (with discussions), Stat. Sc.* **7**, 473–511 (1992).

Gilks, W.R., S. Richardson and D.J. Spielgelhalter. *Markov Chain Monte Carlo in Practice*, Chapman & Hall/CRC, Boca Raton, FL (1996).

Glasserman, P. *Monte Carlo Methods in Financial Engineering* (Stochastic Modelling and Applied Probability), Springer (2003).

Hastings, W.K. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, **57**, 97–109 (1970).

Jain, J.L., S.G. Mohanty and W. Böhm. *A Course on Queueing Models*, Chapman & Hall/CRC, London, Boca Raton, New York (2007).

Kleijnen, J.P.C. *Statistical Techniques in Simulation*, Parts 1 & 2, Marcel Dekker, New York (1974–75).

Knuth, D.E. *The Art of Computer Programming*, **Vol. 2**: *Seminumerical Algorithms*, 2nd ed., Addison-Wesley, Reading (1981).

Kohlas, J. *Stochastic Methods of Operations Research*, CUP (1982).

Law, A.M. and W.D. Kelton. *Simulation Modeling and Analysis*, 2nd ed. (1991), McGraw-Hill, New York, International. 3rd ed., New Delhi (1997).

Marsaglia, G. Random number generation, IN *Encyclopedia of Computer Science and Engineering* (Ed. Ralston, A.) 1260–1264, Van Nostrand, New York (1983).

——— and A. Zaman. A new class of random number generators, *Ann. Appl. Prob.* **Vol. 1** # 3, 462–480, (1991).

Medhi, Deepankar and Ramasamy Karthikeyan. *Network Routing: Algorithms, Protocols and Architecture*, Morgan Kaufmann (ELSEVIER), Amsterdam, London, New York (2007).

Metropolis, N., A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller and E. Teller, Equations of state calculations by fast computing machine, *J. Chem. Phy.*, **21**, 1087–91 (1953).