"There is no result in nature without a cause; understand the cause and you will have no need for the experiment."

Leonardo da Vinci (1452-1519)

Chapter 10

Monte Carlo Methods

In very broad terms one can say that a computer simulation is the process of designing a model of a real or abstract system and then conducting numerical experiments using the computer to obtain a statistical understanding of the system behavior. That is, sampling experiments are performed upon the model. This requires that certain variables in the model be assign random values associated with certain probability distributions. This sampling from various probability distributions requires the use of random numbers to create a stochastic simulation of the system behavior. This stochastic simulation of system behavior is called a Monte Carlo simulation. Monte Carlo simulations are used to construct theories for observed behavior of complicated systems, predict future behavior of a system, and study effects on final results based upon input and parameter changes within a system. The stochastic simulation is a way of experimenting with a system to find ways to improve or better understand the system behavior.

Monte Carlo methods use the computer together with the generation of random numbers and mathematical models to generate statistical results that can be used to simulate and experiment with the behavior of various business, engineering and scientific systems. Some examples of application areas where Monte Carlo modeling and testing have been used are: the simulation and study of specific business management practices, modeling economic conditions, war games, wind tunnel testing of aircraft, operations research, information processing, advertising, complex queuing situations, analysis of mass production techniques, analysis of complex system behavior, analysis of traffic flow, the study of shielding effects due to radiation, the modeling of atomic and subatomic processes, and the study of nuclear reactor behavior. These are just a few of the numerous applications of Monte Carlo techniques.

Monte Carlo simulations usually employ the application of random numbers which are uniformly distributed over the interval $[0,1]$. These uniformly distributed random numbers are used for the generation of stochastic variables from various probability distributions. These stochastic variables can then be used to approximate the behavior of important system variables. In this way one can generate sampling experiments associated with the modeling of system behavior. The statistician can then apply statistical techniques to analyze the data collected on system performance and behavior over a period of time. The generation of a system variable behavior from a specified probability distribution involves the use of uniformly distributed random numbers over the interval $[0,1]$. The quantity of these random numbers generated during a Monte Carlo simulation can range anywhere from hundreds to hundreds of thousands. Consequently, the computer time necessary to run a Monte Carlo simulation can take anywhere from minutes to months depending upon the both the computer system and the application being simulated. The Monte Carlo simulation produces various numerical data associated with both the system performance and the variables affecting the system behavior. These system variables which model the system behavior are referred to as model parameters. The study of the sensitivity of model parameters and their affect on system performance is a large application area of Monte Carlo simulations. These type of studies involve a great deal of computer time and can be costly. We begin this introduction to Monte Carlo techniques with a discussion of random number generators.

Uniformly distributed random numbers

Examine the built in functions associated with the computer language you use most often. Most computer languages have some form of random number generator which can be used to generate uniformly distributed random numbers between 0 and 1. The majority of these random number generators use modulo arithmetic to generate numbers which appear to be uniformly distributed. Consequently, the random number generators are called pseudorandom number generators because they are not truly random. They only simulate the behavior of a uniformly distributed random number on the interval $[0,1]$. The basic equations used in generating these pseudorandom numbers usually have one of the forms

$$X_{i+1} \equiv AX_i \pmod M \qquad \text{A multiplicative congruential generator}$$

or $$X_{i+1} \equiv AX_i + C \pmod M \qquad \text{A mixed congruential generator}$$

where A,C and M are nonnegative integers. The sequence of numbers $\{X_i\}$, for $i = 0, 1, 2, \ldots$, generated by a congruential generator, needs a starting value X_0. The initial value X_0 is called the seed of the random number generator. The quantity X_{i+1} represents one of the integers from the set $\{0, 1, 2, \ldots, M-1\}$. In general $A \equiv B \pmod{M}$ is read A is congruent to B modulo M, where the quantity A is calculated from the relation $A = B - K * M$, where $K = [B/M]$ denotes the largest positive integer resulting from the truncation of B/M to form an integer. The quantity A represents the remainder when B is divided by M. By using modulo arithmetic, the numbers of the sequence $\{X_i\}$ eventually start to repeat themselves and so only a finite number of distinct integers are generated by the above methods. The number of integers generated by the congruential generator before repetition starts to occur is called the period of the pseudorandom number generator.

Using the mixed congruential generator, with a given seed X_0, one can write

$X_1 = AX_0 + C - MK_1$ $\quad$ K_1 is some appropriate constant.

$X_2 = AX_1 + C - MK_2$ $\quad$ K_2 is some appropriate constant.

Substituting for X_1 and simplifying gives

$X_2 = A^2X_0 + C(1+A) - M(K_2 + AK_1)$

$X_3 = AX_2 + C - MK_3$ $\quad$ K_3 is some appropriate constant.

Substituting for X_2 and simplifying gives

$X_3 = A^3X_0 + C(1+A+A^2) - M(K_3 + K_2A + K_1A^2)$

$\vdots$ $\quad$ Continuing in this manner one can show

$X_n = A^nX_0 + C(1+A+A^2+\cdots+A^{n-1}) - MK$

where $K = K_n + K_{n-1}A + \cdots + K_1A^{n-1} = \sum_{i=1}^{n} K_iA^{n-i}$ is some new constant. Knowledge of the geometric series enables us to simplify the above result to the form $X_n = A^nX_0 + \dfrac{C(A^n-1)}{A-1} \bmod M$. Now if for some value of n we have $X_0 = A^nX_0 + \dfrac{C(A^n-1)}{A-1} \bmod M$, then one can write

$$(A^n - 1)\left[X_0 + \frac{C}{A-1}\right] \equiv 0 \bmod M \tag{10.1}$$

The minimum value of n which satisfies the equation (10.1) is the period of the pseudorandom number generator. The special case where $C = 0$ is easier to understand. In this special case one can employ the Fermat theorem that if M is prime and A is not a multiple of M, then $A^M \equiv 1 \bmod M$. Then as a special case of the equation (10.1) $n = M = P$ is the period of the pseudorandom number generator.

The equation (10.1) has been extensively analyzed using number theory and one conclusion is that the quantity M can be selected as a power of 2. The power to which 2 is raised is based upon the number of bits b associated with a computer word size. The largest integer that can be stored using b-bits is $2^b - 1$. Selecting $M = 2^b$ enables the largest possible period to be obtained for the computer system being used. Alternatively, M can be selected as a large prime number compatible with the computer word size. Once M is selected the value of A must satisfy $0 < A < M$. The sequence of values $\{X_i\}$ are then determined by the values M, A, X_0 and C and so the sequence generated will have a period $P \leq M$. That is, $P \leq M$ distinct values will be generated before the sequence starts to repeat itself. For each index i, the uniformly distributed pseudorandom numbers R_i are calculated from the relation $R_i = X_i/M$ and satisfy $0 < R_i < 1$. Optimally, one should select the quantities M, A, X_0, and C to maximize the period of the sequence and reduce the degree of correlation between the numbers generated. However, this is not always achieved.

There are many Fortran language packages created and sold under different brand names. Some Fortran languages treat the random number generator as a subroutine and in others the random number generator is treated as an intrinsic function. In the Fortran examples that follow it is assumed that there exists a subroutine RANDOM(RN) which generates a uniform random number RN between 0 and 1. It is also assumed there is a Fortran subroutine SEED(INTEGER) for setting the seed value for the pseudo random number generator. These subroutines can then be used to create other subroutines which return random variates satisfying various conditions. Some examples are illustrated.

```
      SUBROUTINE RAN1(IX,N)
C     Generate random integer IX
C     Satisfying   0 .LE. IX .LE. N
      CALL RANDOM(RN)
      IX=INT(RN*(N+1))
      RETURN
      END
```

```
      SUBROUTINE RAN2(NMIN,NMAX, N)
C     Generate random integer N
C     satisfying   NMIN .LE. N .LE. NMAX
      CALL RANDOM(RN)
      N=NMIN +INT((NMAX+1-NMIN)*RN)
      RETURN
      END
```

Chi-square χ^2 goodness of fit

The chi-square goodness of fit test is used to compare actual frequencies from sampled data with frequencies from theoretical distributions. The chi-square statistic is calculated from the relation

$$\chi^2 = \sum_{k=1}^{n} \frac{(f_{ok} - f_{ek})^2}{f_{ek}} \tag{10.2}$$

where f_{ok} is the observed frequency of the kth class or interval, f_{ek} is the expected frequency of the kth class or interval due to theoretical considerations, and n is the number of classes or intervals. Let the theoretical distribution function be denoted by $F(x)$. From this theoretical probability distribution one can calculate the probability p_k that a random variable X takes on a value in the kth interval. One finds $f_{ek} = np_k$ as the number of theoretical expected values in the kth class or interval. In using the chi-square goodness of fit test, one is testing the null hypothesis H_0 that there is no significant difference between the frequencies of the sampled data and that expected from theoretical considerations. Ideally, if $\chi^2 = 0$ then the observed frequencies and theoretical frequencies agree exactly. For $\chi^2 > 0$ there is a discrepancy between the observed and theoretical frequencies and one must resort to tables of χ^2 critical values, associated with a significance level and degrees of freedom, to determine if one should accept or reject the null hypothesis. If the computed value of χ^2 is greater than the tabular critical value at some significance level found in tables, then one must reject the null hypothesis.

As an example, apply the chi-square goodness of fit test to the random number generator associated with your computer programming language. Write a computer program to generate 1000 random numbers between 0 and 1. You can then divided the interval 0 to 1 into 10 classes using the intervals $(0, .1), (.1, .2), \cdots, (.9, 1.0)$ and then sort the 1000 random numbers to determine the number in each class. These values are the observed frequencies determined by the experiment. If the pseudorandom number generator is truly uniform, then the theoretical frequency associated with each class would have a value of 100. Note that when using the chi-square goodness of fit test one should always use actual counts for the frequencies. Do not use relative frequencies or percentages. Also the theoretical frequencies associated with each class or interval should number greater than 5. If this is not the case then adjacent intervals or nearest neighbor intervals must be combined into a new class or interval with frequency greater than or equal to 5. The degrees of freedom ν associated a chi-square test is given by $\nu = n - 1 - m$ where n is the number of classes or intervals and m is the number of parameters in the theoretical distribution being tested. For the chi-square goodness of fit test associated with our random number generator test, the value of the degrees of freedom is $\nu = 10 - 1 - 0 = 9$. One would then find a chi-square table of critical values, such as the one on page 497, and look up the critical value associated with a $(1-\alpha)$ significance level by selecting an appropriate column and then select a row of the table which represents the degrees of freedom. The tabulated value found is then compared with the calculated χ^2-value to determine if the random number generator differs significantly from the theoretical values expected.

Example 10-1. (Chi-square test.)

A pseudorandom number generator associated with a certain Fortran computer language was used to generate 1000 random numbers X. The frequencies associated with 10 equal spaced intervals over the range (0,1) where calculated and are given in the accompanying table.

Use a chi-square test to compare the resulting frequencies with theoretical values associated with a uniform distribution of random numbers.

The chi-square statistic is found to be

$$\chi^2 = \sum_{k=1}^{10} \frac{(f_{ok}-100)^2}{100} = 14.90$$

This number is compared with the tabular value of 23.5893 from the $\chi^2_{0.995}$ column and $\nu = 9$ row of the chi-square table of critical values found on page 497. Since the χ^2-value 14.90 is less than the critical value of 23.5893 we can accept the numbers generated by the Fortran computer code as representing a uniform random number generator.

X range	Frequency	Symbol
$X \le 0.1$	106	f_{o1}
$0.1 < X \le 0.2$	110	f_{o2}
$0.2 < X \le 0.3$	79	f_{o3}
$0.3 < X \le 0.4$	99	f_{o4}
$0.4 < X \le 0.5$	88	f_{o5}
$0.5 < X \le 0.6$	98	f_{o6}
$0.6 < X \le 0.7$	125	f_{o7}
$0.7 < X \le 0.8$	107	f_{o8}
$0.8 < X \le 0.9$	97	f_{o9}
$0.9 < X \le 1.0$	91	f_{o10}

■

Discrete and continuous distributions

In constructing Monte Carlo simulations it is important that one know how to generate random variables X which come from a specified probability distribution. Let X denote a random variable with cumulative probability distribution function $F(x)$ and let RN, denote a uniformly distributed random number $0 \le RN \le 1$. If $RN = F(X)$, then the inverse function gives $X = F^{-1}(RN)$. The situation is illustrated in the figure 11-1.

The computer generation of a random variable X associated with a discrete or continuous distribution can be illustrated graphically. Calculate the distribution function $F(x)$ associated with a discrete or continuous probability function or relative frequency function $f(x)$. Then generate a uniform random number RN, $0 \le RN \le 1$ and plot this number on the $F(x)$ axis and then move horizontally until you hit the distribution function curve. Then drop down to obtain the random variable X. This is the inverse function method of generating a random variable X associated with a given distribution.

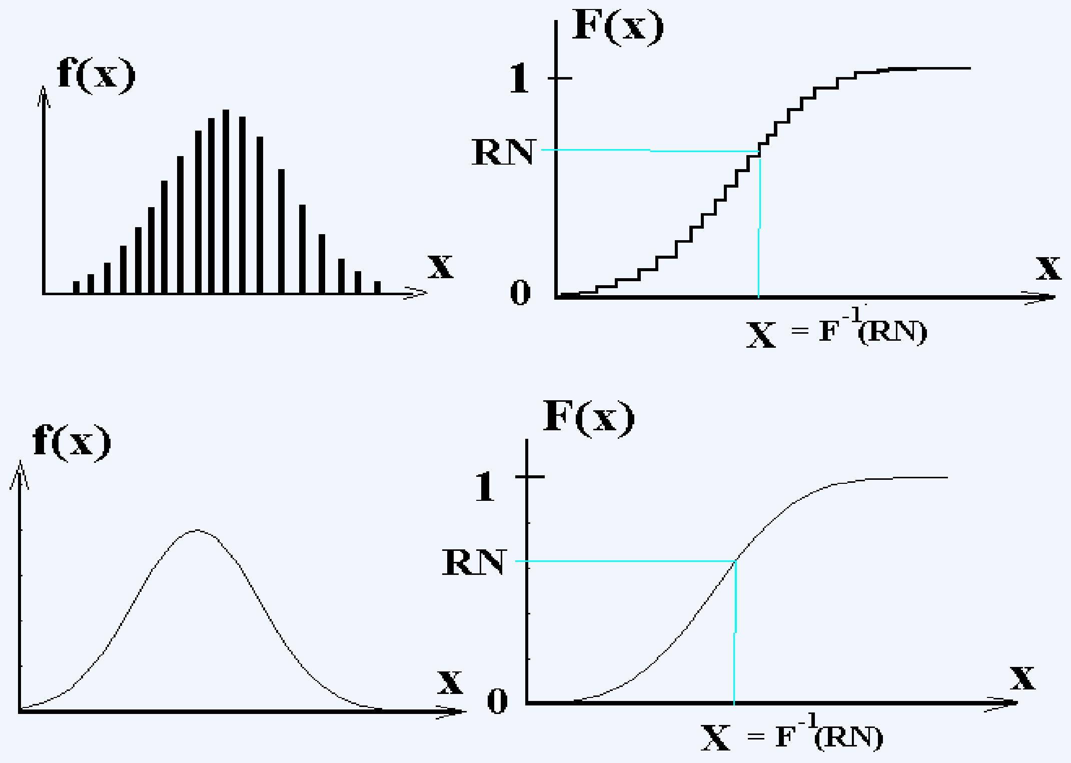

Figure 11-1.
Discrete and continuous distributions associated with random variable X.

The following is a list of some of the more popular discrete and continuous probability distributions that can be used to help model various Monte Carlo simulations.

Discrete Distributions

Discrete uniform distribution

$$f(x) = \begin{cases} \frac{1}{N_2+1-N_1}, & x = N_1, N_1+1, \dots, N_2 \\ 0 & \text{otherwise} \end{cases}$$

N_1, N_2 integers with $N_2 > N_1$.

Poisson distribution

$$f(x) = \begin{cases} e^{-\lambda}\lambda^x/x! & x = 0, 1, 2, \ldots \\ 0 & \text{otherwise} \end{cases} \qquad \lambda > 0.$$

Binomial distribution

$$f(x) = \begin{cases} \binom{N}{x} p^x (1-p)^{N-x}, & x = 0, 1, 2, \ldots, N \\ 0 & \text{otherwise} \end{cases}$$

$0 < p < 1$ and N a positive integer.

Geometric distribution

$$f(x) = \begin{cases} p(1-p)^{x-1}, & x = 1, 2, \ldots \\ 0 & \text{otherwise} \end{cases} \qquad 0 < p < 1.$$

Hypergeometric distribution

$$f(x) = \frac{\binom{M}{x}\binom{N-M}{J-x}}{\binom{N}{J}} \text{ for } x = 0, 1, 2, \ldots, J,$$

N, M, J are integers.

Continuous Distributions

Uniform distribution

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

Normal distribution

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

Exponential distribution

$$f(x) = \lambda\, e^{-\lambda x} \qquad x > 0 \text{ and } \lambda > 0.$$

Beta distribution

$$f(x) = \begin{cases} \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1} & 0 < x < 1 \\ 0 & \text{elsewhere} \end{cases} \qquad \alpha > 0, \beta > 0.$$

Gamma distribution

$$f(x) = \begin{cases} \frac{1}{\Gamma(\alpha)\beta^\alpha} x^{\alpha-1} e^{-x/\beta} & x > 0 \\ 0 & \text{elsewhere} \end{cases}, \quad \alpha > 0, \beta > 0.$$

Weibull distribution

$$f(x) = \begin{cases} \alpha\beta x^{\beta-1} e^{-\alpha x^\beta} & x > 0 \\ 0 & \text{elsewhere} \end{cases} \qquad \alpha > 0, \beta > 0.$$

Log-Normal distribution

$$f(x) = \begin{cases} \frac{1}{\beta\sqrt{2\pi}}\,\frac{1}{x} e^{-(\ln x - \alpha)^2/2\beta^2} & x > 0 \\ 0 & \text{elsewhere} \end{cases} \qquad \beta > 0.$$

Chi-square distribution

$$f(x) = \begin{cases} \frac{1}{2^{\nu/2}\sigma^{\nu}\Gamma(\nu/2)} x^{(\nu/2)-1} e^{-(x/2\sigma^2)} & x > 0 \\ 0 & x < 0 \end{cases}$$

where ν represents the degrees of freedom.

Student's t-distribution

$$f(x) = \frac{1}{\sqrt{n\pi}} \frac{\Gamma[(n+1)/2]}{\Gamma[n/2]} \left(1 + \frac{x^2}{n}\right)^{-(n+1)/2}$$

with $n = 1, 2, 3, \dots$ degrees of freedom.

F-distribution

$$f(x) = \begin{cases} \frac{\Gamma[(m+n)/2]}{\Gamma(m/2)\Gamma(n/2)} (m/n)^{m/2} \frac{x^{(m/2)-1}}{[1+(m/n)x]^{(m+n)/2}} & x > 0 \\ 0 & x < 0 \end{cases}$$

with parameters $m = 1, 2, \dots$ and $n = 1, 2, \dots$

In the discussions that follow we develop programs to generate random variates from only a few select probability distributions. The more complicated random variate generators are left for more advanced simulation courses. In this introduction to Monte Carlo simulations we consider only discrete empirical distributions, binomial distributions, Poisson distributions, normal distributions and exponential distributions as these distributions are easy to work with and are representative of how one employs various discrete and continuous probability distributions for modeling purposes.

Selected discrete distributions

Recall that associated with a discrete sample is the function

$$f(x) = \begin{cases} f_j & x = x_j \quad j = 1, 2, 3, \dots \\ 0 & \text{otherwise} \end{cases} \qquad \sum_{k=1}^{\infty} f_j = 1 \tag{10.3}$$

where f_j are relative frequencies associated with the sample. The function $f(x)$ given by equation (10.3) is also referred to as the probability distribution function of the sample. To calculate the probability $P(a < X \le b)$ one would calculate

$$P(a < X \le b) = \sum_{a < x_j \le b} f(x_j) \tag{10.4}$$

The function $F(x)$ representing the cumulative relative frequency function is given by

$$F(x) = P(X \leq x) = \sum_{x_j \leq x} f(x_j) \quad \text{with} \quad P(X > x) = 1 - F(x) \tag{10.5}$$

and is called the distribution function associated with the sample. Note that the above definition implies

$$P(a < X \leq b) = P(X \leq b) - P(X \leq a) = F(b) - F(a) \tag{10.6}$$

The mean μ associated with a discrete distribution is defined

$$\mu = \sum_j x_j f(x_j)$$

The variance σ^2 associated with a discrete distribution is defined

$$\sigma^2 = \sum_j (x_j - \mu)^2 f(x_j).$$

The positive square root of the variance gives the standard deviation σ. For X a random variable and $g(X)$ any continuous function, then the mathematical expectation of $g(X)$ is defined for discrete distributions as

$$E(g(X)) = \sum_j g(x_j) f(x_j) \quad (X \text{ discrete}).$$

Empirical distributions associated with a sample are generated by collecting data and calculating the frequency, relative frequency and cumulative relative frequency associated with the data.

The **Binomial probability distribution**, which is sometimes referred to as a Bernoulli distribution, is given by

$$f(x) = \begin{cases} \binom{n}{x} p^x (1-p)^{n-x} & \text{for } x = 0, 1, 2, \ldots, n \\ 0 & \text{otherwise} \end{cases}$$

and contains the two parameters p and n. The parameter p is a probability satisfying $0 \leq p \leq 1$ and the parameter $n = 1, 2, 3, \ldots$. This probability distribution is used in application areas where one of two possible outcomes can result. For example, the number x of successes in n independent repeated events in which the probability of success p for each event is governed by the binomial distribution.