C H A P T E R

3 Simulation Basics

"Zeal without knowledge is fire without light."

—Dr. Thomas Fuller

3.1 Introduction

Simulation is much more meaningful when we understand what it is actually doing. Understanding how simulation works helps us to know whether we are applying it correctly and what the output results mean. Many books have been written that give thorough and detailed discussions of the science of simulation (see Banks et al. 2001; Hoover and Perry 1989; Law 2007; Pooch and Wall 1993; Ross 1990; Shannon 1975; Thesen and Travis 1992; and Widman, Loparo, and Nielsen 1989). This chapter attempts to summarize the basic technical issues related to simulation that are essential to understand in order to get the greatest benefit from the tool. The chapter discusses the different types of simulation and how random behavior is simulated. A spreadsheet simulation example is given in this chapter to illustrate how various techniques are combined to simulate the behavior of a common system.

3.2 Types of Simulation

The way simulation works is based largely on the type of simulation used. There are many ways to categorize simulation. Some of the most common include

- Static or dynamic.
- Stochastic or deterministic.
- Discrete event or continuous.

The type of simulation we focus on in this book can be classified as dynamic, stochastic, discrete-event simulation. To better understand this classification, we will look at what these three different simulation categories mean.

3.2.1 Static versus Dynamic Simulation

A *static* simulation is one that is not based on time. It often involves drawing random samples to generate a statistical outcome, so it is sometimes called Monte Carlo simulation. In finance, Monte Carlo simulation is used to select a portfolio of stocks and bonds. Given a portfolio, with different probabilistic payouts, it is possible to generate an expected yield. One material handling system supplier developed a static simulation model to calculate the expected time to travel from one rack location in a storage system to any other rack location. A random sample of 100 from–to relationships were used to estimate an average travel time. Had every from–to trip been calculated, a 1,000-location rack would have involved 1,000 factorial calculations.

Dynamic simulation includes the passage of time. It looks at state changes as they occur over time. A clock mechanism moves forward in time and state variables are updated as time advances. Dynamic simulation is well suited for analyzing manufacturing and service systems since they operate over time.

3.2.2 Stochastic versus Deterministic Simulation

Simulations in which one or more input variables are random are referred to as *stochastic* or *probabilistic* simulations. A stochastic simulation produces output that is itself random and therefore gives only one data point of how the system might behave.

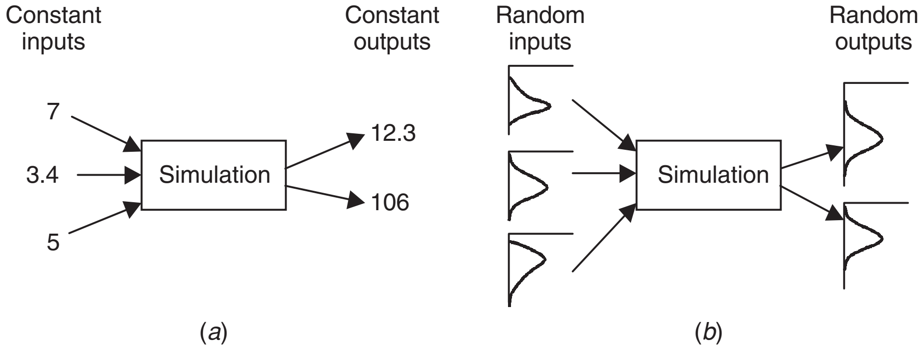
Simulations having no input components that are random are said to be *deterministic*. Deterministic simulation models are built the same way as stochastic models except that they contain no randomness. In a deterministic simulation, all future states are determined once the input data and initial state have been defined.

As shown in Figure 3.1, deterministic simulations have constant inputs and produce constant outputs. Stochastic simulations have random inputs and produce random outputs. Inputs might include activity times, arrival intervals, and routing sequences. Outputs include metrics such as average flow time, flow rate, and resource utilization. Any output impacted by a random input variable is going to also be a random variable. That is why the random inputs and random outputs of Figure 3.1(*b*) are shown as statistical distributions.

A deterministic simulation will always produce the exact same outcome no matter how many times it is run. In stochastic simulation, several randomized runs or replications must be made to get an accurate performance estimate because each run varies statistically. Performance estimates for stochastic simulations are obtained by calculating the average value of the performance metric across all of the replications. In contrast, deterministic simulations need to be run only once to get precise results because the results are always the same.

Figure 3.1

Examples of (a) a deterministic simulation and (b) a stochastic simulation.



3.2.3 Discrete-Event versus Continuous Simulation

A *discrete-event* simulation is one in which state changes occur at discrete points in time as triggered by events. Typical simulation events might include

- The arrival of an entity to a workstation.
- The failure of a resource.
- The completion of an activity.
- The end of a shift.

State changes in a model occur when some event happens. The state of the model becomes the collective state of all the elements in the model at a particular point in time. State variables in a discrete-event simulation are referred to as *discrete-change* state variables. A restaurant simulation is an example of a discrete-event simulation because all of the state variables in the model, such as the number of customers in the restaurant, are discrete-change state variables (see Figure 3.2). Most manufacturing and service systems are typically modeled using discrete-event simulation.

In *continuous* simulation, state variables change continuously with respect to time and are therefore referred to as *continuous-change* state variables. An example of a continuous-change state variable is the level of oil in an oil tanker that is being either loaded or unloaded, or the temperature of a building that is controlled by a heating and cooling system. Some readers are perhaps familiar with a set of nonlinear partial differential equations called Navier-Stokes equations used to model the continuous flow of liquids and other substances. Figure 3.3 compares a discrete-change state variable and a continuous-change state variable as they vary over time.

Continuous simulation products use either differential equations or difference equations to define the rates of change in state variables over time.

FIGURE 3.2

Discrete events cause discrete state changes.

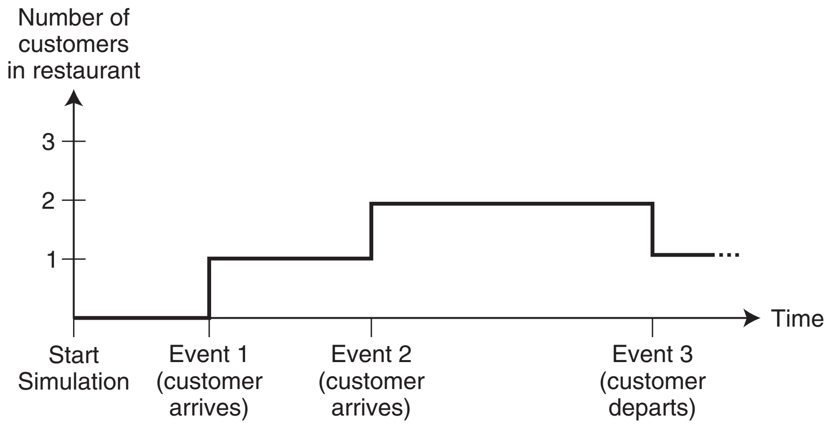
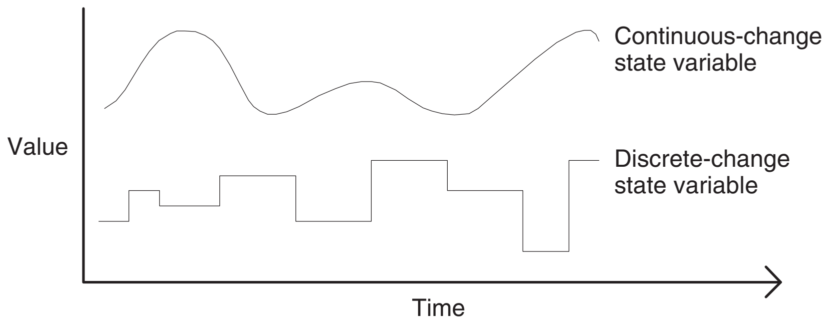


FIGURE 3.3

Comparison of a discrete-change state variable and a continuous-change state variable.



Differential Equations

The change that occurs in some continuous-change state variables is expressed in terms of the derivatives of the state variables. Equations involving derivatives of a state variable are referred to as *differential equations*. The state variable v, for example, might change as a function of both v and time t:

$$\frac{dv(t)}{dt} = v^2(t) + t^2$$

We then need a second equation to define the initial condition of v:

$$v(0) = K$$

On a computer, numerical integration is used to calculate the change in a particular response variable over time. Numerical integration is performed at the end of successive small time increments referred to as *steps*. Numerical analysis techniques, such as Runge-Kutta integration, are used to integrate the differential equations numerically for each incremental time step. One or more threshold values for each continuous-change state variable are usually defined that determine when some action is to be triggered, such as shutting off a valve or turning on a pump.

Difference Equations

Sometimes a continuous-change state variable can be modeled using difference equations. In such instances, the time is decomposed into periods of length t. An algebraic expression is then used to calculate the value of the state variable at the end of period $k + 1$ based on the value of the state variable at the end of period k. For example, the following difference equation might be used to express the rate of change in the state variable v as a function of the current value of v, a rate of change (r), and the length of the time period (Δt):

$$v(k + 1) = v(k) + r\Delta t$$

Batch processing in which fluids are pumped into and out of tanks can often be modeled using difference equations.

Combined Continuous and Discrete Simulation

Many simulation software products provide both discrete-event and continuous simulation capabilities. This enables systems that have both discrete-event and continuous characteristics to be modeled, resulting in a hybrid simulation. Most processing systems that have continuous-change state variables also have discrete-change state variables. For example, a truck or tanker arrives at a fill station (a discrete event) and begins filling a tank (a continuous process).

Four basic interactions occur between discrete- and continuous-change variables:

1. A continuous variable value may suddenly increase or decrease as the result of a discrete event (like the replenishment of inventory in an inventory model).
2. The initiation of a discrete event may occur as the result of reaching a threshold value in a continuous variable (like reaching a reorder point in an inventory model).
3. The change rate of a continuous variable may be altered as the result of a discrete event (a change in inventory usage rate as the result of a sudden change in demand).
4. An initiation or interruption of change in a continuous variable may occur as the result of a discrete event (the replenishment or depletion of inventory initiates or terminates a continuous change of the continuous variable).

3.3 Random Behavior

Stochastic systems frequently have time or quantity values that vary within a given range and according to specified density, as defined by a probability distribution. Probability distributions are useful for predicting the next time, distance, quantity, and so forth when these values are random variables. For example, if an operation time varies between 2.2 minutes and 4.5 minutes, it would be defined in the model as a probability distribution. Probability distributions are defined by specifying the type of distribution (normal, exponential, or another type) and

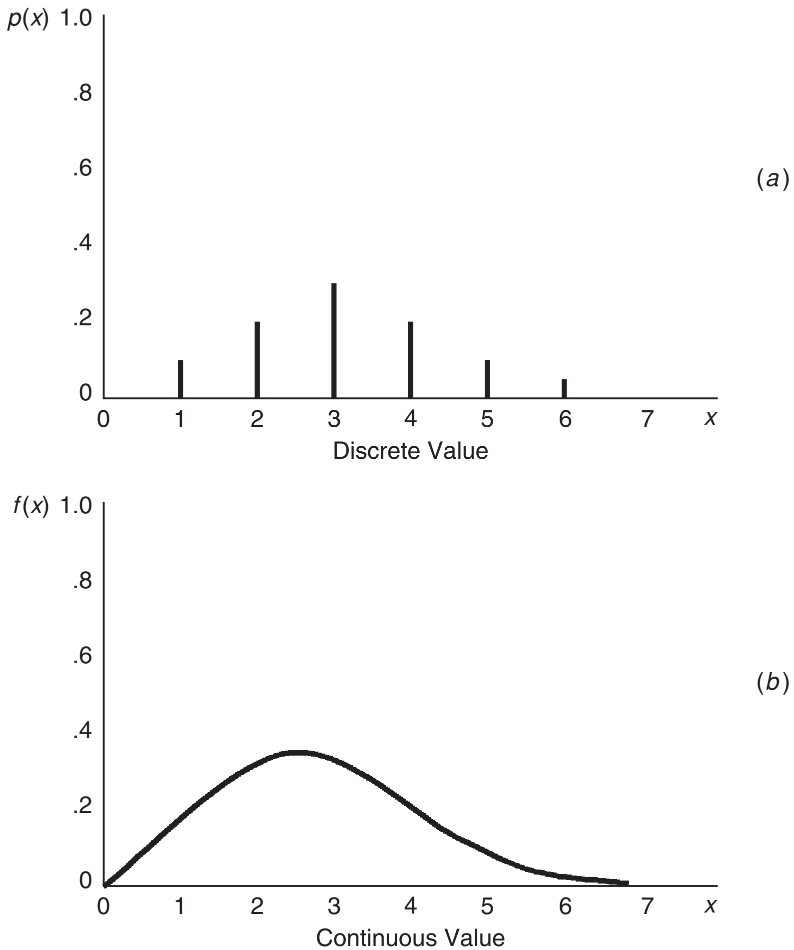


FIGURE 3.4

Examples of (a) a discrete probability distribution and (b) a continuous probability distribution.

the parameters that describe the shape or density and range of the distribution. For example, we might describe the time for a check-in operation to be normally distributed with a mean of 5.2 minutes and a standard deviation of 0.4 minute. During the simulation, values are obtained from this distribution for successive operation times. The shape and range of time values generated for this activity will correspond to the parameters used to define the distribution. When we generate a value from a distribution, we call that value a *random variate*.

Probability distributions from which we obtain random variates may be either discrete (they describe the likelihood of specific values occurring) or continuous (they describe the likelihood of a value being within a given range). Figure 3.4 shows graphical examples of a discrete distribution and a continuous distribution.

A discrete distribution represents a finite or countable number of possible values. An example of a discrete distribution is the number of items in a lot or individuals in a group of people. A continuous distribution represents a continuum of values. An example of a continuous distribution is a machine with a cycle time that is uniformly distributed between 1.2 minutes and 1.8 minutes. An infinite number of possible values exist within this range. Discrete and continuous distributions are further defined in Chapter 5. Appendix A describes many of the distributions used in simulation.

3.4 Simulating Random Behavior

One of the most powerful features of simulation is its ability to mimic random behavior or variation that is characteristic of stochastic systems. Simulating random behavior requires that a method be provided to generate random numbers as well as routines for generating random variates based on a given probability distribution. Random numbers and random variates are defined in the next sections along with the routines that are commonly used to generate them.

3.4.1 Generating Random Numbers

Random behavior is imitated in simulation by using a *random number generator*. The random number generator operates deep within the heart of a simulation model, pumping out a stream of random numbers. It provides the foundation for simulating "random" events occurring in the simulated system such as the arrival time of cars to a restaurant's drive-through window; the time it takes the driver to place an order; the number of hamburgers, drinks, and fries ordered; and the time it takes the restaurant to prepare the order. The input to the procedures used to generate these types of events is a stream of numbers that are uniformly distributed between zero and one ($0 \leq x \leq 1$). The random number generator is responsible for producing this stream of independent and uniformly distributed numbers (Figure 3.5).

Before continuing, it should be pointed out that the numbers produced by a random number generator are not "random" in the truest sense. For example, the generator can reproduce the same sequence of numbers again and again, which is not indicative of random behavior. Therefore, they are often referred

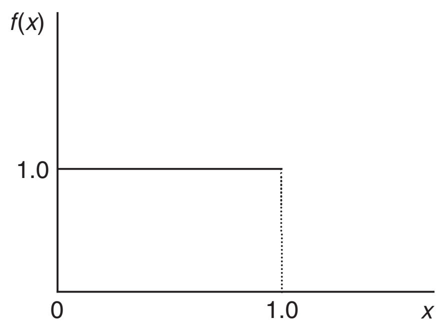
FIGURE 3.5

The uniform(0, 1) distribution of a random number generator.

$$f(x) = \begin{cases} 1 & \text{for } 0 \leq x \leq 1 \\ 0 & \text{elsewhere} \end{cases}$$

$$\text{Mean} = \mu = \frac{1}{2}$$

$$\text{Variance} = \sigma^2 = \frac{1}{12}$$



to as *pseudo-random number generators* (pseudo comes from Greek and means false or fake). Practically speaking, however, "good" pseudo-random number generators can pump out long sequences of numbers that pass statistical tests for randomness (the numbers are independent and uniformly distributed). Thus the numbers approximate real-world randomness for purposes of simulation, and the fact that they are reproducible helps us in two ways. It would be difficult to debug a simulation program if we could not "regenerate" the same sequence of random numbers to reproduce the conditions that exposed an error in our program. We will also learn in Chapter 9 how reproducing the same sequence of random numbers is useful when comparing different simulation models. For brevity, we will drop the *pseudo* prefix as we discuss how to design and keep our random number generator healthy.

Linear Congruential Generators

There are many types of established random number generators, and researchers are actively pursuing the development of new ones (L'Ecuyer 1998). However, most simulation software is based on linear congruential generators (LCG). The LCG is efficient in that it quickly produces a sequence of random numbers without requiring a great deal of computational resources. Using the LCG, a sequence of integers $Z_1, Z_2, Z_3, \ldots$ is defined by the recursive formula

$$Z_i = (aZ_{i-1} + c) \bmod (m)$$

where the constant a is called the multiplier, the constant c the increment, and the constant m the modulus (Law 2007). The user must provide a seed or starting value, denoted Z_0, to begin generating the sequence of integer values. Z_0, a, c, and m are all nonnegative integers. The value of Z_i is computed by dividing $(aZ_{i-1} + c)$ by m and setting Z_i equal to the remainder part of the division, which is the result returned by the mod function. Therefore, the Z_i values are bounded by $0 \le Z_i \le m - 1$ and are uniformly distributed in the discrete case. However, we desire the continuous version of the uniform distribution with values ranging between a low of zero and a high of one, which we will denote as U_i for $i = 1, 2, 3, \ldots$. Accordingly, the value of U_i is computed by dividing Z_i by m.

In a moment, we will consider some requirements for selecting the values for a, c, and m to ensure that the random number generator produces a long sequence of numbers before it begins to repeat them. For now, however, let's assign the following values $a = 21$, $c = 3$, and $m = 16$ and generate a few pseudo-random numbers. Table 3.1 contains a sequence of 20 random numbers generated from the recursive formula

$$Z_i = (21Z_{i-1} + 3) \bmod (16)$$

An integer value of 13 was somewhat arbitrarily selected between 0 and $m - 1 = 16 - 1 = 15$ as the seed ($Z_0 = 13$) to begin generating the sequence of numbers in Table 3.1. The value of Z_1 is obtained as

$$Z_1 = (aZ_0 + c) \bmod (m) = (21(13) + 3) \bmod (16) = (276) \bmod (16) = 4$$

TABLE 3.1 Example LCG $Z_i = (21Z_{i-1} + 3) \bmod (16)$, with $Z_0 = 13$

i	$21Z_{i-1} + 3$	Z_i	$U_i = Z_i/16$
0		13	
1	276	4	0.2500
2	87	7	0.4375
3	150	6	0.3750
4	129	1	0.0625
5	24	8	0.5000
6	171	11	0.6875
7	234	10	0.6250
8	213	5	0.3125
9	108	12	0.7500
10	255	15	0.9375
11	318	14	0.8750
12	297	9	0.5625
13	192	0	0.0000
14	3	3	0.1875
15	66	2	0.1250
16	45	13	0.8125
17	276	4	0.2500
18	87	7	0.4375
19	150	6	0.3750
20	129	1	0.0625

Note that 4 is the remainder term from dividing 16 into 276. The value of U_1 is computed as

$$U_1 = Z_1/16 = 4/16 = 0.2500$$

The process continues using the value of Z_1 to compute Z_2 and then U_2.

Changing the value of Z_0 produces a different sequence of uniform(0, 1) numbers. However, the original sequence can always be reproduced by setting the seed value back to 13 ($Z_0 = 13$). The ability to repeat the simulation experiment under the exact same "random" conditions is very useful, as will be demonstrated in Chapter 9 with a technique called common random numbers.

Note that in ProModel the sequence of random numbers is not generated in advance and then read from a table as we have done. Instead, the only value saved is the last Z_i value that was generated. When the next random number in the sequence is needed, the saved value is fed back to the generator to produce the next random number in the sequence. In this way, the random number generator is called each time a new random event is scheduled for the simulation.

Due to computational constraints, the random number generator cannot go on indefinitely before it begins to repeat the same sequence of numbers. The LCG in Table 3.1 will generate a sequence of 16 numbers before it begins to repeat itself. You can see that it began repeating itself starting in the 17th position. The value of 16 is referred to as the cycle length of the random number generator, which is disturbingly short in this case. A long cycle length is desirable so that each replication of a simulation is based on a different segment of random numbers. This is how we collect independent observations of the model's output.

Let's say, for example, that to run a certain simulation model for one replication requires that the random number generator be called 1,000 times during the simulation and we wish to execute five replications of the simulation. The first replication would use the first 1,000 random numbers in the sequence, the second replication would use the next 1,000 numbers in the sequence (the number that would appear in positions 1,001 to 2,000 if a table were generated in advance), and so on. In all, the random number generator would be called 5,000 times. Thus we would need a random number generator with a cycle length of at least 5,000.

The maximum cycle length that an LCG can achieve is m. To realize the maximum cycle length, the values of a, c, and m have to be carefully selected. A guideline for the selection is to assign (Pritsker 1995)

1. $m = 2^b$, where b is determined based on the number of bits per word on the computer being used. Many computers use 32 bits per word, making 31 a good choice for b.
2. c and m such that their greatest common factor is 1 (the only positive integer that exactly divides both m and c is 1).
3. $a = 1 + 4k$, where k is an integer.

Following this guideline, the LCG can achieve a full cycle length of over 2.1 billion (2^{31} to be exact) random numbers.

Frequently, the long sequence of random numbers is subdivided into smaller segments. These subsegments are referred to as *streams*. For example, Stream 1 could begin with the random number in the first position of the sequence and continue down to the random number in the 200,000th position of the sequence. Stream 2, then, would start with the random number in the 200,001st position of the sequence and end at the 400,000th position, and so on. Using this approach, each type of random event in the simulation model can be controlled by a unique stream of random numbers. For example, Stream 1 could be used to generate the arrival pattern of cars to a restaurant's drive-through window and Stream 2 could be used to generate the time required for the driver of the car to place an order. This assumes that no more than 200,000 random numbers are needed to simulate each type of event. The practical and statistical advantages of assigning unique streams to each type of event in the model are described in Chapter 9.

To subdivide the generator's sequence of random numbers into streams, you first need to decide how many random numbers to place in each stream. Next, you begin generating the entire sequence of random numbers (cycle length) produced by the generator and recording the Z_i values that mark the beginning of each stream. Therefore, each stream has its own starting or seed value. When using the random number generator to drive different events in a simulation model, the previously generated random number from a particular stream is used as input to the generator to generate the next random number from that stream. For convenience, you may want to think of each stream as a separate random number generator to be used in different places in the model. For example, see Figure 9.5 in Chapter 9.

There are two types of linear congruential generators: the mixed congruential generator and the multiplicative congruential generator. *Mixed congruential generators* are designed by assigning $c > 0$. *Multiplicative congruential generators* are designed by assigning $c = 0$. The multiplicative generator is more efficient than the mixed generator because it does not require the addition of c. The maximum cycle length for a multiplicative generator can be set within one unit of the maximum cycle length of the mixed generator by carefully selecting values for a and m. From a practical standpoint, the difference in cycle length is insignificant considering that both types of generators can boast cycle lengths of more than 2.1 billion.

ProModel uses the following multiplicative generator:

$$Z_i = (630{,}360{,}016 Z_{i-1}) \bmod (2^{31} - 1)$$

Specifically, it is a prime modulus multiplicative linear congruential generator (PMMLCG) with $a = 630{,}360{,}016$, $c = 0$, and $m = 2^{31} - 1$. It has been extensively tested and is known to be a reliable random number generator for simulation (Law and Kelton 2000). The ProModel implementation of this generator divides the cycle length of $2^{31} - 1 = 2{,}147{,}483{,}647$ into 100 unique streams.

Testing Random Number Generators

When faced with using a random number generator about which you know very little, it is wise to verify that the numbers emanating from it satisfy the two important properties defined at the beginning of this section. The numbers produced by the random number generator must be (1) independent and (2) uniformly distributed between zero and one (uniform(0, 1)). To verify that the generator satisfies these properties, you first generate a sequence of random numbers $U_1, U_2, U_3, \ldots$ and then subject them to an appropriate test of hypothesis.

The hypotheses for testing the independence property are

H_0: U_i values from the generator are independent
H_1: U_i values from the generator are not independent

Several statistical methods have been developed for testing these hypotheses at a specified significance level α. One of the most commonly used methods is the runs test. Banks et al. (2001) review three different versions of the runs test for conducting this independence test. Additionally, two runs tests are implemented in Stat::Fit—the Runs Above and Below the Median Test and the Runs Up and Runs Down Test. Chapter 5 contains additional material on tests for independence.

The hypotheses for testing the uniformity property are

H_0: U_i values are uniform(0, 1)
H_1: U_i values are not uniform(0, 1)

Several statistical methods have also been developed for testing these hypotheses at a specified significance level α. The Kolmogorov-Smirnov test and the chi-square test are perhaps the most frequently used tests. (See Chapter 5 for a description of the chi-square test.) The objective is to determine if the uniform(0, 1) distribution fits or describes the sequence of random numbers produced by the random number generator. These tests are included in the Stat::Fit software and are further described in many introductory textbooks on probability and statistics (see, for example, Johnson 1994).

3.4.2 Generating Random Variates

This section introduces common methods for generating observations (random variates) from probability distributions other than the uniform(0, 1) distribution. For example, the time between arrivals of cars to a restaurant's drive-through window may be exponentially distributed, and the time required for drivers to place orders at the window might follow the lognormal distribution. Observations from these and other commonly used distributions are obtained by transforming the observations generated by the random number generator to the desired distribution. The transformed values are referred to as variates from the specified distribution.

There are several methods for generating random variates from a desired distribution, and the selection of a particular method is somewhat dependent on the characteristics of the desired distribution. Methods include the inverse transformation method, the acceptance/rejection method, the composition method, the convolution method, and the methods employing special properties. The inverse transformation method, which is commonly used to generate variates from both discrete and continuous distributions, is described starting first with the continuous case. For a review of the other methods, see Law and Kelton (2007).

Continuous Distributions

The application of the inverse transformation method to generate random variates from continuous distributions is straightforward and efficient for many continuous distributions. For a given probability density function $f(x)$, find the cumulative distribution function of X. That is, $F(x) = P(X \leq x)$. Next, set $U = F(x)$, where U is uniform(0, 1), and solve for x. Solving for x yields $x = F^{-1}(U)$. The equation $x = F^{-1}(U)$ transforms U into a value for x that conforms to the given distribution $f(x)$.

As an example, suppose that we need to generate variates from the exponential distribution with mean β. The probability density function $f(x)$ and corresponding cumulative distribution function $F(x)$ are

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

$$F(x) = \begin{cases} 1 - e^{-x/\beta} & \text{for } x > 0 \\ 0 & \text{elsewhere} \end{cases}$$

Setting $U = F(x)$ and solving for x yields

$$U = 1 - e^{-x/\beta}$$
$$e^{-x/\beta} = 1 - U$$
$$\ln(e^{-x/\beta}) = \ln(1 - U) \qquad \text{where ln is the natural logarithm}$$
$$-x/\beta = \ln(1 - U)$$
$$x = -\beta \ln(1 - U)$$

The random variate x in the above equation is exponentially distributed with mean β provided U is uniform(0, 1).

Suppose three observations of an exponentially distributed random variable with mean $\beta = 2$ are desired. The next three numbers generated by the random number generator are $U_1 = 0.27$, $U_2 = 0.89$, and $U_3 = 0.13$. The three numbers are transformed into variates x_1, x_2, and x_3 from the exponential distribution with mean $\beta = 2$ as follows:

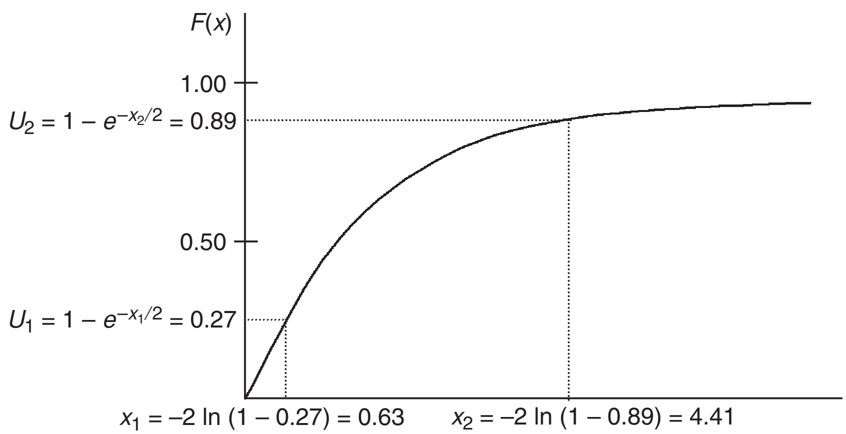
$$x_1 = -2\ln(1 - U_1) = -2\ln(1 - 0.27) = 0.63$$
$$x_2 = -2\ln(1 - U_2) = -2\ln(1 - 0.89) = 4.41$$
$$x_3 = -2\ln(1 - U_3) = -2\ln(1 - 0.13) = 0.28$$

Figure 3.6 provides a graphical representation of the inverse transformation method in the context of this example. The first step is to generate U, where U is uniform(0, 1). Next, locate U on the y axis and draw a horizontal line from that point to the cumulative distribution function $[F(x) = 1 - e^{-x/2}]$. From this point of intersection with $F(x)$, a vertical line is dropped down to the x axis to obtain the corresponding value of the variate. This process is illustrated in Figure 3.6 for generating variates x_1 and x_2 given $U_1 = 0.27$ and $U_2 = 0.89$.

Application of the inverse transformation method is straightforward as long as there is a closed-form formula for the cumulative distribution function, which

FIGURE 3.6

Graphical representation of inverse transformation method for continuous variates.



is the case for many continuous distributions. However, the normal distribution is one exception. Thus it is not possible to solve for a simple equation to generate normally distributed variates. For these cases, there are other methods that can be used to generate the random variates. See, for example, Law (2007) for a description of additional methods for generating random variates from continuous distributions.

Discrete Distributions

The application of the inverse transformation method to generate variates from discrete distributions is basically the same as for the continuous case. The difference is in how it is implemented. For example, consider the following probability mass function:

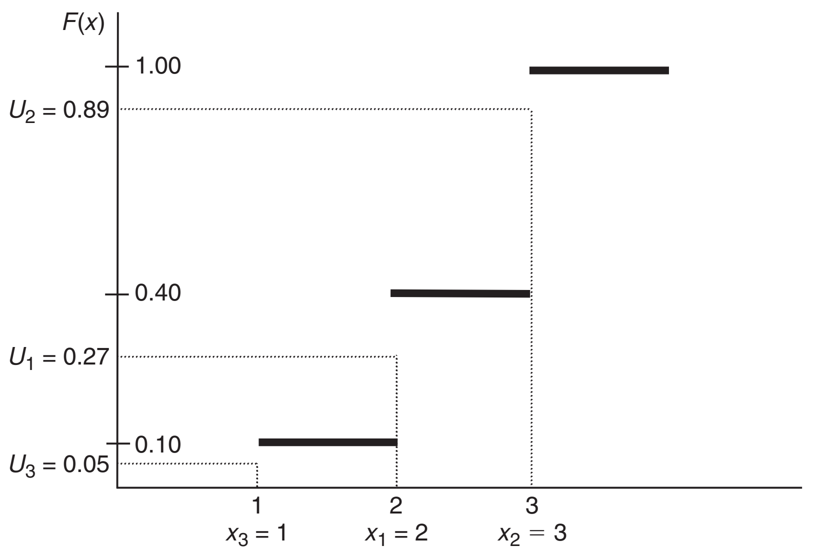
$$p(x) = P(X = x) = \begin{cases} 0.10 & \text{for } x = 1 \\ 0.30 & \text{for } x = 2 \\ 0.60 & \text{for } x = 3 \end{cases}$$

The random variate x has three possible values. The probability that x is equal to 1 is 0.10, $P(X = 1) = 0.10$; $P(X = 2) = 0.30$; and $P(X = 3) = 0.60$. The cumulative distribution function $F(x)$ is shown in Figure 3.7. The random variable x could be used in a simulation to represent the number of defective components on a circuit board or the number of drinks ordered from a drive-through window, for example.

Suppose that an observation from the above discrete distribution is desired. The first step is to generate U, where U is uniform(0, 1). Using Figure 3.7, the value of the random variate is determined by locating U on the y axis, drawing a horizontal line from that point until it intersects with a step in the cumulative function, and then dropping a vertical line from that point to the x axis to read the

FIGURE 3.7

Graphical explanation of inverse transformation method for discrete variates.



value of the variate. This process is illustrated in Figure 3.7 for x_1, x_2, and x_3 given $U_1 = 0.27$, $U_2 = 0.89$, and $U_3 = 0.05$. Equivalently, if $0 \leq U_i \leq 0.10$, then $x_i = 1$; if $0.10 < U_i \leq 0.40$, then $x_i = 2$; if $0.40 < U_i \leq 1.00$, then $x_i = 3$. Note that should we generate 100 variates using the above process, we would expect a value of 3 to be returned 60 times, a value of 2 to be returned 30 times, and a value of 1 to be returned 10 times.

The inverse transformation method can be applied to any discrete distribution by dividing the *y* axis into subintervals as defined by the cumulative distribution function. In our example case, the subintervals were [0, 0.10], (0.10, 0.40], and (0.40, 1.00]. For each subinterval, record the appropriate value for the random variate. Next, develop an algorithm that calls the random number generator to receive a specific value for *U*, searches for and locates the subinterval that contains *U*, and returns the appropriate value for the random variate.

Locating the subinterval that contains a specific *U* value is relatively straightforward when there are only a few possible values for the variate. However, the number of possible values for a variate can be quite large for some discrete distributions. For example, a random variable having 50 possible values could require that 50 subintervals be searched before locating the subinterval that contains *U*. In such cases, sophisticated search algorithms have been developed that quickly locate the appropriate subinterval by exploiting certain characteristics of the discrete distribution for which the search algorithm was designed. A good source of information on the subject can be found in Law (2007).

3.4.3 Generating Random Variates from Common Continuous Distributions

Section 3.4.2 introduced methodologies for generating random variates. In this section, specific equations derived mostly from those methodologies are presented for generating random variates from some common continuous distributions.

Uniform

The continuous uniform distribution is used to generate random variates from all other distributions. By itself, it is used in simulation models when the modeler wishes to make all observations equally likely to occur. It is also used when little is known about the activity being simulated other than the activity's lowest and highest values.

The continuous uniform probability density function is given by

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

with min $= a$ and max $= b$. Using the inverse transformation method, the resulting equation for generating variate *x* from the uniform(a, b) distribution is given by

$$x = a + (b - a)U$$

where *U* is uniform(0, 1).

Triangular

Like the uniform distribution, the triangular distribution is used when little is known about the activity being simulated other than the activity's lowest value, highest value, and most frequently occurring value—the mode.

The triangular probability density function is given by

$$f(x) = \begin{cases} \dfrac{2(x-a)}{(b-a)(m-a)} & \text{for } a \le x \le m \\ \dfrac{2(b-x)}{(b-a)(b-m)} & \text{for } m < x \le b \\ 0 & \text{elsewhere} \end{cases}$$

with min $= a$, max $= b$, and mode $= m$. Using the inverse transformation method, the resulting equation for generating variate x from the triangular(a, m, b) distribution is given by

$$x = \begin{cases} a + \sqrt{(b-a)(m-a)U} & \text{for } 0 \le U \le \dfrac{(m-a)}{(b-a)} \\ b - \sqrt{(b-a)(b-m)(1-U)} & \text{for } \dfrac{(m-a)}{(b-a)} < U \le 1 \end{cases}$$

where U is uniform(0, 1). Note that when the probability density function of the random variable x is defined over separate subintervals with respect to x, then the equation for generating a variate will be defined over separate subintervals with respect to U. Therefore, step one is to generate U. Step two is to determine which subinterval the value of U falls within. And step three is to plug U into the corresponding equation to generate variate x for that subinterval.

Normal

The normal distribution is used to simulate errors such as the diameter of a pipe with respect to its design specification. Some use it to model time such as processing times. However, if this is done, negative values should be discarded.

The normal probability density function is given by

$$f(x) = \left\{ \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/(2\sigma^2)} \qquad \text{for all real } x \right.$$

with mean μ and variance σ^2. A method developed by Box and Muller and reported in Law (2007) involves generating a variate x from the standard normal distribution (mean $\mu = 0$ and variance $\sigma^2 = 1$) and then converting it into a variate x' from the normal distribution with the desired mean and variance using the equation $x' = \mu + \sigma x$.

The method generates two standard normally($\mu = 0, \sigma^2 = 1$) distributed variates x_1 and x_2 and then converts them into normally(μ, σ^2) distributed variates x_1' and x_2'. The method is given by

$$x_1 = \sqrt{-2 \ln U_1} \cos 2\pi U_2$$
$$x_1' = \mu + \sigma x_1$$

and

$$x_2 = \sqrt{-2 \ln U_1} \sin 2\pi U_2$$
$$x_2' = \mu + \sigma x_2$$

where U_1 and U_2 are independent observations from the uniform(0, 1) distribution.

3.4.4. Generating Random Variates from Common Discrete Distributions

Section 3.4.2 introduced methodologies for generating random variates. In this section specific equations derived from those methodologies are presented for generating random variates from some common discrete distributions.

Uniform

Like its continuous cousin, the discrete (consecutive integers) uniform distribution is used in simulation models when the modeler wishes to make all observations equally likely to occur. The difference is that the observations from a discrete uniform distribution are integer values such as the values achieved by rolling a single die. It is also used when little is known about the activity being simulated other than the activity's lowest and highest values.

The discrete uniform probability mass function is given by

$$p(x) = \begin{cases} \dfrac{1}{b - a + 1} & \text{for } x = a, a + 1, \ldots, b \\ 0 & \text{elsewhere} \end{cases}$$

with min $= a$ and max $= b$. Using the inverse transformation method, the resulting equation for generating discrete uniform(a, b) variate x is given by

$$x = a + \lfloor (b - a + 1)U \rfloor$$

where U is continuous uniform(0, 1) and $\lfloor t \rfloor$ denotes the largest integer $\leq t$.

Bernoulli

The bernoulli distribution is used to model random events with two possible outcomes (e.g., a failure or a success). It is defined by the probability of getting a success p. If, for example, the probability that a machine produces a good part (one that meets specifications) is p each time the machine makes a part, the bernoulli distribution would be used to simulate if a part is good or bad.

The bernoulli probability mass function is given by

$$p(x) = \begin{cases} 1 - p & \text{for } x = 0 \\ p & \text{for } x = 1 \\ 0 & \text{elsewhere} \end{cases}$$

with probability of success p. Motivated by the inverse transformation method, the equation for generating bernoulli(p) variate x is given by

$$x = \begin{cases} 1 & \text{for } U \leq p \\ 0 & \text{elsewhere} \end{cases}$$

where U is uniform(0, 1).

Geometric

The geometric distribution is used to model random events in which the modeler is interested in simulating the number of independent bernoulli trials with probability of success p on each trial needed before the first success occurs. In other words if x represents failures, how many failures will occur before the first success? In the context of an inspection problem with a success denoted as a bad item, how many good items will be built before the first bad item is produced?

The geometric probability mass function is given by

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

with x number of independent bernoulli trials each with probability of success p occurring before the first success is realized. Motivated by the inverse transformation method, the equation for generating geometric(p) variate x is given by

$$x = \lfloor \ln U / \ln(1-p) \rfloor$$

where U is uniform(0, 1).

3.5 Simple Spreadsheet Simulation

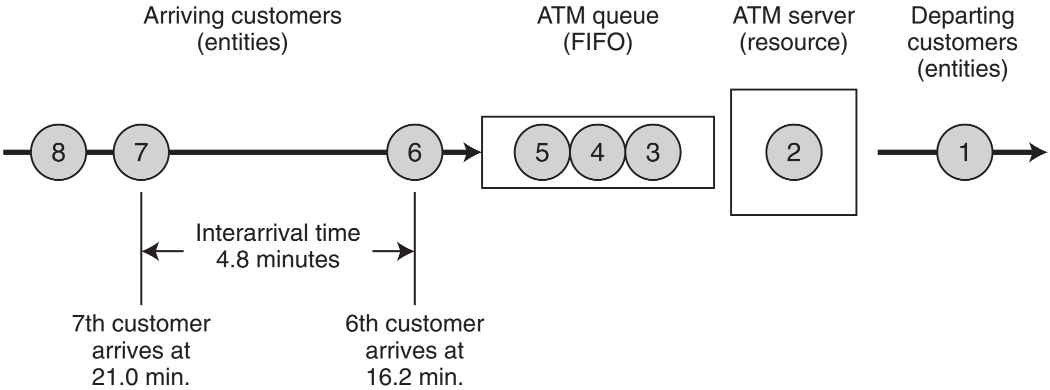
This chapter has covered some really useful information, and it will be constructive to pull it all together for our first dynamic, stochastic simulation model. The simulation will be implemented as a spreadsheet model because the example system is not so complex as to require the use of a commercial simulation software product. Furthermore, a spreadsheet will provide a nice tabulation of the random events that produce the output from the simulation, thereby making it ideal for demonstrating how simulations of dynamic, stochastic systems are often accomplished. The system to be simulated follows.

Customers arrive to use an automatic teller machine (ATM) at a mean interarrival time of 3.0 minutes exponentially distributed. When customers arrive to the system they join a queue to wait for their turn on the ATM. The queue has the capacity to hold an infinite number of customers. Customers spend an average of 2.4 minutes exponentially distributed at the ATM to complete their transactions, which is called the service time at the ATM. Simulate the system for the arrival and processing of 25 customers and estimate the expected waiting time for customers in the queue (the average time customers wait in line for the ATM) and the expected time in the system (the average time customers wait in the queue plus the average time it takes them to complete their transaction at the ATM).

Using the systems terminology introduced in Chapter 2 to describe the ATM system, the entities are the customers that arrive to the ATM for processing. The resource is the ATM that serves the customers, which has the capacity to serve one customer at a time. The system controls that dictate how, when, and where activities are performed for this ATM system consist of the queuing discipline, which is first-in, first-out (FIFO). Under the FIFO queuing discipline the entities (customers) are processed by the resource (ATM) in the order that they arrive to

FIGURE 3.8

Descriptive drawing of the automatic teller machine (ATM) system.



the queue. Figure 3.8 illustrates the relationships of the elements of the system with the customers appearing as dark circles.

Our objective in building a spreadsheet simulation model of the ATM system is to estimate the average amount of time the 25 customers will spend waiting in the queue and the average amount of time the customers will spend in the system. To accomplish our objective, the spreadsheet will need to generate random numbers and random variates, to contain the logic that controls how customers are processed by the system, and to compute an estimate of system performance measures. The spreadsheet simulation is shown in Table 3.2 and is divided into three main sections: Arrivals to ATM, ATM Processing Time, and ATM Simulation Logic. The Arrivals to ATM and ATM Processing Time provide the foundation for the simulation, while the ATM Simulation Logic contains the spreadsheet programming that mimics customers flowing through the ATM system. The last row of the Time in Queue column and the Time in System column under the ATM Simulation Logic section contains one observation of the average time customers wait in the queue, 1.94 minutes, and one observation of their average time in the system, 4.26 minutes. The values were computed by averaging the 25 customer time values in the respective columns.

Do you think customer number 17 (see the Customer Number column) became upset over having to wait in the queue for 6.11 minutes to conduct a 0.43 minute transaction at the ATM (see the Time in Queue column and the Service Time column under ATM Simulation Logic)? Has something like this ever happened to you in real life? Simulation can realistically mimic the behavior of a system. More time will be spent interpreting the results of our spreadsheet simulation after we understand how it was put together.

3.5.1 Simulating Random Variates

The interarrival time is the elapsed time between customer arrivals to the ATM. This time changes according to the exponential distribution with a mean of

Table 3.2 Spreadsheet Simulation of Automatic Teller Machine (ATM)

	Arrivals to ATM			ATM Processing Time			ATM Simulation Logic						
i	Stream 1 ($Z1_i$)	Random Number ($U1_i$)	Interarrival Time ($X1_i$)	Stream 2 ($Z2_i$)	Random Number ($U2_i$)	Service Time ($X2_i$)	Customer Number (1)	Arrival Tone (2)	Begin Service Time (3)	Service Time (4)	Departure Time (5) = (3) + (4)	Time in Queue (6) = (3) − (2)	Time in System (7) = (5) − (2)
0	3			122									
1	66	0.516	2.18	5	0.039	0.10	1	2.18	2.18	0.10	2.28	0.00	0.10
2	109	0.852	5.73	108	0.844	4.46	2	7.91	7.91	4.46	12.37	0.00	4.46
3	116	0.906	7.09	95	0.742	3.25	3	15.00	15.00	3.25	18.25	0.00	3.25
4	7	0.055	0.17	78	0.609	2.25	4	15.17	18.25	2.25	20.50	3.08	5.33
5	22	0.172	0.57	105	0.820	4.12	5	15.74	20.50	4.12	24.62	4.76	8.88
6	81	0.633	3.01	32	0.250	0.69	6	18.75	24.62	0.69	25.31	5.87	6.56
7	40	0.313	1.13	35	0.273	0.77	7	19.88	25.31	0.77	26.08	5.43	6.20
8	75	0.586	2.65	98	0.766	3.49	8	22.53	26.08	3.49	29.57	3.55	7.04
9	42	0.328	1.19	13	0.102	0.26	9	23.72	29.57	0.26	29.83	5.85	6.11
10	117	0.914	7.36	20	0.156	0.41	10	31.08	31.08	0.41	31.49	0.00	0.41
11	28	0.219	0.74	39	0.305	0.87	11	31.82	31.82	0.87	32.69	0.00	0.87
12	79	0.617	2.88	54	0.422	1.32	12	34.70	34.70	1.32	36.02	0.00	1.32
13	126	0.984	12.41	113	0.883	5.15	13	47.11	47.11	5.15	52.26	0.00	5.15
14	89	0.695	3.56	72	0.563	1.99	14	50.67	52.26	1.99	54.25	1.59	3.58
15	80	0.625	2.94	107	0.836	4.34	15	53.61	54.25	4.34	58.59	0.64	4.98
16	19	0.148	0.48	74	0.578	2.07	16	54.09	58.59	2.07	60.66	4.50	6.57
17	18	0.141	0.46	21	0.164	0.43	17	54.55	60.66	0.43	61.09	6.11	6.54
18	125	0.977	11.32	60	0.469	1.52	18	65.87	65.87	1.52	67.39	0.00	1.52
19	68	0.531	2.27	111	0.867	4.84	19	68.14	68.14	4.84	72.98	0.00	4.84
20	23	0.180	0.60	30	0.234	0.64	20	68.74	72.98	0.64	73.62	4.24	4.88
21	102	0.797	4.78	121	0.945	6.96	21	73.52	73.62	6.96	80.58	0.10	7.06
22	97	0.758	4.26	112	0.875	4.99	22	77.78	80.58	4.99	85.57	2.80	7.79
23	120	0.938	8.34	51	0.398	1.22	23	86.12	86.12	1.22	87.34	0.00	1.22
24	91	0.711	3.72	50	0.391	1.19	24	89.84	89.84	1.19	91.03	0.00	1.19
25	122	0.953	9.17	29	0.227	0.62	25	99.01	99.01	0.62	99.63	0.00	0.62
											Average	1.94	4.26

3.0 minutes. That is, the time that elapses between one customer arrival and the next is not the same from one customer to the next but averages out to be 3.0 minutes. This is illustrated in Figure 3.8 in that the interarrival time between customers 7 and 8 is much smaller than the interarrival time of 4.8 minutes between customers 6 and 7. The service time at the ATM is also a random variable following the exponential distribution and averages 2.4 minutes. To simulate this stochastic system, a random number generator is needed to produce observations (random variates) from the exponential distribution. The inverse transformation method was used in Section 3.4.2 just for this purpose.

The transformation equation is

$$X_i = -\beta \ln(1 - U_i) \qquad \text{for } i = 1, 2, 3, \ldots, 25$$

where X_i represents the ith value realized from the exponential distribution with mean β, and U_i is the ith random number drawn from a uniform(0, 1) distribution. The $i = 1, 2, 3, \ldots, 25$ indicates that we will compute 25 values from the transformation equation. However, we need to have two different versions of this equation to generate the two sets of 25 exponentially distributed random variates needed to simulate 25 customers because the mean interarrival time of $\beta = 3.0$ minutes is different than the mean service time of $\beta = 2.4$ minutes. Let $X1_i$ denote the interarrival time and $X2_i$ denote the service time generated for the ith customer simulated in the system. The equation for transforming a random number into an interarrival time observation from the exponential distribution with mean $\beta = 3.0$ minutes becomes

$$X1_i = -3.0 \ln(1 - U1_i) \qquad \text{for } i = 1, 2, 3, \ldots, 25$$

where $U1_i$ denotes the ith value drawn from the random number generator using Stream 1. This equation is used in the Arrivals to ATM section of Table 3.2 under the Interarrival Time ($X1_i$) column.

The equation for transforming a random number into an ATM service time observation from the exponential distribution with mean $\beta = 2.4$ minutes becomes

$$X2_i = -2.4 \ln(1 - U2_i) \qquad \text{for } i = 1, 2, 3, \ldots, 25$$

where $U2_i$ denotes the ith value drawn from the random number generator using Stream 2. This equation is used in the ATM Processing Time section of Table 3.2 under the Service Time ($X2_i$) column.

Let's produce the sequence of $U1_i$ values that feeds the transformation equation ($X1_i$) for interarrival times using a linear congruential generator (LCG) similar to the one used in Table 3.1. The equations are

$$Z1_i = (21Z1_{i-1} + 3) \bmod (128)$$

$$U1_i = Z1_i/128 \qquad \text{for } i = 1, 2, 3, \ldots, 25$$

The authors defined Stream 1's starting or seed value to be 3. So we will use $Z1_0 = 3$ to kick off this stream of 25 random numbers. These equations are used in the Arrivals to ATM section of Table 3.2 under the Stream 1 ($Z1_i$) and Random Number ($U1_i$) columns.

Likewise, we will produce the sequence of $U2_i$ values that feeds the transformation equation ($X2_i$) for service times using

$$Z2_i = (21Z2_{i-1} + 3) \bmod (128)$$
$$U2_i = Z2_i/128 \qquad \text{for } i = 1, 2, 3, \ldots, 25$$

and will specify a starting seed value of $Z2_0 = 122$, Stream 2's seed value, to kick off the second stream of 25 random numbers. These equations are used in the ATM Processing Time section of Table 3.2 under the Stream 2 ($Z2_i$) and Random Number ($U2_i$) columns.

The spreadsheet presented in Table 3.2 illustrates 25 random variates for both the interarrival time, column ($X1_i$), and service time, column ($X2_i$). All time values are given in minutes in Table 3.2. To be sure we pull this together correctly, let's compute a couple of interarrival times with mean $\beta = 3.0$ minutes and compare them to the values given in Table 3.2.

Given $Z1_0 = 3$

$$Z1_1 = (21Z1_0 + 3) \bmod (128) = (21(3) + 3) \bmod (128)$$
$$= (66) \bmod (128) = 66$$
$$U1_1 = Z1_1/128 = 66/128 = 0.516$$
$$X1_1 = -\beta \ln(1 - U1_1) = -3.0 \ln(1 - 0.516) = 2.18 \text{ minutes}$$

The value of 2.18 minutes is the first value appearing under the column, Interarrival Time ($X1_i$) To compute the next interarrival time value $X1_2$, we start by using the value of $Z1_1$ to compute $Z1_2$.

Given $Z1_1 = 66$

$$Z1_2 = (21Z1_1 + 3) \bmod (128) = (21(66) + 3) \bmod (128) = 109$$
$$U1_2 = Z1_2/128 = 109/128 = 0.852$$
$$X1_2 = -3 \ln(1 - U1_2) = -3.0 \ln(1 - 0.852) = 5.73 \text{ minutes}$$

Figure 3.9 illustrates how the equations were programmed in Microsoft Excel for the Arrivals to ATM section of the spreadsheet. Note that the $U1_i$ and $X1_i$ values in Table 3.2 are rounded to three and two places to the right of the decimal, respectively. The same rounding rule is used for $U2_i$ and $X2_i$.

It would be useful for you to verify a few of the service time values with mean $\beta = 2.4$ minutes appearing in Table 3.2 using

$$Z2_0 = 122$$
$$Z2_i = (21Z2_{i-1} + 3) \bmod (128)$$
$$U2_i = Z2_i/128$$
$$X2_i = -2.4 \ln(1 - U2_i) \qquad \text{for } i = 1, 2, 3, \ldots$$

The equations started out looking a little difficult to manipulate but turned out not to be so bad when we put some numbers in them and organized them in a

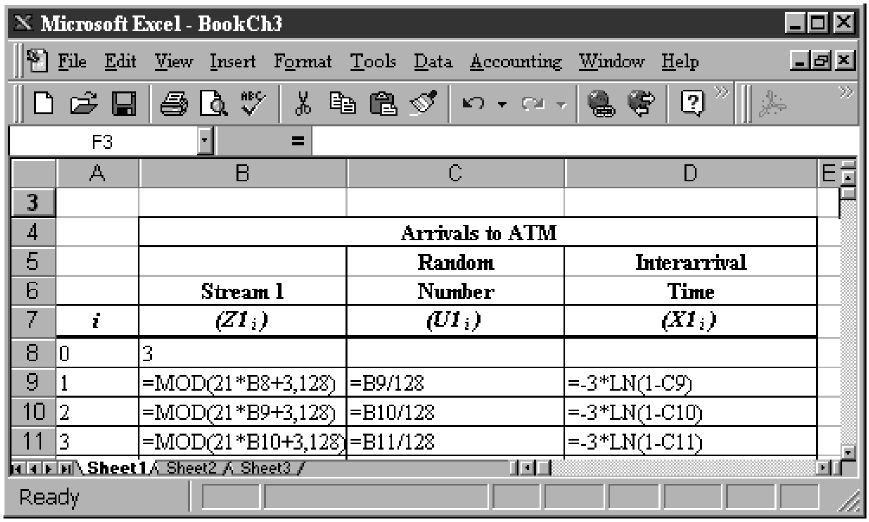


FIGURE 3.9
Microsoft Excel snapshot of the ATM spreadsheet illustrating the equations for the Arrivals to ATM *section.*

spreadsheet—though it was a bit tedious. The important thing to note here is that although it is transparent to the user, ProModel uses a very similar method to produce exponentially distributed random variates, and you now understand how it is done.

The LCG just given has a maximum cycle length of 128 random numbers (you may want to verify this), which is more than enough to generate 25 interarrival time values and 25 service time values for this simulation. However, it is a poor random number generator compared to the one used by ProModel. It was chosen because it is easy to program into a spreadsheet and to compute by hand to facilitate our understanding. The biggest difference between it and the random number generator in ProModel is that the ProModel random number generator manipulates much larger numbers to pump out a much longer stream of numbers that pass all statistical tests for randomness.

Before moving on, let's take a look at why we chose $Z1_0 = 3$ and $Z2_0 = 122$. Our goal was to make sure that we did not use the same uniform(0, 1) random number to generate both an interarrival time and a service time. If you look carefully at Table 3.2, you will notice that the seed value $Z2_0 = 122$ is the $Z1_{25}$ value from random number Stream 1. Stream 2 was merely defined to start where Stream 1 ended. Thus our spreadsheet used a unique random number to generate each interarrival and service time. Now let's add the necessary logic to our spreadsheet to conduct the simulation of the ATM system.

3.5.2 Simulating Dynamic, Stochastic Systems

The heart of the simulation is the generation of the random variates that drive the stochastic events in the simulation. However, the random variates are simply a list of numbers at this point. The spreadsheet section labeled ATM Simulation Logic in Table 3.2 is programmed to coordinate the execution of the events to mimic the processing of customers through the ATM system. The simulation program must keep up with the passage of time to coordinate the events. The word *dynamic* appearing in the title for this section refers to the fact that the simulation tracks the passage of time.

The first column under the ATM Simulation Logic section of Table 3.2, labeled Customer Number, is simply to keep a record of the order in which the 25 customers are processed, which is FIFO. The numbers appearing in parentheses under each column heading are used to illustrate how different columns are added or subtracted to compute the values appearing in the simulation.

The Arrival Time column denotes the moment in time at which each customer arrives to the ATM system. The first customer arrives to the system at time 2.18 minutes. This is the first interarrival time value ($X1_1 = 2.18$) appearing in the Arrival to ATM section of the spreadsheet table. The second customer arrives to the system at time 7.91 minutes. This is computed by taking the arrival time of the first customer of 2.18 minutes and adding to it the next interarrival time of $X1_2 = 5.73$ minutes that was generated in the Arrival to ATM section of the spreadsheet table. The arrival time of the second customer is $2.18 + 5.73 = 7.91$ minutes. The process continues with the third customer arriving at $7.91 + 7.09 = 15.00$ minutes.

The trickiest piece to program into the spreadsheet is the part that determines the moment in time at which customers begin service at the ATM after waiting in the queue. Therefore, we will skip over the Begin Service Time column for now and come back to it after we understand how the other columns are computed.

The Service Time column simply records the simulated amount of time required for the customer to complete their transaction at the ATM. These values are copies of the service time $X2_i$ values generated in the ATM Processing Time section of the spreadsheet.

The Departure Time column records the moment in time at which a customer departs the system after completing their transaction at the ATM. To compute the time at which a customer departs the system, we take the time at which the customer gained access to the ATM to begin service, column (3), and add to that the length of time the service required, column (4). For example, the first customer gained access to the ATM to begin service at 2.18 minutes, column (3). The service time for the customer was determined to be 0.10 minutes in column (4). So, the customer departs 0.10 minutes later or at time $2.18 + 0.10 = 2.28$ minutes. This customer's short service time must be because they forgot their PIN number and could not conduct their transaction.

The Time in Queue column records how long a customer waits in the queue before gaining access to the ATM. To compute the time spent in the queue, we take the time at which the ATM began serving the customer, column (3), and subtract from that the time at which the customer arrived to the system, column (2). The fourth customer arrives to the system at time 15.17 and begins getting service from the ATM at 18.25 minutes; thus, the fourth customer's time in the queue is $18.25 - 15.17 = 3.08$ minutes.

The Time in System column records how long a customer was in the system. To compute the time spent in the system, we subtract the customer's departure time, column (5), from the customer's arrival time, column (2). The fifth customer arrives to the system at 15.74 minutes and departs the system at 24.62 minutes. Therefore, this customer spent 24.62 − 15.74 = 8.88 minutes in the system.

Now let's go back to the Begin Service Time column, which records the time at which a customer begins to be served by the ATM. The very first customer to arrive to the system when it opens for service advances directly to the ATM. There is no waiting time in the queue; thus the value recorded for the time that the first customer begins service at the ATM is the customer's arrival time. With the exception of the first customer to arrive to the system, we have to capture the logic that a customer cannot begin service at the ATM until the previous customer using the ATM completes his or her transaction. One way to do this is with an IF statement as follows:

```
IF (Current Customer's Arrival Time < Previous Customer's
  Departure Time)
THEN (Current Customer's Begin Service Time = Previous Customer's
  Departure Time)
ELSE (Current Customer's Begin Service Time = Current Customer's
  Arrival Time)
```

Figure 3.10 illustrates how the IF statement was programmed in Microsoft Excel. The format of the Excel IF statement is

```
IF (logical test, use this value if test is true, else use this
  value if test is false)
```

The Excel spreadsheet cell L10 (column L, row 10) in Figure 3.10 is the Begin Service Time for the second customer to arrive to the system and is programmed with `IF(K10<N9,N9,K10)`. Since the second customer's arrival time (Excel cell K10) is not less than the first customer's departure time (Excel cell N9), the logical test evaluates to "false" and the second customer's time to begin service is set to his or her arrival time (Excel cell K10). The fourth customer shown in Figure 3.10 provides an example of when the logical test evaluates to "true," which results in the fourth customer beginning service when the third customer departs the ATM.

FIGURE 3.10

Microsoft Excel snapshot of the ATM spreadsheet illustrating the IF statement for the Begin Service Time *column.*

File Edit View Insert Format Tools Data Window Help

L10 = =IF(K10<N9,N9,K10)

ATM Simulation Logic						
Customer Number	Arrival Time	Begin Service Time	Service Time	Departure Time	Time in Queue	Time in System
(1)	(2)	(3)	(4)	(5)=(3)+(4)	(6)=(3)-(2)	(7)=(5)-(2)
1	2.18	2.18	0.10	2.28	0.00	0.10
2	7.91	7.91	4.46	12.37	0.00	4.46
3	15.00	15.00	3.25	18.25	0.00	3.25
4	15.17	18.25	2.25	20.50	3.08	5.33
5	15.74	20.50	4.12	24.62	4.76	8.88

3.5.3 Simulation Replications and Output Analysis

The spreadsheet model makes a nice simulation of the first 25 customers to arrive to the ATM system. And we have a simulation output observation of 1.94 minutes that could be used as an estimate for the average time that the 25 customers waited in the queue (see the last value under the Time in Queue column, which represents the average of the 25 individual customer queue times). We also have a simulation output observation of 4.26 minutes that could be used as an estimate for the average time the 25 customers were in the system. These results represent only one possible value of each performance measure. Why? If at least one input variable to the simulation model is random, then the output of the simulation model will also be random. The interarrival times of customers to the ATM system and their service times are random variables. Thus the output of the simulation model of the ATM system is also a random variable. Before we bet that the average time customers spend in the system each day is 4.26 minutes, we may want to run the simulation model again with a new sample of interarrival times and service times to see what happens. This would be analogous to going to the real ATM on a second day to replicate our observing the first 25 customers processed to compute a second observation of the average time that the 25 customers spend in the system. In simulation, we can accomplish this by changing the values of the seed numbers used for our random number generator.

Changing the seed values $Z1_0$ and $Z2_0$ causes the spreadsheet program to recompute all values in the spreadsheet. When we change the seed values $Z1_0$ and $Z2_0$ appropriately, we produce another replication of the simulation. When we run replications of the simulation, we are driving the simulation with a set of random numbers never used by the simulation before. This is analogous to the fact that the arrival pattern of customers to the real ATM and their transaction times at the ATM will not likely be the same from one day to the next. If $Z1_0 = 29$ and $Z2_0 = 92$ are used to start the random number generator for the ATM simulation model, a new replication of the simulation will be computed that produces an average time in queue of 0.84 minutes and an average time in system of 2.36 minutes. Review question number eight at the end of the chapter asks you to verify this second replication.

Table 3.3 contains the results from the two replications. Obviously, the results from this second replication are very different than those produced by the first replication. Good thing we did not make bets on how much time customers spend in the system based on the output of the simulation for the first

TABLE 3.3 Summary of ATM System Simulation Output

Replication	*Average Time in Queue*	*Average Time in System*
1	1.94 minutes	4.26 minutes
2	0.84 minutes	2.36 minutes
Average	1.39 minutes	3.31 minutes

replication only. Statistically speaking, we should get a better estimate of the average time customers spend in queue and the average time they are in the system if we combine the results from the two replications into an overall average. Doing so yields an estimate of 1.39 minutes for the average time in queue and an estimate of 3.31 minutes for the average time in system (see Table 3.3). However, the large variation in the output of the two simulation replications indicates that more replications are needed to get reasonably accurate estimates. How many replications are enough? You will know how to answer the question upon completing Chapter 8.

While spreadsheet technology is effective for conducting Monte Carlo simulations and simulations of simple dynamic systems, it is ineffective and inefficient as a simulation tool for complex systems. Discrete-event simulation software technology was designed especially for mimicking the behavior of complex systems and is the subject of Chapter 4.

3.6 Summary

Modeling random behavior begins with transforming the output produced by a random number generator into observations (random variates) from an appropriate statistical distribution. The values of the random variates are combined with logical operators in a computer program to compute output that mimics the performance behavior of stochastic systems. Performance estimates for stochastic simulations are obtained by calculating the average value of the performance metric across several replications of the simulation. Models can realistically simulate a variety of systems.

3.7 Review Questions

1. What is the difference between a stochastic model and a deterministic model in terms of the input variables and the way results are interpreted?
2. Give an example of a discrete-change state variable and a continuous-change state variable.
3. For each of the following simulation applications identify one discrete- and one continuous-change state variable.
   a. Inventory control of an oil storage and pumping facility.
   b. Study of beverage production in a soft drink production facility.
   c. Study of congestion in a busy traffic intersection.
4. Give a statistical description of the numbers produced by a random number generator.
5. What are the two statistical properties that random numbers must satisfy?

6. Given these two LCGs:

$$Z_i = (9Z_{i-1} + 3) \bmod (32)$$
$$Z_i = (12Z_{i-1} + 5) \bmod (32)$$

   a. Which LCG will achieve its maximum cycle length? Answer the question without computing any Z_i values.
   b. Compute Z_1 through Z_5 from a seed of 29 ($Z_0 = 29$) for the second LCG.
7. What is a random variate, and how are random variates generated?
8. Apply the inverse transformation method to generate three variates from the following distributions using $U_1 = 0.10$, $U_2 = 0.53$, and $U_3 = 0.15$.
   a. Probability density function:

$$f(x) = \begin{cases} \dfrac{1}{\beta - \alpha} & \text{for } \alpha \leq x \leq \beta \\ 0 & \text{elsewhere} \end{cases}$$

   where $\beta = 7$ and $\alpha = 4$.
   b. Probability mass function:

$$p(x) = P(X = x) = \begin{cases} \dfrac{x}{15} & \text{for } x = 1, 2, 3, 4, 5 \\ 0 & \text{elsewhere} \end{cases}$$

9. How would a random number generator be used to simulate a 12 percent chance of rejecting a part because it is defective?
10. Reproduce the spreadsheet simulation of the ATM system presented in Section 3.5. Set the random numbers seeds $Z1_0 = 29$ and $Z2_0 = 92$ to compute the average time customers spend in the queue and in the system.
    a. Verify that the average time customers spend in the queue and in the system match the values given for the second replication in Table 3.3.
    b. Verify that the resulting random number Stream 1 and random number Stream 2 are completely different than the corresponding streams in Table 3.2. Is this a requirement for a new replication of the simulation?
11. Using the spreadsheet simulation from problem 10 above, replace the exponentially distributed interarrival time with mean 3.0 minutes with an interarrival time that is uniformly distributed (continuous case) with mean 3.0 minutes by setting the uniform distribution's minimum value to 1.0 minute and the maximum value to 5.0 minutes. How did the change influence the average time in queue and average time in system as compared to the second replication results shown in Table 3.3, which are based on the exponentially distributed interarrival time with mean 3.0 minutes?

12. Given $U_t = 0.23$ and $U_z = 0.61$ from a uniform(0, 1) distribution, generate two random variates from each of the distributions following.
    a. Uniform(12, 20) continuous case
    b. Triangular(12, 20, 16)
    c. Normal(16, 10)
    d. Uniform(12, 20) discrete case
    e. Bernoulli(0.50)

References

Banks, Jerry; John S. Carson II; Barry L. Nelson; and David M. Nicol. *Discrete-Event System Simulation.* Englewood Cliffs, NJ: Prentice Hall, 2001.

Hoover, Stewart V., and Ronald F. Perry. *Simulation: A Problem-Solving Approach.* Reading, MA: Addison-Wesley, 1989.

Johnson, R. A. *Miller and Freund's Probability and Statistics for Engineers.* 8th ed. Englewood Cliffs, NJ: Prentice Hall, 2010.

Law, Averill M. *Simulation Modeling and Analysis* 4th ed. New York: McGraw-Hill, 2007.

L'Ecuyer, P. "Random Number Generation." In *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice,* ed. J. Banks, pp. 93–137. New York: John Wiley & Sons, 1998.

Pooch, Udo W., and James A. Wall. *Discrete Event Simulation: A Practical Approach*. Boca Raton, FL: CRC Press, 1993.

Pritsker, A. A. B. *Introduction to Simulation and SLAM II*. 4th ed. New York: John Wiley & Sons, 1995.

Ross, Sheldon M.A. *Simulation.* 4th ed. Burlington, MA: Elsevier Academic Press, 2006.

Shannon, Robert *E. System Simulation: The Art and Science.* Englewood Cliffs, NJ: Prentice Hall, 1975.

Thesen, Arne, and Laurel E. Travis. *Simulation for Decision Making*. Minneapolis, MN: West Publishing, 1992.

Widman, Lawrence E.; Kenneth A. Loparo; and Norman R. Nielsen. *Artificial Intelligence, Simulation, and Modeling*. New York: John Wiley & Sons, 1989.