Proceedings of the 2007 Winter Simulation Conference
S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, eds.

INTRODUCTION TO SIMULATION

David Goldsman

H. Milton Stewart School of
Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, GA 30332, U.S.A.

ABSTRACT

This is an introductory tutorial on the statistical aspects of computer simulation, and is intended to serve as a springboard to many of the other introductory tutorials that appear elsewhere in the *Proceedings*. We present a number of motivational examples, followed by material on random number and random variate generation, input analysis of the random variables that drive a simulation, and output analysis of the random observations that a simulation produces.

1 INTRODUCTION

One of the most useful tools in the arsenal of an operations research / industrial engineering / management science analyst is that of computer simulation. A simulation is simply an imitation of the operation of a real-world system for purposes of evaluating that system. Over the last thirty years, computer simulation has enjoyed a great deal of popularity in the manufacturing, production, logistics, service, and financial industries, just to name a few areas of application. Simulations are often used to analyze systems that are too complicated to attack via analytic methods such as calculus, standard probability and statistics, or queueing theory.

The simulations that we are primarily interested in here are usually

- dynamic — i.e., the system state changes over time
- discrete — i.e., the system state changes as the result of discrete events such as customer arrivals or departures
- stochastic (as opposed to deterministic).

The stochastic nature of simulation prompts our ensuing discussion.

This tutorial is organized as follows. It begins in Section 2 with some simple motivational examples designed to show how one can apply simulation to answer interesting questions about stochastic systems. These examples invariably involve the generation of random variables to drive the simulation, e.g., customer interarrival times and service times. The subject of Section 3 is the development of techniques to generate the necessary random variables. One must take great care when selecting the specific probability distributions to represent the random variables that drive the simulation. For example, a poor choice of distribution for customer interarrival times may render the simulation worthless. Therefore, we discuss the problem of simulation input analysis in Section 4. After a simulation run is completed, the experimenter ought to conduct a rigorous analysis of the resulting output, a task made difficult since simulation output observations — for example, customer waiting times — are almost never independent or identically distributed. Issues related to output analysis are studied in Section 5. A particularly attractive feature of computer simulation is its ability to allow the experimenter to analyze and compare certain scenarios quickly and efficiently. Section 6 discusses methods for reducing the variance of estimators arising from a single scenario, thus resulting in more-precise statements about system performance, at no additional cost in simulation run time. We also extend this work by mentioning methods for selecting the best of a number of competing scenarios.

We point out here that excellent general references for the topic of stochastic simulation are Banks, et al. (2005) and Law (2007). In addition, parts of the current paper appeared in Hines et al. (2003).

2 MOTIVATIONAL EXAMPLES

This section illustrates the use of simulation through a series of simple, motivational examples. The goal is to show how one uses random variables within a simulation to answer questions about the underlying stochastic system.

Example 1 *Coin Flipping.* We are interested in simulating independent flips of a fair coin. Of course, this is a trivial sequence of Bernoulli trials with success probability

$p = 1/2$, but this example serves to show how one can use simulation to analyze such a system. First of all, we need to generate realizations of heads (H) and tails (T), each with probability 1/2. Assuming that the simulation can somehow produce a sequence of independent uniform (0,1) random numbers, $U_1, U_2, \ldots$, we will arbitrarily designate flip i as H if we observe $U_i < 0.5$, and a flip as T if we observe $U_i \geq 0.5$. How one generates independent uniforms is the subject of Section 3. In any case, suppose that the following uniforms are observed.

0.32	0.41	0.06	0.93	0.82
0.49	0.21	0.77	0.71	0.08

This sequence of uniforms corresponds to the outcomes

H H H T T H H T T H.

This type of "static" simulation, in which we simply repeat the same type of trials over and over has come to be known as Monte Carlo simulation, in honor of the European city-state, where gambling is apparently a popular recreational activity. □

Example 2 *Estimate* π. In this example, we will estimate π using Monte Carlo simulation in conjunction with a simple geometric relation. Referring to Figure 1, consider a unit square with an inscribed circle, both centered at (1/2,1/2). If one were to throw darts randomly at the square, the probability that a particular dart will land in the circle is $\pi/4$, the ratio of the circle's area to that of the square.

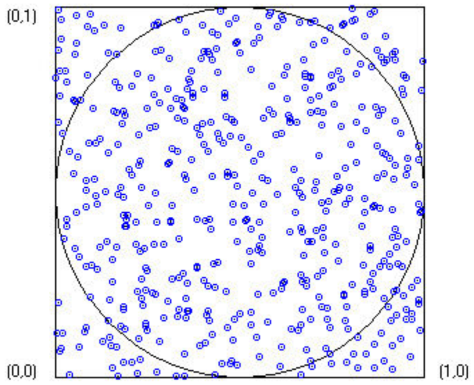

Figure 1: Depiction of dart tosses to estimate π from Example 2.

How can we use this simple fact to estimate π? We shall use Monte Carlo simulation to throw many darts at the square. Specifically, generate independent pairs of independent uniform (0,1) random variables, $(U_{11}, U_{12}), (U_{21}, U_{22}), (U_{31}, U_{32}), \ldots$. These pairs will fall randomly on the square. If, for pair i, it happens that

$$(U_{i1} - 1/2)^2 + (U_{i2} - 1/2)^2 \leq 1/4, \qquad (1)$$

then that pair will fall within the circle. Suppose we run the experiment for n pairs (darts). Let $X_i = 1$ if pair i satisfies inequality (1), i.e., if the ith dart falls in the circle; otherwise, let $X_i = 0$. Now count up the number of darts $Y = \sum_{i=1}^{n} X_i$ falling in the circle. Clearly, Y has the binomial distribution with parameters n and $p = \pi/4$. Then the proportion $\hat{p} = Y/n$ is the maximum likelihood estimate for $p = \pi/4$; and so the maximum likelihood estimator for π is just $\hat{\pi} = 4\hat{p}$. If, for instance, we conducted $n = 1000$ trials and observed $Y = 753$ darts in the circle, our estimate would be $\hat{\pi} = 3.12$. □

Example 3 *Monte Carlo Integration.* Another interesting use of computer simulation involves Monte Carlo integration. Usually, the method becomes efficacious only for high-dimensional integrals, but we will fall back to the basic one-dimensional case for ease of exposition. To this end, consider the integral

$$I = \int_a^b f(x)\,dx = (b-a)\int_0^1 f(a + (b-a)u)\,du. \qquad (2)$$

As in Figure 2, we shall estimate the value of this integral by summing up n rectangles, each of width $(b-a)/n$ centered randomly at point U_i on [0,1], and of height $f(a + (b-a)U_i)$. Then an estimate for I is

$$\hat{I}_n = \frac{b-a}{n} \sum_{i=1}^{n} f(a + (b-a)U_i). \qquad (3)$$

One can show that $\hat{I}_n$ is an unbiased estimator for I, i.e., $\mathsf{E}[\hat{I}_n] = I$ for all n. This makes $\hat{I}_n$ an intuitive and attractive estimator.

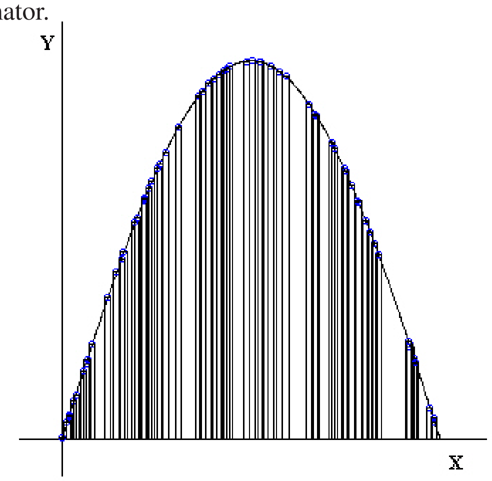

Figure 2: Monte Carlo integration from Example 3.

To illustrate, suppose that we want to estimate the integral

$$I = \int_0^1 [1+\cos(\pi x)]\,dx,$$

and the following $n = 4$ numbers are a uniform (0,1) sample:

0.419 0.109 0.732 0.893

Plugging into Equation (3), we obtain

$$\hat{I}_4 = \frac{1-0}{4}\sum_{i=1}^{4}[1+\cos(\pi(0+(1-0)U_i))] = 0.896,$$

which is "close" to the actual answer of 1. □

Example 4 *A Single-Server Queue.* Now the goal is to simulate the behavior of a single-server queueing system. Suppose that six customers arrive at a bank at the following times, which have been generated from some appropriate probability distribution.

3 4 6 10 15 20

Upon arrival, customers queue up in front of a single teller, and are processed sequentially, in a first-come-first-served manner. The service times corresponding to the arriving customers are

7 6 4 6 1 2

For this example, we assume that the bank opens at time 0, and closes its doors at time 20 (just after customer 6 arrives), serving any remaining customers.

Table 1 and Figure 3 trace the evolution of the system as time progresses. The table keeps track of the times at which customers arrive, begin service, and leave. Figure 3 graphs the status of the queue as a function of time; in particular, it graphs $L(t)$, the number of customers in the system (queue + service) at time t.

Table 1: Tracking customers in the single-server queueing system of Example 4.

customer i	arrival time A_i	begin service B_i	service time S_i	depart time D_i	wait W_i
1	3	3	7	10	0
2	4	10	6	16	6
3	6	16	4	20	10
4	10	20	6	26	10
5	15	26	1	27	11
6	20	27	2	29	7

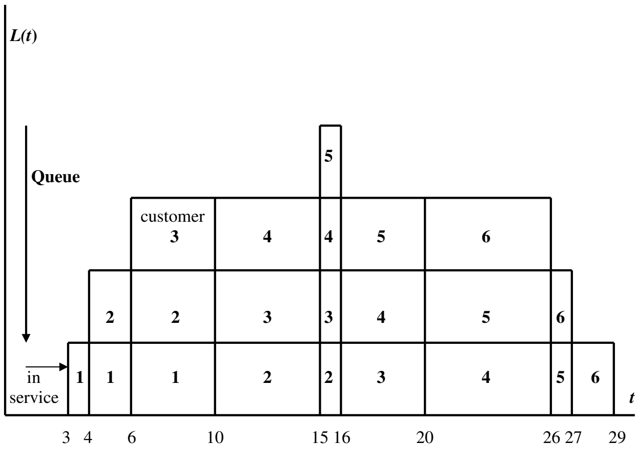

Figure 3: Status of the single-server queueing system of Example 4.

Note that customer i can begin service only at time $\max(A_i, D_{i-1})$, that is, the maximum of his arrival time and the previous customer's departure time. The table and figure are quite easy to interpret. For instance, the system is empty until time 3, when customer 1 arrives. At time 4, customer 2 arrives, but must wait in line until customer 1 finishes service at time 10. We see from the figure that between times 20–26, customer 4 is in service, while customers 5 and 6 wait in the queue. From the table, the average waiting time for the six customers is $\sum_{i=1}^{6} W_i/6 = 44/6$. Further, the average number of customers in the system is $\int_0^{29} L(t)\,dt/29 = 70/29$, where we have computed the integral by adding up the rectangles in Figure 3. Many simulation software packages provide simple ways to model and analyze more-complicated queueing networks. □

Example 5 *(s,S) Inventory Policy.* Customer orders for a particular good arrive at a store every day. During a certain one-week period, the quantities ordered are

10 6 11 3 20 6 8

The store starts the week off with an initial stock of 20. If the stock falls to 5 or below, the owner orders enough from a central warehouse to replenish the stock to 20. Such replenishment orders are placed only at the end of the day and are received before the store opens the next day. There are no customer back orders, so any customer orders that are not filled immediately are lost. This is called an (s,S) inventory system, where the inventory is replenished to $S = 20$ whenever it hits level $s = 5$ (or below).

Table 2 gives a history for this system. We see that at the end of days 2 and 5, replenishment orders were made. In particular, on day 5, the store ran out of stock and lost 14 orders as a result. □

Table 2: A history of the (s,S) inventory system of Example 5.

day	initial stock	customer order	end stock	reorder?	lost orders
1	20	10	10	no	0
2	10	6	4	yes	0
3	20	11	9	no	0
4	9	3	6	no	0
5	6	20	0	yes	14
6	20	6	14	no	0
7	14	8	6	no	0

3 GENERATING RANDOM VARIABLES

All of the examples described in Section 2 required random variables to drive the respective simulations. In Examples 1 through 3, we needed uniform (0,1) random variables; Examples 4 and 5 used more-complicated random variables to model customer arrivals, service times, and order quantities. This section discusses methods to generate such random variables automatically. The generation of uniform (0,1)'s is a good place to start, especially since it turns out that uniform (0,1)'s form the basis for the generation of all other random variables.

3.1 Generating Uniform (0,1) Random Variables

There are a variety of methods for generating uniform (0,1) random variables, among them:

- sample from certain physical devices such as an atomic clock
- look up pre-determined random numbers from a table
- generate *pseudo*-random numbers (PRN's) from a deterministic algorithm

The most widely used techniques in practice all employ the latter strategy of generating PRN's from a deterministic algorithm. Although, by definition, PRN's are not truly random, there are many algorithms available that produce PRN's that *appear* to be perfectly random. Further, these algorithms have the advantages of being computationally fast and repeatable — speed is a good property to have for the obvious reasons, while repeatability is desirable for experimenters who want to be able to replicate their simulation results when the runs are conducted under identical conditions.

Perhaps the most popular method for obtaining PRN's is the linear congruential generator (LCG). Here, we start with a non-negative "seed" integer, X_0, use the seed to generate a sequence of non-negative integers, $X_1, X_2, \ldots$, and then convert the X_i's to PRN's, $U_1, U_2, \ldots$. The algorithm is simple.

1. Specify a non-negative seed integer, X_0.
2. For $i = 1,2,\ldots$, let $X_i = (aX_{i-1} + c) \bmod (m)$, where a, c, and m are appropriately chosen integer constants, and "mod" denotes the modulus function, e.g., $17 \bmod (5) = 2$ and $-1 \bmod (5) = 4$.
3. For $i = 1,2,\ldots$, let $U_i = X_i/m$.

Example 6 Consider the "toy" generator $X_i = (5X_{i-1} + 1) \bmod (8)$, with seed $X_0 = 0$. This produces the integer sequence $X_1 = 1$, $X_2 = 6$, $X_3 = 7$, $X_4 = 4$, $X_5 = 5$, $X_6 = 2$, $X_7 = 3$, $X_8 = 0$, whereupon things start repeating, or "cycling." The PRN's corresponding to the sequence starting with seed $X_0 = 0$ are therefore $U_1 = 1/8$, $U_2 = 6/8$, $U_3 = 7/8$, $U_4 = 4/8$, $U_5 = 5/8$, $U_6 = 2/8$, $U_7 = 3/8$, $U_8 = 0$. Since any seed eventually produces all of the integers $0,1,\ldots,7$, we say that this is a *full-cycle* (or *full-period*) generator. □

Example 7 Not all generators are full-period. Consider another "toy" generator $X_i = (3X_{i-1} + 1) \bmod (7)$, with seed $X_0 = 0$. This produces the integer sequence $X_1 = 1$, $X_2 = 4$, $X_3 = 6$, $X_4 = 5$, $X_5 = 2$, $X_6 = 0$, whereupon cycling ensues. Further, notice that for this generator, a seed of $X_0 = 3$ produces the sequence $X_1 = 3 = X_2 = X_3 = \cdots$, not very random looking! □

The cycle length of the generator from Example 7 obviously depends on the seed chosen, a disadvantage. Full-period generators, such as that studied in Example 6, obviously avoid this problem. A full-period generator with a long cycle-length is given in the following example.

Example 8 The generator $X_i = 16807 X_{i-1} \bmod (2^{31} - 1)$ is full-period. Since $c = 0$, this generator is termed a *multiplicative* LCG, and must be used with a seed $X_0 \neq 0$. Although there are better generators available, this generator has been used in real-world applications, and passes most statistical tests for uniformity and randomness. In order to avoid integer overflow and real-arithmetic round-off problems, Bratley, Fox, and Schrage (1987) offer the following Fortran implementation scheme for this algorithm.

```
FUNCTION UNIF(IX)
K1 = IX/127773
IX = 16807*(IX - K1*127773) - K1*2836
IF(IX.LT.0)IX = IX + 2147483647
UNIF = IX * 4.656612875E-10
RETURN
END
```

In the above program, we input an integer seed `IX` and receive a PRN `UNIF`. The seed `IX` is automatically updated for the next call. Note that, in Fortran, integer division results in *truncation*, e.g., 15/4 = 3; thus, `K1` is an integer. □

3.2 Generating Non-Uniform Random Variables

The goal now is to generate random variables from distributions other than the uniform. The methods we will use to do so always start with a PRN, and then apply an appropriate transformation to the PRN that gives the desired non-uniform random variable. Such non-uniform random variables are important in simulation for a number of reasons. For example, customer arrivals to a service facility often follow a Poisson process; service times may be normal; and routing decisions are usually characterized by Bernoulli random variables.

Inverse Transform Method The most basic technique for generating random variables from a uniform PRN relies on the remarkable *Inverse Transform Theorem.*

Theorem 1 If X is a random variable with continuous cumulative distribution function (CDF) $F(x)$, then the random variable $Y = F(X)$ has the uniform (0,1) distribution.

Proof: For ease of exposition, suppose that X is a continuous random variable. Then the CDF of Y is

$$\begin{aligned} G(y) &= \Pr(Y \le y) \\ &= \Pr(F(X) \le y) \\ &= \Pr(X \le F^{-1}(y)) \\ &\quad \text{(the inverse exists since } F(x) \text{ is continuous)} \\ &= F(F^{-1}(y)) \\ &= y. \end{aligned}$$

Since $G(y) = y$ is the CDF of the uniform (0,1) distribution, we are done. □

With Theorem 1 in hand, it is easy to generate certain random variables. All one has to do is to

1. Find the CDF of X, say $F(x)$.
2. Set $F(X) = U$, where U is a uniform (0,1) PRN.
3. Solve for $X = F^{-1}(U)$.

We illustrate this technique with a series of examples, both for continuous and discrete distributions.

Example 9 Here we generate an exponential random variable with rate λ. Following the recipe outlined above,

1. The CDF is $F(x) = 1 - e^{-\lambda x}$.
2. Set $F(X) = 1 - e^{-\lambda X} = U$.
3. Solving for X, we obtain

$$X = F^{-1}(U) = -\frac{1}{\lambda}\ell\text{n}(1-U).$$

Thus, if one supplies a uniform (0,1) PRN U, we see that $X = -(1/\lambda)\ell\text{n}(1-U)$ is an exponential random variable with parameter λ. □

Example 10 Now we try to generate a standard normal random variable, call it Z. Using the special notation $\Phi(\cdot)$ for the standard normal (0,1) CDF, we set $\Phi(Z) = U$, so that $Z = \Phi^{-1}(U)$. Unfortunately, the inverse CDF does not exist in closed form, so one must resort to the use of standard normal tables (or other approximations). For instance, if we have $U = 0.72$, then a standard normal table yields $Z = \Phi^{-1}(0.72) \approx 0.583$. □

Example 11 We can extend the previous example to generate any normal random variable, i.e., one with arbitrary mean and variance. This follows easily, since if Z is standard normal, then $X = \mu + \sigma Z$ is normal with mean μ and variance σ^2. For instance, suppose we are interested in generating a normal variate X with mean $\mu = 3$ and variance $\sigma^2 = 4$. Then if, as in the previous example, $U = 0.72$, we obtain $Z \approx 0.583$, and, as a consequence, $X \approx 3 + 2(0.583) = 4.166$. □

Example 12 We can also use the ideas from Theorem 1 to generate realizations from discrete random variables. Suppose that the discrete random variable X has probability mass function

$$p(x) = \begin{cases} 0.3 & \text{if } x = -1 \\ 0.6 & \text{if } x = 2.3 \\ 0.1 & \text{if } x = 7 \\ 0 & \text{otherwise} \end{cases}$$

To generate variates from this distribution, we set up Table 3, where $F(x)$ is the associated CDF and U denotes the set of uniform (0,1)'s corresponding to each x-value. To generate a realization of X, we first generate a PRN U, and then read the corresponding x-value from the table. For instance, if $U = 0.43$, then $X = 2.3$. □

Table 3: Values needed to generate variates in Example 12.

x	$p(x)$	$F(x)$	U
-1	0.3	0.3	$[0, 0.3)$
2.3	0.6	0.9	$[0.3, 0.9)$
7	0.1	1.0	$[0.9, 1.0)$

Although the inverse transform method is intuitively pleasing to use, it may sometimes be difficult to apply in practice. For instance, closed-form expressions for the inverse CDF, $F^{-1}(U)$, might not exist, as is the case for the normal distribution; or application of the method might be unnecessarily tedious. We now present a small potpourri of interesting methods to generate a variety of random variables.

Box–Muller method. The Box–Muller (1958) method is an exact technique for generating independent and identically distributed (IID) standard normal (0,1) random variables. The appropriate theorem, stated without proof, is

Theorem 2 Suppose that U_1 and U_2 are IID uniform (0,1) random variables. Then

$$Z_1 \;=\; \sqrt{-2\ell\text{n}(U_1)}\,\cos(2\pi U_2)$$

and

$$Z_2 \;=\; \sqrt{-2\ell\text{n}(U_1)}\,\sin(2\pi U_2)$$

are IID standard normal random variates. (Note that the sine and cosine evaluations must be carried out in *radians*.)

Example 13 Suppose that $U_1 = 0.35$ and $U_2 = 0.65$ are two IID PRN's. Using the Box–Muller method to generate two normal (0,1)'s, we obtain

$$Z_1 \;=\; \sqrt{-2\ell\text{n}(0.35)}\,\cos(2\pi(0.65)) \;=\; -0.852$$

and

$$Z_2 \;=\; \sqrt{-2\ell\text{n}(0.35)}\,\sin(2\pi(0.65)) \;=\; -1.172. \quad \square$$

Central Limit Theorem. One can also use the Central Limit Theorem (CLT) to generate "quick-and-dirty" random variables that are *approximately* normal. Suppose that $U_1, U_2, \ldots, U_n$ are IID PRN's. Then for large enough n, the CLT says that

$$\begin{aligned} \frac{\sum_{i=1}^n U_i - \mathsf{E}\left[\sum_{i=1}^n U_i\right]}{\sqrt{\mathsf{Var}\left(\sum_{i=1}^n U_i\right)}} &= \frac{\sum_{i=1}^n U_i - \sum_{i=1}^n \mathsf{E}[U_i]}{\sqrt{\sum_{i=1}^n \mathsf{Var}(U_i)}} \\ &= \frac{\sum_{i=1}^n U_i - (n/2)}{\sqrt{n/12}} \\ &\approx \text{normal } (0,1). \end{aligned}$$

In particular, the choice $n = 12$ (which turns out to be "large enough") yields the crude but effective approximation,

$$\sum_{i=1}^{12} U_i - 6 \;\approx\; \text{normal } (0,1).$$

Example 14 Suppose we have the following PRN's

0.28	0.87	0.44	0.49	0.10	0.76
0.65	0.98	0.24	0.29	0.77	0.90

Then

$$\sum_{i=1}^{12} U_i - 6 \;=\; 0.77$$

is a realization from a distribution that is approximately standard normal. $\square$

Convolution. Another popular trick involves the generation of random variables via *convolution*, indicating that some sort of sum is involved.

Example 15 Suppose that $X_1, X_2, \ldots, X_n$ are IID exponential random variables with rate λ. Then $Y = \sum_{i=1}^n X_i$ is said to have an *Erlang* distribution with parameters n and λ. It turns out that this distribution has probability density function

$$f(y) \;=\; \begin{cases} \lambda^n e^{-\lambda y} y^{n-1}/(n-1)! & \text{if } y > 0 \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

which readers may recognize as a special case of the gamma distribution.

This distribution's CDF is too difficult to invert directly. One way that comes to mind to generate a realization from the Erlang is simply to generate and then add up n IID exponential(λ) random variables. The following scheme is an efficient way to do precisely that. Suppose that $U_1, U_2, \ldots, U_n$ are IID PRN's. By Example 9, we know that $X_i = -(1/\lambda)\ell\text{n}(1-U_i)$, $i = 1, 2, \ldots, n$, are IID exponential(λ) random variables. Therefore, we can write

$$\begin{aligned} Y \;&=\; \sum_{i=1}^n X_i \;=\; \sum_{i=1}^n \left[-\frac{1}{\lambda}\ell\text{n}(1-U_i)\right] \\ &=\; -\frac{1}{\lambda}\ell\text{n}\left(\prod_{i=1}^n (1-U_i)\right). \end{aligned}$$

This implementation is quite efficient, since it requires only one execution of a natural log operation. In fact, we can even do slightly better from an efficiency point of view — simply note that both U_i and $(1-U_i)$ are uniform (0,1). Then

$$Y \;=\; -\frac{1}{\lambda}\ell\text{n}\left(\prod_{i=1}^n U_i\right)$$

is also Erlang.

To illustrate, suppose that we have three IID PRN's at our disposal, $U_1 = 0.23$, $U_2 = 0.97$, and $U_3 = 0.48$. To generate an Erlang realization with parameters $n = 3$ and $\lambda = 2$, we simply take $Y = -(1/\lambda)\ell\text{n}(U_1U_2U_3) = 1.117$. □

Acceptance-Rejection. One of the most popular classes of random variate generation procedures proceeds by sampling PRN's until some appropriate "acceptance" criterion is met.

Example 16 An easy example of the acceptance-rejection technique involves the generation of a geometric random variable with success probability p. To this end, consider a sequence of PRN's $U_1, U_2, \ldots$. Our aim is to generate a geometric realization X, i.e., one that has probability function

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

In words, X represents the number of Bernoulli trials until the first success is observed. This English characterization immediately suggests an elementary acceptance-rejection algorithm,

1. Initialize $i \leftarrow 0$.
2. Let $i \leftarrow i+1$.
3. Take a Bernoulli(p) observation,
$$Y_i = \begin{cases} 1 & \text{if } U_i < p \\ 0 & \text{otherwise} \end{cases}$$
4. If $Y_i = 1$, then we have our first success, and we stop, in which case we *accept* $X = i$. Otherwise, if $Y_i = 0$, then we *reject*, and go back to step 2.

To illustrate, let us generate a geometric variate having success probability $p = 0.3$. Suppose we have the following PRN's are at our disposal.

0.38 0.67 0.24 0.89 0.10 0.71

Since $U_1 = 0.38 \geq p$, we have $Y_1 = 0$, so we reject $X = 1$. Since $U_2 = 0.67 \geq p$, we have $Y_2 = 0$, so we reject $X = 2$. Since $U_3 = 0.24 < p$, we have $Y_3 = 1$, so we finally *accept* $X = 3$. □

4 INPUT ANALYSIS

A fundamental problem in simulation modeling is that of determining the proper input random variables to drive the simulation. For example, how are the interarrivals of customers distributed? What random variables are appropriate to model service times? Machine breakdown times? Such questions underline the critical nature of correct input analysis. In fact, if we misspecify a particular distribution, the resulting model and any subsequent output we obtain from it could be misleading — garbage-in-garbage-out.

Although an experimenter can never specify the "perfect" model, there are steps that one can take to try to do the best job possible. For instance, a basic game-plan might be as follows:

- Assess what input variables will be needed — arrival times, service times, breakdown times? In addition, are the variables independent of each other?
- Try to collect as much good data as possible, and assess their value. For instance, are the data "hard" or are they simply someone's best guess?
- Make reasonable assumptions about the underlying observations that are obtained. Are they coming from discrete or continuous random variables? Are they independent? Make preliminary guesses about their underlying distributions; or at least construct appropriate empirical distributions.
- If a parametric univariate distribution has been assumed (e.g., exponential, normal, etc.), estimate the relevant unknown parameters. For example, one could use standard maximum likelihood methods to estimate an exponential distribution's rate parameter.
- Conduct a formal goodness-of-fit test (such as a χ^2 or Kolmogorov-Smirnov test) to check whether or not the fitted distribution is reasonable.
- Try to determine whether more-sophisticated models (e.g., nonhomogeneous Poisson processes or autoregressive-moving average time series) are appropriate.

Input modeling is both an art and a science, and can have a significant impact on the viability of the entire simulation study.

5 OUTPUT ANALYSIS

Simulation output analysis is one of the most important aspects of any proper and complete simulation study. Since the input processes driving a simulation are usually random variables (e.g., interarrival times, service times, and breakdown times), we must also regard the output from the simulation as random. Thus, runs of the simulation only yield *estimates* of measures of system performance (e.g., the mean customer waiting time). These estimators are themselves random variables, and are therefore subject to sampling error — and sampling error must be taken into account to make valid inferences concerning system performance.

The problem is that simulations almost never produce convenient raw output that is IID normal data. For example, consecutive customer waiting times from a queueing system…

- Are not independent — typically, they are serially correlated. If one customer at the post office waits in line a long time, then the next customer is also likely to wait a long time.
- Are not identically distributed. Customers showing up early in the morning might have a much shorter wait than those who show up just before closing time.
- Are not normally distributed — they are usually skewed to the right (and are certainly never less than zero).

The point is that it is difficult to apply "classical" statistical techniques to the analysis of simulation output. Thus, our purpose here is to give methods to perform statistical analysis of output from discrete-event computer simulations that bypass the difficulties encountered when applying standard statistical methods. To facilitate the presentation, we identify two types of simulations with respect to output analysis: terminating and steady-state simulations.

1. *Terminating (or transient) simulations*. Here, the nature of the problem explicitly defines the length of the simulation run. For instance, we might be interested in simulating a bank that closes at a specific time each day.
2. *Nonterminating (steady-state) simulations*. Here, the long-run behavior of the system is studied. Presumedly this "steady-state" behavior is independent of the simulation's initial conditions. An example is that of a continuously running production line for which the experimenter is interested in some long-run performance measure.

Techniques to analyze output from terminating simulations are based on the method of independent replications, discussed in Section 5.1. Additional problems arise for steady-state simulations. For instance, we must now worry about the problem of starting the simulation — how should it be initialized at time zero, and how long must it be run before data representative of steady state can be collected? Initialization problems are considered in Section 5.2. Finally, Section 5.3 deals with point and confidence interval estimation for steady-state simulation performance parameters.

5.1 Terminating Simulation Analysis

Here we are interested in simulating some system of interest over a finite time horizon. For now, assume we obtain *discrete* simulation output $Y_1, Y_2, \ldots, Y_m$, where the number of observations m can be a constant or a random variable. For example, the experimenter can specify the number m of customer waiting times $Y_1, Y_2, \ldots, Y_m$ to be taken from a queueing simulation. Or m could denote the random number of customers observed during a specified time period $[0, T]$.

Alternatively, we might observe *continuous* simulation output $\{Y(t) | 0 \le t \le T\}$ over a specified interval $[0, T]$. For instance, if we are interested in estimating the time-averaged number of customers waiting in a queue during $[0, T]$, the quantity $Y(t)$ would be the number of customers in the queue at time t.

The easiest goal is to estimate the expected value of the sample mean of the observations,

$$\theta \equiv \mathsf{E}[\bar{Y}_m],$$

where the sample mean in the discrete case is

$$\bar{Y}_m \equiv \frac{1}{m}\sum_{i=1}^{m} Y_i$$

(with a similar expression for the continuous case). For example, we might be interested in estimating the expected average waiting time of all customers at a shopping center during the period 10 a.m. to 2 p.m.

Although $\bar{Y}_m$ is an unbiased estimator for θ, a proper statistical analysis requires that we also provide an estimate of $\mathsf{Var}(\bar{Y}_m)$. Since the Y_i's are not necessarily IID random variables, it is may be that $\mathsf{Var}(\bar{Y}_m) \ne \mathsf{Var}(Y_i)/m$ for any i, a case not covered in elementary statistics courses.

For this reason, the familiar sample variance,

$$S^2 \equiv \frac{1}{m-1}\sum_{i=1}^{m}(Y_i - \bar{Y}_m)^2,$$

is likely to be highly *biased* as an estimator of $m\mathsf{Var}(\bar{Y}_m)$. Thus, one should *not* use S^2/m to estimate $\mathsf{Var}(\bar{Y}_m)$.

One way around the problem is via the method of *independent replications* (IR). IR estimates $\mathsf{Var}(\bar{Y}_m)$ by conducting b independent simulation runs (replications) of the system under study, where each replication consists of m observations. It is easy to make the replications independent — simply re-initialize each replication with a different pseudo-random number seed.

To proceed, denote the sample mean from replication i by

$$Z_i \equiv \frac{1}{m}\sum_{j=1}^{m} Y_{i,j},$$

where $Y_{i,j}$ is observation j from replication i, for $i = 1, 2, \ldots, b$ and $j = 1, 2, \ldots, m$.

If each run is started under the same operating conditions (e.g., all queues empty and idle), then the replication sample means $Z_1, Z_2, \ldots, Z_b$ are IID random variables. Then the obvious point estimator for $\mathsf{Var}(\bar{Y}_m) = \mathsf{Var}(Z_i)$ is the sample variance of the Z_i's,

$$\hat{V}_R \equiv \frac{1}{b-1}\sum_{i=1}^{b}(Z_i - \bar{Z}_b)^2,$$

where the grand mean is defined as $\bar{Z}_b \equiv \sum_{i=1}^{b} Z_i/b$. Notice how closely the forms of $\hat{V}_R$ and S^2/m resemble each other. But since the replicate sample means are IID, $\hat{V}_R$ is usually much less biased for $\mathsf{Var}(\bar{Y}_m)$ than is S^2/m.

In light of the above discussion, we see that $\hat{V}_R/b$ is a reasonable estimator for $\mathsf{Var}(\bar{Z}_b)$. Further, if the number of observations per replication, m, is large enough, the Central Limit Theorem tells us that the replicate sample means are approximately IID *normal*.

Then basic statistics yields an approximate $100(1-\alpha)\%$ two-sided confidence interval (CI) for θ,

$$\theta \in \bar{Z}_b \pm t_{\alpha/2,b-1}\sqrt{\hat{V}_R/b}, \tag{5}$$

where $t_{\alpha/2,b-1}$ is the $1-\alpha/2$ quantile of the t-distribution with $b-1$ degrees of freedom.

Example 17 Suppose we want to estimate the expected average waiting time for the first 5000 customers in a certain queueing system. We will make five independent replications of the system, with each run initialized empty and idle and consisting of 5000 waiting times. The resulting replicate means are given in Table 4. Then $\bar{Z}_5 = 4.28$ and

Table 4: Five replicate means in a certain queueing system.

i	1	2	3	4	5
Z_i	3.2	4.3	5.1	4.2	4.6

$\hat{V}_R = 0.487$. For level $\alpha = 0.05$, we have $t_{0.025,4} = 2.78$, and Equation (5) gives $[3.41, 5.15]$ as a 95% CI for the expected average waiting time for the first 5000 customers. □

Independent replications can be used to calculate variance estimates for statistics other than sample means. Then the method can be used to obtain CI's for quantities other than $\mathsf{E}[\bar{Y}_m]$, e.g., quantiles. See any of the standard simulation texts for additional uses of independent replications.

5.2 Initialization Problems

Before a simulation can be run, one must provide initial values for all of the simulation's state variables. Since the experimenter may not know what initial values are appropriate for the state variables, these values might be chosen somewhat arbitrarily. For instance, we might decide that it is "most convenient" to initialize a queue as empty and idle. Such a choice of initial conditions can have a significant but unrecognized impact on the simulation run's outcome. Thus, the *initialization bias* problem can lead to errors, particularly in steady-state output analysis.

Some examples of problems concerning simulation initialization are as follows.

- Visual detection of initialization effects is sometimes difficult — especially in the case of stochastic processes having high intrinsic variance such as queueing systems.
- How should the simulation be initialized? Suppose that a machine shop closes at a certain time each day, even if there are jobs waiting to be served. One must therefore be careful to start each day with a demand that depends on the number of jobs remaining from the previous day.
- Initialization bias can lead to point estimators for steady-state parameters having high mean squared error, as well as CI's having poor coverage.

Since initialization bias raises important concerns, how do we detect and deal with it? We first list methods to detect it.

- *Attempt to detect the bias visually* by scanning a realization of the simulated process. This might not be easy, since visual analysis can miss bias. Further, a visual scan can be tedious. To make the visual analysis more efficient, one might transform the data (e.g., take logs or square roots), smooth it, average it across several independent replications, or construct moving average plots.
- *Conduct statistical tests for initialization bias.* Steiger et al. (2005) give an intuitively appealing sequential procedure to detect bias. Various other tests check to see whether the initial portion of the simulation output contains more variation than latter portions.

If initialization bias is detected, one may want to do something about it. Two simple methods for dealing with bias are:

- *Truncate the output* by allowing the simulation to "warm up" before data are retained for analysis. The experimenter would then hope that the remaining data are representative of the steady-state system. Output truncation is probably the most popular method for dealing with initialization bias; and all of the major simulation languages have built-in truncation functions.

But how can one find a good truncation point? If the output is truncated "too early," significant bias might still exist in the remaining data. If it is truncated "too late," then good observations might be wasted. Unfortunately, all simple rules to determine truncation points do not perform well in general. A common practice is to average observations across several replications, and then visually choose a truncation point based on the averaged run. See Welch (1983) for a good visual/graphical approach and Steiger et al. (2005) for a more-rigorous algorithm.

- *Make a very long run* to overwhelm the effects of initialization bias. This method of bias control is conceptually simple to carry out and may yield point estimators having lower mean squared errors than the analogous estimators from truncated data (see, e.g., Fishman 1978). However, a problem with this approach is that it can be wasteful with observations; for some systems, an excessive run length might be required before the initialization effects are rendered negligible.

5.3 Steady-State Simulation Analysis

Now assume that we have on hand stationary (steady-state) simulation output, $Y_1, Y_2, \ldots, Y_n$. Our goal is to estimate some parameter of interest, possibly the mean customer waiting time or the expected profit produced by a certain factory configuration. As in the case of terminating simulations, a good statistical analysis must accompany the value of any point estimator with a measure of its variance.

A number of methodologies have been proposed in the literature for conducting steady-state output analysis: batch means, independent replications, standardized time series, spectral analysis, regeneration, and time series modeling, as well as a host of others. We will examine the two most popular: batch means and independent replications. (Recall: As discussed earlier, the output analysis method of choice for *terminating* simulations usually involves independent replications.)

Batch Means. The method of batch means (BM) is often used to estimate $\mathsf{Var}(\bar{Y}_n)$ or calculate CI's for the steady-state process mean μ. The idea is to divide one long simulation run into a number of contiguous *batches*, and then appeal to a central limit theorem to assume that the resulting batch sample means are approximately IID normal.

In particular, suppose that we partition $Y_1, Y_2, \ldots, Y_n$ into b nonoverlapping, contiguous batches, each consisting of m observations (assume that $n = bm$). Thus, the ith batch, $i = 1, 2, \ldots, b$, consists of the random variables

$$Y_{(i-1)m+1}, Y_{(i-1)m+2}, \ldots, Y_{im}.$$

The ith batch mean is simply the sample mean of the m observations from batch i, $i = 1, 2, \ldots, b$,

$$Z_i \equiv \frac{1}{m} \sum_{j=1}^{m} Y_{(i-1)m+j}.$$

Similar to independent replications, we define the BM estimator for $\mathsf{Var}(Z_i)$ as

$$\hat{V}_B \equiv \frac{1}{b-1} \sum_{i=1}^{b} (Z_i - \bar{Z}_b)^2,$$

where

$$\bar{Y}_n \equiv \bar{Z}_b \equiv \frac{1}{b} \sum_{i=1}^{b} Z_i$$

is the grand sample mean. If m is large, then the batch means are approximately IID normal, and (as for IR) we obtain an approximate $100(1-\alpha)\%$ CI for μ,

$$\mu \in \bar{Z}_b \pm t_{\alpha/2, b-1} \sqrt{\hat{V}_B / b}.$$

This equation is very similar to Equation (5). Of course, the difference here is that the method of batch means divides one long run into a number of batches, whereas independent replications uses a number of independent shorter runs. Indeed, consider the old IR example from Section 5.1 with the understanding that the Z_i's are now regarded as batch means (instead of replicate means); then the same numbers carry through the example.

The technique of BM is intuitively appealing and easy to understand. But problems can come up if the Y_j's are not stationary (e.g., if significant initialization bias is present), if the batch means are not normal, or if the batch means are not independent. If any of these assumption violations exist, poor confidence interval coverage may result — unbeknownst to the analyst.

To ameliorate the initialization bias problem, the user can truncate some of the data or make a long run as discussed in Section 5.2. In addition, the lack of independence or normality of the batch means can be countered by increasing the batch size m.

Independent Replications. Of the difficulties encountered when using BM, the possibility of correlation among the batch means might be the most troublesome. This problem is explicitly avoided by the method of IR, described in the context of terminating simulations in Section 5.1. In fact, the replicate sample means are independent by their construction. Unfortunately, since *each* of the b replications has to be started properly, initialization bias presents more trouble when using IR than when using BM. The usual recommendation, in the context of steady-state analysis, is to use BM over IR because of the possible initialization bias in each of the replications.

6 COMPARISON OF SYSTEMS

One of the fundamental uses of simulation output analysis regards the comparison of competing systems or alternative system configurations. For example, suppose we wish to evaluate two different "re-start" strategies that an airline can evoke following a major traffic disruption such as a snowstorm in the Northeast — which policy minimizes a certain cost function associated with the re-start? Simulation is uniquely equipped to help the experimenter conduct this type of comparison analysis.

There are many techniques available for comparing systems, among them: (i) classical statistical CI's, (ii) common random numbers, (iii) antithetic variates, (iv) and ranking and selection procedures. The following sections discuss each general technique in turn.

6.1 Classical Confidence Intervals

With our airline example in mind, let $Z_{i,j}$ be the cost from the jth simulation replication of strategy i, $i=1,2$, $j=1,2,\ldots,b_i$. For fixed $i=1,2$, suppose $Z_{i,1},Z_{i,2},\ldots,Z_{i,b_i}$ are IID normal with unknown mean μ_i and unknown variance — an assumption that can be justified by arguing that we can

- Get independent data by controlling the random numbers between replications.
- Get identically distributed costs between replications by performing the replications under identical conditions.
- Get approximately normal data by adding up (or averaging) many sub-costs to obtain overall costs for both strategies.

The goal here is to calculate a $100(1-\alpha)\%$ CI for the difference $\mu_1-\mu_2$. To this end, suppose that the $Z_{1,j}$'s are independent of the $Z_{2,j}$'s and define

$$\bar{Z}_{i,b_i} \equiv \frac{1}{b_i}\sum_{j=1}^{b_i} Z_{i,j}, \ \ i=1,2,$$

and

$$S_i^2 \equiv \frac{1}{b_i-1}\sum_{j=1}^{b_i}(Z_{i,j}-\bar{Z}_{i,b_i})^2, \ \ i=1,2.$$

An approximate $100(1-\alpha)\%$ CI is

$$\mu_1-\mu_2 \in \bar{Z}_{1,b_1}-\bar{Z}_{2,b_2} \pm t_{\alpha/2,\nu}\sqrt{\frac{S_1^2}{b_1}+\frac{S_2^2}{b_2}}$$

where the approximate degrees of freedom ν (a function of the sample variances) is given by (cf. Hines et al. 2003)

$$\nu \equiv \frac{\left(\frac{S_1^2}{b_1}+\frac{S_2^2}{b_2}\right)^2}{\frac{(S_1^2/b_1)^2}{b_1+1}+\frac{(S_2^2/b_2)^2}{b_2+1}} - 2.$$

Suppose (as in the airline example) that small cost is good. If the interval lies entirely to the left [right] of zero, then system 1 [2] is better; if the interval contains zero, then the two systems must be regarded, in a statistical sense, as about the same.

An alternative classical strategy is to use a CI that is analogous to a paired-t hypothesis test. Here we take b replications from *both* strategies and set the differences $D_j \equiv Z_{1,j}-Z_{2,j}$ for $j=1,2,\ldots,b$. Then we calculate the sample mean and sample variance of the differences:

$$\bar{D}_b \equiv \frac{1}{b}\sum_{j=1}^{b} D_j \quad \text{and} \quad S_D^2 \equiv \frac{1}{b-1}\sum_{j=1}^{b}(D_j-\bar{D}_b)^2.$$

The resulting $100(1-\alpha)\%$ CI is

$$\mu_1-\mu_2 \in \bar{D}_b \pm t_{\alpha/2,b-1}\sqrt{S_D^2/b}. \tag{6}$$

These paired-t intervals are very efficient if $\mathsf{Corr}(Z_{1,j},Z_{2,j})>0$, $j=1,2,\ldots,b$ (where we still assume that $Z_{1,1},Z_{1,2},\ldots,Z_{1,b}$ are IID and $Z_{2,1},Z_{2,2},\ldots,Z_{2,b}$ are IID). In that case, it turns out that

$$\mathsf{Var}(\bar{D}_b) < \frac{\mathsf{Var}(Z_{1,j})+\mathsf{Var}(Z_{2,j})}{b}. \tag{7}$$

If $Z_{1,j}$ and $Z_{2,j}$ had been simulated *independently*, then we would have had *equality* in expression (7). Thus, our trick may result in relatively small S_D^2 and, hence, small CI length for (6). So how do we evoke the trick?

6.2 Common Random Numbers

The idea behind the above trick is to use *common random numbers* (CRN), i.e., use the same pseudo-random numbers in exactly the same ways for corresponding runs of each of the competing systems. For example, we might use precisely the same customer arrival times when simulating different proposed configurations of a job shop. By subjecting the alternative systems to identical experimental conditions, we hope to make it easy to distinguish which systems are best even though the respective estimators are subject to sampling error.

Consider the case in which we compare two queueing systems, A and B, on the basis of their expected customer transit times, θ_A and θ_B — the smaller θ-value corresponds to the better system. Suppose we have estimators $\hat{\theta}_A$ and $\hat{\theta}_B$ for θ_A and θ_B, respectively. We will declare A as the better system if $\hat{\theta}_A < \hat{\theta}_B$. If $\hat{\theta}_A$ and $\hat{\theta}_B$ are simulated independently, then the variance of their difference,

$$\mathsf{Var}(\hat{\theta}_A - \hat{\theta}_B) \;=\; \mathsf{Var}(\hat{\theta}_A) + \mathsf{Var}(\hat{\theta}_B),$$

could be very large, in which case our declaration might lack conviction. If we could reduce $\mathsf{Var}(\hat{\theta}_A - \hat{\theta}_B)$, then we could be much more confident about our declaration.

CRN sometimes induces a high positive correlation between the point estimators $\hat{\theta}_A$ and $\hat{\theta}_B$. Then we have

$$\begin{aligned} \mathsf{Var}(\hat{\theta}_A - \hat{\theta}_B) &= \mathsf{Var}(\hat{\theta}_A) + \mathsf{Var}(\hat{\theta}_B) - 2\mathsf{Cov}(\hat{\theta}_A, \hat{\theta}_B) \\ &< \mathsf{Var}(\hat{\theta}_A) + \mathsf{Var}(\hat{\theta}_B), \end{aligned}$$

and we obtain a savings in variance.

6.3 Antithetic Random Numbers

Alternatively, if we can induce *negative* correlation between two unbiased estimators, $\hat{\theta}_1$ and $\hat{\theta}_2$, for some parameter θ, then the unbiased estimator $(\hat{\theta}_1 + \hat{\theta}_2)/2$ might have low variance.

Most simulation texts give advice on how to run the simulations of the competing systems so as to induce positive or negative correlation between them. The consensus is that, if conducted properly, common and antithetic random numbers can lead to tremendous variance reductions.

6.4 Selecting the Best System

Ranking, selection, and multiple comparisons methods form another class of statistical techniques used to compare alternative systems. Here, the experimenter is interested in selecting the best of a number of competing processes. Typically, one specifies the desired probability of correctly selecting the best process, especially if the best process is significantly better than its competitors. These methods are simple to use, fairly general, and intuitively appealing. See Bechhofer, Santner, and Goldsman (1995) as well as previous Winter Simulation Conference *Proceedings* for synopses of the most popular procedures.

7 SUMMARY

This tutorial began with some simple motivational examples illustrating various simulation concepts. After this, the exposition turned to the generation of pseudo-random numbers and general random variables. PRN's are numbers that appear to be IID uniform (0,1), and are important because they are used to generate all other random variables, e.g., normal, exponential, and Erlang. We then briefly discussed simulation input analysis — what are appropriate random variables to drive the simulation? We also spent a great deal of time on simulation output analysis — simulation output is almost never IID, so special care must be taken if we are to make statistically valid conclusions about the simulation's results. We concentrated on output analysis for both terminating and steady-state simulations.

Armed with our tutorial's material, the interested reader can now move on to the more-sophisticated tutorials found elsewhere in this *Proceedings*.

ACKNOWLEDGMENT

This material was supported by the National Science Foundation under Grant Number DMI-0400260.

REFERENCES

Banks, J., J. S. Carson, B. L. Nelson, and D. M. Nicol. 2005. *Discrete-Event System Simulation*, 4th edition, Upper Saddle River, New Jersey: Prentice-Hall.

Bechhofer, R. E., T. J. Santner, and D. M. Goldsman. 1995. *Design and Analysis of Experiments for Statistical Selection, Screening, and Multiple Comparisons*. New York: John Wiley and Sons.

Box, G. E. P., and M. F. Muller. 1958. A note on the generation of normal random deviates. *Annals of Mathematical Statistics*, 29:610–611.

Bratley, P., B. L. Fox, and L. E. Schrage. 1987. *A Guide to Simulation*, 2nd edition. New York: Springer-Verlag.

Fishman, G. S. 1978. *Principles of Discrete Event Simulation*. New York: John Wiley and Sons.

Hines, W. W., D. C. Montgomery, D. Goldsman, and C. M. Borror. 2003. *Probability and Statistics in Engineering*, 4th edition. New York: John Wiley and Sons.

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

Steiger, N. M., E. K. Lada, J. R. Wilson, J. A. Joines, C. Alexopoulos, and D. Goldsman. 2005. ASAP3: A batch means procedure for steady-state simulation analysis. *ACM TOMACS*, 15:39–73.

Welch, P. D. 1983. The statistical analysis of simulation results, in *The Computer Performance Modeling Handbook*, ed. S. Lavenberg. Orlando: Academic Press.

AUTHOR BIOGRAPHY

DAVID GOLDSMAN is a Professor in the School of ISyE at Georgia Tech. His research interests include simulation output analysis and ranking and selection. He is an active participant in the Winter Simulation Conference, having been Program Chair in 1995, and having served on the WSC Board of Directors since 2002. His e-mail address is ⟨`sman@gatech.edu`⟩, and his web page is ⟨`www.isye.gatech.edu/~sman/`⟩.