excerpt - CORNs
Sects. 11.1 - 11.2

Simulation Modeling and Analysis

FOURTH EDITION

Averill M. Law

President
Averill M. Law & Associates, Inc.
Tucson, Arizona, USA
www.averill-law.com

Boston Burr Ridge, IL Dubuque, IA New York
San Francisco St. Louis Bangkok Bogotá Caracas Kuala Lumpur
Lisbon London Madrid Mexico City Milan Montreal New Delhi
Santiago Seoul Singapore Sydney Taipei Toronto

CHAPTER 11

Variance-Reduction Techniques

Recommended sections for a first reading: 11.1, 11.2

11.1 INTRODUCTION

One of the points we have tried to emphasize throughout this book is that simulations driven by random inputs will produce random output. Thus, appropriate statistical techniques applied to simulation output data are imperative if the results are to be properly analyzed, interpreted, and used (see Chaps. 9, 10, and 12). Since large-scale simulations may require great amounts of computer time and storage, appropriate statistical analyses (possibly requiring multiple replications of the model, for example) can become quite costly. Sometimes the cost of even a modest statistical analysis of the output can be so high that the precision of the results, perhaps measured by confidence-interval width, will be unacceptably poor. The analyst should therefore try to use any means possible to increase the simulation's efficiency.

Of course, "efficiency" mandates careful programming to expedite execution and minimize storage requirements. In this chapter, however, we focus on *statistical* efficiency, as measured by the *variances* of the output random variables from a simulation. If we can somehow reduce the variance of an output random variable of interest (such as average delay in queue or average cost per month in an inventory system) without disturbing its expectation, we can obtain greater precision, e.g., smaller confidence intervals, for the same amount of simulating, or, alternatively, achieve a desired precision with less simulating. Sometimes such a *variance-reduction technique* (VRT), properly applied, can make the difference between an impossibly expensive simulation project and a frugal, useful one.

As we shall see, the method of applying VRTs usually depends on the particular model (or models) of interest. Therefore, a thorough understanding of the workings of the model(s) is required for proper use of VRTs. Furthermore, it is generally impossible to know beforehand how great a variance reduction might be realized, or (worse) whether the variance will be reduced at all in comparison with straightforward simulation. However, preliminary runs could be made (if affordable) to compare the results of applying a VRT with those from straightforward simulation. Finally, some VRTs themselves will increase computing cost, and this decrease in computational efficiency must be traded off against the potential gain in statistical efficiency [see Glynn and Whitt (1992a) and Prob. 11.1]. Almost all VRTs require *some* extra effort on the part of the analyst (if only to understand the technique) and this, as always, must be considered.

VRTs were developed originally in the early days of computers, to be applied in Monte Carlo simulations or distribution sampling [see Sec. 1.8.3, as well as Hammersley and Handscomb (1964) and Morgan (1984, chap. 7)]. However, many of these original VRTs have been found not to be directly applicable to simulations of complex dynamic systems.

In the remainder of this chapter we will discuss in some detail five general types of VRTs that would appear to have the most promise of successful application to a wide variety of simulations. We refer the reader to Kleijnen (1974), Morgan (1984, chap. 7), and Bratley, Fox, and Schrage (1987, chap. 2) for detailed discussions of other VRTs, such as stratified sampling and importance sampling (see also Sec. 11.6). There is a very large literature on VRTs, and we do not attempt an exhaustive treatment here. Fortunately, there are several comprehensive surveys that provide useful ways of classifying VRTs and also contain extensive bibliographies; Wilson (1984), Nelson (1985, 1986, 1987a, 1987c), L'Ecuyer (1994a), and Kleijnen (1998, sec. 6.3.5 and app. 6.2) are particularly recommended to the reader interested in going further with this subject. In addition, special issues of the journals *Management Science* (Volume 35, Number 11, November 1989, edited by G. S. Fishman) and the *Association for Computing Machinery (ACM) Transactions on Modeling and Computer Simulation* (Volume 3, Number 3, July 1993, edited by P. Glasserman and P. Heidelberger), were devoted to research on VRTs. While our discussion will focus on the different VRTs by themselves, it is possible to use them together; see Kwon and Tew (1994), Avramidis and Wilson (1996), and Yang and Liou (1996).

11.2 COMMON RANDOM NUMBERS

The first VRT we consider, *common random numbers* (CRN), is actually different from the others in that it applies when we are comparing two or more alternative system configurations (see Chap. 10) instead of investigating a single configuration. Despite its simplicity, CRN is the most useful and popular VRT of all. In fact, as mentioned at the end of Sec. 10.2.3, the default setup in most simulation packages is that the same random-number streams and seeds (see Sec. 7.2) are used unless

specified otherwise. Thus, by default, the two configurations would actually use the same random numbers, though not necessarily properly synchronized (see Sec. 11.2.3), which is critical for the success of CRN.

11.2.1 Rationale

The basic idea is that we should compare the alternative configurations "under similar experimental conditions" so that we can be more confident that any observed differences in performance are due to differences in the system configurations rather than to fluctuations of the "experimental conditions." In simulation, these "experimental conditions" are the generated random variates that are used to drive the models through simulated time. In queueing simulations, for instance, these would include interarrival times and service times of customers; in inventory simulations we might include interdemand times and demand sizes. The name of this technique stems from the possibility in many situations of using the *same* basic U(0, 1) random numbers (see Chap. 7) to drive each of the alternative configurations through time. As we shall see later in this section, however, certain programming techniques are often needed to facilitate proper implementation of CRN. In the terminology of classical experimental design, CRN is a form of *blocking*, i.e., "comparing like with like." CRN has also been called *correlated sampling*.

To see the rationale for CRN more clearly, consider the case of *two* alternative configurations, as in Sec. 10.2, where X_{1j} and X_{2j} are the observations from the first and second configurations on the jth independent replication, and we want to estimate $\zeta = \mu_1 - \mu_2 = E(X_{1j}) - E(X_{2j})$. If we make n replications of each system and let $Z_j = X_{1j} - X_{2j}$ for $j = 1, 2, \ldots, n$, then $E(Z_j) = \zeta$ so

$$\overline{Z}(n) = \frac{\sum_{j=1}^{n} Z_j}{n}$$

is an unbiased estimator of ζ. Since the Z_j's are IID random variables,

$$\text{Var}[\overline{Z}(n)] = \frac{\text{Var}(Z_j)}{n} = \frac{\text{Var}(X_{1j}) + \text{Var}(X_{2j}) - 2\,\text{Cov}(X_{1j}, X_{2j})}{n}$$

[see Eq. (4.5) and Prob. 4.13]. If the simulations of the two different configurations are done independently, i.e., with different random numbers, X_{1j} and X_{2j} will be independent, so that $\text{Cov}(X_{1j}, X_{2j}) = 0$. On the other hand, if we could somehow do the simulations of configurations 1 and 2 so that X_{1j} and X_{2j} are *positively* correlated, then $\text{Cov}(X_{1j}, X_{2j}) > 0$, so that the variance of our estimator $\overline{Z}(n)$ is reduced. Thus, when $\overline{Z}(n)$ is observed in a particular simulation experiment, its value should be closer to ζ. CRN is a technique where we try to induce this positive correlation by using (carefully, as discussed in Sec. 11.2.3 below) the *same* random numbers to simulate both configurations. (This does not change the probability distributions of X_{1j} and X_{2j} and, in particular, their means and variances.) What makes this possible is the deterministic, reproducible nature of random-number generators (see Sec. 7.1);

irreproducible gimmicks such as seeding the random-number generator by the square root of the computer's clock value would generally preclude the use of CRN as well as many other valuable VRTs.

11.2.2 Applicability

Unfortunately, there is no completely general proof that CRN "works," i.e., that it will always reduce the variance. Even if it does work, we usually will not know beforehand how great a reduction in variance we might experience. The efficacy of CRN depends wholly on the particular models being compared, and its use presupposes the analyst's (perhaps implicit) belief that the different models will respond "similarly" to large or small values of the random variates driving the models. For example, we would expect that smaller interarrival times for several designs of a queueing facility would result in longer delays and queues for *each* system.

There are, however, some classes of models for which CRN's success *is* guaranteed. Heidelberger and Iglehart (1979) showed this for certain types of regenerative simulations, and Bratley, Fox, and Schrage (1987, chap. 2) derive results indicating conditions under which CRN will work. See also Gal, Rubinstein, and Ziv (1984), Rubinstein, Samorodnitsky, and Shaked (1985), and Glasserman and Yao (1992) for additional results of this type.

Figure 11.1 schematically illustrates the concept in principle, where the horizontal axis shows possible values of a *particular* U_k used for a *particular* purpose in both of the simulations; for instance, this U_k might be used to generate a service time. The curves indicate how the results of the simulations might react, all other things being equal, to possible values of this U_k. In either of the two situations in the top row of plots, both X_{1j} and X_{2j} react monotonically in the same direction to U_k, and we would expect CRN to induce the desired positive correlation, and thus reduce the variance. In the bottom two plots, however, X_{1j} and X_{2j} react in opposite directions to U_k, so CRN could induce negative correlation and thus "backfire," leading to $\text{Cov}(X_{1j}, X_{2j}) < 0$ and an actual *increase* in the variance. Problem 11.2 considers a specific instance of this issue.

Usually, random numbers are first used to generate variates from other distributions (see Chap. 8), which are then used to drive the simulation models. In order to give CRN the best chance of working, we should thus try first to ensure that the generated variates themselves react monotonically to the U_k's in this intermediate variate-generation step; we then must assume that the measures of performance react monotonically to the generated variates. For this reason, the inverse-transform method of variate generation (Sec. 8.2.1) is recommended, since it guarantees monotonicity of the generated input variates to the random numbers; it further provides the strongest possible positive correlation among all variate-generation methods [see Bradley et al. (1987, pp. 53–54) or Whitt (1976)]. Since the inverse-transform method can be slow, however, for some distributions (perhaps involving numerical methods to invert the distribution function), its computational inefficiency could offset its statistical efficiency. For this reason, Schmeiser and

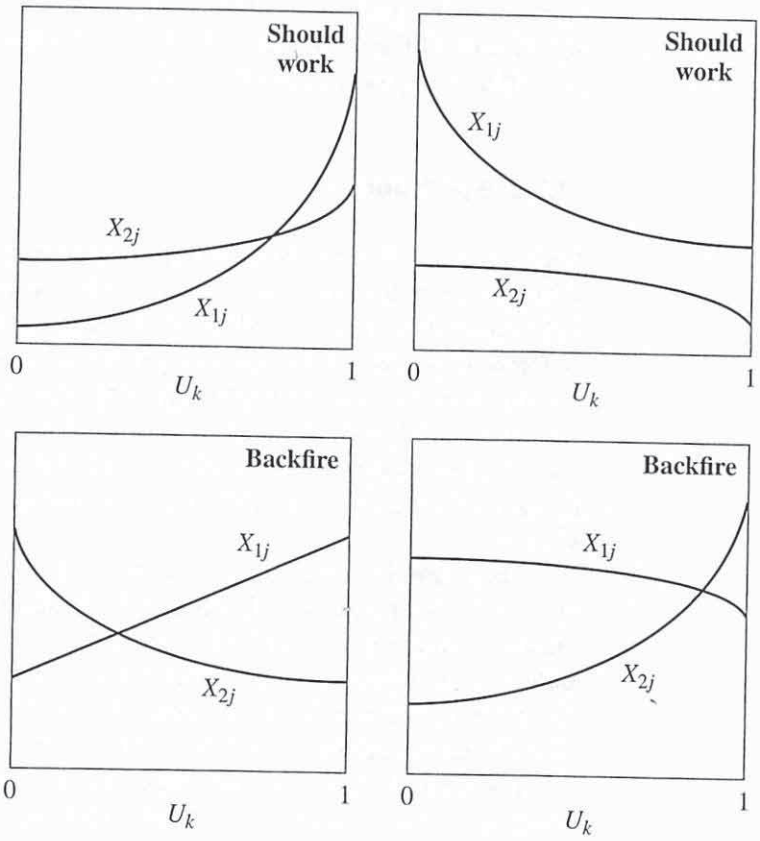


FIGURE 11.1
Model responses for CRN to work (top row) and backfire (bottom row).

Kachitvichyanukul (1990) developed faster non-inverse-transform methods that still induce positive correlation in the generated variates, as required for CRN to work.

If affordable, a small pilot study could provide a preliminary check on the efficacy of CRN for the alternative configurations. In the case of two configurations, make n replications of each, *using* CRN, to obtain output observations X_{1j} and X_{2j} for $j = 1, 2, \ldots, n$. Let $S_1^2(n)$ and $S_2^2(n)$ be the sample variances [using Eq. (4.4)] of the X_{1j}'s and X_{2j}'s, respectively, and let $S_Z^2(n)$ be the sample variance of the differences, $Z_j = X_{1j} - X_{2j}$; since the runs were made using CRN, $S_Z^2(n)$ is an unbiased estimator of the variance of a Z_j under CRN. Regardless of the fact that we used CRN, $S_1^2(n)$ is unbiased for $\text{Var}(X_{1j})$ and $S_2^2(n)$ is unbiased for $\text{Var}(X_{2j})$, so $S_1^2(n) + S_2^2(n)$ is an unbiased estimator of the variance of a Z_j if we *were* to make the runs without CRN. Thus, if CRN is working, we would expect to observe that $S_Z^2(n) < S_1^2(n) + S_2^2(n)$, and the difference estimates how much CRN is reducing the variance of a Z_j. Of course, any extra programming that might be necessary to implement CRN would have to be done for such a pilot study, whether or not CRN is ultimately adopted. While there are some examples of CRN's backfiring, as

observed by Wright and Ramsay (1979) and Koenig and Law (1982) for inventory simulations, we feel that CRN is generally a valuable tool that should be given serious consideration by an analyst faced with the task of comparing two or more alternative configurations.

Another possible drawback to CRN is that formal statistical analyses can be complicated by the induced correlation. Adaptations for standard analysis-of-variance tests in the presence of this correlation are discussed by Heikes, Montgomery, and Rardin (1976) and by Kleijnen (1979). Further issues regarding statistical analysis in the presence of CRN-induced correlation are discussed by Nelson (1987b), who also deals with similar problems created by the antithetic-variates VRT treated in Sec. 11.3. Nozari, Arnold, and Pegden (1987) address the issue of statistical analysis in the general framework of Schruben and Margolin (1978) for correlation induction in simulation experiments, and Tew and Wilson (1992) develop tests for applicability of their methods. Nelson and Hsu (1993) consider CRN with the MCB procedure of Sec. 10.3.3, and Kleijnen (1992) discusses the effect of CRN on regression metamodels of simulations (described in Sec. 12.4). See also the discussion in Sec. 10.4.3 under the heading "Correlation Between Alternatives," relating to CRN in ranking-and-selection procedures.

11.2.3 Synchronization

To implement CRN properly, we must match up, or *synchronize*, the random numbers across the different system configurations on a particular replication. *Ideally*, a specific random number used for a specific purpose in one configuration is used for *exactly the same* purpose in all other configurations. For instance, if a specific U_k is used in the first of two alternative queueing configurations to generate a specific service time, then it should be used in the second configuration to generate the same service time (rather than an interarrival time or some other service time) as well; otherwise, the benefit of CRN could be lost, or (worse) backfiring might occur. In particular, it is generally *not* enough just to start off the simulations of all configurations with the same seed of a random-number stream, which results in all simulations using the same random numbers $U_1, U_2, \ldots$. This is illustrated by the following example.

EXAMPLE 11.1. Recall the two competing designs for the automated teller machine of Examples 10.1 and 10.2. The first configuration (one Zippytel machine) is an *M/M/*1 queue, and the second (two Klunkytels) is an *M/M/*2 queue, both with utilization factor $\rho = 0.9$. The performance measure of interest is the expected average delay in queue for the first 100 customers given that the first customer finds the system empty and idle. Thus, X_{ij} is the average delay in the *M/M/i* queue on the jth replication, for $i = 1, 2$. In Examples 10.1 and 10.2, we generated X_{1j} and X_{2j} independently for the 100 independent replications, but we could have used CRN.

In an attempt to do so, we used a single random-number stream (see Secs. 2.3 and 7.1) to generate both interarrival and service times, and we simply reset the stream's seed back to its original value before simulating the second configuration. For the

TABLE 11.1
Use of the first five random numbers for the *M/M/1* and *M/M/2* simulations

Random number	Usage in *M/M/1*	Time of usage	Usage in *M/M/2*	Time of usage
U_1	A_1	0	A_1	0
U_2	A_2	A_1	A_2	A_1
U_3	S_1	A_1	S_1	A_1
U_4	A_3	A_1+A_2	A_3	A_1+A_2
U_5	A_4	$A_1+A_2+A_3$	S_2	A_1+A_2

M/M/1 case, the program logic of Sec. 1.4 was used. In particular, when a customer arrives, we first generate the time of arrival of the next customer. If the server is idle when a customer arrives, then we generate the customer's service time immediately and schedule the departure event for this customer. On the other hand, if the server is busy, then the arriving customer joins the queue and we do *not* generate his or her service time until the customer enters service, after a delay in queue. The *M/M/2* queue was programmed similarly. To see how the usage of random numbers can get out of synchronization, suppose, e.g., that customers 2 and 3 arrive before customer 1 departs. In Table 11.1 we give the usage of the first five random numbers for the two simulations, as well as the time that these random numbers are used. The random number U_1 is used at time 0 in both simulations to generate A_1, so that the time of the first arrival can be scheduled into the event list. The first customer arrives in both simulations at time A_1, and the random number U_2 is then used to generate A_2 and to schedule the arrival of the second customer. Since there is an idle server in both simulations when the first customer arrives, U_3 is used to generate S_1 and to schedule the time of departure of the first customer. The second customer arrives at time $A_1 + A_2$ (before the first customer departs), at which time U_4 is used to generate A_3. Since there is an idle server when the second customer arrives for the *M/M/2* simulation, U_5 is used at this time to generate S_2. However, in the *M/M/1* simulation, there is no available server when the second customer arrives, and so this customer joins the queue and his or her service time is not generated at this time. The next event in this simulation is the arrival of the third customer at time $A_1 + A_2 + A_3$, at which time U_5 is used to generate A_4. Thus, U_5 is used to generate A_4 and S_2 in the *M/M/1* and *M/M/2* simulations, respectively, and the usage of the random numbers is no longer synchronized.

In Table 11.2 we show how the interarrival times and service times are actually generated from the U_k's. (Recall from Sec. 8.3.2 that an exponential random variate is generated as minus the desired mean times the natural log of a random number.) Note from this table that the correlation between the first service time for the *M/M/1* queue and the first service time for the *M/M/2* queue is a perfect +1 (see Prob. 4.11), which is highly desirable. On the other hand, the correlation between A_4 for the *M/M/1* queue and S_2 for the *M/M/2* queue is also +1. This is not the intended effect, since, e.g., a large value of U_5 will tend to make X_{1j} smaller and X_{2j} larger.

Thus, we cannot in general expect CRN to be implemented properly if we merely recycle the same random numbers without paying attention to how they are used. The poor synchronization in Example 11.1 is due in part to the particular way

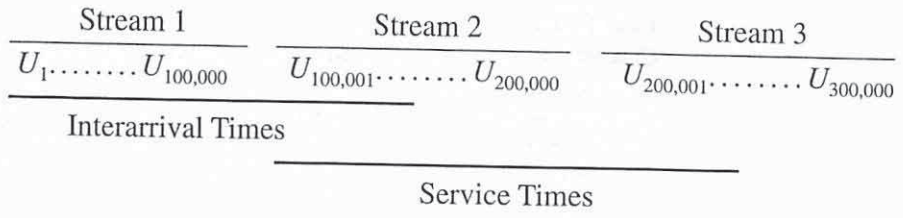
TABLE 11.2
Computation of the A_k's and the S_k's for the *M/M/1* and *M/M/2* queues

Random number	*M/M/1* queue	*M/M/2* queue
U_1	$A_1 = -\ln U_1$	$A_1 = -\ln U_1$
U_2	$A_2 = -\ln U_2$	$A_2 = -\ln U_2$
U_3	$S_1 = -0.9 \ln U_3$	$S_1 = -1.8 \ln U_3$
U_4	$A_3 = -\ln U_4$	$A_3 = -\ln U_4$
U_5	$A_4 = -\ln U_5$	$S_2 = -1.8 \ln U_5$

we programmed the simulations. We did not, however, consciously try to destroy the synchronization, but wrote code in a way that seems reasonable and is in fact correct. The issue of coding for correct synchronization is considered below and in Example 11.4, which also illustrates the statistical consequences of ignoring synchronization.

How difficult it is to maintain proper synchronization in general depends entirely on the model structure and parameters, and on the methods used to generate the random variates needed in the simulations. Several programming "tricks" could be considered to maintain synchronization in a given simulation:

- If there are multiple streams of random numbers available (see Secs. 2.3 and 7.1), or if there are several different random-number generators operating simultaneously, we could "dedicate" a stream (or generator) to producing the random numbers for each particular type of input random variate. In a queueing simulation, for instance, one stream could be dedicated to generating the interarrival times, and a different stream could be dedicated to service times. Stream dedication is generally a good idea, and most simulation packages have facility for separate random-number streams. (The number of different streams readily available, however, may not be entirely adequate for large simulations.) Moreover, since streams are usually just adjacent segments of a single random-number generator's output and thus have a particular length, care should be taken to avoid overlapping them when doing long simulations or when replicating intensively. A back-of-the-envelope calculation might indicate roughly how many random numbers will be used from a stream, and appropriate assignments can then be made. For example, in a simple single-server queueing simulation where about 5000 customers are expected to pass through the system, each will need an interarrival time and a service time. If the inverse-transform method (see Sec. 8.2.1) is used to generate all these variates, we would need about 5000 random numbers from each stream; if we were to replicate the simulation 30 times, we would go through some 150,000 random numbers from each stream. If the streams are, say, 100,000 long and we dedicated stream 1 to interarrival times and stream 2 to service times (as usual), we see that the last 50,000 random numbers used for the interarrival times would actually be the same as the first 50,000 used for the service times, destroying the replications' independence:



The remedy is, of course, to skip some stream assignments: Keep stream 1 (and the first half of stream 2) for the interarrival times, and use, for example, stream 6 (and the first half of stream 7) for the service times.

- The inverse-transform method for generating random variates (see Sec. 8.2.1) can facilitate synchronization since we always need *exactly* one random number to produce each value of the desired random variable. By contrast, the acceptance-rejection method (Sec. 8.2.4), for example, uses a *random* number of U(0, 1) random numbers to produce a single value of the desired random variable. The inverse-transform method, moreover, monotonically transforms the random numbers (see Sec. 11.2.2), and induces the strongest possible positive correlation between the generated variates that then serve as input to the simulations, as discussed in Sec. 8.2.1; this correlation will hopefully propagate through to the simulation output to yield the strongest variance reduction.
- It might be helpful to "waste" some random numbers at certain points in simulating some models. Problem 11.4 gives such an example.
- In some queueing simulations we could generate all of the service requirements of a customer at the time of arrival instead of when the customer actually needs them, and store them as attributes of the customer. Example 11.5 below illustrates this idea for implementing CRN for the alternative job-shop models of Sec. 2.7.3. Also, this approach would have ensured synchronization of the *M*/*M*/1 vs. *M*/*M*/2 systems in Example 11.1, preventing the mix-up of random-number usage depicted in Table 11.1. However, this approach could have the practical disadvantage of requiring a lot of computer memory if there are many attributes per entity and the number of concurrent entities in the simulation becomes large. And the difficulty of taking this approach will generally depend on the model's particular logic; for instance, if there is an inspection of entities at some juncture in the model with the possibility of multiple failures and feedback loops, it might not be clear how to pre-generate and store as attributes all of the inspection times that *might* be required for such an entity (see Prob. 11.19).

It is important when using CRN with multiple replications to make sure that the random numbers are synchronized across the different models for replications beyond just the first one, as will be illustrated by the following example.

EXAMPLE 11.2. Consider the inventory model of Sec. 1.5, and suppose we want to compare the results for $(s, S) = (20, 40)$, which we call model 1, vs. $(s, S) = (20, 100)$, which we call model 2. There are three sources of randomness: the times between successive demands, the sizes of those demands, and the delivery lag when an order is placed to the supplier. If we dedicate stream 1 to generating interdemand times,

stream 2 to generating demand sizes, and stream 3 to generating delivery lags, we will ensure proper synchronization of all random-number usage across models 1 and 2 in replication 1. Furthermore, due to the nature and logic of this model, we will be using the same *number* of random numbers from stream 1 to generate the interdemand times for both models 1 and 2, and the same *number* of random numbers from stream 2 to generate the demand sizes for both models. However, since the reorder point s is 20 for both models, and the order-up-to level S is only 40 for model 1 but 100 for model 2, the order amounts in model 1 will tend to be considerably smaller than in model 2, so we will be placing more (small) orders in model 1 than in model 2, requiring generation of more delivery lags in model 1 than in model 2. As a result, we will use up more of stream 3 in simulating model 1 than we will use in simulating model 2. So if we begin replication 2 of both models where streams 1, 2, and 3 left off after finishing the first replications of models 1 and 2, we will indeed get proper synchronization across the models for the interdemand times (stream 1) and demand sizes (stream 2), but not for the delivery lags (stream 3); thus, we will not have synchronization for delivery lags across the two models for replication 2 and beyond. One possible remedy is to abandon streams 1, 2, and 3 after the first replication, and start the second replication with (say) streams 4, 5, and 6 for interdemand times, demand sizes, and delivery lags, respectively, then move on to streams 7, 8, and 9 for replication 3, etc. Assuming that we do not use up more than a whole stream for any source of randomness on any replication, and that we have enough random-number streams available to carry out the desired number of replications, this will bring the delivery lags back into proper synchronization for all replications, possibly strengthening the effect of CRN. What is another remedy that requires fewer streams (see Prob. 11.17)? Example 11.7 discusses whether delivery lags *should*, in fact, be the same across the two models.

It might be mentioned that the Arena [Kelton et al. (2004, p. 512)], AutoMod [Banks (2004, p. 367)], and WITNESS [Lanner (2006)] simulation packages have special features for facilitating synchronization beyond the first replication (see Sec. 7.3.2).

Even if one is armed with programming tricks such as those discussed above, it may simply be impossible to attain full synchronization across all configurations under study. Also, the extra programming effort, computation time, or storage requirements needed for full synchronization might not be worth the realized variance reduction. Thus, we might consider synchronizing *some* of the input random variates and generating others independently across the various configurations. For instance, it might be convenient to synchronize interarrival times but not service times in a complicated network of queues. In the final analysis, the benefit of using common random numbers and the degree to which we synchronize depend on the situation.

11.2.4 Some Examples

Since the applicability, power, and appropriate synchronization methods for CRN can be quite model-dependent, we will present several examples of its use in particular situations.

EXAMPLE 11.3. We now rework the *M*/*M*/1 vs. *M*/*M*/2 comparison of Example 11.1, but this time we synchronize correctly and present the actual simulation results. Using

TABLE 11.3
Statistical results of CRN for the *M/M/1* queue vs. the *M/M/2* queue

	I	A	S	A & S
$S^2(100)$	18.00	9.02	8.80	0.07
90% confidence-interval half-length	0.70	0.49	0.49	0.04
$\hat{p}$	0.52	0.37	0.40	0.03
$\widehat{\text{Cor}}(X_{1j}, X_{2j})$	−0.17	0.33	0.44	0.995

separate streams to implement CRN, we estimated the effect of various degrees of synchronization in four sequences of 100 pairs of simulations each. In the first sequence, denoted "I" in Table 11.3, all runs were independent; i.e., CRN was not used at all. In the second sequence ("A" in Table 11.3), the interarrival times for the two models were generated using CRN, but we generated the service times independently. For the third sequence ("S"), the interarrival times were independent but the service times were generated using CRN, so both systems experienced the "same" ordered sequence of arriving service demands, with those for the *M/M/2* being exactly twice as great as those for the *M/M/1*. Finally, the fourth sequence ("A & S") was fully synchronized, matching up both interarrival and service times. We can think of these four schemes in physical terms like this:

I — Different customers (in terms of their service requirements) arrive to the two configurations, and at different times.

A — Different customers arrive to the two configurations, but at the same times.

S — The same customers arrive to the two configurations, but at different times.

A & S — The same customers arrive at the same times to both configurations.

From the 100 pairs of simulations in each of the four cases we estimated $\text{Var}(Z_j)$ by the usual unbiased variance estimator $S^2(100)$, from Eq. (4.4) applied to the Z_j's. From this, the half-length of a nominal 90 percent confidence interval for ζ is $1.645\sqrt{S^2(100)/100}$. We computed as well the proportion $\hat{p}$ of the 100 pairs for which the "wrong" decision would be made, i.e., when $X_{1j} < X_{2j}$ [since $E(X_{1j}) > E(X_{2j})$, as discussed in Example 10.1]. As a more direct check on whether CRN is inducing the desired positive correlation, we also estimated the correlation between X_{1j} and X_{2j} by

$$\widehat{\text{Cor}}(X_{1j}, X_{2j}) = \frac{\frac{1}{99}\sum_{j=1}^{100}[X_{1j} - \overline{X}_1(100)][X_{2j} - \overline{X}_2(100)]}{\sqrt{S_1^2(100)S_2^2(100)}}$$

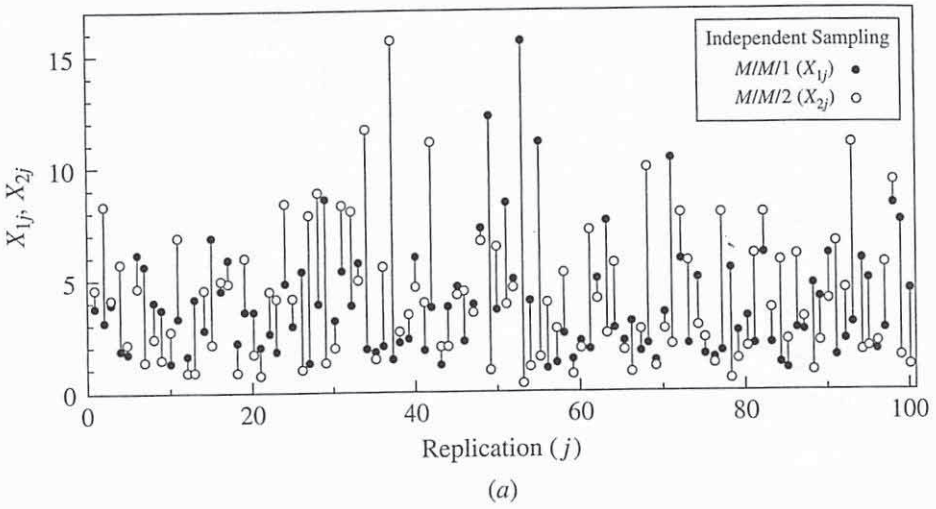
where $\overline{X}_i(100)$ is the sample mean of the X_{ij}'s over j, and $S_i^2(100)$ is the sample variance of the X_{ij}'s over j (see Prob. 4.25).

It is clear from Table 11.3 that the estimated variance reduction attained by full synchronization (A & S) compared with independent sampling (I) is quite significant here, being a reduction of over 99 percent, probably since the two systems are quite similar. Accordingly, the confidence-interval half-length fell from 0.70 (I) to 0.04 (A & S), a reduction of almost 95 percent. Looked at another way, we ask how many replications of each system under independent sampling we would need in order to achieve a precision in our estimator $\overline{Z}(n)$ for ζ (perhaps measured by confidence-interval half-length) equal to that from fully synchronized CRN. If we made n_I replications of each system under

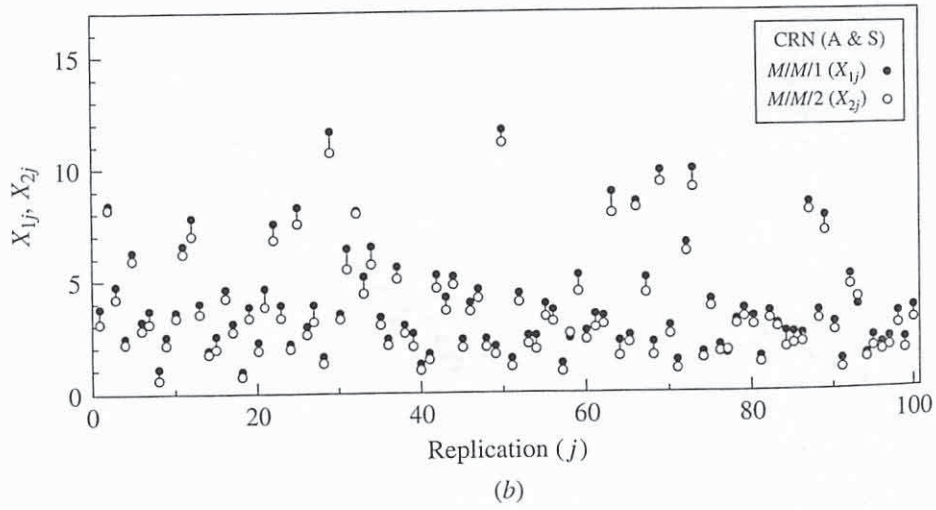
independent sampling, the half-length would be approximately proportional to $\sqrt{18.00/n_I}$ (ignoring degrees of freedom); on the other hand, if we made n_C replications of each system under completely synchronized CRN, the half-length would be approximately proportional to $\sqrt{0.07/n_C}$. Equating the two square roots, we find that $n_I/n_C = 257.14$; i.e., we would need more than 250 times as many replications under independent sampling to get the same precision we would get with fully synchronized CRN.

The estimated probability of making the wrong decision is reduced from 52 percent to 3 percent (see Example 10.1). Looking at the estimated correlations, we see that fully synchronized CRN induced an extremely strong correlation between the two configurations' output measures, explaining the large reduction in variance. For the two partial synchronization schemes, we experienced weaker (but still positive) correlations, and correspondingly weaker variance reductions and only limited drops in $\hat{p}$.

The effect of CRN (completely synchronized) in this example can be expressed graphically in several ways. In Fig. 11.2*a* are the individual-replication results for the



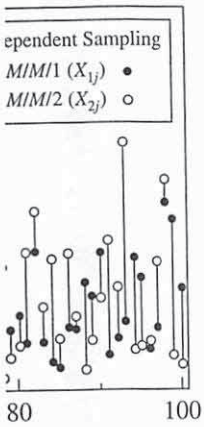
(*a*)

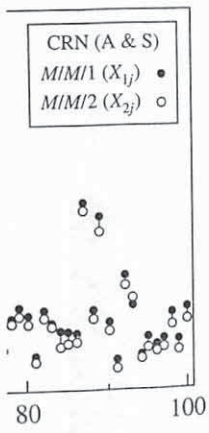


(*b*)

FIGURE 11.2
M/*M*/1 vs. *M*/*M*/2: individual replications.

ortional to $\sqrt{18.00/n_I}$
; replications of each
ıld be approximately
that $n_I/n_C = 257.14$;
er independent sam-
:d CRN.
uced from 52 percent
ions, we see that fully
en the two configura-
e. For the two partial
ive) correlations, and
ps in $\hat{p}$.
ple can be expressed
ication results for the





100 pairs of runs under independent sampling, versus the replication number. The solid circles are the average delays in queue for the *M/M/*1 model (X_{1j}'s), and the hollow circles are for the *M/M/*2 case (X_{2j}'s); for each fixed j, X_{1j} and X_{2j} are connected by a vertical line, whose length is thus $|Z_j|$. Note in particular that there are 52 pairs for which the hollow circle (*M/M/*2) is at the top of a line and the solid circle (*M/M/*1) is at the bottom, which is the wrong order (in terms of their expectations), and corresponds to $\hat{p} = 0.52$ in the "I" column of Table 11.3. Figure 11.2*b* does likewise, but under completely synchronized CRN. The better-behaved nature of the Z_j's is apparent, with none of the very long vertical lines that appear in Fig. 11.2*a*, and with the line lengths being much more consistent within themselves. This is because the variance of $|X_{1j} - X_{2j}|$ is smaller when there is positive correlation, since a large value of X_{1j} tends to be accompanied by a large value of X_{2j}, and small values of X_{1j} and X_{2j} tend to occur together as well. Moreover, there are only three cases in Fig. 11.2*b* where a line has the *M/M/*2 hollow circle at the top and the *M/M/*1 solid circle at the bottom, corresponding to $\hat{p} = 0.03$ in the "A & S" column of Table 11.3.

A direct way of seeing the correlation that CRN induced in this example is shown in Fig. 11.3, where we plot the pairs (X_{1j}, X_{2j}) for both independent sampling (hollow triangles) and completely synchronized CRN (solid triangles). While there is no apparent pattern in the independent pairs, we note an extremely straight (and positively sloping) alignment of the CRN pairs, corresponding to the very strong positive correlation estimate (0.995) in the "A & S" column of Table 11.3.

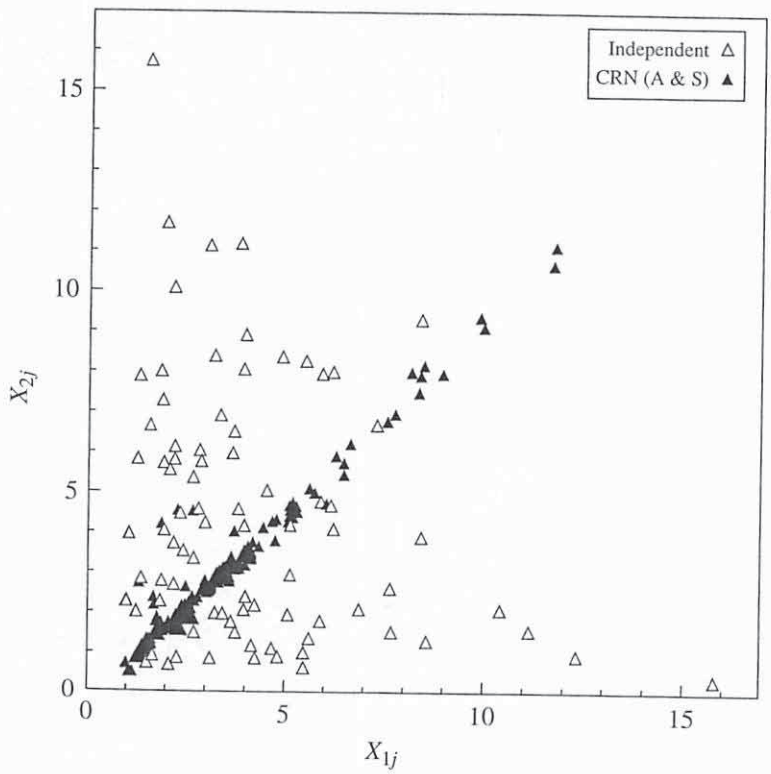


FIGURE 11.3
Correlation plot of *M/M/*2 average delays (vertical axis) vs. *M/M/*1 average delays (horizontal axis).

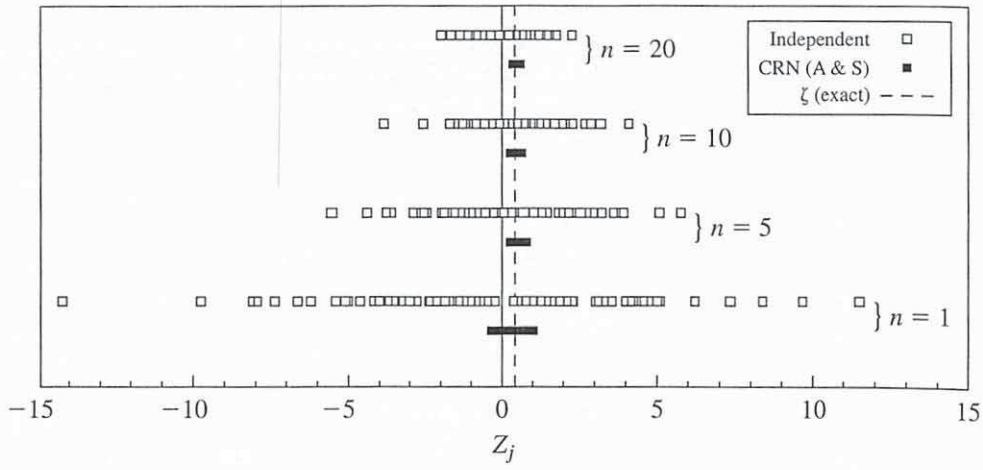


FIGURE 11.4
*M/M/*1 vs. *M/M/*2: differences.

In the "$n = 1$" pair of dot plots at the bottom of Fig. 11.4, each square represents a Z_j, plotted according to the scale shown at the bottom, with the hollow squares being the results of independent sampling, and the solid squares (which run together in the plot) being the results of fully synchronized CRN. The spread of the CRN Z_j's about their expectation (ζ) is much narrower than for the independent-sampling case. We also show in Fig. 11.4 plots of 100 observations on $\overline{Z}(n)$ for $n = 5$, 10, and 20, plotted on the same scale (these correspond to the similar cases in Fig. 10.2 in the case of independent sampling). It is striking to note that we do much better in terms of spread for CRN with $n = 1$ than we do for independent sampling for n as high as 20. Thus, CRN here gave us much better results than we could have gotten by spending more than 20 times as much with independent sampling!

Unlike Example 11.1, we implemented CRN in Example 11.3 with proper synchronization techniques, using separate streams for the two different sources of randomness. The next example illustrates the statistical importance of maintaining proper synchronization.

EXAMPLE 11.4. We reran the experiments of Example 11.3, supposedly using the full "A & S" CRN, except this time we generally ignored synchronization. Our codes are all still "correct," i.e., faithfully simulate the two models, and represent programs that one could actually write, without malice aforethought to disrupt synchronization intentionally. In "Code 1" we generated service requirements upon arrival, and we used the same random-number stream throughout. As it happens, this still results in proper synchronization, as evidenced in Table 11.4. (The numbers differ from the "A & S" column of Table 11.3 since separate streams were used there.) "Code 2" of Table 11.4 uses a single stream but does not generate the service times upon arrival, waiting until a customer enters service to generate the service time; in this case, the random numbers will get mixed up in terms of their usage, as demonstrated in Example 11.1. As seen in Table 11.4, the results are not much better than the "I" case in Table 11.3, since we have lost the benefit (apparently nearly all) of CRN. "Code 3" is the same as Code 2, except that the next interarrival time is generated at the end of the arrival subprogram (see Sec. 1.4), representing yet another valid but nonsynchronized code; again, no benefit is seen.

TABLE 11.4
Statistical results of properly (Code 1) and improperly (Codes 2 and 3) synchronized CRN for the *M/M/1* queue vs. the *M/M/2* queue

	Code 1	Code 2	Code 3
$S^2(100)$	0.07	16.80	12.00
90% confidence-interval half-length	0.04	0.67	0.57
$\hat{p}$	0.07	0.43	0.42
$\widehat{\text{Cor}}(X_{1j}, X_{2j})$	0.997	0.02	−0.03

The examples so far in this chapter (except 11.2) have involved the same *M/M/1* vs. *M/M/2* configurations. These are very simple systems for which properly synchronized CRN is relatively easy to implement. They are also quite similar to each other, probably accounting for the dramatic variance reductions through CRN that we have seen. The next two examples involve models that are considerably more complicated, and consider as well using CRN for steady-state parameters (see Secs. 9.3 and 9.5).

EXAMPLE 11.5. In the manufacturing-system model of Sec. 2.7, recall the discussion at the end of Sec. 2.7.3. As in the last three rows of Table 2.1, let configurations 1, 2, and 3 be, respectively, the manufacturing system obtained by adding a machine to station 1, 2, and 4; and let ν_i be the *steady-state* expected overall average job total delay in configuration i for $i = 1$, 2, and 3. Suppose that we want to estimate $\zeta_{12} = \nu_1 - \nu_2$, $\zeta_{13} = \nu_1 - \nu_3$, and $\zeta_{23} = \nu_2 - \nu_3$ by making simulations of length 100 eight-hour days of each of the three configurations but using the first 10 of these 100 days as a warmup period and collecting data on only the last 90 days (see Prob. 2.7). Let X_{ij} be the overall average job total delay over these 90 days for configuration i as observed on the jth replication, and let $Z_{12j} = X_{1j} - X_{2j}$, $Z_{13j} = X_{1j} - X_{3j}$, and $Z_{23j} = X_{2j} - X_{3j}$. We assume that the 10-day warmup period is sufficient, so that $E(Z_{i_1 i_2 j}) \approx \zeta_{i_1 i_2}$.

We made the runs of the different configurations completely independent of each other (denoted "I" in Table 11.5 below), and also used CRN across the three configurations, as follows. For each configuration, we used the same interarrival times for the jobs and made the sequence of job types the same. Further, when a job arrived at the system and its type was determined, we immediately generated its service requirements that will be needed later as it moves along its route through the system and stored them as additional attributes of this job. Thus, when a job entered service at a particular station, its service time was taken from its appropriate attribute for this station. In this

TABLE 11.5
CRN for the three manufacturing-system configurations

	I	CRN	Variance reduction (%)
$S^2_{12}(10)$	6.27	1.46	77
$S^2_{13}(10)$	10.40	2.35	77
$S^2_{23}(10)$	23.08	0.87	96

way, we used synchronized CRN for all sources of randomness. Note also that in this example all types of random variates are generated at the same instant in simulated time (when a job arrives), so a *single* random-number stream could have been used for all sources of randomness.

Let $S^2_{i_1 i_2}(10)$ be the usual unbiased estimator of $\text{Var}(Z_{i_1 i_2 j})$, which we computed from 10 independent replications using both independent sampling and CRN, as given in Table 11.5. Here, CRN led to variance reductions ranging from 77 percent to 96 percent, depending on which two configurations are being compared. In terms of the required number of replications to achieve a desired confidence-interval half-width, we can proceed as in Example 11.3 to find that independent sampling would need 4.29 (= 6.27/1.46) times as many replications as CRN to compare configurations 1 and 2, 4.43 times as many for configurations 1 and 3, and more than 26 times as many if we are interested in the difference between configurations 2 and 3.

EXAMPLE 11.6. In Example 10.5 we compared two configurations of the manufacturing facility from Example 9.25. In the second configuration, the mean inspection time was smaller. We are again interested in the steady-state mean time in system of parts, so we followed the replication/deletion approach (Sec. 9.5.2) and made moving-average plots for both configurations as in Fig. 10.3. Again letting l_i and m_i be the length (in parts) of the warmup period and the minimum replication length for configuration i, we obtained $m_1 = m_2 = 9445$, $l_1 = 1929$, and $l_2 = 1796$. As in Example 10.5, we made $n = 20$ independent pairs of runs, but we now used separate streams to synchronize the random numbers for all six sources of randomness in this model [interarrival times, machine processing times, inspection times, good/bad decisions, machine operating (up) times, and machine repair (down) times]. Again, we formed a 90 percent confidence interval for the difference between the steady-state mean times in system from these 20 replications, and obtained 1.98 ± 0.18; recall that in Example 10.5 the corresponding interval was 2.36 ± 0.31, from 20 replications of each configuration that were close to the same length and from which close to the same amount of initial data were deleted. Thus, CRN reduced the size of the confidence interval on the difference between the performance measures by 42 percent of its original size, corresponding to an estimated variance reduction of 66 percent. As more direct evidence that CRN is working properly, we estimated the correlation (see Example 11.3) between X_{1j} and X_{2j} to be 0.98.

The next pair of examples of CRN involves an inventory model, for which complete synchronization of all sources of randomness is of debatable correctness; thus, we generated some inputs using properly synchronized CRN and others independently.

EXAMPLE 11.7. For the inventory model (defined in Sec. 1.5) that we considered in Example 10.3, recall that (s, S) was (20, 40) for configuration 1, and was (20, 80) for configuration 2. For each configuration we can arrange for demands of the same size to occur at the same times; i.e., we use CRN (via separate streams) for the demand-size and interdemand-time sources of randomness. Due to the different values of S, however, orders will generally be placed at different times and for different amounts for the two policies, and so the number of orders placed will also differ under the two policies. Thus, it is not clear how we could reasonably match up the delivery-lag random variates (or whether it even makes sense to match them up), so we just generated them independently across the configurations. (In Example 11.2 we attempted to synchronize the delivery lags to illustrate how the random numbers could get out of synchronization on replications 2, 3 . . . .)

As in Example 10.3, we made $n = 5$ independent pairs of simulations, but here with (partial) CRN as just described. We obtained $\overline{Z}(5) = 3.95$ and $\widehat{\text{Var}}[\overline{Z}(5)] = 0.27$,

so that the paired-t 90 percent confidence interval is [2.84, 5.06]. Comparing this with the independent-sampling results of Example 10.3, the estimated variance is reduced by about 89 percent, and the confidence-interval half-length is some 67 percent smaller. Thus, it would take about 45 replications of each system under independent sampling to get a confidence interval as small as the one we got from partial CRN with only 5 replications of each.

The following example illustrates how the multiple confidence-interval methods described in Sec. 10.3 can be sharpened with CRN.

EXAMPLE 11.8. Consider now the five different policy configurations defined in Table 10.4. As in Example 10.7, we first regard policy 1, where $(s, S) = (20, 40)$, as the standard against which the other four are to be compared. We reran the analysis of Example 10.7, again with $n = 5$ replications of each policy configuration, but now using the partial CRN sampling plan described in Example 11.7, across all five policies. Desiring overall confidence of at least 90 percent, we used the Bonferroni inequality (see Sec. 10.3) to form four individual 97.5 percent confidence intervals for $\mu_i - \mu_1$, exactly as described in Example 10.7; here, however, CRN implies that the results on a given replication (j) across the five policies are not independent, precluding use of the Welch approach to confidence-interval formation. Thus, Table 11.6 contains only the paired-t intervals corresponding to Table 10.6. The half-lengths obtained here are all quite a bit smaller than those in Table 10.6. Moreover, comparing the paired-t approaches only, CRN enabled us to identify one more statistically significant difference (between policy 2 and the standard) than we were able to in Example 10.7.

We can also effect the all-pairwise-comparisons analysis of Sec. 10.3.2 using CRN; this was done with independent sampling in Example 10.8, with the results given in Table 10.7. Since there are now 10 individual confidence intervals, we make each at level 99 percent to attain overall confidence of at least 90 percent. The intervals resulting from CRN (again, only the paired-t approach is valid) in Table 11.7 indicate once again that CRN markedly reduced confidence-interval length in all cases except one ($i_2 = 5, i_1 = 2$). In this case the paired-t interval under independent sampling (22.12 ± 3.80) from Table 10.7 is smaller than the CRN interval (23.64 ± 5.02) from Table 11.7. Looking back at Table 10.7, the interval in this case was the smallest one observed, possibly due to simple sampling fluctuation, and at any rate the difference is significant both there and here. Perhaps more important, we see in Table 11.7 that 8 of the 10 CRN-based

TABLE 11.6
Individual 97.5 percent confidence intervals for all comparisons with the standard policy ($\mu_i - \mu_1, i = 2, 3, 4, 5$) using CRN; * denotes a significant difference

		Paired-t	
i	$\bar{X}_i - \bar{X}_1$	Half-length	Interval
2	−3.95	1.83	(−5.78, −2.12)*
3	1.02	2.65	(−1.63, 3.68)
4	5.94	1.41	(4.53, 7.35)*
5	19.69	2.28	(17.41, 21.97)*

TABLE 11.7
Individual 99 percent confidence intervals for all pairwise comparisons ($\mu_{i_2} - \mu_{i_1}$ for $i_1 < i_2$) using CRN; * denotes a significant difference

		Paired-t			
		i_2			
		2	3	4	5
i_1	1	-3.95 ± 2.41*	1.02 ± 3.49	5.94 ± 1.86*	19.69 ± 3.00*
	2		4.97 ± 5.62	9.89 ± 3.68*	23.64 ± 5.02*
	3			4.92 ± 2.39*	18.67 ± 1.69*
	4				13.75 ± 2.03*

intervals miss zero (indicating a statistically significant difference between the corresponding configurations), whereas only 6 or 7 of the 10 in Table 10.7 (depending on the approach) missed zero. Thus, with no more sampling we were able to sharpen our comparisons among these policies.

EXAMPLE 11.9. Consider again the problem of comparing the original and proposed routing policies for the communications network of Example 10.6. We once again made five independent replications of each policy of length $m = 65$ seconds and used a warmup period of length $l = 5$ seconds, but now partial CRN was used across the two policies. For each configuration, we generated outgoing messages for a particular SP at the same times, and with the messages having the same sizes and destinations. However, when each of two links had to be chosen with a probability of 0.5, we used different random numbers for the two configurations. We used Eq. (10.1) to obtain [0.05, 0.12] (in millisecond) as an approximate 90 percent confidence interval for $\nu_1 - \nu_2$. Since the confidence interval does not contain 0, the difference in the two steady-state means *is* statistically significant, whereas it was not in Example 10.6. Furthermore, the sample variance of the Z_j's has been reduced by about 97 percent, the confidence-interval half-length is some 83 percent smaller, and the estimated correlation (see Example 11.3) between X_{1j} and X_{2j} is 0.94.

In summary, we have found that we could realize at least partial synchronization across the different system configurations in many real-world simulation studies that we have performed.

11.3 ANTITHETIC VARIATES

Antithetic variates (AV) is a VRT that is applicable to simulating a *single* system, as are the rest of the VRTs in this chapter. As in CRN, we try to induce correlation between separate runs, but now we seek *negative* correlation.

The central idea, dating back at least to Hammersley and Morton (1956) in the context of Monte Carlo simulation, is to make *pairs* of runs of the model such that a "small" observation on one of the runs in a pair tends to be offset by a "large" observation on the other one; i.e., the two observations are negatively correlated. Then if we use the *average* of the two observations in the pair as a basic data point for analysis, it will tend to be closer to the common expectation μ of an observation