CHAPTER 95

Statistical Analysis of Simulation Results

BARRY L. NELSON
Northwestern University

1. INTRODUCTION

The design and control of many industrial and service systems requires the industrial engineer to account for uncertainty: uncertainty about the demand for products, the reliability of a machine, the arrival of customers, or the availability of components, for example. The language of probability is the primary tool for modeling uncertain or *stochastic* systems. Mathematical analysis, numerical analysis and computer simulation are techniques for analyzing stochastic models. Chapter 83 of the Handbook covers mathematical analysis of stochastic models. Chapters 93 and 94 describe the process of modeling a stochastic system and translating that model into a simulation program, respectively. This chapter provides guidelines for the design and analysis of computer simulation experiments. For a more extensive textbook treatment of these topics see Banks et al. (2000).

Compared to the other techniques, simulation is a brute force approach: using pseudorandom numbers to reproduce the uncertainty in the model, sample model behavior is generated and analyzed,

Handbook of Industrial Engineering: Technology and Operations Management, Third Edition.
Edited by Gavriel Salvendy

much in the same way that sample behavior of the physical system might be collected and analyzed. Simulation is perhaps the most general analysis technique, but the price of generality is that simulations provide only *estimates* of model performance parameters. The best that can be guaranteed is that these estimates converge to the true performance parameters as the simulation becomes infinitely long.

The emphasis in this chapter is on problems that are common in, or unique to, computer simulation relative to statistical experiments as a whole. The reader should refer to Chapters 83 to 87 of the Handbook for basic statistical methods. No specific computer hardware or software is assumed other than a language for programming the simulation experiment and possibly a statistical-analysis package capable of standard procedures such as least-squares regression.

Where possible, algorithms and procedures are provided rather than referenced. In that sense the chapter is self-contained. When references are provided, textbooks are often cited rather than original research papers, since they are more accessible.

Due to space constraints, only one or two methods are included for any problem, which does not imply that they are necessarily the best methods for any particular problem. The methods chosen are theoretically sound, empirically tested, but often conservative. The chapter is up to date, but it is not a guide to current research or emerging techniques.

One omission is the use of animation for simulation output analysis. Animation can be extremely helpful for developing a simulation model, verifying and validating the model, communicating results, and studying unusual model behavior in detail. However, it is not a substitute for a thorough numerical and graphical analysis that encompasses a much longer, and more representative, simulation run than anyone can reasonably view as animation.

The chapter is organized around three examples that are described below. The examples are simple enough that they can be presented in detail (except for the computer code).

2. OVERVIEW

This section is an annotated index to the chapter so that, if desired, the user can find the information of critical interest directly. However, all users should read Section 3, which introduces the three examples that are used as illustrations throughout the chapter, and most users should read Section 4 and Section 5, which describe the source of randomness and the standard statistics available in many simulation languages. A brief description of the other sections in this chapter is given below:

- Section 6 describes estimators for means, probabilities, and quantiles.
- Section 7 describes methods for detecting and reducing the effect of the initial conditions in a simulation experiment that estimates long-run performance.
- Section 8 describes ways to evaluate the goodness of estimates, including standard-error and confidence-interval procedures.
- Section 9 describes methods for evaluating alternative systems.
- Section 10 describes a way to improve the precision of simulation estimators.

Many of the sections contain programming-language-independent algorithms for design and analysis procedures. To understand the algorithms, it is useful to know the following:

The symbol $\leftarrow$ indicates assignment; for example, $a \leftarrow 5$ assigns the value 5 to the variable a.

The scope of *for* and *if* are delimited by *endfor* and *endif,* respectively.

The notation $\lfloor x \rfloor$ means the greatest integer less than or equal to x.

The subscripts on variables that are vectors or arrays are indicated by square brackets; e.g., $y[i,j]$ for y_{ij}.

Addition, subtraction, multiplication and division are indicated by $+$, $-$, $*$ and $/$, respectively.

3. EXAMPLES

Three examples that will be used throughout the chapter to illustrate simulation design and analysis techniques are introduced in this section. Appropriate design and analysis depends on characteristics of the model and on the questions being asked, so these three examples have been chosen to illustrate important classes of problems. Although they are simpler than many practical models—in fact, they are so simple that simulation is not actually needed—the techniques are no more difficult to apply in complex models.

3.1. Static Simulation: Acceptance Sampling

Acceptance sampling is a widely used technique for economically assessing the quality of a "lot" of items. A single-sampling attributes plan of the form (n, c) specifies that n items will be sampled and the lot will be accepted if no more than $c < n$ defective items are discovered. The values of n and c are chosen based on the operating characteristic (OC) curve they imply. The OC curve is the probability of accepting a lot as a function of the lot quality, where quality is usually stated in terms of the probability of a defective item (see Chapter 69 of the Handbook).

OC curves for standard acceptance-sampling plans are derived under the assumption that the quality of items can be modeled as independent and identically distributed (i.i.d.) Bernoulli random variables. Although this model is often plausible, the quality of items produced by some processes exhibit statistical dependence. The goal of this simulation experiment is to estimate the OC curve for sampling plan (10, 1) when item quality is dependent.

Let $X_1, X_2, \ldots, X_n$ represent a sample of n items, where $X_i = 1$ if the ith item is defective and $X_i = 0$ if the ith item is acceptable. The probability that an item is defective is p, and the quality of any item may be dependent on the other items. Specifically, the items are assumed to have a joint Pólya distribution with pairwise correlation $\rho = 0.08$. Standard tables of sampling plans are not appropriate for this situation.

Let $Y = \Sigma_{i=1}^{n} X_i$ be the number of defective items in the sample. The performance parameter of interest is $\theta(p) = \Pr\{Y \leq c|p\}$, the probability that there are c or fewer defective items (i.e., that the lot is accepted) as a function of p.

This simulation experiment is called a *static simulation* because there is no explicit modeling of the passage of time ($X_1, X_2, \ldots, X_n$ need not even be arranged by order of selection). Although static simulations are conceptually the easiest to design and analyze, they nevertheless present important design and analysis problems. In addition, this example illustrates estimating a probability, $\theta(p)$.

3.2. Terminating Simulation: License Plate Renewal

A small city will allow auto owners to renew their license plates by mail, with each owner's renewal taking place during the month of his or her birth. The mail-in renewal applications will be processed by a clerk. It is anticipated that the rate of receipt of applications will increase steadily throughout the month, but that the load during all months is about the same. Mail-in renewals that are postmarked after the 27th day of the month are returned to the applicant without being processed. The city is interested in determining the performance level the clerk must attain, in terms of renewals processed per day, to prevent excessive delays or carryover from month to month.

The arrival of mail-in renewals is modeled as a Poisson process with time-dependent arrival rate $\lambda(t) = t$ renewals per day, where time $0 \leq t \leq 27$ is measured in days. In other words, on the first day of the month applications arrive at a rate of 1 per day, but by the end of the month they are arriving at a rate of 27 per day. The time to process an application is modeled as an exponentially distributed random variable with mean $1/\mu$ days, so that μ is the processing rate in applications per day.

Let Y_i be the delay in processing the ith application received during the month, and let N be the number of applications received during the month. A performance parameter of interest is

$$\theta(\mu) = E_\mu \left[\frac{1}{N} \sum_{i=1}^{N} Y_i \right]$$

the expected average delay for mail-in renewals as a function of the processing rate, μ.

This simulation experiment is called a *terminating simulation* because the parameter of interest is defined with respect to a finite time horizon, the time to process one month of renewal applications in this case. The example illustrates estimating transient or time-dependent performance, and also examining the effect of a continuous design variable, the processing rate μ.

3.3. Steady-State Simulation: Inventory Model

An (s,S) inventory system involves the periodic review of the level of inventory of some discrete unit. If the inventory position (units in inventory plus units on order minus units backordered) at a review is found to be below s units, then enough additional units are ordered to bring the inventory position up to S units. When the inventory position at a review is found to be above s units, no additional units are ordered. One possible goal is to select the values of s and S that minimize inventory cost. The third example is the (s, S) inventory model in Koenig and Law (1985) and Law and Kelton (2000).

TABLE 1 Inventory Policies

ℓ	s	S
1	20	40
2	20	80
3	40	60
4	40	100
5	60	100

Let $\{I_t; t = 1, 2, \ldots\}$ represent the inventory position just after a review at period t, and let $\{X_t; t = 1, 2, \ldots\}$ represent the demand for units of inventory in period t. The inventory position I_t changes in the following way:

$$I_{t+1} = \begin{cases} S, & \text{if } I_t - X_t < s \\ I_t - X_t, & \text{if } I_t - X_t \geq s \end{cases}$$

For convenience, the initial inventory position is taken to be $I_1 = S$; that is, the inventory position is initially at its maximum. The demand $\{X_t; t = 1, 2, \ldots\}$ is modeled as a sequence of i.i.d. Poisson random variables with mean 25 units.

In each period there are costs associated with the inventory position. If $I_t - X_t < s$, then in period $t + 1$ a cost of $32 + 3[S - (I_t - X_t)]$ is incurred, which is a fixed cost plus a per-unit cost of bringing the inventory position up to S. In period $t + 1$, if $I_{t+1} \leq X_{t+1}$, then a holding cost of $(I_{t+1} - X_{t+1})$ dollars is incurred; otherwise a shortage cost of $5(X_{t+1} - I_{t+1})$ dollars is incurred.

Let $Y_t^{(\ell)}$ be the cost incurred in period t under inventory policy ℓ; the inventory policies under consideration are given in Table 1. The performance parameters of interest are the long-run expected cost per period of each inventory policy, with a smaller expected cost being preferred. The assumption behind such a long-run analysis is that $Y_t^{(\ell)}$ converges in distribution to $Y^{(\ell)}$ as $t \to \infty$, which means that the cost per period converges to a limiting random variable whose distribution does not depend on time (period) t. The parameter of interest is $\theta^{\ell} = \mathrm{E}[Y^{(\ell)}]$, the long-run expected cost per period for policy ℓ.

This simulation experiment is called a *steady-state simulation* because the parameter of interest is associated with a limiting random variable. This example illustrates estimating a limiting quantity from a simulation experiment that is (necessarily) finite, and also comparing alternative systems (inventory policies) with the goal of selecting the best system.

4. RANDOMNESS IN SIMULATIONS

Modern simulation languages contain *pseudorandom number generators* that produce sequences of numbers—typically numbers in the interval (0,1)—that are difficult to distinguish from independent and identically distributed (i.i.d.) random numbers. In other words, an analyst subjecting a sequence of pseudorandom numbers to a battery of statistical tests would be unlikely to recognize that they were produced by a deterministic algorithm rather than by a random process. However, if the same simulation program is executed on the same computer at any time, identical results are obtained. This is a useful property because debugging programs would be difficult if results changed randomly.

This method of incorporating randomness into computer simulations has a profound impact on the design and analysis of simulation experiments. Most importantly, it means that different simulation runs will be dependent if they employ the same pseudorandom numbers. This can be good, yielding sharper comparisons between alternative systems, or bad, invalidating the assumptions behind statistical procedures that assume independent observations.

An important feature of most simulation languages is that they permit control of the pseudorandom numbers through random number *streams* or *seeds,* which permit the user to access different, and thus apparently independent, portions of the pseudorandom number sequence. Such control allows the user to induce dependence where desired or obtain independence where necessary.

To be more precise, suppose that a simulation language includes one pseudorandom number generator that produces an ordered sequence of pseudorandom numbers $u_1, u_2, \ldots, u_g$. The length of this sequence is necessarily finite; for many pseudorandom number generators $g < 2^{31} \approx 2$ billion, although generators that produce much longer sequences are becoming common.

Let the seeds or streams be denoted by $s_1, s_2, \ldots, s_h$. The seeds or streams are nothing more than reference points within the sequence $u_1, u_2, \ldots, u_g$. For instance, s_1 might correspond to starting the sequence at u_{2137}. Thus, the number of seeds $h \leq g$, and is typically much less than g with the reference points spaced widely apart. Note that some simulation languages allow the user to specify the *offset* between seeds, rather than having to specify the seeds themselves. In any event, since the

entire sequence $u_1, u_2, \ldots, u_g$ appears to be a sequence of i.i.d. random numbers, the subsequences between seeds also appear to be independent of each other. This property implies an important design and analysis principle:

Design and Analysis Principle 1. The assignment of random number streams or seeds is part of the design of a simulation experiment. Assigning the same streams or seeds to different simulations induces dependence, while assigning different seeds or streams to different simulations induces independence between simulation results.

The simulation-language user seldom works directly with the pseudorandom numbers $u_1, u_2, \ldots, u_g$, but rather specifies the probability distributions of the simulation *inputs*. A convention in this chapter is that input random variables are denoted generically by X. Examples of simulation inputs are:

- The item quality random variables $X_1, X_2, \ldots, X_n$ in the acceptance-sampling model, which have a joint Pólya distribution
- The arrival and processing times of renewal applications in the license-renewal model, which have time-dependent Poisson and exponential distributions, respectively
- The demand for inventory in period t, X_t, in the inventory model, which has a Poisson distribution

Observations or realizations of the inputs are obtained by transforming the pseudorandom numbers. One or more pseudorandom numbers may be required to produce each input, depending on what transformation is used. The inputs, X, are functions of the pseudorandom numbers, u, so they are completely determined by the seed or stream, s, say $X = X(s)$.

The purpose of performing a simulation experiment is to observe model performance. The observed model performance, called the *output,* is derived from realizations of the inputs and the (often complex) logic of the model. A convention in this chapter is that output random variables are denoted generically by Y. Since the outputs are functions of the inputs, they are also functions of the seeds or streams, say $Y = Y[X(s)]$. Examples of simulation outputs are:

- The number of defective items discovered in a sample, Y, in the acceptance-sampling model
- The delay experienced by the ith renewal application, Y_i, in the license-renewal model
- The cost in period t for inventory policy ℓ, $Y_t^{(\ell)}$, in the inventory model

In most simulation experiments a large number of outputs are generated. They are summarized by a *statistic,* which is often a sample average of the outputs, denoted $\overline{Y}$. The value of the statistic is used to estimate system performance, so statistics are also called *estimators*. The statistics are functions of the outputs, so they are also completely determined by the random number seeds or streams.

Perhaps it seems strange that statistical methods are employed to analyze a completely deterministic process, which a simulation is after the seeds are specified (particularly since many users accept the default seeds or streams). However, if the pseudorandom numbers cannot easily be distinguished from random numbers, then treating functions of these numbers—the inputs, outputs and statistics—as random variables will not be misleading.

The role of the pseudorandom number streams or seeds is important. This method of representing randomness, and the corresponding control it permits, is the primary difference between simulation experiments and classical statistical experiments.

5. SIMULATION STATISTICS

Many simulation languages compute statistics and generate reports automatically. To understand these statistics it is necessary to distinguish between within- and across-replication statistics (some simulation languages refer to a replication as a ''run''). To make the difference concrete, consider the license-renewal simulation. One replication or run of this simulation represents one month and generates outputs $Y_1, Y_2, \ldots, Y_N$, which are processing delays for the N applications received during the month. The sample average of these delays is

$$\overline{Y} = \frac{1}{N} \sum_{i=1}^{N} Y_i$$

which is a *within-replication statistic,* meaning a summary of the outputs within one replication or run.

If k months are simulated, then each month yields a sample average delay, say $\overline{Y}_1, \overline{Y}_2, \ldots, \overline{Y}_k$. The sample average of these sample averages is

$$\overline{\overline{Y}} = \frac{1}{k}\sum_{j=1}^{k} \overline{Y}_i$$

which is an *across-replication statistic,* meaning a summary of the statistics across several replications.

Nearly all simulation languages compute within-replication statistics automatically. Many compute across-replication statistics. The standard statistics provided are sample averages, variances (or standard deviations), maximum and minimum observed values, and number of observations.

Some care must be exercised when interpreting the within-replication statistics because the outputs within a replication *are typically neither independent nor identically distributed,* and ''i.i.d. data'' is a common assumption behind many statistical procedures.

For example, in the license-renewal simulation, the within-replication sample variance of the processing delays, $Y_1, Y_2, \ldots, Y_N$, would be calculated by many simulation languages as

$$S^2 = \frac{\Sigma_{i=1}^{N}(Y_i - \overline{Y})^2}{N - 1}$$

Familiarity with classical statistics might lead one to use S^2 as an estimator of $\text{Var}[Y_i]$, the (assumed) common variance of the processing delays, and S^2/N as an estimator of $\text{Var}[\overline{Y}]$, the variance of the average delay.

However, the observed delays within a replication are not identically distributed, since applications received later in the month tend to be delayed longer than applications received earlier in the month. Thus, there is no common variance of the delays within a replication.

Also, the observed delays within a replication are not independent, since applications that experience long (short) delays tend to be followed by applications that experience long (short) delays. The validity of S^2/N as an estimator of $\text{Var}[\overline{Y}]$ rests on the assumption of i.i.d. data, and the estimator can be significantly biased when this assumption is violated. Unfortunately, the estimator is often biased low, which means that $\overline{Y}$ appears to be more precise than it actually is and confidence intervals based on S^2/N are inappropriately narrow.

Finally, the number of applications received during the month, N, is a random variable, so S^2 is a ratio of the random variables $\Sigma_{i=1}^{N}(Y_i - \overline{Y})^2$ and N. The properties of ratio estimators are not straightforward or easy to summarize, and they are typically different from the corresponding estimator when the denominator is not a random variable.

In summary, the within-replication sample variance S^2 is not a useful measure in this example, but there is no way for the simulation language to know that and notify the user.

On the other hand, most simulation languages ensure that statistics across replications are i.i.d. because by default they initialize each replication in the same way, implying identically distributed outputs, but they use different random numbers (by continuing in the same pseudorandom number sequence), implying independent outputs. In the license-renewal simulation the replication averages $\overline{Y}_1, \overline{Y}_2, \ldots, \overline{Y}_k$ are i.i.d., so applying classical statistical analysis is appropriate.

Design and Analysis Principle 2. Outputs within a replication may be neither independent nor identically distributed, and the number of observations may be random. Statistical analysis should therefore typically be based on multiple replications, with each replication supplying one observation of the performance measure of interest.

Rather than using automatically generated statistics, the author favors saving the raw data from a simulation experiment and performing the statistical analysis a posteriori using a general-purpose analysis package that may be separate from, or integrated into, the simulation language. There are three reasons for this preference. First, statistical-analysis packages can perform more procedures than most simulation languages; regression analysis, for instance. Second, using a package on the raw data facilitates exploratory analysis that may suggest what additional analysis is appropriate. For example, the normality of the output data can be empirically checked to see if an inference based on the normal distribution is reasonable. Finally, the raw data may be reanalyzed if new questions arise for which the necessary statistics were not computed during the simulation runs. For example, in a service system, the mean customer delay may be estimated from the individual customer delays. If it is later determined that the probability of a customer being delayed beyond some threshold value is of interest, then this can be estimated without rerunning the simulation if the raw data are available.

The primary arguments against saving all of the data are the storage requirement, which can be excessive, the decrease in simulation execution speed due to electronically writing all of the output data, and the difficulty of transforming the simulation data into a form suitable for an analysis

package. Language designers have addressed these problems by developing integrated database and analysis software and exporting data in standard formats that can be imported by spreadsheet or statistical-analysis software.

6. POINT ESTIMATORS

The decisions resulting from simulation studies are often based on point estimates of system performance parameters. The term *point estimate* means a single number that serves as a best candidate for the unknown performance parameter. Although the point estimates are important, they should always be accompanied by a measure of the error; error estimation is covered in Section 8.

This section describes estimating θ, a parameter of a probability distribution, F. Although F is usually not known, it is assumed that simulation outputs Y can be generated from F. The three cases considered are:

1. When θ is the *mean* of Y; i.e., $\theta = \mathrm{E}[Y] = \int_{-\infty}^{\infty} y dF(y)$
2. When θ is a *probability* associated with Y; e.g., $\theta = \Pr\{a < Y \leq b\} = F(b) - F(a)$ for given values $a < b$
3. When θ is a *quantile* of Y; i.e., $q = \Pr\{Y \leq \theta\} = \int_{-\infty}^{\theta} dF(y)$ for a given probability q

These cases arise in static and terminating simulation experiments. In steady-state simulation, outputs from F cannot be observed directly (because the distribution of interest is a limiting distribution), which introduces bias into the estimators. Estimation of parameters of a steady-state distribution is treated in Section 7.

6.1. Mean Estimation

Perhaps the most commonly reported performance parameter is the mean (expected value) of a system performance measure. The term *expected value,* which defines a parameter, will be used rather than the synonym *mean,* because *mean* may be confused with the *sample mean,* which is an estimator of a parameter.

In the license-renewal simulation, the parameter

$$\theta(\mu) = \mathrm{E}_{\mu}\left[\frac{1}{N}\sum_{i=1}^{N} Y_i\right] \tag{1}$$

is an expected value: the expected average delay for all applications received during one month when the processing rate is μ.

To estimate $\theta(\mu)$ the simulation is designed to generate k i.i.d. replications of one month of renewal processing. Each replication yields a within-replication average

$$\overline{Y} = \frac{1}{N}\sum_{i=1}^{N} Y_i$$

Let $\overline{Y}_j$ be the within-replication average from the jth replication. Since $\theta = \mathrm{E}[\overline{Y}_j]$ for each j by definition, an unbiased estimator of θ is the across-replication average

$$\overline{\overline{Y}} = \frac{1}{k}\sum_{j=1}^{k} \overline{Y}_j$$

Within- and across-replication averages for $k = 5$ replications when $\mu = 20$ applications per day are given in the second column of Table 2.

The standard error of $\overline{\overline{Y}}$ as an estimator of $\theta(\mu)$ is $\sigma/\sqrt{k}$, where σ^2 is the common variance of the replication averages $\overline{Y}_j$. Estimating measures of error is covered in Section 8. The important concept is that the error decreases as the number of replications increases.

Design and Analysis Principle 3. The sample average of i.i.d. outputs is an unbiased estimator of their common expectation. The standard error of the estimate decreases at rate $1/\sqrt{k}$, where k is the number of replications.

Given i.i.d. outputs with a common expectation, mean estimation is straightforward, as illustrated above. However, as discussed in Section 5 some care must be exercised when defining the expected value of interest in terminating simulations because performance parameters may be time dependent.

TABLE 2 Simulation Results for Five Replications of the License-Renewal Simulation with $\mu = 20$ (units are days)

Replication j	$\overline{Y}_j$	Y_{Nj}
1	0.34	1.25
2	0.72	2.20
3	0.32	1.04
4	0.46	1.68
5	0.42	1.08
$\overline{\overline{Y}}$	0.45	1.45

For example, the expected delay of an individual renewal application is not well defined because applications have different expected delays, depending on their order of arrival during the month. The parameter $\theta(\mu)$ in (1) circumvents this problem by averaging all the delays during a month and defining $\theta(\mu)$ to be the expectation of this average. However, $\theta(\mu)$ masks the time-dependent behavior of the delays, which may be important.

A time-dependent parameter that is well defined is $\theta_N(\mu) = \mathrm{E}[Y_N]$, the expected delay of the last application received during the month. To estimate $\theta_N(\mu)$, which is also an expected value, the delay for the last application, Y_N, is the within-replication statistic, and the across-replication average of these delays is an estimator of $\nu_N(\mu)$. The third column of Table 2 gives the delay for the last application received in each of the five replications and the across-replication average of these delays.

6.2. Probability Estimation

Probabilities can be represented as expected values, so the discussion in Subsection 6.1 is applicable to estimating probabilities as well. However, probability estimation is required so frequently that it is worth treating as a separate topic.

In the acceptance-sampling simulation, the parameter of interest is $\theta(p) = \Pr\{Y \leq c|p\}$, where Y is the number of defective items in a sample of n items and the quality of the items has a joint Pólya distribution with marginal probability p of a defective. As a function of p, $\theta(p)$ is the OC curve for sampling plan (n, c).

The key to estimating probabilities is the following principle:

Design and Analysis Principle 4. A probability θ can be represented as the expected value of an indicator (0, 1) random variable, where the value 1 corresponds to occurrence of the event of interest. The variance of an indicator random variable is $\sigma^2 = \theta(1 - \theta)$.

For the acceptance-sampling simulation, define an indicator random variable I as follows:

$$I = \begin{cases} 1, & \text{if } Y \leq c \\ 0, & \text{otherwise} \end{cases}$$

Then $\theta(p) = \mathrm{E}[I]$, and all of the principles of mean estimation can be applied to the indicator random variable, I.

Figure 1 shows an estimate of the OC curve for sampling plan $(n, c) = (10, 1)$ that was obtained by estimating $\theta(p)$ at points $p = 0.01, 0.05, 0.1, 0.15$ and $0.2, 0.3, \ldots, 0.9$. Each point estimate is the average of 10,000 replications of I. Since $\theta(p) = \mathrm{E}[I]$, the point estimators are unbiased, and the standard error of the estimators is $\sqrt{\theta(p)[1 - \theta(p)]/10000}$.

6.3. Quantile Estimation

Quantiles are points on the distribution of a random variable. The most familiar quantile is the *median,* which is the 0.5 quantile. The 0.25 and 0.75 quantiles are also called the *quartiles*. Extreme quantiles, such as the 0.9, 0.95, and 0.99 quantiles, are one way to characterize the tail of a distribution.

Subsection 6.1 described estimating the expected value of $\overline{Y}$, the average delay of renewal applications received during a month. The 0.9 quantile of $\overline{Y}$, denoted $\theta_{0.9}$, is the value such that $\Pr\{\overline{Y} \leq \theta_{0.9}\} = 0.9$. In other words, the average delay in 9 out of 10 months is less than or equal to $\mu_{0.9}$.

The procedure below, called *algorithm quantile estimation,* estimates the q quantile of a random variable given k i.i.d. replications:

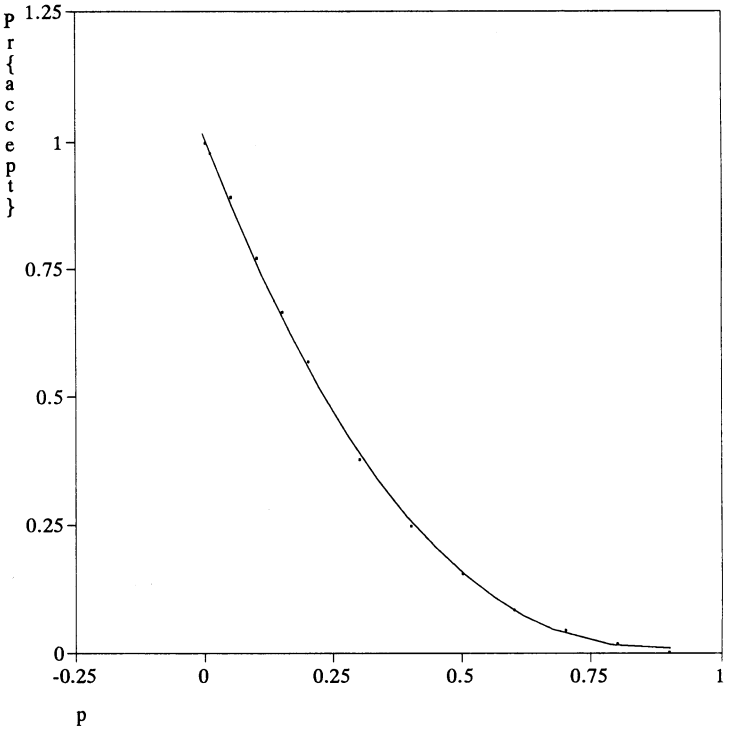


Figure 1 Plot of Estimated OC Curve for Sampling Plans (10, 1) Derived from $k = 10{,}000$ Replications.

1. *Initialization:* Number of replications, k; quantile probability, $0 < q < 1$.
2. *Data:* $y[j]$, output from the jth replication.
3. *Compute order statistics:* sort $y[j]$ from smallest to largest so that $y[1] \leq y[2] \leq \ldots \leq y[k]$ (comment: these sorted values are called the *order statistics* of the sample).
4. $i \leftarrow \lfloor (k + 1)*q \rfloor$.
   if $i < 1$ output $y[1]$
   if $i \geq k$ output $y[k]$
   otherwise $f \leftarrow (k + 1)*q - i$; output $(1 - f)*y[i] + f*y[i + 1]$

Quantile estimators are biased in general, but the bias (as well as the variance) of the estimator decreases as the number of replications increases. The procedure *algorithm quantile estimation* interpolates between order statistics to reduce bias.

For the license-renewal simulation the data are $\overline{Y}_1, \overline{Y}_2, \ldots, \overline{Y}_k$, the average delays from k replications. For $k = 200$ replications and $\mu = 20$ applications per day, an estimate of $\theta_{0.9}$ is $(0.1)*y[180] + (0.9)*y[181] = 0.88$ days.

7. INITIAL-CONDITION BIAS

Performing a steady-state simulation experiment implies that the analyst is interested in long-run performance of the model that is independent of the initial conditions of the simulation replication or run. In the inventory simulation the parameter of interest is $\theta^{(\ell)}$, the long-run expected cost per period of inventory policy ℓ, which does not depend on the inventory position in period 1.

The length of a simulation run is necessarily finite, so residual effects of the initial conditions will often be present in the outputs. These residual effects manifest themselves as *bias* in the statistics, meaning that the expectation of the statistic is not equal to the value of the performance parameter of interest. In some simulations, the effect of the initial conditions can be so overwhelming that the

results are meaningless unless appropriate remedial measures are taken. This section covers mean estimation for steady-state simulation. For more general estimation problems, see Law and Kelton (2000), Schruben (1981), and the comments at the end of this section.

Consider (s, S) inventory policy 1—which is (20, 40)—and let $Y_1^{(1)}, Y_2^{(1)}, \ldots, Y_{200}^{(1)}$ be the observed inventory costs in the first 200 periods of a replication of this policy. An estimator of the long-run expected cost per period, $\theta^{(1)}$, is the sample average

$$\overline{Y}^{(1)} = \frac{1}{200} \sum_{t=1}^{200} Y_t^{(1)}$$

The effect of the initial condition $I_1 = S$ is to cause the expected value of $\overline{Y}^{(1)}$ to be less than $\theta^{(1)}$ because little or no ordering and setup cost is incurred until the inventory has been depleted somewhat. This means that, on average, the estimate will indicate that policy 1 is less costly than it actually is.

The appropriate remedial measure depends on whether it is feasible to perform some preliminary or pilot analysis of the model prior to making the final runs. When a large number of similar systems are to be investigated (e.g., different inventory policies for the same inventory system), then preliminary analysis of one system may be worthwhile since the results can be extrapolated to the other systems. However, if only one system is to be investigated, or if each run is very expensive, then it may be desirable to use the same experiment to remedy the initial-condition bias and estimate the parameters of interest. This section describes both approaches and two remedial measures, data deletion and intelligent initialization.

7.1. Remedial Measures

For most estimation and inference problems, increasing the number of replications is beneficial. The initial-condition bias problem is an exception.

Design and Analysis Principle 5. Increasing the number of independent replications does not decrease the bias of the sample average. If the number of replications is increased at the expense of decreasing the length of each replication, then increasing the number of replications may even increase the bias.

This principle suggests that in the presence of initial-condition bias and a tight budget, the number of replications should be small and the length of each replication should be long. When bias is severe, making only one very long replication may be advantageous. Experiment designs that call for only one replication require more sophisticated strategies for computing a measure of error, as discussed in Section 8.

Even with only a few long replications, remedial measures may be needed. Two of the many remedial measures, which can be used together, are described below.

7.1.1. Data Deletion

If there is a standard remedial measure, it is to delete or discard some of the outputs from the beginning of each replication where the effects of the initial conditions are most pronounced. Returning to the inventory simulation, if $Y_1^{(1)}, Y_2^{(1)}, \ldots, Y_{200}^{(1)}$ are the observed inventory costs in the first 200 periods, then another estimator of $\theta^{(1)}$ is the truncated sample average

$$\overline{Y}^{(1)}(d) = \frac{1}{200 - d} \sum_{t=d+1}^{200} Y_t^{(1)}$$

where d, the number of outputs deleted, could be 0, 1, . . . , 199. Typically, the bias of the truncated sample average decreases as d increases, which is good, but its variance increases with d, which is bad. Methods for choosing d are discussed in a later subsection.

Many simulation languages make it convenient to delete all outputs up to a fixed point in simulated time or up to an event time rather than deleting a fixed number of outputs, d. This makes d a random variable but does not significantly change the properties of the estimators, provided the runs are not too short.

7.1.2. Intelligent Initialization

The initial conditions for a simulation experiment are the starting values assigned to the variables in the model and the events scheduled at the beginning of each run or replication. Since the parameters of interest in a steady-state simulation do not depend on the initial conditions, initial conditions are often chosen for programming convenience. In the inventory simulation the initial inventory position is $I_1 = S$, for example. Some care in setting the initial conditions can greatly reduce the bias they cause.

Any model that has a steady-state distribution has a ''correct'' initialization procedure so that there is no initial-condition bias. This result is of more theoretical interest than practical importance because knowing the correct procedure typically implies already knowing the values of the parameters of interest. However, setting initial conditions close to steady-state mean or mode conditions, while not correct, can be very effective in reducing bias. There are at least two ways to approximate steady-state conditions:

1. *Approximate models:* Steady-state distributions and parameters are known for many stochastic processes; e.g., queueing, inventory, Markov chains. These results can be used to approximate the simulation model. For example, a service system can be approximated by a Markovian queue to determine the expected number of customers in the system. This value can be used to set the initial number of customers in the system for the simulation, rather than using the (convenient) initial condition of an empty system. Chapter 81 of the *Handbook* is a good source of approximations. Even cruder approximations, such as replacing a random quantity by its expectation, can also be used.
2. *Pilot runs:* The debugging runs or preliminary pilot runs of the simulation can provide information for setting initial conditions more intelligently. For instance, printing out the average inventory position during debugging runs of the inventory simulation showed that taking $I_1 = 36$ is closer to steady-state conditions than $I_1 = 40$ for inventory policy 1.

The problem of setting optimal initial conditions has been studied, but there is no single recommendation. Significant improvement is possible even if the initial conditions are not optimal, especially when the convenient initial conditions are far from optimal. The point is to use all available knowledge about the model, including knowledge gained from experimentation, to refine the initial conditions.

7.2. Detection of Initial-Condition Bias

The methods described in this subsection can be used to detect the presence of initial-condition bias and to determine the amount of data to delete.

7.2.1. Mean Plot

Let $\theta_t^{(1)} = \mathrm{E}[Y_t^{(1)}]$ be the expected cost of inventory policy 1 *in period t.* The assumption behind steady-state simulation is that $\theta_t^{(1)} \to \theta^{(1)}$ as t increases. The mean plot estimates $\theta_t^{(1)}$ as a function of t so that the rate of convergence can be determined. This method is appropriate when some preliminary study of the bias is possible.

An estimate of $\theta_t^{(1)}$ can be obtained by making a number of replications of the experiment, averaging across replications, and plotting the resulting values. To be specific, suppose that 30 replications of 200 periods each were generated for inventory policy 1. Let $Y_{tj}^{(1)}$ be the cost in period t for replication j, $t = 1, 2, \ldots, 200$ and $j = 1, 2, \ldots, 30$. Then the across-replication averages

$$\overline{Y}_t^{(1)} = \frac{1}{30} \sum_{j=1}^{30} Y_{tj}^{(1)}$$

are estimates of $\theta_t^{(1)}$, since $\overline{Y}_t^{(1)}$ is the average cost in period t across 30 replications. A plot of these averages as a function of t is given in Figure 2. Since the data are still quite variable, a smoothing curve was fitted to make the mean function more apparent (a smoothing spline was used in Figure 2; Welch 1983 recommends a moving average). Another way to smooth the plot is to increase the number of replications. The procedure below, called *algorithm mean-plot,* calculates the across-replication averages needed for the plot:

1. *Initialization:* number of replications, k; length of each replication, m; array $a[t]$ of length m; array $s[t] \leftarrow 0$ for $t \leftarrow 1, 2, \ldots, m$
   (comment: k and m are problem dependent and require experimentation)
2. *Data:* $y[t, j]$, the tth observation from replication j
3. *Calculate averages:*
   (a) for $t \leftarrow 1$ to m:
   for $j \leftarrow 1$ to k: $s[t] \leftarrow s[t] + y[t, j]$, endfor
   endfor
   (b) for $t \leftarrow 1$ to m: $a[t] \leftarrow s[t]/k$, endfor
4. *Output:* plot $a[t]$ vs. t, smoothing the plot if necessary

The mean plot in Figure 2 shows that bias is clearly present—although it dissipates rapidly in this example—and the early cost values have a negative bias (biased low). A deletion point d can

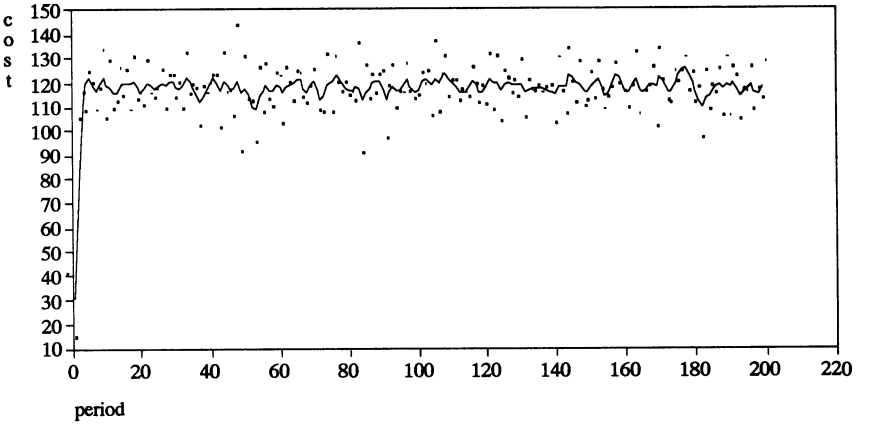


Figure 2 Mean Plot for Inventory Example.

be obtained by looking for a point where the curve seems to become nearly horizontal, perhaps at $d = 20$ periods in this example. The selection of d using the mean plot is subjective.

7.2.2. Cusum Plot

Schruben derived a plot that can be formed from a single, long replication and that is particularly sensitive to the presence of initial-condition bias (Barton and Schruben 1989). In terms of the inventory simulation, define $S_0 = 0$, and let

$$S_j = \sum_{t=1}^{j} (\bar{Y}^{(1)} - Y_t^{(1)})$$

for $j = 1, 2, \ldots, m$, where m is the number of periods simulated. The *cusum plot* is a graph of S_j vs. j.

Notice that $S_0 = S_m = 0$. If there is no initial-condition bias, then $E[S_j] = 0$ for all values of j in between, so the plot of S_j will tend to cross zero several times. However, when bias is present the values of S_j will all tend to be on one side of zero, depending on whether the bias is positive or negative.

Figure 3 shows such a plot for $m = 200$ periods of the inventory simulation before and after 20 periods of output were deleted (this plot was formed after averaging across several replications to make the characteristic behavior even more pronounced). In the first case the plot tends to be above zero, indicating negative bias, while after deletion no bias appears to be present.

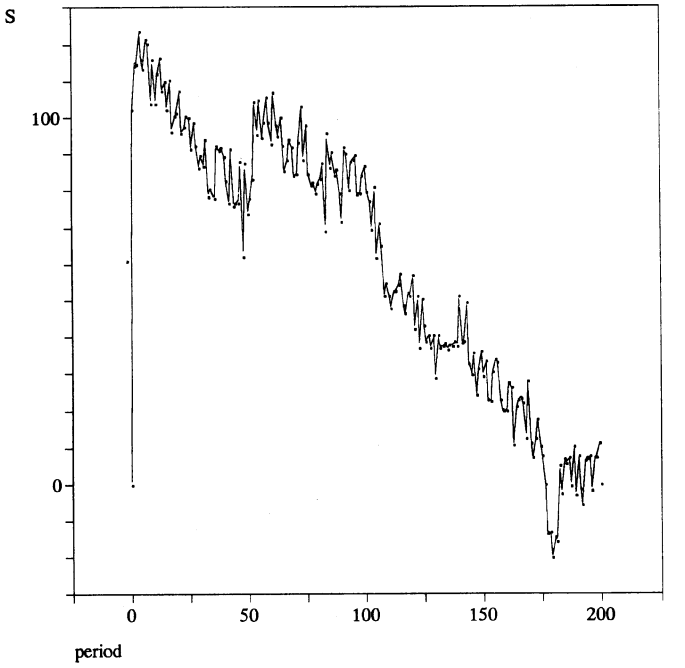
To smooth the plot, batching can be used. This is done in *algorithm cusum:*

1. *Initialization:* length of the replication, m; batch size b; number of batches $k \leftarrow \lfloor m/b \rfloor$; $a \leftarrow 0$; array $s[j] \leftarrow 0$ for $j \leftarrow 0, 1, \ldots, k$
(comment: m is problem dependent but should be large; Schruben recommends $b \leftarrow 5$)

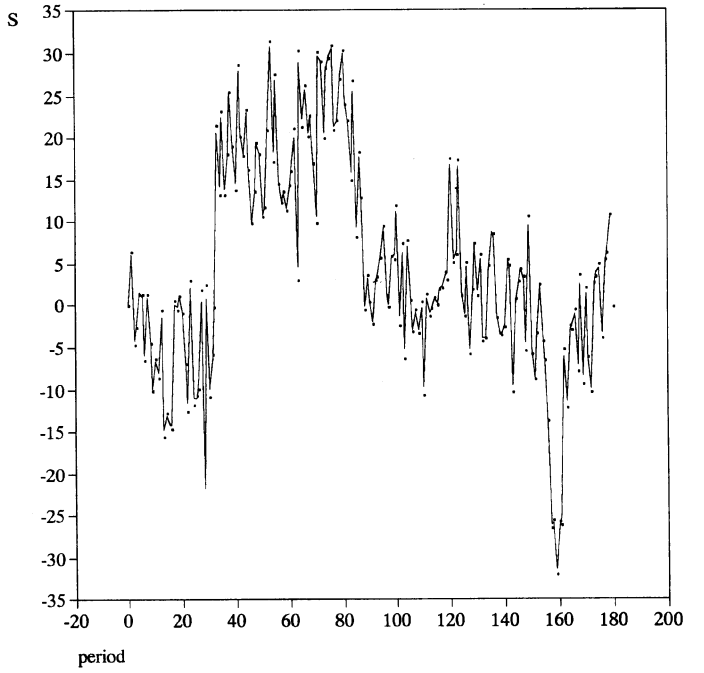
2. *Data:* $y[t]$, the tth observation from a single replication

3. *Batch the outputs:*
(a) for $j \leftarrow 1$ to k:
$y[j] \leftarrow y[(j-1)*b + 1]$
for $t \leftarrow 2$ to b: $y[j] \leftarrow y[j] + y[(j-1)*b + t]$, endfor
endfor
(b) for $j \leftarrow 1$ to k: $y[j] \leftarrow y[j]/b$, endfor
(comment: batch averages are now stored in $y[1], y[2], \ldots, y[k]$; if the original data must be saved then a separate array should be used)

4. *Compute overall average:*
for $j \leftarrow 1$ to k: $a \leftarrow a + y[j]$, endfor
$a \leftarrow a/k$



(*a*)



(*b*)

Figure 3 Cusum Plot for Inventory Model before and after Deleting Outputs.

5. *Generate cusum plot:* for $j \leftarrow 1$ to k: $s[j] \leftarrow s[j-1] + a - y[j]$, endfor

6. *Output:* plot $s[j]$ vs. j

The procedure *algorithm cusum* assumes that all of the outputs $y[t]$ are available before generating the plot. If an additional batch average, $y[k+1]$, is generated, then the plot values can be updated using the following steps:

$s[j] \leftarrow s[j] + j*(y[k+1] - a)/(k+1)$ for $j \leftarrow 1, 2, \ldots, k$
$s[k+1] \leftarrow 0$
$a \leftarrow (k * a + y[k+1])/(k+1)$

In the next subsection a test for initial-condition bias based on the characteristic behavior of the cusum plot is given.

7.2.3. Initial-Condition-Bias Test

Schruben (1982) used the characteristic behavior of the cusum plot to derive a statistical test for the presence of initial-condition bias. The test would typically be performed on an output process after remedial measures, such as data deletion, have been applied. The null hypothesis of the test is that there is no negative initial-condition bias (biased low) in the mean of the output process.

Loosely speaking, the test works as follows. The output of a single replication is divided in half. If the run is long enough, then any bias is most prevalent in the first half. The cusum values from each half are compared in terms of the location and magnitude of their maximum deviation from zero. If the behavior of the first half is significantly different from the second half, then the hypothesis of no initial-condition bias is rejected.

There are several versions of this test that have power against specific alternatives (Schruben et al. 1983; Goldsman et al. 1989). The version given in *algorithm bias test* is conservative in the sense that it tries to ensure that all of the asymptotic assumptions behind the test will be valid:

1. *Initialization:* length of the replication, m; batch size b; number of batches $k \leftarrow \lfloor m/b \rfloor$; arrays $a[j] \leftarrow 0$, $s[j] \leftarrow 0$, $smax[j] \leftarrow 0$ and $\ell[j] \leftarrow 0$ for $j = 1, 2$
(comment: m is problem dependent but should be large; Schruben recommends $b \leftarrow 5$)

2. *Data:* $y[t]$, the tth observation from a single replication
(comment: test assumes negative bias; if positive bias, modify step 6 as indicated; both tests can be performed if the direction of bias is uncertain)

3. *Batch the outputs:*
(a) for $j \leftarrow 1$ to k:
$y[j] \leftarrow y[(j-1)*b + 1]$
for $t \leftarrow 2$ to b: $y[j] \leftarrow y[j] + y[(j-1)*b + t]$, endfor
endfor
(b) for $j \leftarrow 1$ to k: $y[j] \leftarrow y[j]/b$, endfor
(comment: batch means are now stored in $y[1], y[2], \ldots, y[k]$; if the original data must be saved, then a separate array should be used)

4. $n \leftarrow \lfloor k/2 \rfloor$
(comment: n is half the batched process; the algorithm ignores the last batch if k is odd)

5. *Calculate sample mean of each half:*
for $i \leftarrow 1$ to n:
$a[1] \leftarrow a[1] + y[i]$
$a[2] \leftarrow a[2] + y[n+i]$
endfor
for $i \leftarrow 1$ to 2: $a[i] \leftarrow a[i]/n$, endfor

6. Locate maximum from each half:
(comment: if positive bias is suspected, replace all $>$'s with $<$'s in this step)
for $i \leftarrow 1$ to n:
$s[1] \leftarrow s[1] + a[1] - y[i]$
if $s[1] > smax[1]$
$\ell[1] \leftarrow i$ and $smax[1] \leftarrow s[1]$
endif
$s[2] \leftarrow s[2] + a[2] - y[n+i]$
if $s[2] > smax[2]$
$\ell[2] \leftarrow i$ and $smax[2] \leftarrow s[2]$
endif
endfor

7. Calculate test statistic:
(comment: if $\ell[1] = 0$ test for bias of the opposite sign; if $smax[2] = 0$ then m is too small)
$f \leftarrow \ell[2] * (n - \ell[2]) * smax[1] * smax[1] / ((\ell[1] * (n - \ell[1]) * smax[2] * smax[2])$
8. If $f > 9.28$, then reject the hypothesis of no initial-condition bias
(comment: 9.28 is the critical value for a 5% significance level; use 5.39 or 29.5 for 10% or 1% significance levels, respectively; to obtain other significance levels use the F distribution with (3,3) DOF).

Applying the test to the output process in Figure 3 before data deletion yields an f value of 88.66, which is significant even at the 1% level. However, after deleting the first 20 outputs from the process the f value is only 2.93, which is not significant.

7.3. A Comment on Probabilities and Quantiles

Long-run probabilities and quantiles of an output process can be estimated from a single replication of a steady-state simulation using the estimators given in Section 6. However, the convergence of probabilities and quantiles to their limiting values is often much slower than convergence of the mean. Thus, approximate convergence of the mean cannot be used as a substitute for establishing the convergence of probabilities and quantiles if they are the parameters of primary interest.

8. MEASURES OF ERROR

Design and Analysis Principle 6. A point estimator should be accompanied by a measure of its potential error.

A simple example illustrates why this principle is so important: Measures of error can be interpreted as the number of meaningful digits in a point estimate. If the estimated expected cost of an inventory policy is 1.1 million, it probably matters whether or not the second digit is meaningful. Without a measure of error, however, the analyst cannot even be sure if the *first* digit has meaning! A true expected cost of 2.7 million may mean the company is out of business.

This section describes measures of error for estimates of a performance parameter, θ, that can be represented as a mean (expected value), probability or quantile; point estimators for such parameters are covered in Section 6.

Two measures of error are the *standard error* and *confidence interval*. Whether or not it is difficult to derive a measure of error is related to whether or not the experiment design calls for i.i.d. replications: static and terminating experiments always specify replications, but steady-state simulations may not, depending on the severity of the initial-condition bias (see Section 7).

8.1. Standard Error

The *standard error* of a point estimator is an average error, where the average is with respect to repeated experiments of the same type. More precisely, it is the standard deviation of the point estimator. This subsection begins with a basic result for estimating the standard error of a mean or probability estimate and then illustrates it using the examples.

Let $Y_1, Y_2, \ldots, Y_k$ be i.i.d. output random variables with mean (expected value) θ, and let $\overline{Y}$ be their average as defined in Section 6. Then the standard error of $\overline{Y}$ as an estimator of θ is $\sigma/\sqrt{k}$, where σ^2 is the common $\mathrm{Var}[Y_i]$. An estimator of $\sigma/\sqrt{k}$ is

$$\frac{S}{\sqrt{k}} = \sqrt{\frac{\sum_{i=1}^{k} (Y_i - \overline{Y})^2}{k(k-1)}} = \sqrt{\frac{\sum_{i=1}^{k} Y_i^2 - (\sum_{i=1}^{k} Y_i)^2/k}{k(k-1)}} \tag{2}$$

where $S^2 = (k-1)^{-1} \sum_{i=1}^{k} (Y_i - \overline{Y})^2$ is the sample variance, an estimator of σ^2.

The estimator $S/\sqrt{k}$ is also valid when θ is a probability and the outputs are indicator variables I_i (as defined in Section 6), in which case (2) reduces to

$$\frac{S}{\sqrt{k}} = \sqrt{\frac{\overline{I}(1-\overline{I})}{k-1}}$$

For example, in the acceptance-sampling simulation I_i could be an indicator of whether or not the ith sample of size 10 was accepted (had 1 or fewer defectives), so that $\overline{I}$ estimates $\theta(p)$, the probability of accepting the sample. Table 3 gives the estimated standard errors for estimates of $\theta(p)$, where each estimate is based on $k = 10{,}000$ i.i.d. replications as described in Section 6.

TABLE 3 Estimated Points on OC Curve and Associated Standard Error of the Estimate

p	$\bar{I}$	$S/\sqrt{10000}$
0.00	1.00	0.000
0.01	0.98	0.001
0.05	0.89	0.003
0.10	0.77	0.004
0.15	0.67	0.005
0.20	0.57	0.005
0.30	0.38	0.005
0.40	0.25	0.004
0.50	0.16	0.004
0.60	0.09	0.003
0.70	0.05	0.002
0.80	0.02	0.001
0.90	0.01	0.001

Design and Analysis Principle 7. The first nonzero digit in the standard error indicates an uncertain digit in the corresponding decimal place of the point estimator.

This principle is conservative; it does not imply that the first uncertain digit in the point estimator has no meaning, but only that it may not be correct. Table 3 displays two decimal places for each $\bar{I}$ because the first nonzero digit of the standard error is in the third decimal place; however, the third decimal place of $\bar{I}$ was used for rounding.

Formula (2) is applicable when the point estimator is an average across k i.i.d. replications. As discussed in Section 5, it is seldom applicable to outputs within a single replication because they may be neither independent nor identically distributed. This causes no problem in static or terminating simulations because the natural experiment design is to generate independent replications.

Replications can also be generated in steady-state simulations, although remedial measures (such as deletion, see Section 7) must be applied to each replication. When initial-condition bias is not severe, replications are recommended and formula (2) can be applied to the across-replication statistics.

When only a single replication is generated from a steady-state simulation, an estimator of the standard error can be derived using the method of *batch means*. To illustrate the discussion, let $Y_1^{(1)}, Y_2^{(1)}, \ldots, Y_m^{(1)}$ be the output from a single replication of the inventory simulation for policy 1 *after* the first 20 periods have been deleted; thus, $Y_t^{(1)}$ is the observed cost in period $t + 20$. The sample average, $\bar{Y}^{(1)}$, is used to estimate $\theta^{(1)}$, the long-run expected cost per period for inventory policy 1. An estimate of the standard error of $\bar{Y}^{(1)}$ is required. For convenience, drop the superscript (1) and let the output and statistic be simply Y_t and $\bar{Y}$, respectively.

Let $k \leq m$ be the *number of batches,* $b = \lfloor m/k \rfloor$ the *batch size,* and define the jth batch mean (average) to be

$$\bar{Y}_j(k) = \frac{1}{b} \sum_{t=1}^{b} Y_{(j-1)b+t}$$

for $j = 1, 2, \ldots, k$. The principle behind batch means is that $\bar{Y}_1(k), \bar{Y}_2(k), \ldots, \bar{Y}_k(k)$ are more nearly independent than the original outputs, $Y_1, Y_2, \ldots, Y_m$, for most simulation output processes. If the remedial measures for initial-condition bias have been effective, then the batch means are also nearly identically distributed. Thus, the batch means are (approximately) i.i.d. statistics, even though they are within-replication statistics, and formula (2) applies. The batch means are written as functions of k because selecting the number of batches is a decision that the analyst must make.

The procedure below, called *algorithm batch_means,* computes an estimate of the standard error of a within-replication average $\bar{Y}$. It can be used to compute the standard error for the across-replication average of k replications (formula (2)) by omitting step 3(a) and setting $m = k$:

1. *Initialization:* length of the replication, m; number of batches k; batch size $b \leftarrow \lfloor m/k \rfloor$; *sum* $\leftarrow 0$; and *sumsq* $\leftarrow 0$.
   (comment: m is problem dependent but should be large; choosing k is discussed below)
2. *Data:* $y[t]$, the tth observation from a single replication *after* applying remedial measures.

3. *Batch the outputs:*
(a) for $j \leftarrow 1$ to k:
$y[j] \leftarrow y[(j-1)*b+1]$
for $t \leftarrow 2$ to b: $y[j] \leftarrow y[j] + y[(j-1)*b+t]$, endfor
endfor
(b) *Compute the standard error:*
for $j \leftarrow 1$ to k:
$y[j] \leftarrow y[j]/b$
$sum \leftarrow sum + y[j]$
$sumsq \leftarrow sumsq + y[j]*y[j]$
endfor
(comment: batch means are now stored in $y[1], y[2], \ldots, y[k]$; if the original data must be saved, then a separate array should be used)

4. *Output:* sqrt{$(sumsq - sum*sum/k)/(k(k-1))$}.

To illustrate the effect of batching, 2020 periods of the inventory simulation were generated and the first 20 periods of data were deleted, leaving $m = 2000$ outputs. The sample mean of the outputs was $\overline{Y} = 118.48$ dollars per period. The estimated sample correlation between successive periods, $\text{Corr}[Y_t, Y_{t+1}]$, was -0.51, clearly showing dependence. After batching the cost data into $k = 20$ batches of size $b = 100$, the estimated correlation between successive batch means, $\text{Corr}[\overline{Y}_j(20), \overline{Y}_{j+1}(20)]$, was -0.09, which appears less correlated. The standard error of $\overline{Y}$ calculated by *algorithm batch means* was 0.29, an average error of 29 cents.

The most difficult design decision when using the method of batch means is choosing k, the number of batches. Several studies have concluded that no matter how long the replication is, it should be divided into a relatively small number of batches, say $10 \leq k \leq 30$. Diagnostic tests, such as computing the correlation between successive observations, can be employed to verify that batching has been effective. Formal batching algorithms are also available (Bratley et al. 1987; Fishman 1978; Fishman and Yarberry 1997; Law and Kelton 2000).

8.2. Confidence Intervals

Confidence intervals are constructed in hopes that they contain the unknown performance parameter of interest, θ. The ''confidence'' associated with a confidence interval is, strictly speaking, a statement about the *procedure* used to construct the interval, not the interval itself. The confidence interval either contains θ or it does not. The width of the confidence interval is a measure of error.

There are many confidence-interval procedures based on a variety of different assumptions, too many to describe here (see Chapter 86 of the Handbook or any introductory statistics text). This subsection contains some general comments regarding the use of confidence-interval procedures for estimating means, and a specific confidence-interval procedure for quantile estimation.

8.2.1. Confidence Intervals for Means

The validity of a confidence-interval procedure frequently depends on a number of conditions. One such condition is that the output data are i.i.d. As discussed in Subsection 8.1, i.i.d. outputs can be obtained by generating replications, or, in steady-state simulation, by using the method of batch means.

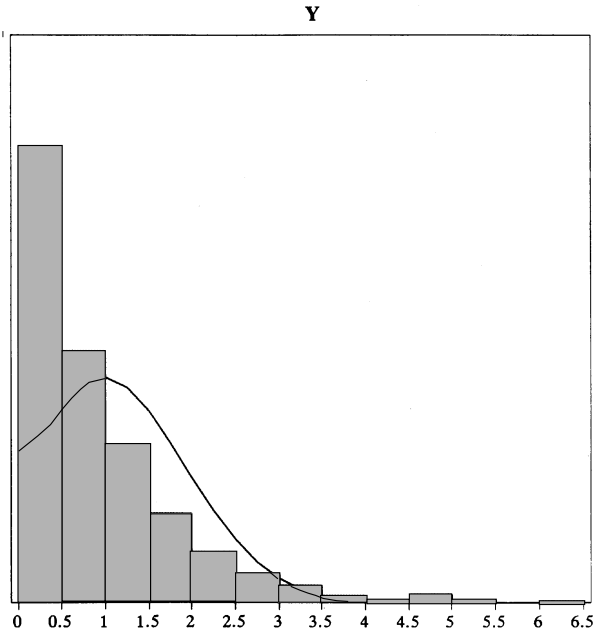
A second common condition is that the simulation outputs have a normal distribution. The assumption of normality can be severely violated in simulations. Diagnostic checks using histograms, quantile plots, or hypothesis tests are recommended to verify the normality of the output data.

The normality of the output data can be improved by using the method of batch means, even if the data are already i.i.d. This is because averages tend to be normally distributed, as shown in the central limit theorem. For example, Figure 4 shows two histograms for the same 500 i.i.d. simulation outputs. The first histogram shows raw data, while the second displays the same data after batching them into $k = 100$ batch means (batch size $b = 5$). In both cases a normal curve is superimposed over the histogram. Clearly the sample distribution of the batch means is more nearly normal.

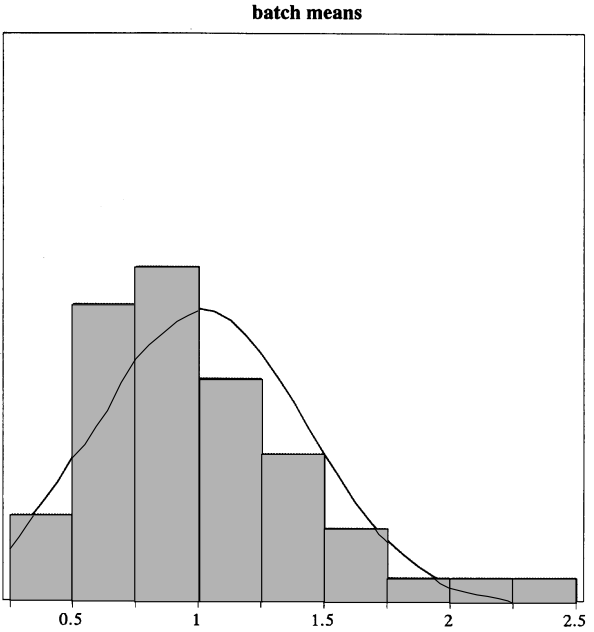
Although batching reduces the number of degrees of freedom for the confidence-interval procedure—since there are fewer batch means than total outputs—the penalty is slight provided that the number of batch means is not too small. Schmeiser (1982) showed that, for the standard t confidence-interval procedure, batching to $10 \leq k \leq 30$ batch means does not degrade the performance of the procedure substantially *even if the original data are precisely i.i.d. normal.* On the other hand, when the data are not normal, batching can improve the performance of the procedure in terms of delivering the requested level of confidence.

8.2.2. A Confidence Interval for Quantiles

A point estimator for the q quantile, θ_q, of a random variable Y was given in Section 6. The point estimator and the confidence-interval procedure given here assume that k i.i.d. replications of Y are



(*a*)



(*b*)

Figure 4 Simulation Outputs before and after Batching.

TABLE 4 Standard Normal Quantiles

$1 - \alpha$	$z_{1-\alpha/2}$
0.90	1.6449
0.95	1.9600
0.99	2.5758

available, denoted generically by $Y_1, Y_2, \ldots, Y_k$. Let the sorted values of Y, called the order statistics, be denoted $Y_{(1)} \le Y_{(2)} \le \ldots \le Y_{(k)}$.

Although a standard error estimate is difficult to derive, deriving a confidence interval for θ_q is relatively easy. In addition, this confidence-interval procedure is nonparametric, meaning that the distribution of Y is not a factor.

A $(1 - \alpha)100\%$ confidence interval for θ_q is $(Y_{(\ell)}, Y_{(u)})$, where $1 \le \ell < u \le k$ are integers such that the binomial probability

$$\sum_{j=\ell}^{u-1} \binom{k}{j} q^j (1-q)^{k-j} \approx 1 - \alpha$$

The confidence interval is nonparametric since the constants ℓ and u depend only on k and q. Determining ℓ and u is difficult when the number of replications, k, is large (and k should be large for quantile estimation), but large k allows a normal approximation to the binomial distribution. The approximation leads to setting

$$\ell = \lfloor kq - z_{1-\alpha/2}\sqrt{kq(1-q)} + 1/2 \rfloor \text{ and} \tag{3}$$

$$u = \lfloor kq + z_{1-\alpha/2}\sqrt{kq(1-q)} + 1/2 \rfloor + 1 \tag{4}$$

where $z_{1-\alpha/2}$ is the $1 - \alpha/2$ quantile of the standard normal distribution. The standard normal quantiles for 90%, 95%, and 99% confidence intervals are given in Table 4.

Section 6 illustrated estimating the 0.9 quantile, $\theta_{0.9}$, of Y, the average delay for renewal applications received during a month in the license-renewal simulation. Based on 200 replications, the point estimate was 0.88 days. Substituting $k = 200$ and $q = 0.9$ into (3) and (4) gives $\ell = 172$ and $u = 189$ for a 95% confidence interval. The corresponding interval is $(Y_{(172)}, Y_{(189)}) = (0.82, 0.98)$.

8.3. A Note on Experiment Planning

The discussion above assumes that the analyst is interested in computing a measure of error *after* completing the simulation replications or runs. Experiment planning includes determining the number of replications required to achieve a prespecified level of error.

Consider the standard error. If the process variance σ^2 is known, then it is easy to determine the number of replications (or batch means) required to achieve a standard error less than or equal to, say, δ, by solving

$$\frac{\sigma}{\sqrt{k}} \le \delta \tag{5}$$

for k. Typically, σ^2 is not known, but it may be estimated during debugging or preliminary pilot runs and the estimate can be substituted into (5).

Law and Kelton (2000) contains a discussion of methods for determining k sequentially (while the simulation is in progress) to guarantee a prespecified level of error.

9. OPTIMIZATION AND SENSITIVITY

Industrial engineers frequently use simulation experiments to compare the performance of alternative systems and, ideally, to optimize system performance. When a system is modeled as a stochastic process, the objective is often to optimize expected performance, where "expected" means the mathematical expectation of a random variable. This section describes methods for optimization via simulation, using the problem of selecting the inventory policy that minimizes long-run expected cost per period as an illustration.

Even if optimization is not desired or feasible, the analyst may be interested in the sensitivity of system performance to certain controllable factors. This section also discusses sensitivity analysis, using the sensitivity of the expected average delay $\theta(\mu)$ to the processing rate μ in the license-renewal simulation as an illustration.

The scope of this section is limited because methods for optimization and sensitivity analysis are still evolving. See Andradóttir (1998).

9.1. Multiple Comparisons

A common optimization problem is to choose the system with maximum or minimum expected performance from among a small number of systems, say 2 to 10. Assuming that estimates of expected performance are available from simulation experiments, the question of which system to select is easily answered: all else being equal, select the system with the sample best performance. A more relevant question is whether the observed ranking of the systems is due to estimation error or actual differences in performance. Thus, optimization is an extension of the problem of estimating and controlling error, as discussed in Section 8.

The case of comparing two systems is covered in standard statistics texts. For three or more systems, two classes of statistical procedures are widely used: ranking-and-selection procedures and multiple-comparison procedures.

Ranking-and-selection procedures treat the optimization problem as a decision problem, typically either deciding which system is the best (indifference-zone ranking) or on a subset of systems that contains the best system (subset selection). The decisions are guaranteed to be correct with a prespecified probability. Achieving this goal often requires two-stage sampling, which means restarting simulation experiments after initial runs of all systems. A summary of the many ranking and selection procedures is given by Bechhofer et al. (1995), Gupta and Panchapakesan (1979), and Law and Kelton (2000). Extensions of these procedures have been made specifically to stochastic simulation (Koenig and Law 1985; Clark and Yang 1986; Goldsman 1985; Goldsman and Nelson 1998; Iglehart 1977; Sullivan and Wilson 1989).

Multiple-comparison procedures, on the other hand, treat the optimization problem as an inference problem on the performance parameters of interest. An important property of multiple-comparison procedures is that inference about the relative performance of all systems is provided. In addition, multiple-comparison procedures can be implemented in a single-stage of sampling. This subsection describes a multiple-comparison procedure that is useful for optimization.

In the inventory simulation smaller $\theta^{(\ell)}$ is preferred, where $\theta^{(\ell)}$ denotes the long-run expected cost per period for policy ℓ, $\ell = 1, 2, \ldots, 5$. For policy ℓ, the parameter $\theta^{(\ell)} - \min_{i \neq \ell} \theta^{(i)}$ is policy ℓ cost minus the minimum of the other policies' costs. The parameters $\theta^{(\ell)} - \min_{i \neq \ell} \theta^{(i)}$, for $\ell = 1, \ldots, 5$, are the appropriate parameters to estimate for optimization. If $\theta^{(\ell)} - \min_{i \neq \ell} \theta^{(i)} < 0$, then policy ℓ is the best because all other policies have larger expected cost. On the other hand, if $\theta^{(\ell)} - \min_{i \neq \ell} \theta^{(i)} > 0$, then policy ℓ is not the best, since there is another policy with smaller expected cost. Even when $\theta^{(\ell)} - \min_{i \neq \ell} \theta^{(i)} > 0$, if $\theta^{(\ell)} - \min_{i \neq \ell} \theta^{(i)} < \Delta$, where Δ is a positive number, then policy ℓ is within Δ of the best. Simultaneous statistical inference on $\theta^{(\ell)} - \min_{i \neq \ell} \theta^{(i)}$, for $\ell = 1, \ldots, 5$, is termed *multiple comparisons with the best* (MCB). In problems where larger expected performance is preferred, the parameter of interest is $\theta^{(\ell)} - \max_{i \neq \ell} \theta^{(i)}$, so that $\theta^{(\ell)} - \max_{i \neq \ell} \theta^{(i)} > 0$ indicates the best system.

For the general case of r systems, Hsu (1984) derived simultaneous $(1 - \alpha)100\%$ confidence intervals for $\theta^{(\ell)} - \max_{i \neq \ell} \theta^{(i)}$ (equivalently $\theta^{(\ell)} - \min_{i \neq \ell} \theta^{(i)}$) for $\ell = 1, 2, \ldots, r$. The form of the intervals is

$$\left[(\overline{Y}^{(\ell)} - \max_{i \neq \ell} \overline{Y}^{(i)} - H)^-, (\overline{Y}^{(\ell)} - \max_{i \neq \ell} \overline{Y}^{(i)} + H)^+ \right]$$

where $\overline{Y}^{(\ell)}$ is the sample average of the outputs from system ℓ, $x^- = \min\{0, x\}$, $x^+ = \max\{0, x\}$ and H is a quantity that depends on r, k and the sample standard error.

The assumptions behind these confidence intervals, and their interpretation in simulation experiments, are listed below. To aid the explanation let $Y_j^{(\ell)}$ be generic for the jth output from system ℓ.

- For each system ℓ, the output data consist of k i.i.d. outputs $Y_1^{(\ell)}, Y_2^{(\ell)}, \ldots, Y_k^{(\ell)}$ with common expected value $\theta^{(\ell)}$. This will be true if the outputs are from across replications, or approximately true if they are batch means from within a single replication (see Section 8).
- The outputs from different systems $\ell = 1, 2, \ldots, r$ are independent of each other. This will be true if different random number streams or seeds are specified for the simulation of each system (see Section 4).
- All the outputs $Y_j^{(\ell)}$ $j = 1, 2, \ldots, k$ and $\ell = 1, 2, \ldots, r$ are normally distributed. This may be approximately true if $Y_j^{(\ell)}$ is an average of many outputs, but it should be verified through graphical or statistical analysis.
- $\mathrm{Var}[Y_j^{(\ell)}]$ is the same for all j and ℓ. This may be true if the different systems are similar, but it should be verified through graphical or statistical analysis. If the variances across systems are widely different then variance-stabilizing transformations can be applied.

The procedure below, called *algorithm mcb*, calculates MCB intervals for the maximization case; to obtain results for the minimization case replace all the >'s with <'s in step 6 of the algorithm as indicated.

1. *Initialization:* number of independent observations, k; number of systems, r; arrays $sum[\ell] \leftarrow 0$ and $sumsq[\ell] \leftarrow 0$ for $\ell = 1, 2, \ldots, r$; $s \leftarrow 0$; array $ybar[\ell]$ of length r; critical value $d \leftarrow d^{\alpha}_{r(k-1),r}$ (see Table 5)
2. *Data:* $y[\ell, j]$, the jth independent observation from system ℓ
3. *Compute sums and sums of squares:*
   for $\ell \leftarrow 1$ to r:
      for $j \leftarrow 1$ to k:
         $sum[\ell] \leftarrow sum[\ell] + y[\ell, j]$
         $sumsq[\ell] \leftarrow sumsq[\ell] + y[\ell, j]*y[\ell, j]$
      endfor
   endfor
4. *Compute sample averages and pooled standard error estimate:*
   for $\ell \leftarrow 1$ to r:
      $ybar[\ell] \leftarrow sum[\ell]/k$
      $s \leftarrow s + sumsq[\ell] - sum[\ell]*sum[\ell]/k$
   endfor
   $s \leftarrow \text{sqrt}\{s/(r(k-1))\}$
5. *Confidence interval halfwidth:* $h \leftarrow d * s * \text{sqrt}(2/k)$
6. Find first- and second-best sample performance:
   (comment: for minimization problems replace all >'s by <'s in this step)
   $findex \leftarrow 1$; $first \leftarrow ybar[1]$
   $sindex \leftarrow 2$; $second \leftarrow ybar[2]$
   if $ybar[2] > ybar[1]$
      $findex \leftarrow 2$; $first \leftarrow ybar[2]$
      $sindex \leftarrow 1$; $second \leftarrow ybar[1]$

TABLE 5 Critical Values $d^{0.05}_{r(k-1),r}$ for MCB Intervals at 95% Confidence[a]

	Number of Systems, r							
k	3	4	5	6	7	8	9	10
2	2.938	2.885	2.849	2.827	2.815	2.809	2.807	2.807
3	2.337	2.417	2.466	2.502	2.532	2.558	2.581	2.601
4	2.180	2.287	2.356	2.407	2.448	2.482	2.511	2.537
5	2.108	2.227	2.305	2.362	2.408	2.445	2.478	2.506
6	2.067	2.193	2.274	2.335	2.384	2.424	2.458	2.488
7	2.041	2.170	2.255	2.318	2.368	2.410	2.445	2.476
8	2.022	2.154	2.241	2.306	2.357	2.400	2.436	2.467
9	2.008	2.142	2.230	2.296	2.349	2.392	2.429	2.461
10	1.998	2.133	2.222	2.289	2.342	2.386	2.424	2.456
11	1.989	2.126	2.216	2.284	2.337	2.382	2.419	2.452
12	1.982	2.120	2.211	2.280	2.333	2.378	2.416	2.449
13	1.977	2.115	2.207	2.275	2.330	2.375	2.413	2.446
14	1.972	2.111	2.203	2.272	2.327	2.372	2.411	2.444
15	1.968	2.107	2.200	2.269	2.324	2.370	2.408	2.442
16	1.964	2.104	2.197	2.267	2.322	2.368	2.407	2.440
17	1.961	2.101	2.195	2.265	2.320	2.366	2.405	2.439
18	1.959	2.099	2.193	2.263	2.319	2.365	2.404	2.438
19	1.956	2.097	2.191	2.261	2.317	2.363	2.402	2.437
20	1.954	2.095	2.189	2.260	2.316	2.362	2.401	2.435
30	1.941	2.084	2.180	2.251	2.308	2.355	2.394	2.429
40	1.935	2.078	2.175	2.246	2.304	2.351	2.391	2.426
50	1.931	2.075	2.172	2.244	2.301	2.349	2.389	2.424
60	1.928	2.073	2.170	2.242	2.300	2.347	2.388	2.423
120	1.922	2.067	2.165	2.238	2.296	2.344	2.384	2.420
∞	1.916	2.062	2.161	2.234	2.292	2.341	2.381	2.417

[a]These values were calculated using a program provided by Jason C. Hsu

```
    endif
    for ℓ ← 3 to r:
      if ybar[ℓ] > first
        sindex ← findex; second ← first
        findex ← ℓ; first ← ybar[ℓ]
      else
        if ybar[ℓ] > second
          sindex ← ℓ; second ← ybar[ℓ]
        endif
      endif
    endfor
```

7. *Output:* For system ℓ, the lower endpoint, point estimate, and upper endpoint for $\theta^{(\ell)} - \max_{i \neq \ell} \theta^{(i)}$ (or $\theta^{(\ell)} - \min_{i \neq \ell} \theta^{(i)}$)

```
    for ℓ ← 1 to r:
      point ← ybar[ℓ] − first
    if ℓ = findex then point ← ybar[ℓ] − second
      lower ← min{point − h , 0}
      upper ← max{point + h, 0}
      output ℓ, lower, point, upper
    endfor
```

The critical values $d^{\alpha}_{r(k-1),r}$ for 95% confidence intervals ($\alpha = 0.05$) are given in Table 5. Critical values for other confidence levels can be found in Table 4 of Appendix 3 in Hochberg and Tamhane (1987).

MCB intervals are constrained intervals, meaning that they either contain 0 or one of their endpoints is 0. In a maximization problem, if the confidence interval for $\theta^{(\ell)} - \max_{i \neq \ell} \theta^{(i)}$ contains 0 it means that system ℓ is not significantly different from the best system, and may be the best. If the upper endpoint of the interval is 0, then system ℓ is not the best system. However, if the lower endpoint is 0, then system ℓ is the best system; at most one system will have lower endpoint 0. These statements are made with confidence level $1 - \alpha$.

The minimization case is illustrated by the inventory example. For each inventory policy a single replication of 2020 periods was generated and the first 20 periods of data were deleted to reduce initial-condition bias (see Section 7). A different random number stream was used to generate demands for each policy so that the results across policies are independent. To obtain approximately i.i.d. outputs from within each replication, the 2000 outputs were batched into $k = 20$ batch means, and the batch means were used as the basic data for each inventory policy (see Section 8). Thus, the critical value from Table 5 is $d^{0.05}_{5(19),5} = 2.189$, which comes from the $k = 20$ row and $r = 5$ column. The 95% simultaneous confidence intervals for $\theta^{(\ell)} - \min_{i \neq \ell} \theta^{(i)}$ computed by *algorithm mcb* are given in Table 6.

Since the lower endpoints for policies 3, 4, and 5 are 0, they are significantly more expensive than the least-cost policy; in other words, the difference between the cost of any one of these policies and the minimum cost of the other policies is greater than 0. Policy 2 is the sample best policy, but it is not statistically significantly different from policy 1 since the intervals for both policies 1 and 2 contain 0. However, if policy 2 is not the best, the upper endpoint of its interval indicates that it is no more than 1.836 per period more expensive than the best. Similarly, policy 1 is no more than 2.526 from the best. If these differences are practically insignificant, then policy 2 should be selected. If more precision is required, then additional replications or batches will sharpen the comparison.

9.2. Metamodels

In some simulation studies, the analyst is interested in determining the effects of controllable system factors, perhaps to optimize system performance with respect to the controllable factors or to examine

TABLE 6 MCB Intervals for $\theta^{(\ell)} - \min_{i \neq \ell} \theta^{(i)}$ in the Inventory Simulation

ℓ	Policy	Lower Endpoint	Point Estimate	Upper Endpoint
1	(20,40)	−1.836	0.345	2.526
2	(20,80)	−2.526	−0.345	1.836
3	(40,60)	0	15.409	17.590
4	(40,100)	0	18.574	20.755
5	(60,100)	0	34.064	36.245

their relative impact. For example, in the license-renewal simulation, the application processing rate, μ, may be controllable through the equipment and staff available to the clerk, and the analyst may be interested in its effect on the average processing delay for applications.

The sample performance of a simulated system can be viewed as a complicated function of the controllable factors, which include the random number seeds or streams assigned to the experiment. A *metamodel* is a simpler function that approximates the relationship between system performance and the controllable factors. The primary benefit of a metamodel is that it can be studied using straightforward mathematical analysis.

One approach that is often used to derive a metamodel is to propose a function that has some unknown parameters and then estimate the parameters via least-squares regression. For example, the following metamodel might be postulated to describe the relationship between the average monthly delay for renewal applications, $\overline{Y}$, and the processing rate, μ:

$$\overline{Y} = \beta_0 + \beta_1\mu + \beta_2\mu^2 + \epsilon(s) \tag{6}$$

where β_0, β_1, and β_2 are unknown parameters and $\epsilon(s)$ is a mean 0 random variable that represents the variability in the system. Notice that $\epsilon(s)$ is a function of s, the random number stream or seed assigned to the simulation experiment. Metamodel (6) implies that $\theta(\mu)$, the expected average delay when the processing rate is μ, can be approximated by the function

$$\theta(\mu) = \mathrm{E}_\mu[\overline{Y}] \approx \beta_0 + \beta_1\mu + \beta_2\mu^2$$

Metamodels, such as (6), become useful when appropriate values are assigned to the unknown parameters. Experimental design—choosing the levels of the controllable factors at which simulation experiments will be performed—is needed to ensure good parameter estimates; see Chapter 85 of the Handbook for guidance. Once the experiments have been performed, least-squares regression is a standard procedure for estimating the parameters; see Chapter 87 of the Handbook.

The methods and procedures discussed in Chapters 85 and 87 are as relevant to simulation experiments as they are to statistical experiments in general. There is one experiment design and analysis issue that is unique to simulation, however:

Design and Analysis Principle 8. Statistical inference on a metamodel is typically based on the assumption that outputs from different design points (factor settings), and the replications at a single design point, are statistically independent. Thus, different random number streams or seeds should be assigned to each design point unless the analyst plans to account directly for the dependence induced by using common streams.

Although there are potential advantages from using common streams or seeds across design points, methods for statistical analysis under such designs are still evolving (Hussey, et al. 1987a, b; Kleijnen 1988; Nozari, et al. 1987; Schruben and Margolin 1978). See also Section 10.

Table 7 shows the experiment design used to fit (6). Ten independent replications were made at each design point, and the parameters were estimated via least-squares regression. The resulting metamodel for the expected average delay is

$$\theta(\mu) \approx 12.01 - 0.97\mu + 0.02\mu^2 \tag{7}$$

Because each design point was simulated independently, standard tests for significance of the parameters and lack of fit could be used to validate the fitted model.

The derivative of (7) with respect to μ gives the approximation

TABLE 7 Experiment Design for Fitting License-Renewal Processing Metamodel

Design Point	Processing Rate, μ	Random Number Stream
1	15	3
2	20	1
3	25	4

$$\frac{d\theta(\mu)}{d\mu} \approx -0.97 + 0.04\mu$$

The derivative of a metamodel is one way to examine the sensitivity of the expected average delay to the processing rate. For example, at $\mu = 20$ the derivative is -0.17, implying that an increase of one application per day in the processing rate should result in a decrease of 0.17 days in the expected average delay.

The extent to which a metamodel is valid is necessarily limited. Extrapolating a metamodel beyond the factor settings used to fit the model should always be done with caution. For instance, the metamodel (7) predicts that the expected average delay will begin to increase if the processing rate exceeds 25 applications per day!

Many methods, in addition to metamodel estimation, have been proposed for examining the sensitivity of system performance to controllable factors. For recent surveys see Andradóttir (1998), Glynn (1989), and Jacobson and Schruben (1989) and the references therein.

10. VARIANCE REDUCTION

The goal of variance reduction is to decrease the error of a point estimate, which should lead to a smaller estimated standard error or a narrower confidence interval (see Section 8 for a discussion of measures of error). This section describes the variance reduction technique known as *common random numbers* (CRN), which is useful for reducing the error in comparing the expected performance of two or more systems. The presentation is based on Nelson (1987).

For simplicity, suppose that there are two competing systems, and let $\theta^{(1)}$ and $\theta^{(2)}$ be the (unknown) expected performance of system 1 and system 2, respectively. If there are more than two systems, then the methods described below can be applied to each pair of systems. When comparing the performance of two systems, the single parameter $\theta^{(1)} - \theta^{(2)}$ often captures the essential information about their relative performance.

For example, consider comparing the expected cost per period of inventory policies 1 and 2 of the inventory example. Since a smaller expected cost is preferred, $\theta^{(1)} - \theta^{(2)} < 0$ implies that inventory policy 1 is superior, while $\theta^{(1)} - \theta^{(2)} > 0$ implies that inventory policy 2 is better. The absolute value of $\theta^{(1)} - \theta^{(2)}$ is the amount by which one of the policies is better than the other.

Generically, let $Y_1^{(\ell)}, Y_2^{(\ell)}, \ldots, Y_k^{(\ell)}$ be outputs from k i.i.d. replications (or possibly batch means, see Section 8) from system ℓ, for $\ell = 1, 2$, so that $\theta^{(\ell)} = \mathrm{E}[Y_j^{(\ell)}]$; that is, $Y_j^{(\ell)}$ is the sample performance of system ℓ on replication j. Let $D_j = Y_j^{(1)} - Y_j^{(2)}$ for $j = 1, 2, \ldots, k$, the difference between the sample performance of the two systems on the jth replication. A point estimator of $\theta^{(1)} - \theta^{(2)}$ is

$$\overline{D} = \frac{1}{k}\sum_{j=1}^{k} D_j = \overline{Y}^{(1)} - \overline{Y}^{(2)}$$

An estimator of the standard error of $\overline{D}$ is $S_D/\sqrt{k}$, and an approximate $(1 - \alpha)100\%$ confidence interval for $\theta^{(1)} - \theta^{(2)}$ is

$$\overline{D} \pm t_{1-\alpha/2,k-1} S_D/k$$

where

$$S_D^2 = \frac{1}{k-1}\sum_{j=1}^{k} (D_j - \overline{D})^2$$

is the sample variance of the differences and $t_{1-\alpha/2,k-1}$ is the $1 - \alpha/2$ quantile of the t distribution with $k - 1$ degrees of freedom (a table of t-distribution quantiles can be found in any statistics text; if no such text is available, the normal distribution quantiles $z_{1-\alpha/2}$ from Table 4 can be used provided $k \geq 30$). The condition that makes this confidence interval valid is that the D_j are i.i.d. normally distributed random variables. If the $Y_j^{(\ell)}$ are actually within-replication averages or batch means, then this condition may be approximately satisfied.

The size of the estimated standard error or width of the confidence interval determines whether or not a difference in system performance can be detected in the presence of the random variation in the simulation results. To be able to detect a difference, it is necessary that the potential error in the point estimate $\overline{D}$ be small with respect to the difference $\theta^{(1)} - \theta^{(2)}$.

The true standard error of $\overline{D}$ can be written as $\sigma_D/\sqrt{k}$, where $\sigma_D^2 = \mathrm{Var}[D_j]$, the common variance of the differences. Thus, the size of the estimated standard error and the width of the confidence interval depend on the underlying variance σ_D^2. The variance of the difference is

$$\sigma_D^2 = \text{Var}\,[Y_j^{(1)}] + \text{Var}\,[Y_j^{(2)}] - 2\text{Cov}[Y_j^{(1)}, Y_j^{(2)}]$$
$$= \sigma_1^2 + \sigma_2^2 - 2\rho\sigma_1\sigma_2 \quad (8)$$

where $\sigma_\ell^2 = \text{Var}\,[Y_j^{(\ell)}]$ and ρ is the correlation between pairs of observations from the two systems on the same replication, $Y_j^{(1)}$ and $Y_j^{(2)}$. Typically, the variances σ_1^2 and σ_2^2 are properties of the systems being simulated and are not easily altered. However, the correlation ρ may be controlled, and Eq. (8) shows that making $\rho > 0$ will result in a variance reduction. The following design and analysis principle describes how to accomplish this:

Design and Analysis Principle 9. Positive correlation between the sample performance of two systems can often be induced by assigning the same ("common") random number streams or seeds to the simulation of each system (see Section 4 for a description of random number streams and seeds). The magnitude of the correlation can be increased further by synchronizing the pseudorandom numbers, as described below.

The intuition behind the use of CRN is straightforward: a fairer comparison between competing systems is obtained if they are both subjected to the same experimental conditions. Assigning the same random number streams or seeds to each system means that the influence of randomness is similar for both systems and the observed differences in performance are due to differences in the systems, not in the pseudorandom numbers.

The goal of synchronization is to guarantee that each pseudorandom number is used *for the same purpose* in the simulation of each system. The effect of CRN can be greatly enhanced by synchronizing the pseudorandom numbers. Synchronization is achieved through the random number stream assignments to the input processes.

Simulations contain one or more input processes, which are frequently i.i.d. sequences of random variables (see Section 4 for the definition of simulation inputs). For example, in the inventory simulation there is one input process, the sequence of demands in each period. In the license-renewal simulation there are two input processes, the successive times between arrivals of applications and the successive processing times for applications.

One of the best ways to synchronize is to dedicate a distinct random number stream or seed to each input process and then to use the same stream assignment for all simulated systems. In the license-renewal simulation, assigning different random number streams to the two input processes guarantees that exactly the same sequence of application arrival times is experienced by all systems and that the processing times will be positively correlated for all processing rates, μ. In the inventory simulation, CRN guarantees that each inventory policy experiences exactly the same sequence of demands. Synchronization is the primary reason for having more than one random number stream in a simulation language.

Table 8 illustrates the effect of CRN on the inventory simulation when the goal is to estimate $\theta^{(1)} - \theta^{(2)}$, the expected difference between the cost per period of inventory policies 1 and 2. The experiment design is the same one described in Section 9, so the basic data for each inventory policy are $k = 20$ approximately i.i.d. normal batch means (for the purposes of this illustration they can be thought of as k i.i.d. replications). The table gives the point estimate, estimated standard error, and a 95% confidence interval for the expected difference.

The estimated correlation between outputs when using CRN was 0.58, which is positive as desired. Notice that the estimated standard error and the width of the confidence interval are reduced relative to the experiment with independent runs (different random number streams for each policy).

More critically, the point estimate derived without CRN is positive, indicating that policy 2 yields the smallest expected cost, while the point estimate derived with CRN is negative, indicating that policy 1 is superior. For this simple model it can be shown that policy 1 does have the smaller expected cost, so the use of CRN leads to the correct decision. However, in both experiments the 95% confidence interval for the difference contains 0, so it cannot be stated conclusively that policy 1 is best based on the data. Additional batches are needed to reduce the error even further.

TABLE 8 Estimated Difference in Expected Cost With and Without CRN

Design	D	Standard Error	95% Confidence Interval
Independent	0.3	0.57	(−0.85, 1.54)
CRN	−0.3	0.31	(−0.98, 0.35)

ACKNOWLEDGEMENTS

The author gratefully acknowledges the helpful comments and recommendations of Gordon Clark, Lynne Goldsman, Jason C. Hsu, W. David Kelton, A. Alan B. Pritsker, Charles Reilly, Bruce Schmeiser, Lee Schruben, Fataneh Taghaboni, Mingjian Yuan, and the 1990 IND ENG 854 class at Ohio State on the 2d edition. He also acknowledges the support of NSF Grant DMI-9622065.

REFERENCES

Andradóttir, S. (1998), "Simulation Optimization," in *Handbook of Simulation,* John Wiley & Sons, New York.

Banks, J., Carson, J. S., Nelson, B. L., and Nicol, D. M. (2000), *Discrete-Event System Simulation,* 3rd Ed., Prentice Hall, Upper Saddle River, NJ.

Barton, R. R., and Schruben, L. W. (1989), "Graphical Methods for the Design and Analysis of Simulation Experiments," in *Proceedings of the 1989 Winter Simulation Conference,* E. MacNair, K. Musselman, and P. Heidelberger, Eds., pp. 51–61.

Bechhofer, R. E., Santner, T. J., and Goldsman D. (1995), *Design and Analysis for Statistical Selection, Screening and Multiple Comparisons,* John Wiley & Sons, New York.

Bratley, P., Fox, B. L., and Schrage, L. E. (1987), *A Guide to Simulation,* 2nd Ed., Springer, New York.

Clark, G. M., and Yang, W. (1986), "A Bonferroni Selection Procedure when Using Common Random Numbers with Unknown Variances," in *Proceedings of the 1986 Winter Simulation Conference,* J. Wilson, J. Henriksen, and S. Roberts, Eds., pp. 313–315.

Fishman, G. S. (1978), *Principles of Discrete Event Simulation,* John Wiley & Sons, New York.

Fishman, G. S., and Yarberry, L. S. (1997), "An Implementation of the Batch Means Method," *INFORMS Journal on Computing* Vol. 9, pp. 296–310.

Glynn, P. W. (1989), "Optimization of Stochastic Systems via Simulation," in *Proceedings of the 1989 Winter Simulation Conference,* E. MacNair, K. Musselman, and P. Heidelberger, Eds., pp. 90–105.

Goldsman, D. (1985), "Ranking and Selection Procedures Using Standardized Time Series," in *Proceedings of the 1985 Winter Simulation Conference,* D. Gantz, G. Blais, and S. Solomon, Eds., pp. 120–124.

Goldsman, D., and Nelson, B. L. (1998), "Comparing Systems via Simulation," in *Handbook of Simulation,* John Wiley & Sons, New York.

Goldsman, D., Schruben, L. and Swain, J. J. (1989), "Tests for Transient Means in Simulated Time Series," *Naval Research Logistics,* Vol. 41, pp. 171–187.

Gupta, S. S. and Panchapakesan, S. (1979), *Multiple Decision Procedures: Theory and Methodology of Selecting and Ranking Populations,* John Wiley & Sons, New York.

Hochberg, Y., and Tamhane, A. C. (1987), *Multiple Comparison Procedures,* John Wiley & Sons, New York.

Hsu, J. C. (1984), "Constrained Two-Sided Simultaneous Confidence Intervals for Multiple Comparisons with the Best," *Annals of Statistics,* Vol. 12, pp. 1136–1144.

Hussey, J. R., Myers, R. H., and Houck, E. C. (1987a), "Correlated Simulation Experiments in First-Order Response Surface Design," *Operations Research,* Vol. 35, pp. 744–758.

Hussey, J. R., Myers, R. H., and Houck, E. C. (1987b), "Pseudorandom Number Assignment in Quadratic Response Surface Designs," *IIE Transactions,* Vol. 19, pp. 395–403.

Iglehart, D. L. (1977), "Simulating Stable Stochastic Systems, VII: Selecting the Best System," *TIMS Studies in the Management Sciences,* Vol. 7, pp. 37–49.

Jacobson, S. H., and Schruben, L. W. (1989), "Techniques for Simulation Response Optimization," *Operations Research Letters,* Vol. 8, pp. 1–9.

Kleijnen, J. P. C. (1988), "Analyzing Simulation Experiments with Common Random Numbers," *Management Science,* Vol. 34, pp. 65–74.

Koenig, L. W., and Law, A. M. (1985), "A Procedure for Selecting a Subset of Size m Containing the l Best of k Independent Populations, with Applications to Simulation," *Communications in Statistics—Simulation and Computation,* Vol. 14, pp. 719–734.

Law, A. M., and Kelton, W. D. (2000), *Simulation Modeling and Analysis,* 3rd Ed., McGraw-Hill, New York.

Nelson, B. L. (1987), "Variance Reduction for Simulation Practitioners," in *Proceedings of the 1987 Winter Simulation Conference,* A. Thesen, H. Grant, and W. Kelton, Eds., pp. 43–51.

Nozari, A., Arnold, S. F., and Pegden, C. D. (1987), "Statistical Analysis for Use with the Schruben and Margolin Correlation Induction Strategy," *Operations Research,* Vol. 35, pp. 127–139.

Schmeiser, B. (1982), "Batch Size Effects in the Analysis of Simulation Output," *Operations Research,* Vol. 30, pp. 556–568.

Schruben, L. (1981), "Control of Initialization Bias in Multivariate Simulation Response," *Communications of the ACM,* Vol. 24, pp. 246–252.

Schruben, L. (1982), "Detecting Initialization Bias in Simulation Output," *Operations Research,* Vol. 30, pp. 569–590.

Schruben, L. W., and Margolin, B. H. (1978), "Pseudorandom Number Assignment in Statistically Designed Simulation and Distribution Sampling Experiments," *Journal of the American Statistical Association,* Vol. 73, pp. 504–525.

Schruben, L. W., Singh, H., and Tierney, L. (1983), "Optimal Tests for Initialization Bias in Simulation Output," *Operations Research,* Vol. 31, pp. 1167–1178.

Sullivan, D. W., and Wilson, J. R. (1989), "Restricted Subset Selection Procedures for Simulation," *Operations Research,* Vol. 37, pp. 52–71.

Welch, P. D. (1983), "The Statistical Analysis of Simulation Results," in *The Computer Performance Modeling Handbook,* S. Lavenberg, Ed., Academic Press, New York, pp. 286–328.