Proceedings of the 2001 Winter Simulation Conference
B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, eds.

SOME MYTHS AND COMMON ERRORS IN SIMULATION EXPERIMENTS

Bruce W. Schmeiser

School of Industrial Engineering
Purdue University
West Lafayette, IN 47907–1287, U.S.A.

ABSTRACT

During the more than fifty years that Monte Carlo simulation experiments have been performed on digital computers, a wide variety of myths and common errors have evolved. We discuss some of them, with a focus on probabilistic and statistical issues.

1 INTRODUCTION

Of the many meanings and purposes of *simulation*, we consider only that of the 1940's Manhattan Project: on a digital computer, perform a sampling experiment on a given *model* with the purpose of obtaining a statistical *point estimate* of a *performance measure* whose value is unknown. We refer to such experiments as Monte Carlo Simulation and include dynamic (in time) models as well as static.

We discuss various myths and common errors associated with Monte Carlo simulation experiments. Myths cause needless extra effort or expense; errors cause needless degradation in the distribution of $\hat{\theta}$, typically measured with bias, standard deviation, or root mean squared error. Some are fact; many are opinion. Some are well known; many are little known, at least among the general practitioner community. No references are given because the best, or even a good, reference for each myth and error often is not obvious or sometimes is too embarrassing.

The discussion begins with a world view in Section 1.1 and a list of error sources in Section 1.2. Specific myths and errors follow in Section 2. Due to a page limit, discussion of each myth and error is, with only a few exceptions, restricted to a single paragraph.

1.1 A World View

Our discussion uses a world view, some vocabulary, and (just a bit of) notation that evolved from Barry Nelson's Ph.D. dissertation in the early 1980s. Let θ denote the performance measure's unknown value and let $\hat{\theta}$ denote the point estimator of θ; in general both are vectors, with each component corresponding to an aspect of performance deemed interesting by the simulation practitioner.

The experiment can be illustrated as

$$G \rightarrow U \rightarrow X \rightarrow Y \rightarrow \hat{\theta}.$$

Each of the four arrows is a deterministic function, with the *randomness generator* G used to create a set of *random numbers* U, which is used to generate a set of *input data* X, which is used to compute a set of *output data* Y, which is used to compute the point estimator $\hat{\theta}$. With few exceptions in practice, G is a pseudorandom-number generator with associated seed value(s) and U contains random variables with values between zero and one that are assumed to be uniformly and independently distributed. Neither G nor U contain information about the model; many simulation languages, for example, have a default choice of random-number generator that can be used for all applications.

The model, which is given, is composed of two parts. The *input model*, corresponding to the arrow between U and X, is the given probability model for the (possibly dependent) random variables in X; observed values of the input data are referred to as the *random variates*. The *logic model*, corresponding to the arrow between X and Y, is the given relationship between the input data and the output data. The output data in Y are from an unknown probability model for which the unknown performance measure θ is a property. Typically the output data are identically distributed; often the output data are not independent.

In what sense is Y's probability model, and in turn θ, unknown? Although the input-model and logic-model functions are given, the transformation from random numbers to input data to output data can be so complicated that the probability model of the output data is unknown in the sense that it cannot be determined easily. In general, there is no reason to expect that there is an easier way to express the output-data probability model than to state the input model and the logic model. Therefore, as in inferential

statistics in general, the performance measure θ is used to summarize the output-data probability model.

Simulation and experimentation that do not fit into this world view lie outside the boundaries of this paper. In particular, we don't consider other important types of simulation, such as training simulators, virtual reality, or numerical solution of differential equations, which undoubtedly have their own myths and common errors. We also don't include experimentation in general, although some of our comments apply to inferential-statistics experiments with real-world (that is, not computer generated) data.

Why discuss the world view of simulation experiments in a paper about myth and errors? First, of course, is to introduce some notation and terminology that simplify the discussion of specific myths and common errors below. More generally, though, many of the errors and some of the misperceptions underlying myths are avoided when practitioners have a clear world view in mind as they perform simulation experiments.

1.2 Sources of Analysis Error

Suppose that a practitioner has a specified a model (that is, both an input model and a logic model) and therefore a performance measure θ whose value is to be estimated via a simulation experiment. This is a big supposition, of course, but one that underlies all of probability modeling. If the given model does not correspond well to the real world, then the value of θ will not have a useful real-world interpretation. Some *modeling error* is unavoidable for complex problems, whether performed analytically, numerically, or via Monte Carlo experimentation.

(Aside. All three are analysis methods. There is no such thing as a "simulation model". Any probability model can be analyzed in any of the three ways, or in a combination of the three ways, although certainly sometimes the practitioner knows that Monte Carlo simulation will be the analysis method. When the model is specified using software designed for simulation analysis, the phrase "simulation model" becomes natural, even if imprecise.)

Other than modeling error, all error is *analysis error*, the difference between θ and the point estimator $\hat{\theta}$ (again, regardless of the method of analysis). While abstracting a model from the real world is very much an art, with many ways to err as well as to be correct, analysis of the model is more of a science, and therefore easier, both to teach and to do. Analysis is also easier because the degree of success is so easily measured by $\theta - \hat{\theta}$. Finally, analysis is easier because there are only five sources of error. We briefly discuss each source.

1. Coding error. The code wrong, in commercial software or the practitioner's specification of the model. Sometimes called *verification error* (analogous to modeling error being called *validation error*).
2. Numerical error. Computer arithmetic is not real-number arithmetic; computers can store only a finite set of numbers. Examples include numbers close to zero being denser than numbers close to one, floating-point comparisons being suspect because of rounding, and combinatorial calculations overflowing.
3. Random-number error. Pseudorandom numbers are not truly random numbers. As computers become faster, sample sizes become larger, and sensitivity to random-number error increases.
4. Random-variate error. Methods to generate random variates are sometimes approximations. For example, the reasonably good standard normal inverse transformation $x = (u^{0.135} - (1-u)^{0.135})/0.1975$ truncates at about five standard deviations from the mean.
5. Sampling error. Monte Carlo simulation analysis is fundamentally a statistical-inference method; therefore, sampling error is unavoidable. Sampling error is typically measured by standard error, the standard deviation of the point estimator, which is often inversely proportional to the square root of the sample size.

In addition to modeling error and these five kinds of analysis error, there are many possible practical implementation errors (such as failure to involve decision makers, inadequate documentation, and budget variances). These are not errors of experimentation, however, so we do not discuss them here.

2 MYTHS AND COMMON ERRORS

In the next ten subsections, we discuss various myths and common errors. Inclusion is based on the author's experience in teaching, research, and consulting; others certainly have their own favorites. Except for the careful choice of subsection titles, the organization into ten subsections is arbitrary.

2.1 Simulation as a Last Resort

Maybe the greatest myth about Monte Carlo simulation, perpetuated even within some textbooks, is that simulation is a method of last resort. The argument is that simulation requires extensive effort and provides minimal insight. Certainly examples of needless, or needlessly detailed, simulation experiments are easy to find; a back-of-the-envelope calculation often can provide substantial insight with little effort. A well-trained probabilist with a complex, but nicely structured, model is a wonder to behold.

Many practitioners' backgrounds and many models' structures, however, do not lend themselves to any analysis other than Monte Carlo experimentation. In addition, the availability of easy-to-use modeling software with automatic animation of Monte Carlo output data dramatically changes the effort/insight tradeoff. Further, many well-trained probabilists first simulate, at least mentally, a problem to ensure that the problem is well defined. Monte Carlo simulation is also useful to a well-trained probabilist as a quick method to disprove a conjecture or to verify complicated analyses; the simulation process forces attention to model details that can be missed with pencil and paper.

2.2 Input Modeling

Common errors abound in input modeling, the process of determining the probability model for the input data. The input model is based on "what-if" conjectures, expert opinions, or real-world data.

Maybe the most common error is to ignore the physics of the system to be modeled. Modeling a service time with a normal distribution, for example, ignores the physical fact that a service time cannot be negative. Although the modeler might rationalize that zero is many standard deviations below the mean, later experimentation might use a different normal distribution for which the probability of being less than zero is non-negligible.

A second error is to ignore statistical dependencies, either within a random vector or through time. Most commercial simulation software provides no support to a practitioner who wishes to include dependence within an input model. Assuming that processing times are independent, both at different servers for the same part or at the same server for different parts, is common; more reasonable might be a random vector in the former case and a time series in the latter case.

A third error is to use only classical distributions, those named functional forms that are typically included in distribution-fitting software. The problem is that many of these distribution families provide very similar probability models, but almost all have unimodal or U-shaped density functions. An application requiring multiple modes requires a mixture of such distributions, but little commercial-software support exists to support mixture models.

A fourth error is rejecting an adequate model because it fails a lack-of-fit test. In a world with some huge real-world data sets, the power of such tests is quite high. Similarly, small data sets lead to low power, but failure to reject does not imply that the input model is adequate. Much better to check visual fit and conformance with physical properties. We elaborate on the shortcomings of hypothesis testing in Section 2.6.

A myth is that maximum-likelihood estimation (MLE) should be used to fit distributions to data. The myth arises from MLE's wonderful limiting properties. For huge data sets, where the limiting properties are relevant, the computational hassles of MLE estimation can be substantial, including both large computation time and numerical problems. Often method-of-moments estimators, for example, are easier to compute with only minimal, if any, loss of statistical information. Also arguing against MLE is that it requires the user to choose a family of distributions before fitting begins; the optimal limiting properties assume that the choice is correct.

2.3 Methods for Random Variates

Random-variate generation is the conversion of U(0,1) random numbers into observations from the input model. Most classical distribution families have several fast, exact methods, many of which can be found on the internet as public-domain algorithms. There are four fundamental ideas in random-variate generation: the inverse cumulative distribution function, acceptance-rejection, composition, and special properties.

The most-common error in commercial simulation software is the failure to use the inverse transformation. Despite providing multiple random-number streams, non-inverse routines continue to be used. Although non-inverse routines are sometimes faster and easier to code than numerical inverse routines, their use destroys the variance-reduction purpose of the multiple random-number streams. Although common random numbers will work when distributions are unchanged, other correlation-induction techniques, such as antithetic variates and external control variates, will not.

A myth is that the alias method is a particularly good method for generating discrete random variables when the number of positive-probability values is finite, say k. The alias method, which provides a random variate with one U(0,1) random number and two "if" statements, is indeed fast and the idea (an application of composition) is creative. The set-up, unfortunately, is not simple, involving identifying $k-1$ two-point distributions. Simpler, requiring about the same memory, and almost as fast is using index tables to implement the inverse transformation.

2.4 U(0,1) Random Numbers

The discussion in this subsection assumes the usual situation: Pseudo-random number generators implementing linear-congruential logic in positive integer (or equivalent) arithmetic, converting to the U(0,1) interval by dividing by the largest relevant integer.

The leading error is probably incorrect implementation. Various computer-dependent numerical problems can occur, with a leading problem being a compiler that defines an integer with too-few bits.

A second error is choosing a bad generator. The most-famous example is RANDU, Fortran code distributed for use with IBM 360 computers in the IBM Scientific Subroutine Package. Because RANDU, like many generators, is only a few lines of code, its half-life is long. A recurring bad-generator idea involves implementations that are based on floating-point logic, sometimes with references to chaotic behavior; floating-point logic is computer dependent and chaotic behavior doesn't imply the desirable properties of a random-number generator.

A myth is that statistical analysis is inappropriate for simulation output data because there is nothing random in a Monte Carlo simulation experiment; digital computers are deterministic. This reasoning is false, however, because there is indeed a truly random component. The choice of random-number generator and the choice of initial seed(s) is a decision made by humans. This choice is made without regard to the model to be analyzed (although the choice might be a default). Therefore, although all later logic is indeed deterministic, the output data are the result of true randomness.

A second myth is that the initial integer, the *random-number seed*, should be odd. The origin of the myth is that one kind of linear-congruential generator, now little used, partitions the integers into four equal-size sets, two of odd integers and two of even integers. One of the two sets of even integers contains zero, the only value that causes the generator to degenerate. Choosing an odd integer ensures that such a generator does not degenerate. Unless using such a generator, even seeds are fine.

A third myth is that a separate stream of random numbers should be used for each "kind" of random process. For example, arrivals would use stream 1 and service times at server i would use stream $i + 1$. Commercial software commonly provides multiple streams, but the only purpose of multiple streams is to support correlation-induction schemes (e.g., common random numbers, antithetic variates, and external control variates) for reducing the variance of the point estimator. The myth is that separate streams are needed to provide independence between random variates. If the random-number generator is good, however, independence occurs automatically, using only one stream. In fact, using multiple streams raises the (small) concern of using the same random number for two different purposes. Using one stream is best unless variance-reduction is to be attempted.

A third myth is that one should search for a good seed. This myth borders on being an error, in that if one somehow defines and finds a favorite seed, a source of true randomness disappears. At worst, the definition of "good" could be used to obtain a desired bias in the point estimator.

2.5 Logic Modeling

The single dominate error in logic modeling is excessive detail. With the impressive success of commercial software to simplify the process of expressing the logic model, and with animation increasingly used to "sell" the model, the temptation to include visible but unimportant details mounts. Sometimes the inclusion of such details results from poorly defined experiment objectives. Sometimes the model is written to be ready for unspecified future use. The tendency to include logic-model detail provides an interesting comparison to the tendency to omit details (e.g., dependence, time non-homogeneity) in input models.

2.6 Analysis of Output Data

Recall that the purpose of Monte Carlo simulation, by definition, is to estimate the value of a performance measure θ, which is composed of properties (e.g., mean, standard deviation, quantile, or correlation) of the output data's unknown distribution. We assume that the output data, denoted here by $Y_1, Y_2, \ldots, Y_n$, are identically distributed but not necessarily independent. For simplicity we assume that the sample size n is a constant (although in many simulation experiments it is a random variable.)

Analysis of the output data comprises two activities. First, point estimation is based on the function of the output data that provides the estimator $\hat{\theta}$. Second, sampling-error estimation is based on the function of the output data that provides the estimator of the quality of $\hat{\theta}$. Sampling error is often measured by the standard error, the standard deviation of $\hat{\theta}$, or its square, $\text{var}(\hat{\theta})$. The standard error is reported directly or used to compute other measures, such as confidence intervals.

Under this definition of output analysis, in which the output data are assumed to be identically distributed, warming up a steady-state simulation by discarding initial data is not part of output analysis. Because of this distinction, which is only a matter of semantics, we discuss the initial-transient problem in the subsections "Ill-Posed Problems" and "New Experiment".

A common myth is that the point estimator should depend upon the dependencies among the output data. In fact, such dependencies should be ignored. Whatever point estimator is appropriate for independent and identically distributed (iid) data should be used for simulation output data. For example, a probability should be estimated by the fraction of times the event occurs, a mean should be estimated by the sample average, and a variance should be estimated by a sample variance.

An associated common error is to assume that the quality of the point estimator does *not* depend upon the dependencies among the output data. The standard error of $\hat{\theta}$ can differ by orders of magnitude depending upon

the dependencies, which are typically measured by the correlations $\text{corr}(Y_i, Y_j)$. Substantial research is devoted to estimating the standard error for steady-state output data, especially when θ is a mean. Assuming that the output data are independent is a serious error, because correlations are often positive, which leads to underestimating the standard error, which leads the practitioner to conclude that the point estimator is more precise than it is.

A second, and maybe more-common, error is to ignore sampling error. Here we refer to the practitioner who blindly assumes that essentially $\theta = \hat{\theta}$. (A practitioner who knows that the sampling error is negligible or knows the approximate value of the standard error from similar experiments is not ignoring standard error.)

A third error is to use bootstrapping, jackknifing, and other computationally intensive methods to estimate the standard error of $\hat{\theta}$ for simulation output data. These methods are fine for real-world data, where the budget for collecting additional data is separate from the computational budget. In simulation experiments, however, computation spent estimating standard error could be spent increasing the sample size, and the thereby decreasing the standard error. Batching provides a default $O(n)$ standard-error estimation method.

A fourth error is to estimate standard-error with a batching method with very few or very many degrees of freedom. Very few degrees of freedom correspond to a large standard-error-estimator variance; in many contexts high variance corresponds to low bias, but here high variance causes negative bias (because of the square-root required to convert the variance estimate to a standard-error estimate). Very many degrees of freedom corresponds to small variance, but at a cost of bias arising from violating analysis assumptions.

A myth is that batches need to be (essentially) independent to obtain a good standard-error estimator. In fact, a class of estimators with good statistical properties uses overlapping batches, with adjacent batch statistics asymptotically having correlation one. That independence is not needed follows because variance estimators are averages (of squared differences), averages are sums, and the expected value of a sum is the sum of expected values.

A fifth error is to estimate θ when its value is infinity. The error arises out of a fundamental weakness of Monte Carlo simulation: it provides a finite estimate regardless of whether θ is finite. A classic example is the gambling game underlying the St. Petersburg Paradox. Specifically, I will pay you 2^X dollars, where X is the number of coin flips until the coin lands with head up. (The paradox is that although the minimum payoff is 2 dollars and the expected value is infinite, few people will pay more than 5 dollars to play.) Relevant to us here is that Monte Carlo simulation is unable to tell a practitioner that $\theta = \infty$.

A sixth, similar, error is to analyze output data when its value is undefined. The distribution of the ratio of two independent standard normal random variables is Cauchy, which is symmetric about zero with heavy tails. Although the mean is undefined, a Monte Carlo simulation experiment will provide a finite answer, except in the unlikely case that the denominator is computationally zero for one or more trials. Even when stated as the ratio of two normal random variables, the practitioner can be mislead. More subtle is the same problem when posed as a spinning radar whose expected intercept point is to be estimated.

A seventh, similar, error is to analyze output data as if they are identically distributed when they are not. A classic example is a non-stable queueing system, with arrivals overwhelming service capacity, for which no steady state exists and queues grow without bound. In an interesting example, a Ph.D. student estimated the steady-state effect of having workers move tools to desired locations whenever the system was stopped due to breakdown. Careful attention was paid to the experiment's design and deletion of initial data. The Monte Carlo output data indicated that moving tools improved performance. In fact, there was no steady-state improvement, because the system was an irreducible Markov process and therefore steady-state behavior was independent of the initial state. The correct conclusion was that moving the tools improved system performance for a long time, and therefore was worthwhile, despite the effect not lasting forever.

Hypothesis testing—hoping to reject a null hypothesis in favor of an alternative hypothesis—is considered by many people to be a fundamental topic for analyzing simulation output data. We provide five reasons why hypothesis testing is, with few exceptions, inappropriate for analyzing simulation output data.

Reason 1: Confusion between *statistical* significance and *practical* significance. The simulation experiment can produce statistically significant conclusions because a large sample size n, and therefore large power, is available by simply running the computer longer. Rejecting a null hypothesis with a p value close to zero or one is not evidence that the null hypothesis is far from false. Rather, it is evidence that the computer was run long enough to conclude with substantial confidence that the null hypothesis is not true.

Reason 2: The simple null hypothesis is known to be false before the experiment is run. In medical experimentation, for example, it is possible that a novel treatment and a placebo treatment have the same effect, which would be the null hypothesis; the mysteries inside the human body are not well understood. In simulation experimentation the practitioner has built the model; there are no mysteries inside the two systems being compared. If the null hypothesis is known to be false before analyzing the data, then there is not need to ask the data for help in concluding whether the null hypothesis is false. Most of ANOVA is inappropriate for simulation analysis.

Reason 3: The choice of α, the probability of rejecting the null hypothesis given the that the null hypothesis is true, is usually arbitrary. But if α is arbitrary, the conclusion is arbitrary. This error, or course, applies to most contexts, not only to simulation analysis.

Reason 4: The tendency to interpret the p value as the probability that the null hypothesis is true. In fact, the p value is the probability of no event. First, p is a random variable, so at best it only could estimate the probability of an event. Second, if indeed the null hypothesis is true, p has a $U(0,1)$ distribution, so it would be a poor estimator.

Reason 5: Cascading tests of hypotheses. For example, failing to reject normality the practitioner tests for independence with a test that assumes normality; failing to reject independence, the practitioner tests for equality of means with a test that assumes normality and independence. Such procedures are frail structures, as is obvious by their dependence upon arbitrary choices of α and n. Whether the next step in the cascade is appropriate depends upon the purpose of the next step, yet the previous step's logic (typically) does not consider the logic of the next step. This error maybe is better categorized as a myth because it is widespread and because there is little real-world evidence that the severity of the error is great. Because simulation experiments often have large sample sizes, the more common error is probably too much power forcing a practitioner to conclude that continuing down the cascade is inappropriate.

2.7 Tactical Issues

An issue is *tactical* if it concerns how to exercise the model to obtain the output data. We focus on the sample-size decision.

A myth is that the practitioner needs a faster computer. Point-estimator standard error is, with only a few exceptions, $O(n^{-1/2})$. Therefore, requesting one additional significant digit in $\hat{\theta}$ requires increasing the sample size by a factor of 100. A computer that is faster by a factor of ten accomplishes little.

An error is to seek a point-estimator standard error that is far below the modeling error. As sample size goes to infinity the standard error goes to zero, but analysis precision that exceeds modeling precision is wasted, and maybe misleading, effort. Cutting the sample size by a factor of ten is often more appropriate than asking for a faster computer.

A second error is to seek point-estimator standard error smaller than needed for the purpose of the experiment. For example, thesis students regularly run week-long simulation experiments using all available computing, despite the purpose of simply identifying the best of a small set.

A second myth is that sample size is related to the real-world system's expected lifetime. For example, a practitioner might object to a steady-state fifty-year observation of a semiconductor fab. Such an objection is valid, of course, if a steady-state analysis is inappropriate. If, however, the definition of θ is based on steady-state behavior then running for fifty (or fifty thousand) years is valid.

2.8 Ill-Posed Research Problems

Research in simulation methodology results in underlying theory and methods, typically algorithms that solve a commonly encountered problem. We discuss here some ill-posed problems.

First, the initial-transient problem is well known, fundamental to estimating steady-state performance. Because the model's initial state cannot be chosen from the steady-state distribution, $Y_1, Y_2, \ldots, Y_{d^*}$ are not from the steady-state distribution; in many models the value of d^* must be quite large to ensure that the data are, practically, steady state. The issue is that a point estimator based on initial-transient data will be biased. The classic solution is to warm up the simulation experiment by discarding some initial data $Y_1, \ldots, Y_d$. The ill-posed problem is to determine a good value of d.

The initial-transient problem is ill posed because almost without exception authors providing methods for determining d do not state the nature of a good solution. The problem revolves around bias in the point estimator, but choosing d to minimize bias would set $d = n - 1$ if θ is a mean. A reasonable objective would be to minimize the mean squared error (mse) of the point estimator. No method has been proposed to minimize mse, probably because estimating bias is difficult if not impossible. Because the problem is ill posed, algorithmic methods remain heuristic, continue to be produced, and continue (often) to fail.

Second, a confidence interval is the goal set by many researchers in estimating sampling error. Confidence intervals are widely studied in first-year probability and statistics, so the goal seems reasonable. Researchers regularly suggest algorithms for computing confidence intervals.

The confidence-interval problem is ill posed because with only few exceptions authors providing methods for computing confidence intervals do not pursue an objective. Nominally, the stated—and defining— objective is that the interval should cover θ with confidence $1 - \alpha$, where α is the probability that θ is not included. In batching methods, blindingly pursuing this objective leads to one degree of freedom, with corresponding confidence-intervals lengths with large mean and variance. Procedures with such long, unstable intervals are generally considered to be bad, despite their good coverage probability. Sometimes a researcher will claim that a procedure works well because the coverage probability is greater than $1 - \alpha$, but if greater coverage probability is the goal and interval length is unimportant, then the best interval is $(-\infty, \infty)$, a trivial unacceptable solution.

Fundamentally, developing confidence-interval procedures is a multiple-objective problem. Procedures are developed and the objectives estimated for various scenarios, with the result usually being ambiguous because of the multiple objectives. A well-posed formulation might involve determining an efficient frontier of solutions.

For several reasons, there is little need to pose the confidence-interval problem well. First, as with hypothesis testing, the choice of α is usually arbitrary. Second, there is insufficient room for commercial software to report lower bound, upper bound, and point estimator for each θ, because there are often tens or hundreds of performance measures. Third, practitioners (in general, not just simulation) widely misunderstand the meaning of a confidence interval, wanting to conclude that the probability that θ lies within the *observed* interval is $1 - \alpha$.

An alternative to the ill-posed confidence-interval problem is to pose the problem as minimizing the mse of the estimated standard error. Such a problem has no magic parameter such as α, requires little room to report, and is difficult to misunderstand. (Some practitioners might not understand the estimated standard error, but they will not misunderstand it.)

2.9 Optimization

Simulation optimization is searching for an optimal design given only the ability to perform a simulation experiment at any design point, a performance measure to optimize, and a feasible region, which might involve the design parameters or other performance measures. Because the simulation at any design point has a finite sample size, the algorithm has available only point estimates of the various performance measures, as well as associated standard errors. *Prospective* algorithms mimic deterministic algorithms by examining one design point at a time; *retrospective* algorithms search many design points using common random numbers before moving to new random numbers.

An error, common to much published research, is to assume that "observation" is a well defined term. In simple distribution-sampling experiments, the natural definition is the vector of values returned from a single trial, for example in a reliability simulation, a zero or one indicating whether the system failed or worked. One could, of course, define one observation to be the average of five, or ten, single-trial observations. Similarly, in a steady-state simulation experiment, an observation is completely arbitrary, possibly corresponding to an hour, a shift, a day, a year, or a century. Algorithm performance can differ dramatically based upon the definition of "observation".

A second error is to assume that the point estimator should have the same standard error, or be based on the same number of observations, throughout a prospective algorithm's progress through design points. A bit of thought shows that small sample sizes are appropriate when far from the optimal solution, whereas large sample sizes are required when near the optimal solution. Similarly, retrospective algorithms should be based on monotonically increasing sample sizes.

A myth is that a limit theorem (guaranteeing convergence in some sense) implies good, or even adequate, performance. Certainly having a limit theorem is better than not having one, but sometimes ideas for algorithm improvements are discarded because the resulting logic complication would destroy the convergence proof.

The converse myth is that a method without a convergence proof is not useful. Various versions of the Nelder-Mead algorithm provide counter examples.

2.10 New Experiment

We take as the *original experiment* any experiment—input model, logic model, sampling procedure, and point estimator—that the practitioner creates to estimate a performance-measure value θ. We take as the *new experiment* any other experiment whose purpose is to estimate θ; any or all of the four experimental components might differ from the original experiment.

The purpose of substituting a new experiment for the original experiment is efficiency, which has three components: (1) human effort, (2) computer effort, and (3) quality of the point-estimator distribution. This substitution is usually referred to as *variance-reduction* (but there is no "V" in the word "simulation").

The first error, which is minor because it is only a terminology issue, is confusion about which variance is being reduced. The variance is that of (each component of) the point estimator. For example, a variance of the output-data distribution, which is a function of the input and logic model, is a possible performance measure, not a measure of experiment efficiency.

The second error is to think that variance reduction has been achieved because the standard-error estimate from the new experiment is smaller than that from the original experiment. This error arises when the practitioner assumes that the estimated standard error is essentially equal to the true standard error.

The third error, the most important, is to mistakenly substitute an experiment whose performance measure is not θ. Such a substitution occurs, for example, when a practitioner mistakenly changes the input model to use only mean values; the new experiment then has a biased, zero-variance point estimator.

The fourth error, also quite important, is the chance that the new experiment is incorrect because of implementation error. Typically the new experiment is more complex than the original experiment, which is usually based on simply mimicking real-world behavior. For example, con-

trol variates involve more-complicated point estimation and stratified sampling involves more-complicated sampling.

A myth, widespread among thesis students but not among real-world practitioners, is that a sophisticated experiment is more desirable than a simple experiment. Reducing point-estimator variance by increasing sample size should be the first-thought strategy.

ACKNOWLEDGMENTS

I thank Michael R. Taaffe and Laurel Travis for inviting me to write this paper, which is based on various of my presentations with a similar theme. The ideas and opinions within this paper grew from interactions with people within the Monte Carlo simulation community. Although they might not agree with various points in this paper, David Goldsman, Alan Pritsker, Lee Schruben, Mike Taaffe, and Jim Wilson, as well as all of my thesis-level students, both former and current, have provided many stimulating and thoughtful discussions.

AUTHOR BIOGRAPHY

BRUCE W. SCHMEISER is a professor in the School of Industrial Engineering at Purdue University. He received his Ph.D. from the School of Industrial and Systems Engineering at Georgia Tech in 1975; his undergraduate degree in the mathematical sciences and master's degree in industrial engineering are from The University of Iowa. His interests lie in applied operations research, with emphasis in stochastic models, especially the probabilistic and statistical aspects of stochastic simulation. He is an active participant in the Winter Simulation Conference, including being Program Chair in 1983 and chairing the Board of Directors during 1988–1990.