"Some Tactical Problems in Digital Simulation" for the Next Ten Years

Barry L. Nelson
Northwestern University
Lancaster University Management School

October 30, 2015

Abstract

In his influential 1963 paper "Some Tactical Problems in Digital Simulation," Conway identified important issues that became the pillars of research in simulation analysis methodology. Naturally these "problems" were a product of the applications of interest at the time, as well as the state of simulation and computing, much of which has changed dramatically. In light of those changes, we attempt to identify the tactical problems that might occupy simulation researchers for the next ten years.

1 Introduction

Nance and Sargent (2002) credit Conway (1963) with creating the stochastic simulation research area we now call "analysis methodology" (AM). See also the earlier paper by Conway et al. (1959) that provided a foundation for the later work. AM addresses all of the statistical aspects of the simulation experiment that remain once we are given a valid simulation model. Conway separated tactical issues, which relate to efficiency, from strategic issues of experiment design, but AM (and this paper) encompass both. Conway recognized that design and analysis issues were important, perhaps just as important as proper construction of the simulation model which was the dominant concern at that time.

Conway (1963) was remarkably prescient. Nelson (2004) traces its influence on eight award-winning papers that appeared in *Management Science* through 1994, and it is virtually impossible to attend a conference or workshop on stochastic simulation without hearing echoes of Conway's tactical problems. Of course, Conway (1963) reflected the state of simulation practice and the type of computing that was available at that time. The reader will require no convincing that these have changed, and we argue below that they have changed in ways that raise entirely new problems and perspectives. The goal of this paper is to identify key AM problems for the next fifty years, or, more realistically, for the next ten. We

consider dynamic, stochastic simulations—discrete event or agent based—that are employed for system design, evaluation and control, but not simulation as used for training.

Our new tactical problems grow primarily out of three observations:

- Data storage is cheap and effectively unlimited, which means that we can exploit more of the simulation-generated data than we typically do today.
- Parallel simulation is becoming easy to do, and any simulation experiment that requires multiple replications or multiple scenarios can benefit dramatically from parallel simulation.
- More and more current and potential simulation users are interested in risk analysis, prediction and control, rather than in system design.

We begin in Section 2 with a brief review of the context that gave rise to Conway's tactical problems, and follow that with a discussion of the changes from then until now in Section 3. Sections 4–6 present our thoughts about how the observations above should reshape AM research and practice. We close with a look at a few other emerging tactical problems in Section 7.

2 How we got here

Stochastic simulation existed before the advent of computers, but computers made it possible to solve problems of substantial scale. Computing in the sixties, and for may years thereafter, was characterized by batch runs on single-processor machines with limited memory, slow, nonexistent, or inconvenient mass storage of results, and possibly a need to schedule time on the computer in advance. In such an environment it was natural to try to anticipate everything you might want from a simulation run, to compile performance estimates on the fly (as the simulation was running), and to program for computational efficiency. As Conway noted:

> In general, Phase 3 [tactical] involves questions of efficiency of execution. The use of very large sample sizes can overwhelm virtually all of these difficult tactical questions, but I do not believe this is a satisfactory or practical answer. Since computer time is not a free good and the running time to execute a complex simulation is disappointingly great, sample sizes tend to be rather modest even with very powerful computers (Conway, 1963, p. 48).

Early applications of simulation were often in manufacturing or telecommunication systems design, problems that had already been the focus of queueing theory. There is little doubt that queueing theory, which is generally credited to A. K. Erlang in the early 1900s, preceded and influenced the development of computer simulation in two important ways: The first, by representing manufacturing and telecommunication systems as networks of

queues, and the second, by emphasizing long-run average performance measures of stochastically stationary systems. As a result, many simulation programming languages adopted a queueing network representation of the world (e.g., GPSS, QGERT), and books emphasized techniques for simulating stationary queueing systems (e.g., Tocher 1967, which is generally considered to be the first textbook on stochastic system simulation, contains two chapters on simulating queues but only one on "general simulation problems").

Taken in this light, Conway's tactical problems are not surprising: (i) establishing equilibrium (the time it takes the model to "warm up"), and (ii) consideration of variability and sample size. Today we would recognize "equilibrium" as the steady-state simulation problem, which is the simulation version of long-run average results for mathematically tractable queues. Conway proposes one of the first data truncation rules, as well as intelligent selection of initial conditions, to address the "warm up" problem. For considerations of variability and sample size when computing resources are dear, Conway expounds on the benefits of inducing correlation between alternative scenarios via common random numbers. However, he also notes that such correlation violates the assumptions of classical analysis of variance, and therefore provides motivation for research in new ranking-and-selection procedures and multiple comparisons. Further, the need for efficient computation suggests employing a single long replication (truncating data only once), rather than multiple independent replications. This introduces a new research problem: error estimation from dependent data. Conway proposes estimation of (what we would now call) the asymptotic variance constant and the use of batching. More subtle, but also apparent in Conway (1963) and the AM literature in general, is the assumption that simulation is used primarily for performance-measure estimation in the service of system design and optimization.

We feel confident that, if counted, the number of the theses, papers, presentations and algorithms that have attacked these tactical problems would be staggering! However, the assumptions behind Conway's problems are no longer as relevant, as we elucidate next.

3 From then to now

What has changed since 1963 that should influence AM research and practice today?

On the computational side, expensive and slow storage of data (think magnetic tape) has been replaced by effectively unlimited storage that is also cheap, and in the case of solid-state storage, amazingly fast. Dramatically increased data storage has been a driver of big data analytics, whose motto could be "collect and save *everything*, then figure out what is useful later." The value of saving everything is that when you have enough of "everything" then you can derive fine-grained, *conditional* statements, such as what product to recommend to someone whose home is in A, gender is B, has purchased C, D and E in the past, and has a projected income of F. It is worth noting that many of the tools for big data analytics were initially developed outside of the field of statistics while statisticians remained committed to the classical paradigm of squeezing the maximum possible information out of a small sample of expensive data (Efron, 2010). This should be a cautionary tale for AM researchers. We examine the implications of retaining the entire simulated sample path in Section 4.

Also on the computational side we have gone from expensive batch runs on a single-processor computer to personal computing that is virtually free in terms of cost, although not in terms of time. This has had an obvious impact on simulation language interfaces and has made animation of simulation runs routine. Nevertheless, because simulation runs do take time, the efficient-sequential-experiment-on-a-single-processor mindset is still pervasive in AM research; two- and four-core personal computers have not yet provided enough parallelism to cause substantial rethinking. On the horizon, however, is the relatively seamless renting of thousands of processors through service providers. This will flip the time vs. cost tradeoff, making the simulation time required for many applications negligible, but at a financial cost. In Section 5 we consider the tactical problems that this creates for AM.

There has also been a change in who uses simulation, and for what it is used. At the time of Conway (1963) the simulation user was a rare bird who had probably invested significant time and effort in learning how to build and execute simulation models. Simulation was not a general-use tool. In fact, access to simulation software was not universal, and many if not most simulations were programmed in lower-level languages, necessitating additional expertise. Simulation is now taught to undergraduate engineers and management scientists the world over, many of whom could not imagine programming a simulation in a lower-level language. Further, they are applying simulation in the widely diverse fields in which they are employed, and simulation is just one of many modeling and analysis tools that they know how to use, but perhaps not so much about how they work.

At the same time, an interconnected world is pushing operations research in general, and simulation in particular, to address systems-of-systems problems that are large in scale, broad in scope, complex in behavior, and have multiple competing objectives. Further, we have learned from financial crises, if nothing else, that the quantification of risk is often more important for day-to-day operation and decision making than long-run average performance. In fact, reaching a long-run average may be considered a rare event as real-world systems have to evolve and adapt rapidly to be competitive. In Section 6 we speculate about how AM research could support decision making for the users and applications described here.

4 Simulation analytics

Big data analytics are used to explore vast quantities of transactional data to discover anticipated and unanticipated relationships, with a focus on conditional statements, prediction and understanding. Dynamic stochastic simulations, both discrete-event and agent-based, generate large quantities of *synthetic* transactional data, but what simulation users typically do with the data is entirely different: (i) They do not retain the simulated inputs and state changes, making it difficult to find relationships with the outputs; (ii) they average everything across time, masking time-dependent, dynamic effects; (iii) they create high-level summaries, making it difficult to evaluate risk or predict actual system behavior; and (iv) they assume that every performance measure or relationship of interest can be anticipated *before* running the simulation, rather than being *discovered* afterwards. We believe that the overwhelming success of data analytics in business and industry will lead simulation users

to expect the same sort of fine-grained, conditional analysis from their simulations, which suggests that there is a need for a field of *simulation analytics.*

Simulation analytics, as we envision it, is more than just "saving all the simulation data" and then applying modern data-analysis tools (although this capability alone would be beneficial). Instead, simulation analytics research should address and exploit the differences between simulation-generated data and real-world transactional data, as well as the differences in problem context that necessitate a simulation as opposed to a field data analysis. Some features that distinguish the simulation analytics context from the big data analytics context are the following:

- Simulation data are clean, complete, and (could be) conveniently organized (but, we contend, should not be summarized) as generated. Further, within computational budget constraints, the quantity of data is under our control.

- The probability models that describe the underlying stochastic inputs are *known* and also under our control, as are the random-number assignments that generate realizations.

- The logical structures that cause state transitions (e.g., entity networks or agent rules) are also *known*, and can be manipulated.

- Simulations generate sample paths, as opposed to transaction instances, that are (often) nonstationary and strongly dependent; further the outputs are more often ratio (real valued, measured from zero), rather than attribute or categorical data.

- Alternative scenarios are simulated, often to optimize the performance of systems that do not yet exist. Thus, simulations produce multiple sample paths from within each alternative (i.e., replications) and across different alternatives. These need to be stored, analyzed and compared.

As noted in Section 2, the focus of AM research and practice has been on precise, efficient estimation of system performance measures. Simulation analytics could provide a more comprehensive view of system performance by facilitating a more complete understanding of the system behavior that might actually occur if a simulated system design is implemented. Simulation is often touted as a "what if?" tool because any change in system design that can be conceived can be simulated. Simulation analytics could add a "what matters?" and "how will it behave?" capability by looking more deeply at the simulated sample paths. We could ask questions such as, what system conditions or behavior predict simulated periods of poor system performance, and what conditions seem to mitigate them? This moves simulation beyond system design and toward gaining insight into how to manage and control the system without having to anticipate all possible management or control actions up front and then simulate them as "what if's?"

Here are five general objectives for simulation analytics:

1. To generate dynamic conditional statements: relationships of inputs and system state to outputs; and outputs to other (possibly time-lagged) outputs.

2. To generate dynamic distributional statements: full characterization of the observed output behavior, marginally at a point in time, and dynamically across time.

3. To generate statements on multiple time scales: high-level aggregation and drilling down to individual event times.

4. To generate comparative statements: a comprehensive understanding about how and why alternative system designs differ, and how they will behave if implemented.

5. To generate inverse conditional statements: relationships of outputs to inputs or the system state.

We provide an example of #2 below.

How does an analytics view change simulation experiment design? Most simulation textbooks contain advice that goes something like this: Before you build a simulation model, or design an experiment on that model, identify all of the questions that the simulation study is required to answer. This fixes the level of detail of the model, the performance measures that should be instrumented into the model, and the precision that needs to be achieved. Such advice makes sense when we think of the simulation as a time-consuming way to optimize system performance.

However, when generating and saving simulated data is fast and cheap, a generic objective can be to explore the data to find relationships that might be useful for system control, to understand how the system will actually behave over time, or to assess risk. The simulation then becomes a way to fill a database with many detailed sample paths, paths that can be explored post-run using a variety of tools. Viewed in this way, we might want the simulation to include more details than are strictly needed to answer specific design questions, and we might simulate a host of scenarios that fill some feasible system-design space, rather than directly optimizing. Of course this may require more model building effort. After learning from this data, additional simulations could be targeted to achieve a more narrow objective, attain a specific level of precision, test system control strategies, or optimize. In AM we are unaccustomed to thinking about designing simulations for this purpose.

To facilitate simulation analytics, some hard thinking about what it means to "save the entire sample path" will be required. For some thoughts on this from as far back as the 1980's see Pritsker (1990) Chapter 7.3 and references therein. We do not expect the *amount* of data to be the bottleneck as even complicated simulations generate far less than what is considered "big data" today. However, the desire to recreate and explore every aspect of a high-dimensional, time-dependent, dynamic process may make some strategies better than others. The generalized semi-Markov process view of discrete-event simulation (see, for instance Glasserman 1991) might be a good place to start.

Here is an example that illustrates the change in perspective relative to #2 above: Smith and Nelson (2015) consider the problem of estimating a summary measure (e.g., mean) of

the virtual waiting time of a customer arriving at time (or a collection of times) t to a complex queueing system with a time-varying arrival rate. By "virtual waiting time" they mean the waiting time a customer would expect to experience if they arrived at time t, an event that may never actually happen even in multiple replications of the simulation. They approach the problem by defining time buckets (e.g., hourly), and instrumenting the simulation to accumulate and calculate statistics for waiting times of customers who arrive during the bucket, rather than by the order in which they depart. The focus of the paper is on assessing the error in the virtual waiting time estimates, but the point of relevance here is that the approach to modeling and experiment design is traditional: the performance measures are known in advance, the time buckets are prespecified, and the statistics are accumulated dynamically as the simulation runs, giving point and error estimates at the end.

Clearly if detailed sample paths of all customer arrivals from all replications had been retained then precisely the same analysis could be achieved; but so much more is possible. The time-bucket size could be adjusted to deal with the bias-variance tradeoff, and even varied throughout the simulated day to reflect the intensity of arrivals. If excessive waiting times for arrivals around time t were observed, then they could be decomposed by customer and by stage of the service process to identify the pinch points. Is the arrival rate around time t the only reason for the long delays? This question could be explored by looking for other predictors of long delay. And, of course, the entire distribution of virtual waiting time could be estimated, not just a predetermined summary measure, for any time t we desired.

The analysis described above is not straightforward, since it deals with multivariate, time-dependent, highly correlated outputs across multiple replications. But it is a problem of significant practical importance: when one goes to the airport for a 10 AM flight the long-run average check-in time matters not at all, but the 99th percentile of my check-in time is very relevant. A simulation analytics perspective opens the door to a more complete understanding of system dynamics through time, but we need appropriate output analysis tools to estimate and explore them.

5 Parallel simulation

There are at least three reasons to perform a stochastic simulation experiment:

Feasibility: We have a plan, will it work? "Work" could mean produce enough profit, complete a project before a deadline, or be able to serve 80% of the demand in less than one week, for instance.

Sensitivity: We are not really sure about everything, so how much does it matter? The things we are not sure about could include environmental conditions, the validity of our input models, or how well we can actually set the control parameters.

Optimization: What are the good options, and how good are they? This category includes any situation that requires comparing simulated alternatives, whether we need to find

a unique optimal or not.

How are these three tasks affected by the availability of parallel simulation, meaning the capability to execute multiple runs simultaneously? For the discussion in this section, we suppose that in each case the problem would be solved if we knew a performance parameter $\theta(\mathbf{x})$, possibly as a function of parameters/decision variables/environmental conditions $\mathbf{x}$. We call $\mathbf{x}$ a scenario. Simulation can generate an estimator $\widehat{\theta}(\mathbf{x}; T, n, \mathbf{U})$ that is itself a function of three additional quantities:

- A stopping time, T, that is the simulation clock time when a replication ends; it might be fixed or random, part of the definition of the problem or part of the experiment design.
- A number of independent and identically distributed (i.i.d.) replications, n, which might also be fixed, but could be random if $\widehat{\theta}$ is created by a sequential procedure.
- The pseudorandom numbers, $\mathbf{U}$, which we think of as both random—the reason why $\widehat{\theta}(\mathbf{x}; T, n, \mathbf{U})$ is treated as a random variable—but also as part of the experiment design because they are in fact pseudorandom and therefore can be assigned.

The only thing we assume about the estimator is that for any fixed $\mathbf{x}$, the quality of $\widehat{\theta}(\mathbf{x}; T, n, \mathbf{U})$, as measured by mean squared error (MSE), improves as the number of replications n increases and as the run length T increases, when T can be treated as a design variable. For instance, in the steady-state simulation setting increasing n reduces variance, while increasing T reduces variance and bias.

Whether checking feasibility, assessing sensitivity or optimizing, it is the nature of simulation to solve the problem by choosing instances of $(\mathbf{x}, T, n, \mathbf{U})$ and executing them. Feasibility focuses on a specific scenario $\mathbf{x}_0$, so the experiment design decisions are n and possibly T; we only require that $\mathbf{U}$ come from a pseudorandom number generator that has good properties. Sensitivity may also focus on a single $\mathbf{x}_0$—if $\widehat{\theta}(\mathbf{x}; T, n, \mathbf{U})$ is an estimator of the gradient with respect to $\mathbf{x}$, for instance—or it might require simulations of scenarios $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k\}$ if we are interested in more than just local sensitivity. Whenever there are multiple $\mathbf{x}$'s then the choice of $\mathbf{U}$ comes into play because we have the option of assigning distinct or common $\mathbf{U}$'s to different scenarios. Optimization always involves simulating a collection $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_K\}$, but K might be random if we are searching over a large scenario space that we cannot exhaust.

The computing paradigm within which AM tends to think about solving feasibility, sensitivity and optimization problems is one in which instances $(\mathbf{x}_i, T_i, n_i, \mathbf{U}_i)$ are executed one, or a few, at a time until the problem is solved using processors that are effectively free. Since *time* is the consumed quantity, this paradigm puts a premium on reducing the number of scenarios $\mathbf{x}$ that need to be simulated as well as the effort (n, T) expended on them. There will always be problems for which this approach makes sense. However, we believe that computing with the characteristics described below will become increasingly accessible and affordable:

- Instances $(\mathbf{x}_i, n_i, T_i, \mathbf{U}_i)$ can be conveniently, even automatically, assigned to distinct processors, and the number of available processors, P, will be large.

- The monetary cost of renting one processor to simulate an instance $(\mathbf{x}_0, n_0, T_0, \mathbf{U}_0)$ will be almost identical to the cost of renting n_0 processors to simulate instances $(\mathbf{x}_0, 1, T_0, \mathbf{U}_j)$, $j = 1, 2, \ldots, n_0$ in parallel.

- *When no coordination is required and n_0 is less than the number of processors* P, the time to simulate $(\mathbf{x}_0, n_0, T_0, \mathbf{U}_j)$ on a single processor will be approximately $n_0\times$ the time to simulate all $(\mathbf{x}_0, 1, T_0, \mathbf{U}_j)$, $j = 1, 2, \ldots, n_0$ in parallel.

- For applications that require multiple replications or scenarios, the overhead to distribute and coordinate a single instance $(\mathbf{x}_0, 1, T_0, \mathbf{U}_j)$ across multiple processors will typically make that approach less attractive than distributing whole instances to distinct processors.

The tactical implications of this sort of computing environment are profound, and should drive research and practice in new ways.

The most basic question is: how should multiple processors be deployed? The answer depends on, among other things, whether there are greater or fewer processors P than scenarios k to be simulated. As the number of processors that one can rent grows (and we are aware of services that can provide 10,000), then there will be problems for which $\mathsf{P} \gg k$. If, in addition, $\mathsf{P} \geq kn$, where n is the average number of replications needed, then the time to solve many feasibility and sensitivity problems, and some small- to medium-scale optimization problems, is effectively the time to execute one replication of one scenario! This happy situation opens up a host of new applications for simulation, including using it for real-time decision support, a topic that has attracted little attention in AM.

However, all is not quite as simple as we just made it appear. Achieving the maximal speedup means avoiding coordination. Therefore, the simulation effort (n_i, T_i) that is needed for scenario $\mathbf{x}_i$ to obtain acceptable MSE must be known up front; otherwise some sort of sequential, coordinated experiment is required. This suggests that methods for robust *a priori* planning—as in Whitt (1989, 2006) for queueing simulation—or for planning based on results from previous similar experiments (see, for instance, Feng and Staum 2015), will become more important. *Stated differently, there will be applications for which the practical benefits of being able to execute the entire experiment using only one pass through the* P *processors will make it more valuable to be able to predict in advance an (n_i, T_i) that is large enough, rather than sequentially finding an (n_i, T_i) that is no larger than necessary.*

When $\mathsf{P} \gg k$, for instance when there is only $k = 1$ scenario, it seems clear that single-replication experiment designs, or designs for which the next simulation run depends strongly on completion of the previous one, will rarely make sense. For example, in steady-state (equilibrium) simulation problems the standard recommendation of $n = 1$ and T as large as possible—which goes back at least as far as Conway (1963)—looks foolish. Why idle so many processors? However, the reason that small n and large T is recommended is that we have not been as successful in solving the warm-up problem as we have in deriving error

estimates from dependent data. Thus, we anticipate that the initialization-bias problem may re-emerge, motivating a desire for *a priori* planning tools or empirical solutions that can exploit multiple processors, as in Hsieh and Glynn (2009).

One of the most striking benefits of massively parallel simulation is that it facilitates optimization by exhausting the collection of feasible scenarios, rather than searching over a subset. This is clearly the case when the number of feasible scenarios is finite, but it also makes it attractive to solve continuous-decision-variable optimization problems by discretizing $\mathbf{x}$ and simulating all feasible combinations. Point-to-point searches that move in improving directions are unappealing for the same reason that single-replication designs for steady-state simulation are: they waste cheap parallel processing.

Unfortunately, statistical guarantees such as a lower bound on the probability of correct selection are hampered by the curse of multiplicity, which is that the computational effort required to guarantee that a statement is simultaneously true for all instances grows dramatically with the number of instances it has to cover. We will need new paradigms for statistical inference that make sense for the simulation of millions of scenarios when it is computationally possible to do so. Related problems motivated by large-scale statistical testing have also arisen (Efron, 2010).

Parallel computing breaks the traditional relationship between simulation time and the number of scenarios, number of replications and run length to simulate them; the level of coordination matters much more. Algorithms that save replications by careful coordination and synchronization are likely to give way to algorithms that perhaps waste some replications and instances rather than let processors idle. Balancing that tradeoff will be a research and practical challenge, particularly when $k \gg \mathsf{P}$.

As a case in point, consider the KN algorithm for ranking and selection (Kim and Nelson, 2001). This algorithm provides a guaranteed probability of correct selection when comparing the means of k scenarios. The algorithm coordinates in two ways: (i) after an initial run of n_0 replications from all k scenarios the sample variances of all pairs of differences are computed; and (ii) from then on, decisions about which scenarios can be eliminated are based on all pairs of differences among the sample means after obtaining a single additional replication from each scenario still in contention. In addition, a different sort of synchronization is necessary if common random numbers (CRN) are employed to insure that each scenario is assigned the same $\mathbf{U}$.

The rapid elimination of inferior scenarios, enhanced by CRN, is the reason KN is so efficient *in terms of replications consumed.* But comparisons of all pairs of differences after each incremental replication can create a significant *coordination* bottleneck: When $\mathsf{P} > k$ not all available processors will even be used; and when $k \gg \mathsf{P}$ all processors will be repeatedly idle during the nontrivial work of doing $k(k-1)/2$ comparisons. Contrast this with a simple two-stage procedure, such as Rinott (1978), that can be implemented to avoid *any* coordination at all because the number of replications generated from scenario $\mathbf{x}_i$ depends only on scenario $\mathbf{x}_i$. The penalty is that substantially more replications are needed to select the best for typical problems, which may limit the size of problem that can be solved in a timely manner, or greatly increase the cost for processors when the number of

scenarios k is huge and much larger than P.

Mitigating the coordination bottleneck in parallel simulation seems certain to be an important tactical problem going forward, but it has to be done with some care. Consider modifying a procedure like KN. As a first cut one might consider making elimination decisions using whatever outputs have been returned from the processors, rather than waiting to coordinate. Unfortunately, a pitfall affecting this sort of approach was discovered some time ago by Heidelberger (1988) and Glynn and Heidelberger (1991). They noted that there is frequently dependence between the *time* that it takes for a replication to execute and the *response value* that the simulation returns. For instance, in a queueing simulation replications for which the system is more congested will tend to execute more slowly because there are a greater number of events to execute. This implies that the simulation results from multiple processors observed in the order that they are completed are not identically distributed (and, as it turns out, not independent either). Heidelberger (1988) and Glynn and Heidelberger (1991) only considered a single scenario with replications assigned to multiple processors, but the statistical problems get worse when there are multiple scenarios (Luo et al., 2015).

The bottom line is that both computational efficiency and statistical validity play a role when deriving optimization strategies for parallel simulation. There would seem to be two general approaches:

1. Preserve the statistical validity of existing procedures by using the simulation outputs in the order that they were requested, not returned, while efficiently managing what is requested and how it is manipulated; see Ni et al. (2014).

2. Efficiently use the output data in the order that it is returned, but derive procedures that are statistically valid anyway, as in Luo et al. (2015).

Finally, it is AM folklore, again going back at least to Conway (1963), that optimization should employ CRN across scenarios to obtain improved efficiency essentially for free. However, CRN works best when all scenarios receive the same number of replications and are paired; in fact, it can be counterproductive otherwise. In algorithms for parallel optimization we may find that the benefits of CRN do not outweigh the overhead of coordination and synchronization, or perhaps new approaches can be invented that strategically combine CRN and independent replications.

6 Simulation to support decisions

Regardless of whether a simulation study is conducted to check feasibility, assess sensitivity, or to optimize, there is virtually always one or more underlying decisions motivating it. Often the simulation results will support the decision rather than produce the decision itself. But even when the simulation provides the decision—e.g., when it is a simulation optimization—additional insight about predicted system behavior or the availability of other good scenarios will aid in implementing and managing the decision. AM research and practice can, and

should, do more to support decisions for current applications and users in at least two areas: the reporting of results and the evaluation of model risk.

6.1 Reporting of results

As a research community AM has had a historical focus on measures of error, while decisions should usually be based on measures of risk. By "risk" we mean the actual dynamic behavior that might be realized, as opposed to a summary performance measure that is averaged over the detailed behavior. For instance, a supply chain replenishment policy may minimize long-run average cost, but do so by encouraging wide swings in period-by-period costs that would be intolerable in practice.

Unfortunately, users often misinterpret error as risk. A confidence interval (CI) is a measure of error, and measures of error answer the question "have we done enough simulation so that our estimate of performance is statistically good?" The width of a CI says nothing about the natural system variability; that is, it says nothing about realized behavior or risk. A prediction interval (PI) is a measure of risk; it answers the question "what performance will we most likely see when we implement the decision?" Measures of risk should be standard, but they are not. And when reporting either type of measure, statistically meaningless precision should be avoided so that it is not interpreted as meaningful. If a simulation outputs a long-run average supply chain cost of \$10,247,381.175, and this is reported, then it may be *believed* even though only the first two digits, say, are statistically meaningful.

Nelson (2008), Wieland and Nelson (2009) and Song and Schmeiser (2009) present some ideas about reporting simulation results so as to distinguish risk and error and avoid misleading precision, but they assume either risk or error is measured *across* i.i.d. replications, or *within* a replication of a stationary process (think steady-state queueing simulation). However, what if the performance we need to evaluate is indexed by time, as in the example of virtual waiting time in Section 4?

We are willing to argue that whenever a system has time-varying load, or the performance of interest is defined with respect to a finite planning horizon, then it will be more meaningful to have performance measures indexed by time, rather than averaged across time. A simulation analytics perspective would provide the necessary data, but the research challenges are in estimating and displaying measures of risk and error for time-dependent outputs. And "displaying" should not be dismissed as trivial: simulations produced high-dimensional, dependent (through time and across series) output processes, possibly from multiple scenarios, so inventing displays that capture the relevant information, yet can still be interpreted and explored, is not easy. See Luboschik et al. (2014) for one innovative idea.

A different reporting issue arises when simulations are executed in an informal, exploratory manner that is guided by the user, as opposed to following a classical experiment design. Since simulations are not physical experiments, data are often cheap, and results can be collected and displayed immediately, user-driven exploration is natural. How should we summarize a user-guided sequence of experiments on different scenarios to prevent them from missing something or being misled, as well as insuring statistical validity of conclusions?

Hong and Nelson (2007) show, in a more narrow problem, how difficult it is to maintain an overall probability of correct selection in simulation optimization when users choose the next scenario based on the results from previous runs. Tracking, guiding and warning users seems difficult, but progress in doing so would be a boon to effective simulation analysis.

A big step forward in the early development of simulation software was instrumenting it to collect standard performance measures automatically: If there is a queue in the model, then the minimum, maximum and average waiting time, queue length and server utilization will be recorded from within each replication, and then averaged across replications as well. When these pre-packaged statistics are what a user needs for their decision then they are in great shape.

However, when users want to go beyond these canned measures, as they should be encouraged to do, our software and methods are often quite demanding of them. We expect the user to know what other performance measure they want, what procedures to use, which scenarios to run, and to provide settings like run length, warm-up period, number of replications, confidence level, indifference zone, etc. This is not feasible for much of the current user population, as described in Section 3. Simulation will be applied more effectively if the user can *focus on the decision, not the method*. Stated differently, the user should tell the software what they want to know, not how to get it. Here are some examples:

- The user says, "From which of these would a small change have the biggest impact?" The software computes derivatives or does factor screening.
- The user says, "What can I promise and be right at least 85% of the time?" The software computes quantiles.
- The user says,"Which of these is related to the process slowing down?" The software reports correlations.
- The user says, "What is likely to happen if I actually implement this set up?" The software displays prediction intervals.
- The user says, "Get rid of the scenarios that are not very good." The software performs multiple comparisons or subset selection.

Software that works in this way would be a radical change in our thinking about how users interact with a simulation, but it might also be the biggest advance in the use of simulation since graphical interfaces. From an AM perspective it challenges us to create methods and procedures that have sensible and well-justified defaults, are robust and adaptive, have useful displays to aid in interpreting results, and generate warnings when they may have failed.

6.2 Model risk

The validity of any mathematical, computer or physical model is important if one is to have confidence in using it to inform decisions. See Sargent (2013) for a recent survey of tools and techniques for validating stochastic simulation models.

A simulation model is never expected to be a perfect representation of reality, so a valid model is usually defined to be one that is accurate enough for the decisions that it was created to support, which is a difficult concept to quantify. Further, even a careful validation exercise cannot hope to vet every aspect of a model, nor can it prevent "black swan" events that are outside of the experience of the simulation modeler or the data upon which the model is based. Even if a model is deemed "invalid" it may still be used if nothing better is available. In the end, validation cannot be asked to do more than provide some assurance that good modeling practices were used, that simplifications were carefully considered, and that simulated output data are plausible.

Taken collectively, we refer to the exposure due to an imperfect simulation model as *model risk*. There is increasing demand by businesses and governments that rely on models for analysis that goes beyond validation and toward quantifying as much of the model risk as possible. Knowing that a model has been "validated" is reassuring, but if you can assign a number to the model risk then you can perhaps hedge against it. Model risk management is now a staple offering of many consulting firms.

Two forms of model risk for which quantification seems promising are input uncertainty and calibration error. We address calibration later. Input uncertainty refers to the risk that is derived from using estimated input models in the simulation, where "input models" are the fully specified stochastic models from which observations are directly generated to drive the simulation; think arrival processes in queues; bed occupancy times in hospitals; and machine failures in manufacturing. By "estimated" we mean either fit to real-world data or based on subjective judgment.

For input uncertainty due to fitted distributions there is a substantial literature; see the surveys of Barton (2012) and Song et al. (2014). Much of it focuses on propagating uncertainty about the estimated distributions—a topic that is well studied in both classical and Bayesian statistics—to the output performance measures that the simulation estimates. For instance, some methods seek to inflate the standard simulation CIs so that they account for input distribution, as well as stochastic sampling, error. When the input models are simple, i.e., independent, parametric, univariate distributions, there are several useful and rigorously justified methods. Input uncertainty from multivariate input distributions, including time series, random vectors, and nonstationary arrival processes, are the next obvious topics on the horizon. And in the spirit of assessing risk rather than performance, PIs for actual performance that account for input uncertainty would be more useful than CIs in many contexts.

Less work has been done on subjectively specified input distributions; instead it is usually treated as sensitivity analysis with respect to, say, the mean of the distribution. The concern with this approach is that the mean is not the only aspect of the input distribution that matters; unfortunately, a more comprehensive sensitivity analysis to the mean, standard deviation, upper and lower bounds, certain percentiles, etc. seems cumbersome at best. Also, being sensitive does not necessarily imply that an error is likely: input uncertainty involves both sensitivity of the output to the input, and also the level of uncertainty about the input itself. Formal Bayesian treatments both in eliciting the distributions and in propagating the

uncertainty would be welcome here.

Simply quantifying input uncertainty (or model risk more generally) still leaves open the decision of whether and how to hedge against it. A direct approach is to formulate and solve the decision problem including the model risk. Expanding our optimization framework from Section 5, let the performance measure we want to maximize be $\theta(\mathbf{x};\boldsymbol{\vartheta})$, where $\boldsymbol{\vartheta}$ is the set of parameters of all of the input models. Ideally we would maximize over $\mathbf{x}$, given the true input parameters, denoted $\boldsymbol{\vartheta}^{\text{true}}$; instead, what we have is an estimate of $\boldsymbol{\vartheta}$ based on data.

Suppose that, via frequentist or Bayesian reasoning, we can define an uncertainty set $\mathcal{U}$ such that $\{\boldsymbol{\vartheta}^{\text{true}} \in \mathcal{U}\}$ with high probability. Then one of many input-uncertainty-robust formulations of the optimization problem is

$$\mathbf{x}^\star = \text{argmax}_{\mathbf{x}} \min_{\boldsymbol{\vartheta}\in\mathcal{U}} \theta(\mathbf{x};\boldsymbol{\vartheta}). \tag{1}$$

In words, we want the decision $\mathbf{x}^\star$ that maximizes over the worst case in the parameter uncertainty set. Of course, (1) has all of the difficulties that the generic simulation optimization problem has, plus the minimization over the uncertainty set. A critical challenge here is defining the uncertainty set in a way that is practically sensible and over which the problem can be solved. For instance, an uncertainty set that is too large could lead to decisions that are too conservative to be useful. Careful thought about meaningful formulations of simulation optimization under model risk, as well as solution techniques for those formulations, will provide useful research problems in the next decade.

A related problem that has received little attention in AM, but substantial attention in the world of deterministic computer experiments, is calibration error. One of the most-cited references in computer experiments is Kennedy and O'Hagan (2001).

We first contrast input modeling to calibration. Input modeling uses real-world input data to fit families and parameters of input distributions that are then used in the simulation. Calibration, on the other hand, adjusts simulation model parameters—which could be input-distribution or other tuning parameters—so that the simulation output closely matches the real-world system output; this is a type of inverse problem. Calibration has seemed less important because we so often optimize, meaning relative performance matters, and we tend to believe (perhaps incorrectly) that model errors affect relative performance less than absolute performance. However, as simulation plays an increasing role in assessing risk and making accurate predictions of behavior, calibration will be desirable.

Calibration is also needed when input modeling is not possible. Many systems automatically collect a wealth of process data, but these are more often output (performance) data than they are the basic input data needed to model uncertainty in the system. Even when we have a wealth of output data on the existing system, if we want to use simulation to change and improve system performance then we need input models to construct the simulation.

As a stylized illustration, consider a system that is actually an $M/M/\infty$ queue with arrival rate λ and service rate μ, but for which no input data are available to estimate λ and μ. However, real-world *output* data have been collected for the number of customers in the system Q and the number of customers departing the system per unit time, D. Since

for this system we know that, for large t, $\mathrm{E}(Q) = \lambda/\mu$ and $\mathrm{E}(D) = \lambda$, we can use the observed output data to estimate the unknown input-distribution parameters via inverse maps: service rate $\widehat{\mu} = \bar{D}/\bar{Q}$ and arrival rate $\widehat{\lambda} = \bar{D}$. In this case the input model risk comes only from the finite output sample used to estimate Q and D, because the inverse model and the input-distribution families are known.

In general neither the underlying distribution families nor the inverse model will be available, implying at least two sources of risk: the finite output sample used to estimate the input distributions, and the approximation used for the inverse model. When will invertible relations exist between measurable outputs and the unknown input parameters? What are practical tools for determining invertibility? How can approximations to the inverse be constructed? What can be said about the correctness or uniqueness of the approximations? How can we incorporate inverse model uncertainty with the other sources of uncertainty to characterize overall model risk? These are difficult questions that need answers if we want to use calibrated simulation models. For a neat approach that applies to a more realistic queueing example see Goeva et al. (2014).

7 Other emerging tactical problems

Our approach has been to look at three trends that affect more than simulation—cheap storage of big data, parallel, cloud computing, and a societal need for risk analysis, prediction and control—and think through the broad simulation design and analysis issues that they imply. In this closing section we describe a few other tactical problems that seem likely to be important, but that do not fit as neatly into our framework.

Our vision for simulation analytics might also be expressed as "data analytics for systems that do not yet exist," since we suggest applying the philosophy of field data analytics to synthetically generated sample paths. We hinted that a detailed analysis of simulated sample paths could be useful for suggesting system control strategies, but that is not the same as optimal control, since it is passive observation. Dynamic policy optimization is a weakness for simulation methodology; we are much better at optimizing static assignments, like allocating capacity to manufacturing cells, staff to work shifts, and beds to hospital clinics. The methodology we have developed is not effective for state-dependent policy optimization of complex systems. Breakthroughs in this area would greatly extend the reach of stochastic simulation.

There are two other ways in which we touch, or are touched by, the data analytics trend.

- The standard input modeling paradigm in stochastic simulation is to (a) collect input data; (b) fit many candidate distributions using something like maximum likelihood; (c) evaluate using goodness-of-fit tests and conformance plots; and (d) select the "best fit" for use in the simulation. This makes sense when data are scarce or expensive, but probably does not when we have thousands to millions of observations; in fact, a goodness-of-fit test is likely to reject any standard distribution when we have enough data since real data do not come from distributions. Reusing the data themselves seems

very attractive, particularly when the inputs are multivariate or time-dependent, but resampling, say, 30 empirical distributions each of 40 million data points is probably unworkable. This suggests that we need more parsimonious representations of big input data that do not smooth out the unusual or rare features that really exist, but facilitate use in a simulation.

- By the very nature of the tools and models that are used, machine learning with big data tends to be much better at interpolation than extrapolation. That is, the tools do well in predicting properties of new cases whose attributes are within the space of observed attributes that trained the tool. Simulation can perhaps work in concert with machine learning for extrapolation from big data.

Of course, there will be a host of AM problems that we have not anticipated, including some that may refine our understanding of the foundations of simulation itself. In fact, work at the interface of simulation and applied probability is thriving, and our bet is that some of the most exciting, and (we hope) most practically important tactical issues a decade from now will arise there. In any event it seems likely that the problems we want to solve will continue to outpace the computing and analysis tools we have available to solve them, providing compelling challenges for AM for some time to come.

Acknowledgements

While the author bears sole responsibility for the opinions expressed in this article, he received insight and feedback from discussions with Russell Barton, Shane Henderson, Jeff Hong, Emily Lada, Yujing Lin, Dennis Pegden, Justice Pettigrew, Eunhye Song, Jeremy Staum and Dave Sturrock. This research was partially supported by the National Science Foundation of the United States under Grant Number CMMI-1537060. Portions of Section 4 were based on the grant proposal, and portions of Section 6 were based on Taylor et al. (2013).

References

Barton, R. R. (2012). Tutorial: Input uncertainty in outout analysis. In *Proceedings of the 2012 Winter Simulation Conference*, pp. 1–12. IEEE.

Conway, R. W. (1963). Some tactical problems in digital simulation. *Management Science 10*(1), 47–61.

Conway, R. W., B. M. Johnson, and W. L. Maxwell (1959). Some problems of digital systems simulation. *Management Science 6*(1), 92–110.

Efron, B. (2010). *Large-Scale Inference: Empirical Bayes Methods for Estimation, Testing, and Prediction*. Cambridge University Press.

Feng, M. and J. Staum (2015). Green simulation designs for repeated experiments. In *Proceedings of the 2015 Winter Simulation Conference*, pp. in press. IEEE.

Glasserman, P. (1991). *Gradient Estimation via Perturbation Analysis*. Springer Science & Business Media.

Glynn, P. W. and P. Heidelberger (1991). Analysis of parallel replicated simulations under a completion time constraint. *ACM Transactions on Modeling and Computer Simulation 1*(1), 3–23.

Goeva, A., H. Lam, and B. Zhang (2014). Reconstructing input models via simulation optimization. In *Proceedings of the 2014 Winter Simulation Conference*, pp. 698–709. IEEE.

Heidelberger, P. (1988). Discrete event simulations and parallel processing: Statistical properties. *SIAM Journal on Scientific and Statistical Computing 9*(6), 1114–1132.

Hong, L. J. and B. L. Nelson (2007). Selecting the best system when systems are revealed sequentially. *IIE Transactions 39*(7), 723–734.

Hsieh, M.-H. and P. W. Glynn (2009). New estimators for parallel steady-state simulations. In *Proceedings of the 2009 Winter Simulation Conference*, pp. 469–474. IEEE.

Kennedy, M. C. and A. O'Hagan (2001). Bayesian calibration of computer models. *Journal of the Royal Statistical Society, Series B, Statistical Methodology 63*(3), 425–464.

Kim, S.-H. and B. L. Nelson (2001). A fully sequential procedure for indifference-zone selection in simulation. *ACM Transactions on Modeling and Computer Simulation 11*(3), 251–273.

Luboschik, M., S. Rybacki, F. Haack, and H.-J. Schulz (2014). Supporting the integrated visual analysis of input parameters and simulation trajectories. *Computers & Graphics 39*, 37–47.

Luo, J., L. J. Hong, and B. L. Nelson (2015). Fully sequential procedures for large-scale ranking-and-selection problems in parallel computing environments. *Operations Research*, in press.

Nance, R. E. and R. G. Sargent (2002). Perspectives on the evolution of simulation. *Operations Research 50*(1), 161–172.

Nelson, B. L. (2004). 50th anniversary article: Stochastic simulation research in Management Science. *Management Science 50*(7), 855–868.

Nelson, B. L. (2008). The MORE plot: Displaying measures of risk & error from simulation output. In *Proceedings of the 2008 Winter Simulation Conference*, pp. 413–416. IEEE.

Ni, E. C., S. G. Henderson, and S. R. Hunter (2014). A comparison of two parallel ranking and selection procedures. In *Proceedings of the 2014 Winter Simulation Conference*, pp. 3761–3772. IEEE.

Pritsker, A. A. B. (1990). *Papers, Experiences, Perspectives*. Systems Publishing Corporation.

Rinott, Y. (1978). On two-stage selection procedures and related probability-inequalities. *Communications in Statistics-Theory and Methods 7*(8), 799–811.

Sargent, R. G. (2013). Verification and validation of simulation models. *Journal of Simulation 7*(1), 12–24.

Smith, J. S. and B. L. Nelson (2015). Estimating and interpreting the waiting time for customers arriving to a non-stationary queueing system. In *Proceedings of the 2015 Winter Simulation Conference*, pp. in press. IEEE.

Song, E., B. L. Nelson, and C. D. Pegden (2014). Advanced tutorial: Input uncertainty quantification. In *Proceedings of the 2014 Winter Simulation Conference*, pp. 162–176. IEEE.

Song, W. T. and B. W. Schmeiser (2009). Omitting meaningless digits in point estimates: The probability guarantee of leading-digit rules. *Operations Research 57*(1), 109–117.

Taylor, S. J., S. Brailsford, S. E. Chick, P. L'Ecuyer, C. M. Macal, and B. L. Nelson (2013). Modeling and simulation grand challenges: An OR/MS perspective. In *Proceedings of the 2013 Winter Simulation Conference*, pp. 1269–1282. IEEE.

Tocher, K. D. (1967). *The Art of Simulation*. London, U.K.: English Universities Press.

Whitt, W. (1989). Planning queueing simulations. *Management Science 35*(11), 1341–1366.

Whitt, W. (2006). Analysis for design. *Handbooks in Operations Research and Management Science 13*, 381–413.

Wieland, J. R. and B. L. Nelson (2009). How simulation languages should report results: A modest proposal. In *Proceedings of the 2009 Winter Simulation Conference*, pp. 709–715. IEEE.