Proceedings of the 2008 Winter Simulation Conference
S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, J. W. Fowler eds.

ANALYTICAL SIMULATION MODELING

Lee Schruben

Industrial Engineering and Operations Research
University of California, Berkeley
Berkeley, CA 94720, U.S.A.

ABSTRACT

Simulation modeling methodology research and simulation analysis methodology research have evolved into two nearly separate fields. In this paper, ways are shown how simulation might benefit from modeling and analysis becoming more closely integrated. The thesis of this paper is that simulation analysis and simulation modeling methodologies, considered together, will result in important advancements in both. Some examples demonstrate how dramatically more efficient discrete event simulation models can be designed for specific *analytical* purposes, which in turn enable more powerful analytical procedures that can exploit the special structures of these models. A series of increasingly difficult analytical problems, and models designed to solve them, are considered: starting with simple performance estimation, and progressing to dynamic multiple moment response surface meta-modeling.

1 INTRODUCTION

Simulation modeling and simulation analysis have diverged. Most university courses and textbooks focus mainly on either simulation modeling or on simulation analysis. Undergraduate simulation courses are primarily software-based, where the modeling approach and analysis are limited by the language used. On the other hand, many advanced PhD simulation courses are essentially model free. ("Graduate students should be able learn a simulation language on their own.") At major conferences, simulation modeling and analysis tend to have distinct, conflicting tracks with little overlap in the participants. There are even distinct conferences focusing on one aspect of simulation or the other. See, for examples, the Simulation Solutions Conferences, `<www.simsol.org>`, and the I-SIM conference, `<www.insead.edu/issrw/>`.

In simulation analysis research papers, there is the implication that the data are generated by an actual simulation program. Other than tacitly presuming simulations can produce arbitrary, unlimited amounts of relatively cheap data, little further consideration is given to the details of the simulation model's design or to the computer program that would be supplying the data. The work required to produce the output used in a simulation analysis procedure is typically measured only coarsely by sample size: the number of replications, number of observations in a run, number of regenerative cycles, etc.

Simulation modeling decisions can play a significant role in the performance of analytical procedures. How a simulation model is designed and coded can enable, inhibit, or even invalidate analytical procedures and methodology research results.

Computer simulation is among the most widely used engineering and scientific methodologies; however, much of simulation's use is in qualitative applications involving animations, graphics, and "what-if" scenario studies. Developing quantitative analysis methodologies, explicitly in the context of discrete event simulation models, presents new opportunities for meaningful research and more efficient modeling.

This paper is motivated in part by a long-term concern among simulation analysis researchers that new methodologies have not been widely applied in practice. More compelling demonstrations are needed on the value of new simulation analysis methodologies to simulation software vendors who control the adoption of simulation analysis research results.

There have been numerous sessions at national conferences on the disconnection between simulation research and simulation practice. Two such panels, a decade apart, are (Glynn. et. al. 1995) and (Andradottir et. al. 2005). As expected from the earlier panel, the direction of discrete event simulation software development has been on animation at the expense of analysis. Excellent insight into simulation software development is in the recent paper by Pidd and Carvalho (Pidd and Carvalho 2006) where they coin the phrase "package bloat", to describe modern commercial simulation/animation software products.

2 THREE MODELS

Three G/G/s simulation models with different analytical properties are used to provide the context for this paper. These models are defined using the language of Event Relationship Graphs (ERGs) (Schruben 1983). ERGs are concise, completely general abstractions of causality in stochastic discrete event system models that are independent of any particular simulation language or modeling "world view" (Savage et. al. 2005). Each node in an ERG simulation model represents the state changes that might happen when an event occurs. Directional arcs in an ERG explicitly specify at what times and under what conditions an event might cause another event to occur.

2.1 Modeling the Q(t) process

A simulation model for the number of jobs in a G/G/s system at time *t*, *Q(t),* is shown in Figure 1. This model will be referred to as ERG_1.

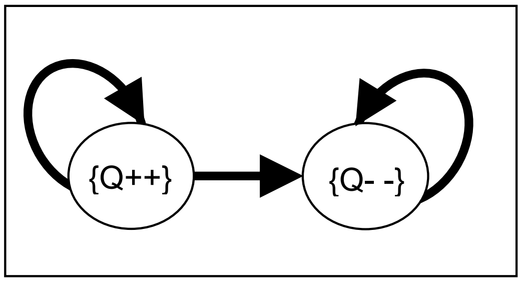

Figure 1: ERG_1 for the Q(t) process of a G/G/s system

Figure 1 uses a simplified ERG notation where bold arcs represent time delays between event executions, and thin arcs (none are used here) will represent instantaneous conditional event sequences (Law 2006). In Figure 1, the event node on the left increments *Q(t)* when a job enters the system and the event node on the right decrements *Q(t)* when a job finishes. In this simple model, the two event nodes are labeled with their state changes. (C/Java/Matlab syntax is used throughout this paper.)

Simulation model ERG_1 is small and fast. The system state is the single integer, Q. More importantly, the execution speed of ERG_1 is insensitive to queue congestion; the model runs at the same speed if the system has 10 jobs or 10 million jobs. However, the execution speed of ERG_1 slows inversely with the number of busy servers. The simulation model, ERG_1, is appropriate for simulating the *Q(t)* process with non-stationary, heavy, or unstable traffic.

2.2 An alternative Q(t) model

Several simulation texts present ERG_2 in Figure 2 as an alternative simulation model for a G/G/s queue process Q(t) (Seila, et. al. 2003, Law 2006). Figure 2 does not use the ERG shorthand in Figure 1, but fully specifies the relationships among the three events and two difference equations in this discrete event simulation model, including delay times and arc conditions (/) that must be true to schedule events. An initial value of Q (here assumed zero) along with the input processes {t_A = interarrival times} and {t_S = service times} are a complete description of the dynamic behavior for this simulated system.

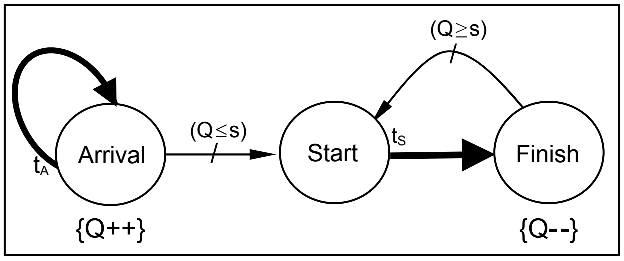

Figure 2: ERG_2 for the Q(t) process of a G/G/s system (t_A and t_S are the interrival and service time input processes)

It is generally accepted that the *Start* service event in ERG_2 is worse than superfluous. It does not change *Q(t)*, but requires the scheduling and execution of a another event for every job in a run. ERG_1 and ERG_2 are *behaviorally* equivalent for *Q(t)*, but ERG_2 is slower (Schruben and Yücesan 1992). However, ERG_2 is *structurally* different from ERG_2 and has an analytical advantage over ERG_1 for studying job delays.

2.3 Modeling job delays, {Di}

An important weakness in simulation model ERG_1 is that it does not simulate the job delay process, {D_i}. Simulating job delays usually requires much more work than simulating *Q(t)*. This is because information on job delays is typically recorded and stored for an interval of time, while queue sizes can be observed instantaneously.

ERG_1 is easily modified to simulate job delays directly using a conventional priority queue abstract data type with operations INSERT (here called PUT) and DELETEMIN (here called GET) (Aho, et. al. 1985). The PUT operation records job arrival times and the GET operation computes each job's waiting time. PUT and GET are implemented here using functions that always return a value of 1, so the *Q(t)* process is identical to that for ERG_1. The resulting model, ERG_3, simulates both *Q(t)* and {D_i} and is shown in Figure 3.

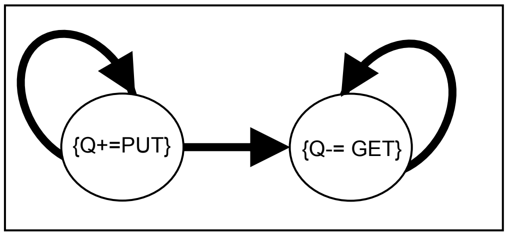

Figure 3: ERG_3 for Q(t) and $\{D_i\}$ of a G/G/s system

The execution speed of simulation model ERG_3 is, unfortunately, now sensitive to the number of jobs in the system as well as the number of busy servers. Implementations of ERG_1 and ERG_3 in Arena show that ERG_3 can take several orders of magnitude longer to simulate the same numbers of jobs than ERG_1. (ERG_1 was simulated in Arena by having jobs arriving to an empty queue *become* a server (an Arena *resource*) for the duration of the busy period. Jobs joining a queue are simply counted and discarded.) A simple experiment to illustrate this is to have a huge batch of arrivals such as might occur at a hospital after a massive disaster, a denial of service attack (or suddenly popularity) of a web site, a produce harvest, or regulatory batch quarantine releases in a biopharmaceutical supply chain. When the batch size was increased 60 fold, the run times increased 1,000 fold. Experiments were run using Arena (v. 11) Professional Edition on an AMD 2700+ (2.16 gigahertz) processor with 512 megabytes of RAM. The runs were conducted at the *High Speed* setting in Arena. Arena is used here to illustrate the language independence of ERG models, any number of other languages could be used with similar results.

ERG_3 runs slowly for congested systems because it creates a record for each job. This is how most commercial software languages work. (For examples, see the AutoMod User's Manual v 10.0, Brooks Automation (2001), Chelmsford, MA; Harrell, Charles R., Royce Bowden, and Birman K Ghosh, (2004) Simulation Using Promodel with CD ROM, McGraw-Hill Professional; or Kelton, W. David, Randall P. Sadowski, and David T. Sturrock (2003), Simulation with Arena, McGraw-Hill Professional.)

This approach is popular with simulation software largely because it maps directly into an animation – the objects that move on the screen during an animation are the active entities in the simulation code. Fast execution is critical to a simulation model being of practical analytical value in experimenting with highly-congested systems. Animation is not free.

2.4 Practical relevance

Before continuing, we note that the methods to be presented later in this paper apply to considerably more realistic simulation models than the simple ERGs used for illustration. By adding only a single arc, ERG_3 can enriched to model queueing networks of *any size*, processing *any number* of different classes of jobs (each having different routes, timing, and priorities) being served by *any number and types* of parallel servers having multiple failure modes with variable-sized batched arrivals and services possible at each station. This is done using parametric event vertices. The ERG models shown here should be considered as elements in a high-dimensional array of ERGs (see, for example, Fig 7.8, p. 108 in (Schruben and Schruben 2006)).

Situations where the modeling methodologies to be presented can be applied include many interesting real service, production, and communications systems such as call centers, transportation networks, semiconductor fabrication facilities, biopharmaceutical supply chains, and the internet. A call center simulator using these methods was developed for a major bank that ran much faster than a competing simulator. A simulation of an actual semiconductor fabrication plant used these modeling techniques to execute almost two orders of magnitude faster than the most popular simulation/animation software (Schruben and Roeder 2003). A validated biopharmaceutical production and supply chain simulator has recently been developed using these modeling methods that can simulate one year's production in about a second on a laptop computer, again this is dramatically faster than any commercial simulator.

3 ANALYTICAL SIMULATION MODELING

In this section, four increasingly difficult analysis problems are considered to demonstrate how they might influence simulation model design.

3.1 Indirect estimation of expected job delays

If one is interested only in estimating the mean job delay, E[D], then it would appear better to run the much faster simulation model, ERG_1, to estimate E[Q] directly and then apply Little's Law to estimate E[D]. However, the literature on the variance reduction technique of indirect estimation (IE) recommends just the opposite (Carson and Law 1980, Glynn and Whitt 1989). The recommendation in the literature is to run the slower simulation model, ERG_3, to estimate E[D] directly and indirectly estimate E[Q] via Little's Law. This recommendation is based on the assumption that an equivalent amount of computation is required to simulate $\{D_i\}$ as $Q(t)$.

Not considering different models, the mathematically possible variance reductions of IE are modest. Whereas accounting for the work done using different models, IE can increase variances dramatically. Furthermore, the mathematical variance advantage of IE disappears for highly congested queues, while the efficiency of model ERG_1 relative to model ERG_3 increases as congestion increases. Most interesting queueing systems, at least occasionally, become congested. This is an example of where the results from rigorous simulation analysis research, taken out of the simulation modeling context, are likely to be misleading most of the time.

3.2 Direct estimation of job delay probabilities

Assume one is interested in estimating more than the mean job delay. It is probably more realistic, particularly when simulating service systems, to be interested in estimating the probability that a job will be delayed less than some service performance standard. For a particular performance threshold, τ, this problem is to estimate the value of the cumulative distribution function of D at τ,

$$\theta(\tau) = F_D(\tau) = \text{Prob}\{D_i \le \tau\}.$$

Using the slower model, ERG_3, to simulate job delays is the customary approach. The typical estimator of $\theta(\tau)$ is the average of indicators, $I_{[D_i \le \tau]}$ (equal to one when $D_i \le \tau$ and zero otherwise)

$$\hat{\theta}(\tau) = \frac{1}{n}\sum_{i=1}^{n} I_{[D_i \le \tau]}.$$

Here n is the number of simulated jobs, perhaps after a warm up period.

Since the process $\{D_i\}$ is converted into a sequence of indicator function values to estimate $\theta(\tau)$, it is possible to save work by designing our simulation model to generate $I_{[D_i \le \tau]}$ directly.

If we assume that jobs are served in the order they arrive, then simulating this delay indicator is easily done by adding another "superfluous" event to simulation model ERG_2. This new event simply counts the number of jobs that have arrived more than τ time units before the current *Start* event. This simulation model, ERG_4, is in Figure 4.

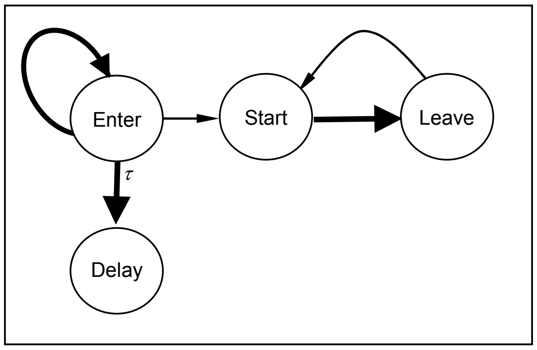

Figure 4: ERG_4, implicit estimation of job delay

At the time of each job Start event, if the count of Delay events is greater than the count of Start events, then we know that the job starting service has spent less than τ time waiting in line. Job arrival times do not need to be recorded and stored.

At first glance, it looks like simulation ERG_4 is merely using the master list of scheduled events to effectively store the arrival times of each job. However, the Delay event in ERG_4 is implicit: it does not schedule any other events and does not change the system state; therefore, it is not essential that this event be either scheduled or executed. A technique for implementing implicit events in an actual large-scale semiconductor fabrication facility simulation was developed by Roeder, resulting in nearly two orders of magnitude faster model execution than their current simulation software (Roeder 2004).

The structural analytical advantage of model ERG_2 over ERG_1 mentioned earlier applies when modeling parallel servers. If an implicit job delay counting event were added to model ERG_1 instead of ERG_2 in a multiple server simulation, there would be an error due to job overtaking. Overtaking occurs when jobs are depart in a different order than they arrived. One does not have to read very far in some queueing texts to see the complexity overtaking adds to queueing analysis (Baccelli and Bremaud 2003, page 1). For FIFO queues, there is no overtaking between the Delay and Start events in ERG_4 while a job waits in line.

If more points on the probability distribution function for D are desired, (or there are different job priority classes, but FIFO within each class), then a parameterized set of implicit job delay events can be defined for a larger number of different arguments.

If the queueing discipline is not first-come-first-served, then a "kanban" control can be implemented that limits the number of concurrent implicit job delay events "tagged" by kanbans to one. The Delay events and Start events corresponding to tagged jobs will then execute in the same sequence. Overtaking among the tagged jobs does not occur provided the job order in the queue will not change except at event times. This includes all the usual analytical queue model disciplines, but does not cover external queue reordering, say due to the exogenous changing some of the job due dates.

3.3 Response surface meta-modeling

If one wants to study the behavior of the expected delay over a range of different system parameters and factors, then fitting a meta-model to the average simulation response surface is appropriate. The usual approach is to run some screening experiments followed by a more carefully designed experiment to estimate important effects and interactions. The simulation outputs are then used to fit a regression of the average response delay to the system parameter and factor values (Kleijnen 2008). Running a large number of experiments covering a large factor space requires a fast simulation.

Using conventional methodology, a fast model like ERG_1 might be appropriate for initial factor screening (using Little's Law), while ERG_3 might be appropriate for the later, more refined experiments. If Response Surface Methodology (RSM), or similar methods, are used for response optimization, then ERG_1 might be used for RSM-Phase I (where a coarse search is done using low-resolution experimental designs to fit linear meta-models) and ERG_3 used for RSM-Phase II (where a higher-resolution experiment is run to fit a higher-degree polynomial meta-model in the neighborhood of a local optimum) (Myers and Montgomery 2002).

If the study goal is system ranking and selection using a 2-phase procedure, then a fast model like ERG_1 might be more effective for the first phase, while a slower running, but more informative model like ERG_3 might be better suited during the second phase. (Goldsman et. al. 2002). A two-phase ranking and selection procedure becomes a 4-phase method when only two different models are considered. Different levels of model detail are also probably appropriate for the different candidate systems, depending on how likely they are to be among the eventual winners. The options grow as powers of the numbers of possible models.

3.3.1 Embedding experiments into models

Regardless of the purpose of the study, the simulation models are typically run sequentially. However, a simulation model can be designed to replicate the entire experiment simultaneously. To illustrate: consider a situation where one wants to choose the most efficient of four service systems with, say, different numbers and/or types of servers. Figure 5 is an ERG that simulates all four systems simultaneously; all the events for all the systems are run concurrently on 4 integrated "copies" of ERG_1.

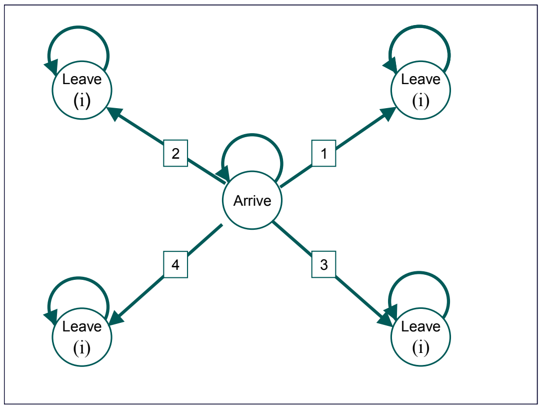

Figure 5 : Single ERG of four competing systems (ERG_5)

In ERG_5 all four systems share a common job arrival process, but each Leave(i) event is passed a different parameter value that identifies the system being modeled. Simulation model ERG_5 replicates a complete experiment for all the competing systems. Perfect positive correlation in the arrival processes is automatic since all systems have an identical arrival event.

Extending this notion, any number of replicates of the full experimental design for any number of competing systems can be run simultaneously with a single ERG model. This is accomplished by using a second event parameter to identify the replicate to which an event belongs. The ability to control initialization bias in such a multiple-system, multiple-replicate ERG model is enhanced since the warm-up periods of all replicates are observable together rather than sequentially.

A simulation model can also be used to adaptively design the experiment since the number of systems being considered does not need to be fixed throughout the run. New systems can enter or be dropped out of contention while the model is running. New event parameter values can be passed to sub-ERGs whenever any new system factors look promising, or old event parameter values can be discontinued when they no longer look competitive; all during a single run.

Somewhat less obvious is that each candidate system in the experiment does not need to use the same time scale, nor do these time scales need to be constant during a run. Systems that are doing comparatively poorly can have their *simulated* time scales dilated and systems that are performing better can have their *simulated* time scales contracted. The result is that more CPU cycles are devoted to simulating the better performing systems than to simulating the losers (Schruben 1997).

A detailed algorithm for implementing this methodology is given in the PhD thesis by Paul Hyden (2003). There this approach is compared to some of the leading commercial simulation "optimizers" using the results of a production experiment reported by Law and McComas (2002). The results were dramatic. The full factorial time-dilated ERGs obtained better answers with far fewer simulated jobs. The "Cost of Decision" (using the costing in Law and McComas's case study) for the time dilation ERG was more than an order of magnitude smaller than any of commercial competitors. At least an order of magnitude fewer simulated parts were needed. A much better decision was made much faster. (Hyden and Schruben 2000).

A generalized implementation of this adaptive model design concept, using MATLAB, is outlined in Hyden's PhD thesis cited earlier. Developing this into a practical general selection or response optimization procedure, perhaps embedded within nested partitions (Shi, Chen, and Yucesan 1999) will require further research.

While considering the analytical purpose of a simulation model generally enriches ranking and selection procedures, it can also lead to simplifications. As shown with ERG_5, parametric models can simulate k different candidate systems simultaneously. This model design effectively eliminates the cost of model switching that was studied in (Hong and Nelson 2005). Simultaneous Ranking and Selection procedures using integrated models of all the alternatives also warrents further research.

3.3.2 Dynamic meta-modeling

There are other analytical advantages to integrating the experimental design and analysis into a simulation model. A common reason for conducting a simulation study is to fit a response surface regression meta-model over different values of system factors. Simulation models can be specifically designed for this analytical purpose.

The usual approach to meta-modeling is to replicate simulation models sequentially at different input factor settings (called design points) in a designed experiment. A linear regression model is then fitted to the average responses and evaluated. If the precision or accuracy of the meta-model is felt to be insufficient, more simulation runs are made at perhaps augmented or abridged design points. The process is complicated by several problems: what design points should be simulated? how many replicates should be run at each design point? how long should each replicate be? how should each replicate be initialized? and what is an appropriate (not confounded) meta-model?

When using an ERG model like ERG_5, which simultaneously replicates the entire experimental design, the output can be the meta-model itself – not just data that could be used later to fit and evaluate a meta-model.

To be specific, assume a model like ERG_5 is used to simulate *k* different combinations of factor settings in a designed experiment with *n* observations taken at each factor setting, producing here the waiting times, $Y_{i,j}$, for job $i = 1,2,\ldots n$ in system $j = 1,2,\ldots k$. The parameter estimators for linear regression meta-models are linear functions of the average responses at each design point. Since all the responses for all replications at all design points are observed throughout a single run of the model, the output from the simulation can be the fitted meta-model parameters.

In general, a linear regression meta-model parameter, $\beta^{(p)}$, is estimated as a linear weighting, say with weights $\alpha^{(p)}$, of the average responses at the *k* design points $(\overline{Y}_1, \overline{Y}_2, \ldots, \overline{Y}_k)$. Switching the order of the summations, this becomes

$$\hat{\beta}^{(p)} \doteq \sum_{j=1}^{k} \alpha_j^{(p)} \overline{Y}_j = \frac{1}{n}\sum_{j=1}^{k} \alpha_j^{(p)} \sum_{i=1}^{n} Y_{i,j} = \frac{1}{n}\sum_{i=1}^{n} \hat{\beta}_i^{(p)}$$

In the last equality, the estimator of the meta-model parameter is represented as its algebraic equivalent average of a time series of regression parameter estimates $\{\hat{\beta}_i^{(p)}\}$, defined as $\hat{\beta}_i^{(p)} = \sum_{j=1}^{k} \alpha_j^{(p)} Y_{i,j}$. The simulation output consists of the time series of these parameter estimates $\{\hat{\beta}_i^{(p)}\}$ for all of the meta-model parameters, which are updated continuously throughout the run.

The weighted sum to estimate the ith meta-model parameter, $\hat{\beta}_i^{(p)}$, requires values of the ith observation, $Y_{i,j}$, for *all* k design points in the simultaneous simulation. One way to accomplish this is to store partial accumulations of incomplete weighted sums in reusable memory (say, dynamic arrays). These accumulators are initiated at the pace of the fastest of the k systems, but are completed at the rate of the slowest system. The active storage needed might become significant if the systems in the simultaneous simulation process jobs at very different speeds. However, candidate systems that make sense to be include in a simultaneous simulation model are presumably viable competitors or to cover a reasonably sized design space, so they might not be hugely different.

The key analytical advantages in this approach are that the accuracy and precision of the meta-model can be evaluated while the model is being run. In addition, the (perhaps intentionally induced) correlations among the parameter estimators can be estimated directly from the time series of meta-model parameter estimates using the method in (Schruben and Margolin 1978). Solving the problems of controlling initialization bias, determining run durations, estimating correlations in the parameter estimators, and meta-model estimation can all take advantage of the fact that a current, up to date, meta-model for the entire experiment is observable throughout a single run.

When the experiment is part of the simulation model, the design also can be changed during the run by adding or removing factor settings as noted earlier. Sir R. A. Fisher, the inventor of statistical experimental design, who is often quoted: “The best time to design an experiment is after you’ve run it.” might agree that the best time to design a *simulation* experiment is *while* you’re running it.

3.4 Multiple-moment meta-models

Perhaps one is not only interested in fitting a meta-model to the mean response, but also interested in meta-models for higher-order moments. For example, for robust queuing system design, a meta-model for both the response mean and variance is required (Kleijnen, et. al. 2003, Govind, et. al. 2004). The direct approach is to create two different regression response surface meta-models for the mean and the variance.

It may be possible to use a single simulation model that generates a single meta-model for multiple response moments. A simple example is given here to illustrate the approach.

Laplace transforms (as well as characteristic and moment-generating functions) may be potentially useful as simulation response surface meta-models. These functions provide not only mean responses, as do typical regression meta-models, but also higher other moments as functions of input parameter values. Laplace transforms can be sampled directly with model, ERG_4, by allowing the delay times τ to be random variables.

If the implicit job delay, τ, in ERG_4 is exponentially distributed with different means, then it is possible to generate samples of the Laplace transform of the delay probability distributions directly. Observe that the Laplace transform for the job delay probability distribution, $f_{D,}$ is equal to the probability that a job delay, *D*, is less than an independent exponential random variable, $X(\tau)$ with mean $1/\tau$.

$$L_D(\tau) \doteq E[e^{-\tau D}] \doteq \int_0^\infty e^{-\tau x} f_D(x)dx$$
$$= Pr\{D \le X(\tau)\}.$$

ERG_4, with exponential delays, can be used to generate the indicator function values directly that can be used to estimate the Laplace transform for the delay times for a G/G/s queue with parameters, say, $(\lambda, \mu, s, \ldots)$. The fitted, or estimated Laplace transform of, $L_D(\tau : \lambda, \mu, s \ldots)$, might then be used as a response surface meta-model for queueing system delay moments as a function of the input parameters, $(\lambda, \mu, s \ldots)$. Of course, different values of τ could be run simultaneously as in ERG_5.

A small experiment with an M/M/1 queue illustrates how an estimated Laplace transform might be generated by ERG_4. Kanban job tagging discussed earlier to prevent overtaking is necessary here. In this example, the mean service time is 1 and the average arrival rate is 0.9. Kanbans are generated according to a Poisson (rate .1) process (sampling the time-average queue by appealing to the PASTA principle (Wolff 1989)). These kanbans were used to tag the next arrival only if no jobs in the queue are already tagged, to avoid the possibility of job overtaking. Empirically, a small bias seemed to be present possibly because it is overly likely that kanbans will tag the first customer in each busy period or may tag too few customers when congestion is high. Kanbans that arrived while the server was idle were discarded to reduce this potential for their biasing the output data. Research is needed before recommending a job tagging strategy that exploits the trade-offs between a higher tagging frequency, improving estimator precision, and the increased risk of job overtaking, which would reduce estimator accuracy.

Eight independently-seeded replicates were run at each of 12 different exponential delays (τ is replaced by $X(\tau)$ in ERG_4). Each replicate had an expected number of 100000 tagged jobs after an arbitrary warm-up period of 50 jobs. The estimated Laplace transform appears in Figure 6 and closely matches the plotted true Laplace transform for job delays.

We can design any simulation model to generate consistent empirical Laplace transform estimates directly for all (in the limit) values of its argument simultaneously.

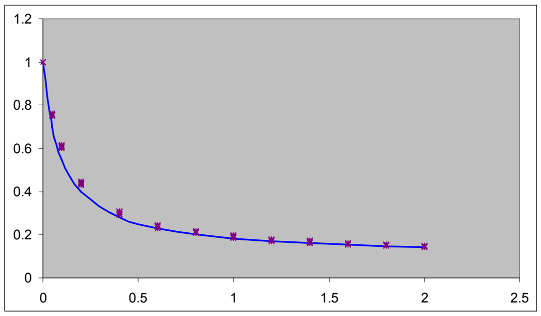

Figure 6: Sampled (using ERG_4) and known Laplace transforms for M/M/1 queue delays

The Laplace transform of the density function of any non-negative random variable, *D*, is the complimentary distribution function of the random variable, *X/D*, where *X* is a standard (mean equal to one) exponential random variable.

$$L_D(\tau) = Pr\{D \le X/\tau\} = Pr\{X/D \ge \tau\}$$
$$= 1 - F_{X(D)}(\tau) \doteq \overline{F}_{X(D)}(\tau).$$

The most common estimator is the complementary empirical distribution function, which is an average of indicator functions. The estimator of $L_D(\tau)$ for argument τ is then

$$\hat{L}_D(\tau) = \frac{1}{n}\sum_{i=1}^{n} I_{[X_i(D_i)>\tau]}\ .$$

In general, any simulation that can generate samples of any non-negative random variable, *D*, can generate samples of exponentials with random means *D*, as *X/D*. As in Section 3.2, each simulated job thus generates the indicator function used in the empirical Laplace transform estimator $\hat{L}_D(\tau)$ at all values of τ simultaneously.

To illustrate, three independently seeded replications of ERG_3, augmented to output the randomized Laplace transform argument S_i were made (with no warm up period). The empirical complimentary distribution functions of *S* for these runs are plotted along with the true theoretical Laplace transforms in Figure 7 at two different traffic intensities.

Comparing Figure 6 with Figure 7 shows that the randomized argument approach produced as good or better results than the direct (kanban) Laplace transform estimation method. Furthermore, the number of jobs simulated was vastly reduced. The directly estimated Laplace transforms at 12 specific argument values in Figure 6 required more than 1.2 million tagged jobs. The randomized Laplace argument estimates in Figure 7 used only 50 thousand total jobs to provide estimates over the *full* range of observed argument values. Here the direct job tracking model, ERG_3, required dramatically fewer simulated jobs than the faster but less informative ERG_4 model.

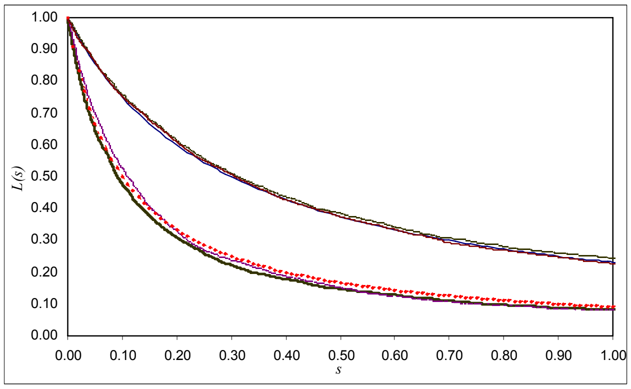

Fig. 7: Empirical Compliment CDFs of the Laplace transform for an M/M/1 queue (with their theoretical values) at ρ=.7 and .9

Some possible uses for multiple moment meta-models include controlling run initialization bias - by testing when the slopes of (simultaneously) replicated samples of the empirical Laplace transforms are close - and obtaining moment estimates needed for interval estimators. It might also be reasonable to estimate the gradient of a response with respect to its system parameters using mixed partial derivatives of fitted multiple-moment meta-models. First the slopes of the multiple-moment meta-models with respect to τ is estimated at zero, and used to fit the mean response as a function of the system parameters. The slopes of these fitted slopes can in turn be estimated with respect to one or more of the other system parameters to estimate the derivatives of the response.

More study of multiple-moment meta-models would need to be done before recommending using them in practice. The point here is merely to illustrate how simulation models that are both fast and effective might be designed to generate a particular analytical output directly, not necessarily to recommend doing so. Research would be needed on designing experiments and estimators for the different types of multiple-moment meta-models, and to develop algorithms for using them in practical simulation studies.

4 SUMMARY

The analytical properties of a simulation model may make the difference between a successful study and an inconclusive one. Simulation analysis methodologies that explicitly exploit the trade-offs in different levels of abstraction, control of uncertainty and time, and explicit causality in simulation models could produce better results. The potential of simulation models that design their own experiments and analysis while they are running is huge. Synergies between simulation modeling research and simulation analysis research, like those suggested in this article, could lead to significant advances in both.

REFERENCES

.

Aho, A., J. Hopcroft, and J. Ullman, 1985, *Data Structures and Algorithms*, AddisonWesley, pp. 135-8.

Andradottir S., D. Goldsman, B. Schmeiser and L. Schruben, and E. Yücesan, 2005, Analysis Methodology: are we done? *Proc. of the 2005 Winter Simulation Conf.*, Orlando FL, pp. 790-796,

Baccelli, Francois and Pierre Bremaud, 2003, *Elements of Queueing Theory: Palm Martingale Calculus and Stochastic Recurrences*, Springer, (see page 1).

Carson, John S. and Averill M. Law, 1980, Conservation Equations and Variance Reduction in Queueing Simulations. *Operations Research*, 28, pp. 535-526.

Glynn, P. W., and W. Whitt, 1989, Indirect Estimation via L=λW, *Operations Research*, 37.1, pp. 82-102.

Glynn, P. W., J. O. Henriksen, C. D., Pegden, B. W. Schmeiser, and L. Schruben, 1995, The Interface Between Simulation Output Analysis Research and Practice, *Proc. of the 1995 Winter Simulation Conference*, Alexandria, VA, December 3-6, p. 346.

Goldsman, D., S-H Kim, W. Marshall, and B. L. Nelson, 2002, Ranking and Selection for Steady-State Simulation: Procedures and Perspectives,' INFORMS Journal on Computing, Vol. 14, pp. 2-19.

Govind, N., R. Barton, D. J. Medeiros, and L. Schruben, 2004, Variance Response Surface Estimation for Robust Design: A Framework for Queueing Systems, *Proc. of the Industrial Engineering Research Conference*, May 2004.

Hong, L. J. and B. L. Nelson, 2005, The tradeoff between sampling and switching: New sequential procedures for indifference-zone selection. *IIE Transactions,* Vol. 37, pp. 623-634.

Hyden, P., and L. Schruben, 2000, Improved decision processes through simultaneous simulation and time dilation. *Proc. of the 2000 Winter Simulation Conference*, pp. 743-748.

Hyden, Paul, 2003, Time dilation: Decreasing time to decision with discrete-event simulation, PhD thesis, School of Operations Research and Industrial Engineering, Cornell.University. Ithaca, NY.

Kleijnen, J. P. C., B. Bettonvil, and J. F. Persson, 2003, Robust solutions for supply chain management: Simulation, optimization, and risk analysis, under review for publication.

Kleijnen, Jack P.C., 2008, *Design and Analysis of Simulation Experiments,* Springer.

Law, A. M., 2006, *Simulation Modeling and Analysis 4th ed.,* McGraw-Hill, pp 58-9.

Law, A.M., and M.G. McComas, 2002, Simulation-based optimization, *Proc. the 2002 Winter Simulation Conf.*, pp. 41-44.

Myers, Raymond H., and Douglas C. Montgomery, 2002, *Response Surface Methodology: Process and Product Optimization Using Designed Experiments* (2nd ed.), Wiley-Interscience.

Pidd M. and M. A. Carvalho, 2006, Simulation software: not the same yesterday, today or forever. *J. of Simulation*, Vol. 1. No. 1, pp 7-20.

Roeder, T., 2004, An Information Taxonomy for Discrete Event Simulations, PhD. Thesis, Dept. of IEOR, UC Berkeley.

Savage, E.., L. Schruben, and E. Yücesan, 2005, On the Generality of Event-Graph Models *INFORMS J. on Computing*, Vol. 17, No. 1, Winter 2005, pp. 3–9.

Schruben, D. A., and L. Schruben, 2006, *Simulating Dynamic Systems with Event Relationship Graphs,* Custom Simulations,

Schruben, L. and B. Margolin, 1978, Pseudorandom Number Assignment in Statistically Designed Simulation and Distribution Sampling Experiments, *Journal of the American Statistical Association*, **73** (363), pp. 504-525.

Schruben, L., 1997, Simulation Optimization Using Simultaneous Replications and Event Time Dilation. *Proc. of the 1997 Winter Simulation Conference*, pp. 177-180.

Schruben, L., and T. M. Roeder, 2003, Fast Simulations of Large-Scale Highly Congested Systems, *Simulation*, Vol. 79, No. 3, pp. 1-11.

Schruben, L., 1983, Simulation Modeling with Event Graphs, *Comm. of the ACM*, 26.11, pp. 957-63.

Schruben, L., 2008, Simulation Modeling for Analysis, under review for possible publication.

Seila, Andrew, V. Ceric, and P. Tadikamalla, 2003, *Applied Simulation Modeling,* Duxbury Press.

Shi L., C-H Chen, and E. Yucesan, 1999, Simultaneous simulation experiments and nested partition for discrete resource allocation in supply chain management, *Proc. 1999 Winter Simulation Conf.* pp. 395-401.

Wolff, R. W., 1989, *Stochastic modeling and the theory of queues,* Prentice-Hall.

Yücesan, E. and L.Schruben, 1992, Structural and Behavioral Equivalence of Simulation Models, *ACM Trans. on Modeling and Computer Sim*, Vol. 2 No.1, pp. 82-103.

ACKNOWLEDGMENTS

The author wishes to acknowledge his colleagues and students for their formative discussions of the topics in this paper, and the continued support of Genentech and the National Science Foundation, grant DMI-0323765.

AUTHOR BIOGRAPHY

LEE W. SCHRUBEN is a Chancellor's Professor in the Department of Industrial Engineering and Operations Research at the University of California, Berkeley. He holds degrees from Cornell, the University of North Carolina, and Yale and is an Informs Fellow. Prior to coming to Berkeley, he held the Andrew Schultz Endowed Professorship of Operations Research and Industrial Engineering at Cornell. His research interests are in simulation modeling and analysis methodologies and simulation applications in service and production systems, with particular emphasis on biopharmaceutical production and supply chain management. Prof. Schruben is a cofounder of the Bioproduction Group. email: `<LeeS.at.Berkeley.edu>`