Proceedings of the 1998 Winter Simulation Conference
D.J. Medeiros, E.F. Watson, J.S. Carson and M.S. Manivannan, eds.

INTERACTIVE IMPLEMENTATION OF OPTIMAL SIMULATION EXPERIMENT DESIGNS

Russell C. H. Cheng
John D. Lamb

Canterbury Business School
The University, Canterbury
Kent CT2 7NF, United Kingdom

ABSTRACT

An attractive feature of many simulation packages is their availability on desktop computers and their potential for allowing the user to run a simulation model under different conditions in a highly interactive way. Such a way of studying a system is attractive because of its immediacy and the direct control it offers the user. However, partly as a consequence of this, good practice in the use of the methodology of the design of experiments is not always followed. As a result the efficiency and effectiveness of the overall simulation study may not be as good as it should be. In this paper we investigate how design of experiments methodology can be explicitly incorporated into interactive desktop studies. In particular we show how the optimal design of experiments methodology proposed by Cheng and Kleijnen (1998) for studying queues with highly heteroscedastic output can be used to provide a front-end advisory interface for controlling and conducting the study of an actual system. To illustrate our discussion, we show how the interface can be set up for the SIMUL8 simulation package and show its use in the actual analysis of a particular queueing model.

1 INTRODUCTION

Use of discrete-event simulation is now a well established methodology (Carson et al 1984, Law and Kelton 1991). There are now many attractive and easy-to-use simulation packages (see Banks, 1996 for a review) that are designed to be operated from a desktop computer in an interactive way. Such packages enable models of complex systems to be rapidly built up and run. Most such packages contain features for saving and presenting the results of simulation experiments, and indeed such packages usually offer basic methods of analysing simulation output once this has been obtained. However, the determination of how best to conduct the runs that make up a simulation experiment is not usually catered for. Such decisions on how to make runs, and at what parameter settings, is usually left to the user. It is tempting, therefore, not to pay so much attention to design of experiment issues, especially when the model is being used interactively. This is despite the fact that there exist powerful design of experiments methodology which, if utilised, can greatly improve the efficiency of an overall simulation study. See for example Fedorov (1972), Whitt (1989), Ermakov and Melas (1995) and Vollebregt (1996).

In this paper we consider how the methodology of the design of simulation experiments can be implemented in the form of a front-end interface that can be attached to a simulation model. This interface is directly linked to the model. The interface receives the output from the model when it is run and automatically processes this output using an appropriate analysis methodology. This analysis provides information about the form and variability of the simulation output, and gives guidance about how best to continue the simulation study. The interface thus can be used interactively either to give advice to the user conducting the simulation study, or else to directly control the conduct of the runs.

We discuss the features needed in such an interface, and what kind of diagnostic information it might produce during the course of the simulation of some particular system.

Cheng and Kleijnen (1998) consider queueing systems where the simulation output response of interest is dependent on a design variable, where both the expected value and the variance of the output varies substantially as the design variable varies, and where the objective of the simulation is to measure this response over a range of design variable values. They describe a sequential method of optimally allocating the computing effort, during the study, to different selected design points. Such a sequen-

tial technique is particularly suitable for implementation in an interactive interface. We shall discuss such an implementation for a typical package, SIMUL8, a simulation package designed specifically to allow models to be very easily constructed and run in a desktop environment (SIMUL8, Guide, 1998) Another very good package is ARENA (Kelton et al., 1998), and the methods that we discuss should be readily implementable using this, though we have not yet tried ourselves.

In Section 2 we give an outline of the design of simulation experiments methodology given by Cheng and Kleijnen. In Section 3 we describe the overall structure of the interface. We also discuss in rather more detail the key modules and routines contained in the interface. In Section 4 we demonstrate use of the interface in conducting the simulation of a particular queueing model. We show how its use enables the user to control the allocation of runs to different design points in an informed and efficient way.

2 OPTIMAL ALLOCATION OF RUNS

Suppose we have a simulation model and wish to investigate its behaviour. We are interested in an output (response) variable y, which may represent for example a waiting time or queue length. We wish to model how y varies with a vector $\mathbf{x} = (x_1, ..., x_t)$ of independent input variables. These may represent parameters such as arrival rates, service rates and number of channels. And we are interested in the behaviour of y over a range of values of $\mathbf{x}$ given by $x_i \in [x_{\mathrm{L}i}, x_{\mathrm{U}i}]$ $(i = 1, \ldots, t)$. We suppose, generalising Cheng and Kleijnen (1998) that the input-output relationship can be described by the following regression metamodel:

$$y_{ij} = \left(\sum_{i=0}^{k} \beta_i q_i(\mathbf{x}_j) \right) f(\mathbf{x_j}) + \varepsilon_{ij}$$

for $i = 1, \ldots, n$, $j = 1, \ldots, m_i$, where

1. f and $q_0, \ldots, q_k$ are known functions (see next paragraph);
2. ε_i is the approximation error of the metamodel, with mean 0 and variance σ_i^2;
3. $\beta_0, \ldots, \beta_k$ are unknown parameters representing input effects;
4. we make the simulation runs at only $n+1$ distinct input values $\mathbf{x}_0, \ldots, \mathbf{x}_n$, with m_i observations (replications) at each point $\mathbf{x}_i$.

We call $\mathbf{x}_0, \ldots, \mathbf{x}_n$ the *design points*. We are principally interested in how best to choose the m_is.

The functions q_i may be any functions that we think will give a sensible model of the behaviour of the response variable. Typically we can choose functions by looking at the general behaviour (e.g. increasing or decreasing) and the limiting behaviour of y. The function f allows us to construct regression models that have unbounded responses; in particular it allows us to model saturated queueing situations. For a single server queue where x is the traffic intensity, a typical regression model would be one of the form

$$y_{ij} = \left(\sum_{i=0}^{k} \beta_i x_j^i \right) / (1 - x_j) + \varepsilon_{ij}.$$

Typically we will be interested in ranges of $\mathbf{x}$ where the response will be highly heteroscedastic. Thus we will have to consider the variance of the response. We can assume that the variance is given by $\mathrm{Var}(\varepsilon) = [g(\mathbf{x})\sigma]^2$ where $g(\mathbf{x})$ may or may not be known. In this case Kiefer and Wolfowitz (1959) point out that homogeneity of variance can be restored in the regression metamodel simply by dividing by $g(\mathbf{x})$:

$$z_{ij} = \left(\sum_{i=0}^{k} \beta_i q_i(\mathbf{x}_j) \right) r(\mathbf{x_j}) + \delta ij \qquad (1)$$

where

$$r(\mathbf{x_i}) = f(\mathbf{x}_i)/g(\mathbf{x}_i).$$

Then $\mathrm{Var}(\delta_{ij}) = \sigma^2$, a constant, independent of $\mathbf{x}$. We call

$$z_{ij} = \frac{y_{ij}}{g(\mathbf{x}_i)} \qquad (2)$$

the *transformed response variable* and model its behaviour with Equation (1). If $g(\mathbf{x})$ is unknown, it can be estimated at each design point from the results of the simulation runs.

It is easy to show that we should choose the number of design points $n+1$ equal to the number of functions $k+1$ in the regression model. So two problems remain: how to choose the design points, and how much computing effort to concentrate at each of them. Cheng and Kleijnen (1998) suggest that the results of our simulation experiments are not likely to be improved substantially by choosing design points optimally rather than uniformly within the range of possible values. So we will assume that we have chosen some design points and functions, and concentrate on the problem of how many runs to make at each point.

We use a generalisation of the method of Cheng and Kleijnen (1998) to find the optimal number of runs m_i for each design point $\mathbf{x}_i$. We leave the details of the method for another paper, but note that the most complex part of the calculation, involving matrix inversion needs to be done only once, at the start of the simulation experiment,

even if (see next paragraph) we need to recalculate the m_is.

Since we are using the regression model of Equation (1) to estimate $\beta_0, \ldots, \beta_k$, we are effectively using the transformed response variable z of Equation (2). If the form of the variance, $g^2(\mathbf{x})$ is known exactly, then we can find $m_0, \ldots, m_k$ before carrying out any runs. In practice, it is unlikely that we would know $g(\mathbf{x})$. But we can still approximate $m_0, \ldots, m_k$ with the following procedure.

1. Let N be the total number of runs to be made. Initially make m_{I} pilot runs at each design point $\mathbf{x}_i$ with $km_{\mathrm{I}} \ll N$. The number of initial runs is not critical, but should be enough to estimate $g(\mathrm{x}_i)$.

2. Cycle through the design points in a fixed order $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_k$ say. For each design point $\mathbf{x}_i$, we can calculate easily π_i, the proportion of runs that should be made at $\mathbf{x}_i$ using the best available estimate of $g(\mathbf{x}_i)$. If m_i runs have already been made at $\mathbf{x}_i$, make sufficient extra runs so that the new value of m_i will exceed $\pi_i N$.

3. Repeat step 2 until N runs have been made.

3 THE INTERFACE

Although it is more efficient to implement our methodology as (say) a C programme, we have chosen to use the SIMUL8 simulation package together with Microsoft Excel and VBA. The main advantage of the approach is that it enables simulation runs to be carried out within the framework of design of experiments methodology, but where instead of having to write a controlling programme to implement such a methodology, the framework is already built into the interface. Thus, once the simulation model is built, it can be linked to this interface very easily. The interface can then be used to control the simulation runs. The interface can either carry out the necessary runs of a fairly elaborate simulation experiment as specified by the design of experiments prescription, or it can be used interactively, with the user exercising ultimate control over how runs should be carried out, but with the interface providing diagnostic, run-time information during the progress of the experiment for the user to make use of as he or she thinks fit.

The main reasons for our choice of packages/software are the following. First, SIMUL8 is a very flexible and widely used Windows based package. So it is easy to set up any model we wish to analyse or to modify an existing model. No special skill or programming knowledge is required. Second, Excel is a widely used standard spreadsheet package with good graphical facilities. It provides an easy flexible way of storing, exploring and displaying the results of our analysis. It is also easy to use with SIMUL8. Third, use of VBA allows the interface to be developed within a Windows environment that enables all three main software components, *viz.* (i) SIMUL8 model, (ii) Excel Spreadsheet holding model and runtime information, and (iii) the interface itself, to be linked in a unified, coherent way.

In practice individual experiments might well need special setting up. Our approach, using an interface is aimed at minimising the programming effort needed for carrying out such changes. In practice, modification of the interface will need to be carried out during the overall experiment so the interface needs to be reasonably transparent and accessible

The interface should not add any substantial overhead to computing time. Our experience is that most of the run time is taken up by SIMUL8 even for very simple simulation models. Thus there is no great loss of efficiency in controlling the simulation experiment in this interactive way, particularly for more complex simulation models.

We will now describe the interface between SIMUL8, Excel and our VBA programme. Figure 1 shows a block diagram of the relationship between the components.

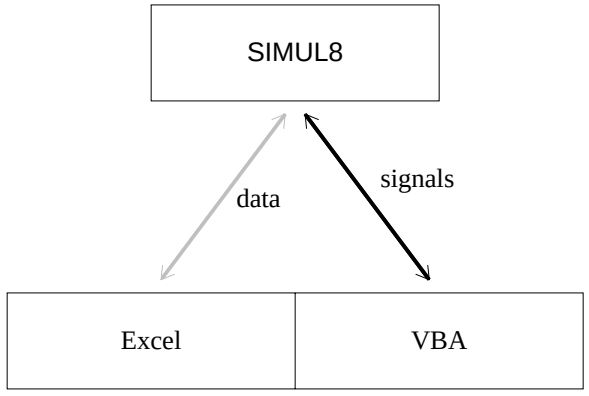

Figure 1: Relationship between SIMUL8, Excel and VBA

The interface must be able to (i) calculate the details of an appropriate design of experiments setup, in particular the design points at which runs are to be made, and the amount of computing effort needed at each such design point, (ii) display this information in Excel in appropriate windows, (iii) control the running of the simulation model using this information, (iv) receive runtime information back from the SIMUL8 model and display this also in Excel, possibly modifying the design in consequence, and (v) enable the user possibly to intervene in an interactive way as the overall simulation experiment progresses. VBA (Visual Basic for Applications) being a Windows-based programming language that comes as part of Microsoft Excel, is well

suited for satisfying these requirements. It is the language that Excel uses for its macros and has commands that allow it to use Excel as if it were a human user. It can be used as a programming language to implement the sometimes complex mathematical calculations required for our simulation methodology. It can use Windows dialog boxes to communicate with the user. It can also send signals to SIMUL8 using a module supplied with the SIMUL8 package. We use these signals to set parameters in SIMUL8 (e.g. mean service time for a work center), to control run lengths and warm-up times and also to collect information when a run is complete.

SIMUL8 sends signals to VBA. The most important one used by our programme is a signal that tells VBA when a simulation run is complete. VBA is then able to ask SIMUL8 for information about the run, e.g. average length of a queue or average waiting time in the system. We use Excel to store information about the parameters for a set of simulation runs and also information about the runs themselves. We also use it as a convenient way of displaying the results of our analysis. SIMUL8 can store data directly in and collect information directly from Excel. However, we do not use this facility.

We now describe how one can use the modelling system. First, we use SIMUL8 to build the model we want to investigate. Then we make a few minor modifications to allow us to use the Excel/VBA interface. We need to tell SIMUL8 that it will use the Excel worksheet containing our VBA programme and to send it a signal to indicate the end of a simulation run. The only other modification we need to make is to give a name to any distribution whose parameters we wish to modify.

Figure 2 shows how it appears for a small simulation model. The screen displays three main elements. The top third holds the SIMUL8 display, including the model (M/M/1 in the Figure). The main window in the bottom left is the Excel worksheet displaying details such as initial estimates of likely response variability and the suggested design point information. The remaining smaller window in the bottom right is the Simulation Model Control dialog box holding information about suggested run length and warmup period at each design point.

When the SIMUL8 model is reset, it loads the Excel worksheet and programme. We then use dialogs to collect information about the simulation runs we will make. This includes the following.

1. Run length and warm-up time
2. Total number of runs and number of initial runs
3. Parameters to be changed by the model
4. Parameter to be measured in the model

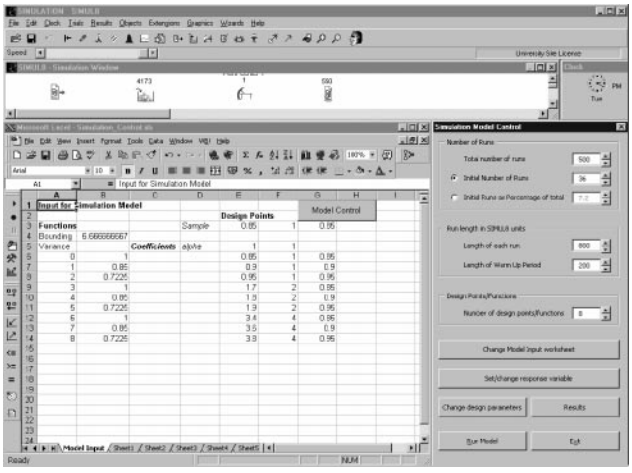

Figure 2: Appearance Of The Modelling System

5. Number of simulation points and values of parameters at them
6. Functions to be used to build the regression model

When the model has been set up, we use the Run Model buuton on the Simulation Model Control dialog box to instruct the programme to carry out the runs in SIMUL8. It does this, collects the data and stores it in Excel, and produces a summary of the results on an Excel worksheet.

4 QUEUE SIMULATION EXAMPLE

This section describes the results of applying our programme to a simple queueing simulation. The system we chose to model was an M/M/c queue. We modelled it for a range of traffic intensities from 0.85 to 0.95 and for 1–4 channels, and measured the average queue length for each run we made. We chose this model because it has two parameters, is highly heteroscedastic over the range of the parameters and is one for which the average queue length can be obtained theoretically.

The SIMUL8 model comprises a work entry point, a queue, a work center and a work exit point. There are two distributions: an arrivals distribution at the work entry point, which we set as exponential with arrival rate λ, and a service distribution at the work center, which we set as exponential with service rate μ. We wish to vary the traffic intensity $\rho = \lambda/(c\mu)$; so we set $\mu = 1$ and $\lambda = c\rho$ for each simulation point (ρ, c) we consider. We vary c by varying the number of replicates of the work center in SIMUL8.

We chose as a response variable y the average length of the queue. We modelled the transformed response z

(see Equation 1) as

$$z = \left(\sum_{i=0}^{8} q_i(\rho, c) \right) \frac{f(\rho, c)}{g(\rho, c)}$$

where $f(\rho, c) = 1/(c(1-\rho))$, $g^2(\rho, c)$ is the variance of the average queue length and $q_i(\rho, c)$ and the simulation points are given in Table 1. The choice of functions reflects the behaviour of y which should increase with ρ, decrease with c, tend to ∞ as ρ tends to 1 and tend to 0 as ρ tends to 0

Table 1: Regression Functions and Simulation Points

i	$q_i(\rho, c)$	ρ	c
0	1	0.85	1
1	ρ	0.90	1
2	ρ^2	0.95	1
3	c	0.85	2
4	$c\rho$	0.90	2
5	$c\rho^2$	0.95	2
6	c^2	0.85	4
7	$c^2\rho$	0.90	4
8	$c^2\rho^2$	0.95	4

We first used our programme to make 500 runs with 4 runs initially at each point and the remaining runs chosen optimally. As we would expect, most of the runs were made at the points $(0.95, 1)$ and $(0.9, 1)$. For comparison, we repeated the experiment with equal numbers of runs at each simulation point. To do this we made 504 runs with an initial 56 at each of the same nine simulation points.

The estimates for the average queue lengths obtained from each of our two simulation experiments are shown for $c = 1$ and $c = 4$ in Figures 3 and 4. The estimates with optimised distribution of runs are shown as 'optimal' while those with equal distributions are shown as 'non-optimal'. The 'exact' curves are obtained theoretically and show the true average queue lengths for the given parameters. The plots for $c = 2$ and $c = 3$ give similar results, showing that using our methodology produces much better estimates for the response than the naïve approach of making equal numbers of runs at each simulation point.

5 CONCLUSIONS

The approach suggested in this paper is aimed at trying to bridge the gap between existing simulation practice and the substantial design of experiments methodology now available for improving the efficiency of simulation experiments. Interactive use of simulation packages should become ever more commonplace, much like present day to day use of spreadsheet packages. The development of interfaces like the one described in this paper should enable such desktop use of simulation to become much more efficient and effective.

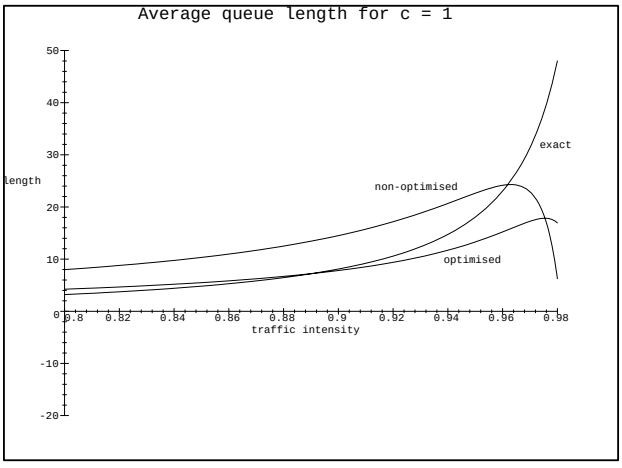

Figure 3: Results for M/M/C queue with $C = 1$

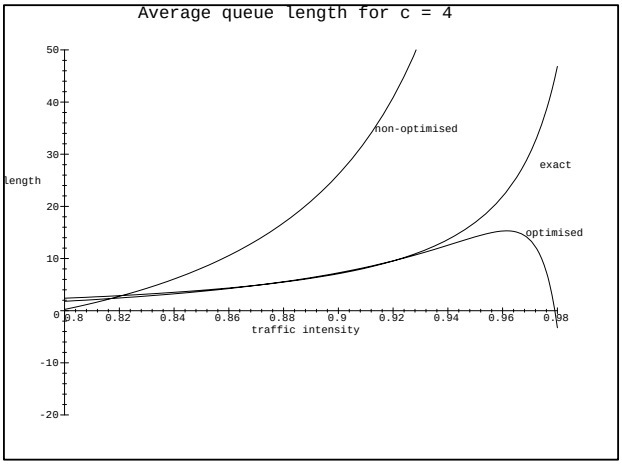

Figure 4: Results for M/M/C queue with $C = 4$

REFERENCES

Banks, J., Carson II, J. S. and Nelson, B. L. (1984), *Discrete-Event Simulation, (2nd Edn)*. Upper Saddle River, NJ: Prentice Hall.

Banks, J. (1996), Software for Simulation, in *Proceedings of the 1996 Winter Simulation Conference*, ed. J. M. Charnes, D. J. Morrice, D. T. Brunner and J. J. Swain. IEEE, Piscataway, 31-38.

Cheng, R. C. H., Holland, W. and Hughes, N. A. (1996) Selection of input models using bootstrap goodness-of-fit. In *Proceedings of the 1996 Winter Simulation Conference*, eds J. M. Charnes, D. J. Morrice, D. T. Brunner and J. J. Swain. IEEE, Piscataway, 317-322.

Cheng, R. C. H. and Kleijnen, J. P. C. (1998), Improved Design of Queueing Simulation Experiments with Highly Heteroscedastic Responses, *Operations Research*, Accepted.

Ermakov, S. M. and Melas, V. B. (1995), *Design and Analysis of Simulation Experiments*. Kluwer Academic Publishers, Dordrecht, Netherlands.

Fedorov, V. V. (1972), *Theory of Optimal Experiments*. Academic Press, New York.

Kelton, W. D., Sadowski, R. P. and Sadowski, D. A. (1998) *Simulation with Arena* WCB Boston: McGraw-Hill

Kiefer, J. and Wolfowitz, J (1959), Optimum designs in regression problems. *Annals Mathematical Statistics*, 30, pp. 271-294.

Law, A. M. and Kelton, W. D. (1991), *Simulation Modeling and Analysis 2nd Edition*. New York: Mc-Graw-Hill.

SIMUL8 Manual and Simulation Guide (1998) Glasgow: Visual Thinking International Limited.

Vollebregt, T. A. J. (1996), *Experimental design for simulation*, PhD Dissertation, Dept. of Management, Univ. of Canterbury, New Zealand.

Whitt, W. (1989), Planning queueing simulations, *Management Science*, 35, no. 11, November, pp. 1341-1366.

AUTHOR BIOGRAPHIES

RUSSELL C. H. CHENG is Professor of Operational Research at the University of Kent at Canterbury. He has an M.A. and the Diploma in Mathematical Statistics from Cambridge University, England. He obtained his Ph.D. from Bath University. He is Chairman of the U.K. Simulation Society, a Fellow of the Royal Statistical Society, Member of the Operational Research Society. His research interests include: variance reduction methods and parametric estimation methods. He is Joint Editor of the IMA Journal on Mathematics Applied to Business and Industry, and an Associate Editor for *Management Science*.

JOHN D. LAMB is Lecturer in Operational Research at the University of Kent at Canterbury. He has a BSc in Mathematics from the University of Glasgow and an MSc in Mathematics from the University of Nottingham. He also obtained his PhD from the University of Nottingham. He is a member of the Operational Research Society.