Proceedings of the 2003 Winter Simulation Conference
S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, eds.

SPREADSHEET SIMULATION

Andrew F. Seila

Terry College of Business
The University of Georgia
Athens, GA 30602-6273, U.S.A.

ABSTRACT

"Spreadsheet simulation" refers to the use of a spreadsheet as a platform for representing simulation models and performing the simulation experiment. This tutorial explains the reasons for using this platform for simulation, discusses why this is frequently an efficient way to build simulation models and execute them, describes how to setup a spreadsheet simulation, and finally suggests when a spreadsheet is not an appropriate platform for simulation.

1 INTRODUCTION

A simulation is a sampling experiment that is done on the computer (Fishman 1996). At the core of any simulation is a model that involves quantities whose values are unpredictable and therefore must be sampled from an appropriate population of observations. The model is represented using a computer program, and the program actually samples the random variables, performs the computations of the model and reports the outcome, usually in the form of one or more observations. All simulation models fit this description. "Spreadsheet simulation" simply involves the use of a spreadsheet to represent the model, do the sampling and perform the computations.

The idea of an electronic spreadsheet for storing information and performing calculations dates back to Mattesich (1961) when computers were mainframes, but spreadsheets as we know them are a product of the microcomputer age. The earliest spreadsheet, VisiCalc, had limited functionality but introduced the world to the concept of the interactive electronic spreadsheet, i.e., a program that stores data and computations in a rectangular array of cells. Later spreadsheets such as Lotus 1-2-3, Microsoft Excel and Quattro Pro greatly expanded the features and developed the spreadsheet into an effective modeling, prototyping, analysis and presentation tool.

Today, spreadsheets are available for all of the major operating systems: Windows, Unix/Linux and Mac OS. Since Mac OS X is a derivative of BSD Unix, all of the Unix spreadsheets (as well as other software) will eventually be available for this platform. The most prevalent spreadsheet today is Microsoft Excel, which comes bundled with Microsoft Office. Most other spreadsheets operate similarly and have similar features. In this paper, we will refer to Excel menus and other features. However, the concepts and techniques to be discussed apply to virtually all other spreadsheets.

Spreadsheets are used by many people. Since the idea of the spreadsheet started in the context of accounting, it is natural that they are widely used in business. Most business users, however, employ a very small subset of the available features. Engineers have been slow to adopt spreadsheets as a computation and analysis platform, perhaps because they have been trained to use other software tools such as MATLAB that were written specifically for mathematical modeling. These tools are very powerful and can certainly be used to develop a broad range of simulation models, but in many cases a spreadsheet is simpler and more intuitive to use.

2 WHY USE A SPREADSHEET FOR SIMULATION MODELING?

We will use the term "simulation platform" to refer to the software environment used to develop, test and run a simulation experiment. First, let's examine what features a simulation platform needs. Following is a list of the capabilities that must be available:

1. A way to represent mathematical and logical relationships between variables in the form of computations and assignment of values, and algorithms that describe how to do a series of computations.
2. A way to generate random numbers and use them to sample observations from various distributions.
3. A means to repeat a series of computations, thus implementing replications.

This list is minimal. All of these features are necessary for the platform to be used for simulation. Most popular spreadsheets have these features. In addition, the following

features are available in most spreadsheets to make the process quick and reliable:

1. A large number of functions to do mathematical, statistical, database, date/time, financial and other calculations.
2. Database representations and database access.
3. Charting and graphing.
4. Display and documentation features such as fonts, colors and geometric shapes to improve presentation.
5. Automation through scripting languages such as VBA (in the case of Excel).

The table structure of spreadsheets allows the developer to organize the computations and results in a natural and intuitive manner. Spreadsheets are ubiquitous - almost everybody has one - and file formats have become standardized, so files written by one spreadsheet can usually be imported by others. As a result, developers and users can easily pass simulation models from one to another. For these reasons, the spreadsheet is an attractive platform for simulation.

There are a number of publications that discuss spreadsheet simulation. See (Winston 1996) and (Seila, Ceric, and Tadikamalla 2003), Chapters 2 through 4 for very readable tutorials.

3 WHEN IS SIMULATION USING A SPREADSHEET APPROPRIATE?

Certain modeling situations lend themselves well to implementation in a spreadsheet. Indeed, any set of calculations in a spreadsheet can be considered a model. Usually, these models have parameters or variables whose values are unknown and assumed.

3.1 Stochastic Models

In some cases, the unknown parameters are actually random variables whose value cannot be predicted, i.e., the models are stochastic models. Many stochastic models in finance (including real estate and insurance), logistics and engineering can be conveniently setup in a spreadsheet for simulation. Spreadsheets are frequently used by actuaries, for example, to evaluate insurance rating methods. Consider, for example, an inventory model in which the demand for the product is stochastic. In order to evaluate a particular replenishment policy, this value must be sampled when the simulation experiment is run. An experiment would consist of sampling demand for the product and applying the inventory policy over a long period of time to compute observations of the periodic costs resulting from excess inventory and shortages associated with the policy. These observations would then be used to estimate the mean cost for the policy. The experiment would be repeated for several policies to find the inventory policy that produces the lowest mean cost. This is a typical stochastic model that can be analyzed using simulation. Interesting spreadsheet implementations of queueing simulations have also been developed (Grossman 1999).

3.2 Sensitivity Analysis for Spreadsheet Models

Another situation where spreadsheet simulation is useful involves doing a "what-if" or sensitivity analysis for models having unknown parameters that are not necessarily random. It is often the case in spreadsheet models that modelers frequently want to determine how sensitive the performance measure is to variations in these parameters. For example, in a model that concerns the leasing or purchase of a piece of real estate, the mortgage interest rate at the time the contract is signed is an unknown parameter. The present value of each decision (lease vs. buy) will depend upon this parameter value. If only one or two parameters are involved, modelers can use the "Table" command of the "Data" menu to evaluate the performance measure when each parameter value in a collection of possible values is substituted into the model. Excel and other spreadsheets will support this calculation with one unknown parameter and many performance measures, or with two unknown parameters and one performance measure. For example, one could vary the interest rate from 5.5% to 11.0% in steps of .5% and, for each value, compute the present value of the lease decision and the present value of the buy decision. Or, one could vary the interest rate and also vary the value of the property over a discrete set of values, and for each combination of these two values, compute the difference between the present values of the two decisions.

In real spreadsheet models, there are normally many unknown parameters, as well as multiple performance measures. This type of what-if analysis can become unwieldy when the number of parameters is more than a few. For example, suppose that the number of unknown parameters in the model is 10, and the number of possible values for each of these parameters is 3, denoting the minimum, most likely and maximum values. Then, the number of recalculations that must be performed in order to assess all combinations of these possible values is $3^{10} = 59,049$. Clearly, this is possible only if the process of recalculation is automated, and then it is rather time-consuming. If the number of parameters increases to just 15, the number of recalculations grows to about 14 million, an infeasible amount of computing on most desktop systems. The solution to this conundrum is simulation. By sampling these unknown values from appropriate distributions, one can do a "what-if" analysis on a model with a large number of unknown parameters. In fact, 1000 replications is generally enough observations to assess the variation in the output measures, regardless of the number of combinations of values of unknown parameters.

Thus, simulation is a useful technique when the number of unknown parameters is moderate or large.

The mechanics of setting up and running a spreadsheet simulation are very much the same in both of these cases, but there is one important difference in the way the output data are analyzed: When simulating a stochastic model, you are usually interested in using the output data to estimate an unknown performance measure for the model; when doing sensitivity analysis, you are interested in using the output data to assess the amount of variation in one or more output quantities.

4 HOW DOES ONE SETUP A SPREADSHEET SIMULATION?

Generally, each cell in a spreadsheet model can be classified as containing one of three types of quantities:

- *Inputs to the model*. These cells can contain parameters that are part of the model, such as unit costs or mean demand. The contents can also be sampled values of the random variables that represent uncertain quantities in the model such as demand or price paid, or they can be assumed values of unknown parameters when one is doing a sensitivity analysis.
- *Intermediate computations*. These cells contain calculations that are involved in the model. For example, in an inventory model, they might compute the inventory levels or backlogs at the end of each period.
- *Outputs from the model*. These cells contain the observations on quantities of interest one seeks from the model. For example, in an inventory model, these observations could be the costs incurred during each period.

Most models that can be organized in this way can be simulated in a spreadsheet. The following steps are described in more detail in Chapter 2 of (Seila, Ceric, and Tadikamalla 2003).

4.1 Setup the Model

The first step is to build the model in the spreadsheet using definite values for all parameters and other inputs. This allows one to check the computations and assure the correctness of the model before the simulation-specific components are added.

Second, replace the values in the cells that represent random or unknown quantities with formulas that sample these quantities from appropriate distributions. Appropriate formulas can be found in any reference on random variate generation. See (Cheng 1998) for example. At this point, all random variates will be resampled each time the spreadsheet is recalculated.

Third, identify the "output data" for the model. Actually, the modeler should know these desired performance measures when the model is created. For example, in an inventory model, one might use the mean cost per period as a performance measure, so the output data for the model would be the costs incurred in each period. Here, you want to identify those cells that contain the values of these performance measures. At this point, you can observe the values of these cells change (i.e., being sampled) each time the spreadsheet is recalculated.

4.2 Create the Simulation Run

It is useful to distinguish two types of simulation experiments: (1) static simulations that replicate the experiment independently, producing independent, identically distributed observations and (2) dynamic simulations that produce a time series of dependent observations. The setup for each of these types is different.

Where independent replications are performed, the model computations are normally contained in some region or group of regions of the spreadsheet. Since a recalculation produces a replication, we need to do a series of recalculations of the spreadsheet and save the outputs after each recalculation to perform the replications. There are several ways to accomplish this. If the model computations can be put in a single row, we can just copy this row the appropriate number of times, so each recalculation produces all replications at once. If the computations in the model are more involved and cannot be placed on a single row, we can use the Table command in the Data menu to tell the spreadsheet to go through an iterative recalculation, storing the values of the outputs between each recalculation. To setup the data table, first create a column of numbers having values from 1 to the number of replications you will perform. Excel and most other spreadsheets have an easy way to create a column or row of consecutive numbers. Above each adjacent column to the right, place a formula that will produce the value of a specific output. The design is for each row of this table to contain the replication number in the first column and the outputs, i.e., observations on each performance measure, for that replication in the adjacent columns.

Actually running the replications involves using a command such as the Data-Table command to tell the spreadsheet to run the recalculations. This command was originally created to perform "what-if" scenarios as described above by substituting each value in the first column into a specific cell, recalculating the spreadsheet, and recording the values of other cells that depend upon the substituted value adjacent to the substituted value. In our case, the contents of each cell in the first column of the data table is just a replication number, and since the replications are independent and identically distributed, the replication number is not actu-

ally used in recalculation. However, the recalculation will cause all random variate sampling formulas to re-execute, producing a new value for all random variates and thus new values for all outputs that are independent of those for all other replications. The Data-Table command produces a dialog asking where you want to put the input value. You can select any unused cell and click "OK". The data table containing the results of the replications will fill quickly. When it is finished, each column of this table except the first will contain all of the observations on a specific performance measure. It is not difficult to do thousands, or even tens of thousands, of replications in this way.

In a dynamic simulation, the output values are observed periodically over time. For example, in an inventory model, the costs incurred might be observed at the end of each week. In addition, each output observation will depend in some way on the previous outputs. If each period's computations can be placed in a single row, then the next period's computations are constructed from the contents of the cells in the previous row. Once the computations for a representative set of periods are setup, i.e., once a representative row is entered, the row(s) can be copied, thus extending the time span of the model and producing the desired number of periodic observations. As a result, the sequence of dependent output observations in the simulation will be contained in one or more columns of the spreadsheet. Chapter 3 of (Seila, Ceric, and Tadikamalla 2003) has some examples of dynamic models implemented in a spreadsheet.

4.3 Analyze the Data

Each simulation has its own analysis requirements (Alexopoulos and Seila 1998). For stochastic models, analysis normally involves applying statistical procedures to compute estimates of population parameters as well as confidence intervals for these estimates. When sensitivity analysis is the objective, data analysis is concerned with evaluating the *range of values* of the output data. This can involve computing extreme values of the data such as quantiles and graphically displaying the distribution of the data. Most spreadsheets have formulas for computing the sample mean, sample variance and quantiles of well known distributions such as the normal distribution, so the usual confidence interval formulas can be applied. Spreadsheets also have a rich selection of other statistical computations such as regression analysis and quantile computation, which can be applied too.

Presentation generally includes some tables and graphs. Spreadsheets have extensive facilities that make it easy to produce these types of presentations in high quality.

Some example spreadsheet models can be found at `http://seila.terry.uga.edu/spreadsheetSim`. These models illustrate the concepts and techniques just discussed.

5 SIMULATION ADD-INS FOR SPREADSHEETS

The process of developing and running a simulation in a spreadsheet can be simplified somewhat by using one of the available commercial add-in packages for Excel such as @RISK (`http://www.palisade.com`) or Crystal Ball (`http://www.decisioneering.com`). PopTools (`http://www.cse.csiro.au/poptools/`) is a free Excel add-in. Another free add-in for Excel called SIMTOOLS.XLA by Professor Roger Myerson is available at `http://home.uchicago.edu/~rmyerson`. These packages provide several features that are not included in the basic spreadsheet:

- Random number generation using documented and tested algorithms.
- Extensive functions for generating random variates from a variety of distributions.
- Features to automate the setup and running of the simulation experiment.
- Features to automate analysis and presentation of the output data from the simulation experiment.
- Optimization procedures for the model.

The random number function, which is called "RAND()" in most spreadsheets, produces a pseudo random sample from a uniform distribution between 0 and 1. Unfortunately, many spreadsheet publishers do not document the algorithm used in RAND(). Frequently, these are just the functions that are distributed as part of the C or C++ compiler. Research has shown that some algorithms for generating random numbers have better statistical properties than others (Fishman 1996, L'Ecuyer 1998). Thus, using the built-in RAND() function carries some risk that the random numbers will not behave as truly independent, random numbers. In @RISK, Crystal Ball and PopTools, the random number generators have been tested and documented, and therefore are recommended over RAND().

It is easy to write functions that generate observations from some distributions such as the triangular, exponential and normal distributions, starting from independent uniform random variates (Cheng 1998). However, observations from some distributions such as the Gamma and Weibull are difficult or impossible to generate using just the built-in functions of the spreadsheet. These add-ins provide easy, intuitive functions for all common distributions.

If you use the Data-Table method described above to run replications, some effort is required to set it up and the method uses space in the spreadsheet to store the results. These packages implement their own iterative procedure to run replications and store the resulting summary statistics or raw data. Often, you do not need to store all of the raw data. Only the summary statistics are needed. Thus, these add-ins can simplify the problems of setting up and running simulations, and analyzing the output. Examples of the use of these add=ins can be found on their websites

and examples of the use of @RISK for financial modeling can be found in Chapter 3 of (Seila, Ceric, and Tadikamalla 2003).

6 WHEN IS SPREADSHEET SIMULATION NOT ADVISED?

Spreadsheets are powerful, convenient tools for simulation modeling, but they do have four important limitations: (1) Only simple data structures are available, (2) complex algorithms are difficult to implement, (3) spreadsheets are slower than some alternatives and (4) data storage is limited. Let's examine each of these limitations.

The spreadsheet consists of a group of pages, each of which has a table consisting of rows and columns of cells. Each cell can contain data or a formula. One can treat a column or row of cells as a vector, and a two-dimensional range of cells can be treated as a two-dimensional array, or matrix. In some simulation models, more elaborate data structures such as lists and trees are needed. One case in point is that of discrete event simulation, where lists are needed for the event list and waiting lines. These structures can be built in a spreadsheet, but they are contrived and inefficient.

For the most part, formulas in spreadsheet cells are static computations that are executed once when the cell is recalculated. Spreadsheets do not have convenient facilities to implement a while-loop or a for-loop. These can also be implemented, but the implementation is often inefficient and inflexible. For example, if a computation needs to be done 10 times, it can be implemented in a column or row of 10 cells. But, what if it needs to be done 100,000 times? Most spreadsheets do not allow a column this long. Moreover, how would you implement a loop that must be executed until a particular value is obtained? For example, in an actuarial ruin model, the value of the firm is computed until it is negative. Since you do not know the maximum number of periods to guarantee ruin (it might even be infinite), you do not know how many cells to include. VBA in Excel can be used to implement more complex logic, but this is a more advanced tool that is seldom used by casual spreadsheet users. So, models that require complex loops and other conditional computations may not work well in a spreadsheet.

Consider what a spreadsheet must do to recalculate. Formulas are stored in "source code." That is, the spreadsheet must interpret the formula before it can be executed. This interpretation action normally takes much longer than the execution. Some spreadsheets are sophisiticated enough to store the executable code so the interpretation does not have to be repeated each time, but it is nevertheless a much less efficient setup than one would have with a compiled language. Moreover, spreadsheets use much more of the computer's resources to support the elaborate user interface and provide all of the features. Thus, spreadsheets are inefficient in their use of memory. A model that is very large and/or requires long simulation runs would need to be programmed in a compiled language in order to execute in a feasible length of time or use a reasonable amount of main memory.

Finally, since the output data must be stored in the spreadsheet, usually in a column, the length of the output series is limited. In many spreadsheets, column lengths can be tens of thousands or even hundreds of thousands of cells. However, some models such as models that evaluate the reliability of highly reliable systems, require very large sample sizes - in the millions of observations. There are ways to circumvent this restriction. One could use multiple columns to store output data for the same performance measure, or one could accumulate sample statistics without actually storing the raw data. All of these solutions require a more complex approach to the simulation and result in more inefficiency in the execution of the simulation. When this is the case, it is appropriate to ask if another platform would be a better solution.

These four limitations seem to restrict considerably the range of models that can be implemented in a spreadsheet. However, many models are not subject to these restrictions, and they are often done to get "quick and dirty" results. This is the place where a spreadsheet really earns its bars. Prototypes can be quickly built and run in a spreadsheet. If the prototype shows that the simulation does not work well in the spreadsheet, then it can be moved to a more appropriate platform.

7 CONCLUSIONS

Spreadsheets provide a useful platform for many simulation models. The attractiveness of this platform comes from its availability, intuitive interface, ease of use and powerful features. Of the situations where a simulation can provide valuable information for decision making, a simulation actually built and used in only a small percentage of cases. The reasons for this underutilization of simulation are many. Sometimes, the managers or analysts are not familiar with simulation software. Spreadsheets provide a means to use familiar, intuitive software to do simulation, and if the computations for the model are already implemented in a spreadsheet, this approach avoids the trouble of moving them to another simulation platform.

Many simulations do not need to be extensive. They are designed to provide ball-park estimates and to show general system behavior. This is often true of financial models. These models can usually be implemented most efficiently in a spreadsheet. Simulation problems for which spreadsheets are a useful platform also include prototype models which are relatively small and designed to provide a proof of concept.

Commercial and free spreadsheets are continuing to be developed. Future versions will undoubtedly allow larger worksheets and perform computations more efficiently. As computing power continues to grow, the limitations to spreadsheet simulation will be removed and this platform will be even more attractive. Excel comes bundled with an optimization tool (solver). Perhaps there will be a time when it also comes bundled with a simulation tool!

REFERENCES

Alexopoulos, C., and A. F. Seila. 1998. Output data analysis. In *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice*, ed. J. Banks. New York: John Wiley.

Cheng, R. C. H. 1998. Random variate generation. In *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice*, ed. J. Banks. New York: John Wiley.

Fishman, G. S. 1996. *Monte Carlo concepts, algorithms and applications*. New York: Springer.

Grossman, T. A. 1999. Spreadsheet modeling and simulation improves understanding of queues. *Interfaces* 29 (3): 99–103.

L'Ecuyer, P. 1998. Random number generation. In *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice*, ed. J. Banks. New York: John Wiley.

Mattesich, R. 1961. Budgeting models and system simulation. *The Accounting Review* 36:384–397.

Seila, A. F., V. Ceric, and P. Tadikamalla. 2003. *Applied simulation modeling*. Belmont, California: Brooks-Cole.

Winston, W. L. 1996. *Simulation modeling using @RISK*. Belmont, California: Duxbury.

AUTHOR BIOGRAPHY

ANDREW F. SEILA is a Professor in the Department of Management Information Systems in the Terry College of Business at the University of Georgia. He has been involved with simulation teaching and research for more than 30 years, and has attended the Winter Simulation Conference since 1977, serving as Program Chair for the 1994 meeting in Orlando. Dr. Seila is a Fulbright Fellow and recognized authority on the use of spreadsheets for model representation and simulation. His professional interests include statistical methodology for simulation, modeling methodology and simulation applications, especially those in healthcare and finance. He is the author of over 50 refereed papers and co-authored the chapter on output analysis in the Handbook of Simulation. He has been a consultant to numerous businesses, consulting firms and public institutions. In recent years, Dr. Seila has expanded his interests to Internet Technology. He is currently Director of the Internet Technology Program at the University of Georgia. He is a member of the American Statistical Association, INFORMS and the Healthcare Information Management Systems Society. His e-mail address is <andy@ms.terry.uga.edu>.