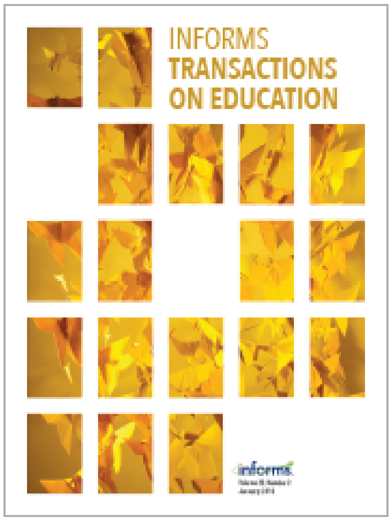

INFORMS Transactions on Education

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Spreadsheet Modeling of the G/G/c Queuing System without Macros or Add-Ins

Stephen C. Hora,

Spreadsheet Modeling of the G/G/c Queuing System without Macros or Add-Ins

Stephen C. Hora
School of Business and Economics
University of Hawaii at Hilo
200 W. Kawili St., Hilo, HI, 96720

Hora@Hawaii.edu

ABSTRACT

Spreadsheets are have become a popular tool for teaching business students about waiting line systems. Modeling queuing systems with multiple servers, however, is somewhat difficult and ordinarily accomplished by using macros or by having multiple templates, each template with a different number of servers. Here, it is shown how native Excel array functions can be used to create a spreadsheet model with the number of servers being an adjustable parameter.

Editor's note: This is a pdf copy of an html document which resides at http://ite.pubs.informs.org/Vol3No3/Hora/

Spreadsheet programs such as Microsofts Excel are widely used in teaching decision modeling to both graduate and undergraduate business students. There are a number of reasons why spreadsheets have become popular as a tool for teaching modeling. Among these reasons are:

1. Spreadsheets provide easier and more rapid to access to model development for the non-technical student by raising the algebraic curtain (Savage, 1997).
2. Spreadsheet skills provide a set of tools adaptable to a far wider set of situations than can be accommodated by specialized software (Powell, 2001)
3. Most managers will have only one analytic tool available on their personal computer and that tool will be a spreadsheet. (Winston, 1996)

An area of decision modeling that has partially lent itself to spreadsheet modeling is the simulation of waiting line or queuing systems. In this note, we limit our interest to the process-driven simulation where each row in the spreadsheet represents the history of an object or customer passing through the system (Evans and Olson, 1998). The changes in the system state reflect the arrival or departure of a unit. Templates are readily available for handling the G/G/1 model which has a single queue, a single server, and general arrival and service time distributions. Such templates can be found in a large number of textbooks such as Anderson, Sweeney, and Williams (1999).

The jump to the G/G/c model which has c servers can be accomplished in one of several ways. First, macros or user supplied functions can be programmed in a language such as VBA (Visual Basic Applications) or C++. Albright (2001) provides such an example using Excel and VBA. The companion CD-Rom for the Hillier and Hillier (2003) also contains such a model. Moreover, the VBA code is readily accessible so that the instructor and students have the ability to explore the underlying logic if desired. Limited student versions of commercial products such as Arena (Camm and Evans, 2000) or ServiceModel (Fitzsimmons and Fitzsimmons, 2001) are sometimes bundled with texts. Unfortunately, the complexity of these products and the time required to learn to use them often limits their use to the demonstration of applications developed outside the classroom.

Yet another approach, one which does not require programming or add-ins, is to build a spreadsheet where each server is represented by a set of columns in the spreadsheet and IF() statements are used to direct the object or customer to the appropriate server. An example of a twelve server spreadsheet can be found at http://www.bus.ualberta.ca/aingolfsson/simulation (Grossman and Ingolfsson, 2002).

In the case of programming or add-ins, the non-technical student does not have the experience of creating the missing logic needed to simulate the c server system while in the case of representing servers by additional columns, the template becomes inflexible with respect to the number of servers.

It is, however, possible to program the G/G/c system with c as a parameter in an Excel spreadsheet using only the native functions supplied with Excel and without special macros or functions. The logic required to do this is available through the use of array formulas. An array formula can perform multiple calculations and then return either a single result or multiple results. Array formulas act on two or more sets of values known as array arguments. Each array argument must have the same number of rows and columns. One creates an array formula in the same way that one creates other formulas, except the keys CTRL+SHIFT+ENTER are pressed simultaneously to enter the formula. Array formulas are briefly documented in the Excel help files. Further explanation is given in Walkenbach (2001).

For example, suppose that one wishes to average all strictly positive values in a range. To accomplish this, one would enter the formula

$$= AVERAGE(IF(range > 0, range))$$

into a cell and when finished CTRL+SHIFT+ENTER is pressed rather the just ENTER.Here, range is a reference to a group of cells such as the twenty seven cells A1:C9. After completing the entry, the expression shown on the formula bar will be enclosed in brackets {...} indicating that this is an array formula. Array formulas can be copied as any other formulas with absolute and relative addressing maintained. When an array formula is edited, however, it must always be entered with CTRL+SHIFT+ENTER lest it revert to a normal formula. Moreover, one cannot insert the brackets manually: this must be done with the required multiple key entry.

We are now ready to discuss the logic that makes implementation of the G/G/c possible. The needed insight is that one must look backward in time (or upward in the table) to determine how many units are currently in the system. Now, suppose that the n^{th} unit arrives at time T_n and that the preceding $n-1$ units were scheduled to end processing at times $P_1, \ldots, P_{n-1}$. We count the number of Pi that are greater than T_n using an array formula to find how many units are currently in the system. Call this number x. If this x is less than c, then a server is free and the current unit can seize a server and begin processing immediately. If not, the unit must be scheduled to seize a server at a time dictated by the first-come first-serve queue discipline. In order to find this time, we compute x - c which gives us the rank in ascending order of the x end of processing times greater than T_n. We then retrieve the value of this particular end of processing time and place it in the cell which shows the beginning processing time for the current observation. Retrieving the value of the Pi having the rank x - c is accomplished using the PERCENTILE function in Excel.

Suppose that the simulation begins with the first units history in row 2 of the spreadsheet and suppose that the arrival time of this unit is in column H and the end processing times are in column M. (These row and column designations correspond to an Excel implementation found at http://www2.Hawaii.edu/ hora/queuing. Note that this address is case sensitive.) Using these locations, the number of units already in the system when the tenth unit arrives, for example, is given by

$$Cell\ H11: \ COUNTIF(\$M\$11:M19, \text{“} > \text{”}\&H3)$$

Suppose that this value is stored in cell I11. The time that this tenth unit will begin processing is computed by

$$(= IF(I11 >= servers,$$
$$PERCENTILE(IF(\$M\$2:M10 > H11, \$M\$2:M10),$$
$$(I11 - servers)/)(I11-1)), H11)$$

where servers is the range name for the cell whose value is the number of servers. This formulation employs a combination of absolute and relative addressing so that the formula may be copied down with other formulas in the same row to simulate any number of arrivals. The PERCENTILE function is used in this implementation because it provides a convenient way to find the value in a set of service completion times that has a given rank. The logic of the above Excel formula is as follows:

- The outer IF statement, IF(I11 ¿= servers,., determines if the servers are all occupied.
  - If the servers are all occupied, the percentile function is used to calculate the time when a server will become available for this customer
  - If the servers are not all occupied, then service starts immediately (time H11).

Next, the logic of using the PERCENTILE function is explained.

- The PERCENTILE function will return the value in a range of n values which has the rank k by using the arguments PERCENTILE(range, (n - k)/(n-1)).

- The first argument in the PERCENTILE function of the formula is $IF(\$M\$2 : M10 > H11, \$M\$2 : M10)$ which is evaluated in the array mode and returns the subset of end of service times later than the H11. The cardinality of this subset is stored in cell I11 as explained above.
- The second argument determines rank of the end of service times that will be the arriving customers start of service time. For example, suppose that there are c = 3 servers and 8 customers in the system, 3 in service and 5 in the queue at the time the next customer arrives. The arriving customer will not begin service until the time when the customer with the end of service time ranked $3^r d$ (largest numerically or latest) among the 8 finish times.

A drawback to the formulation given here is that as more rows are added, the formula in each successive row must look back to the first arrival to determine the number of units in the system at the time of arrival and thus the number of end of service times to be considered for the nth arrival is $n-1$. When the number of arrivals simulated becomes large, such as one−thousand, there is a noticeable increase in the recalculation time. One way to work around this problem is to restrict the number of end of service times to be examined. For example, one might determine that it is not necessary to look further back than the last fifty units arriving before the current unit. For the $51^s t$ unit, the number of units in the system would be calculated as

$$(= IF(I52 >= servers,$$
$$PERCENTILE(IF(M2 : M51 > H11, M2 : M51),$$
$$(I52 - \$A\$8)/)(I52 - 1)), H52)$$

This formula can be copied downward so that all beginning of service times after the $50^t h$ are calculated using the end of service times of the most recent 50 arrivals.

The spreadsheet simulation of the G/G/c model with the above logic found at http://www2.hawaii.edu/ hora/queuing/ provides summary statistics of operating characteristics estimated from the simulation, and graphs of system state, queue length, service and wait times, and server utilization. The spreadsheet may be used as a pedagogical tool to achieve the following:

- The spreadsheet model can be used to demonstrate the G/G/c model and give students the ability to experiment with arrival rates, service time distributions, as well as the number of servers. However, add-ins and stand alone simulation packages can also be used in this way.
- Because the spreadsheet is coded in accessable native Excel functions, the spreadsheet can be used to teach the logic of the queuing model and allow students with even limited spreadsheet skills to modify the system. The student can then learn by guided exploration and experimentation.
- The spreadsheet also provides a venue for the introduction of Excel array functions. Students are rarely introduced to such functions and many instructors are not aware of their power.

With the logic given, the G/G/c model with an infinite queue and first-come first-served queue discipline is programmed in Excel in a straightforward manner. Balking, reneging, and finite queue length can also be incorporated into the model as discussed by Grossman (1999). Other features such as batch scheduling of appointments can also be accommodated. The author has been unsuccessful at creating such a spreadsheet for the finite calling population model. It is an open issue as to whether the native Excel functions are sufficiently rich to support such behavior. Perhaps not. Moreover, the author has not attempted to introduce other queue disciplines such as priority queues into the spreadsheet.

REFERENCES

Albright, S.C. (2001), *VBA for Modelers*, Duxbury, Pacific Grove, CA.

Anderson, D.R., D.J. Sweeney, and T.A. Williams (1999), *Contemporary Management Science with Spreadsheets*, Cincinnati, Ohio: South-Western College Publishing.

Camm, J.C. and J.R. Evans (2000), *Management Science and Decision Technology,* South-Western College Publishing, Cincinnati, Ohio.

Evans, J.R. and D.L. Olson (1998), *Introduction to Simulation and Risk Analysis*, Prentice Hall, Upper Saddle River, New Jersey.

Fitzsimmons, J.A. and M.J. Fitzsimmons (2001), *Service Management Operations, Strategy, and Information Technology*, McGraw-Hill, New York.

Grossman, T.A. (1999), Teachers Forum: Spreadsheet Modeling and Simulation Improves Understanding of Queues, *Interfaces*, Vol. 9, No. 3, pp. 88-103.

Grossman , T.A. and A. Ingolfsson (2002), "Graphical Spreadsheet Simulation of Queues" *INFORMS Transactions on Education*, Vol. 2, No. 2, http://ite.pubs.informs.org/Vol2No2/IngolfssonGrossman/

Hillier, F.S. and M.S. Hiller, *Introduction to Management Science A Modeling Case Studies Approach with Spreadsheets*,McGraw-Hill, New York.

Powell, S.G. (2001), Teaching Modeling in Management Science, *Informs Transactions on Education*, Vol. 1, No. 2, http://ite.pubs.informs.org/Vol1No2/Powell/Powell.php

Savage, S.L. 1997, Weighing the Pros and Cons of Decision Technology in Spreadsheets, *OR/MS Today*, Vol. 24, No. 1, pp.42-45.

Walkenbach, J. (2001), *Excel 2002 Formulas*,M&T Book, New York.

Winston, W.L. (1996), Issue in Education: Software Decisions, *ORMS Online Edition*, Vol. 23, No. 5, http://lionhrtpub.com/orms/orms-10-96/education.html