Proceedings of the 1995 Winter Simulation Conference
ed. C. Alexopoulos, K. Kang, W. R. Lilegdon, and D. Goldsman

USING *MATHEMATICA* TO AID SIMULATION ANALYSIS

Paul A. Savory

Industrial and Management System Engineering
University of Nebraska
Lincoln, Nebraska 68588-0518, U.S.A.

ABSTRACT

As computer power has increased, so has the capability of software developers to write programs that assist people with time-consuming tasks. *Mathematica* is such a program. The objective of this paper is to demonstrate how *Mathematica*, a symbolic programming environment, can be used to aid simulation analysis. In addition to a general discussion of *Mathematica's* uses, advantages, and disadvantages, several examples will be presented. The examples include using *Mathematica* for distribution fitting, queueing analysis, random number generation, and creating a surface plot for optimization.

1 INTRODUCTION

Computer software and hardware advances have had an important impact on discrete-event simulation. One result of these advances is *Mathematica*, a general software system and language for mathematical applications. Wolfram (1991) describes *Mathematica* as "A System for Doing Mathematics by Computer." *Mathematica* is an ideal general-purpose analysis tool since it integrates several features into a unified, interactive environment: numerical and symbolic calculations, functional, procedural, rule-based, and graphics programming. Additional features include:

- manipulating complicated symbolic expressions,
- graphing and animation,
- performing numerical calculations to arbitrary precision,
- its own programming language for constructing elegant and efficient programs,
- portability of programs to a wide range of computer platform without any modification.

The ability to perform symbolic as well as numerical manipulations places *Mathematica* with programs such as Derive, Macyma, Maple, and Reduce (Swain 1989). In contrast, TK!Solver and MathCAD are primarily numerical in their manipulations.

Mathematica is an interpreted language - it reads and evaluates an expression and then computes and prints out a result. Wickham-Jones (1994) describe that *Mathematica* is split into two parts. The kernel is the computational engine of the system that receives and evaluates all expressions sent to it. The front-end provides the program interface to the user and is concerned with such issues as how input is entered and how the results are displayed. Even though the front-end differs between computer platforms, the underlying kernel provides essentially the same set of functions.

The objective of this paper is to illustrate how *Mathematica* can aid a simulation analysis. The next two sections present numerical and graphical simulation examples. The paper concludes with some final comments on *Mathematica* and offers sources for further information.

2 NUMERICAL EXAMPLES

2.1 Kolmogorov-Smirnov Goodness of Fit Test

The first example uses *Mathematica* to test whether a set of data is modeled by a probability distributions. Figure 1 presents output resulting from testing a sample data set.

```
Mean of data: 33.88
SD of data: 1.87427
Length of data: 10
D+ : 0.138257
D- : 0.178486
D-Value = 0.178486
KS critical value is: 0.409
The data does follow the NORMAL distribution
```

Figure 1: Output from executing function KSTest[] which tests whether a set of ten data points (31, 31.4, 33.3, 33.4, 33.5, 33.7, 34.4, 34.9, 36.2, 38) follow a normal distribution. The function concludes that there is not enough evidence to reject the normality assumption

This output was generated using the function KSTest[], which applies the one-sample Kolmogorov-Smirnov Goodness of Fit test for determining if a data set follows a normal distribution. The function works by asking the user to enter the name of an ASCII data file and a level of significance for the hypothesis test. The function next computes K-S test statistics and calls the function KSTable[] to look up the critical K-S value. Be aware that KSTest[] only checks whether a data set follows a normal distribution. Minimal effort is required to extend the routine to test for additional distributions. Appendix A contains the code for the KSTest[] and KSTable[] functions.

2.2 Queueing Theory

One of *Mathematica's* advantages is its ability to effectively manage equations. To see this, consider the function Queue[] which implements the queueing formulas for describing an M/M/S queue. The function works by prompting a user to specify the arrival rate (e.g., 0.1), the service rate (e.g., 0.2) and the number of servers (e.g., 3) to a queue. Using this information, the function computes M/M/S summary statistics. Figure 2 presents the results from running function Queue[]. The *Mathematica* code is given in Appendix B. It is significant to note that only fourteen lines of code is required. Programming the same algorithm in a traditional programming language would required approximately 100 lines.

```
Lambda = 0.1  Mu = 0.2  S = 3
rho = 0.166667
W = 5.0303   L = 0.50303
Wq  = 0.030303    Lq  = 0.0030303
```

Figure 2: Output resulting from running the function Queue[]. The arrival rate, service rate, and number of servers are specified as 0.1, 0.2, and 3, respectively.

3 GRAPHICAL EXAMPLES

Mathematica represents graphics as expressions that can be manipulated. Sometimes one can achieve a desired graphical effect by using the options of the built-in plotting commands; at other times, the only way to accomplish a goal may be to modify the expressions returned by a plotting command or even to create an expression from scratch.

3.1 RANDU Generator

As discussed by Law and Kelton (1991), the RANDU random number generator,

$$Z_i = 65{,}539 Z_{i-1} (\text{mod } 2^{31})$$
$$Z_0 = 123{,}456{,}789$$

is biased when random numbers are used in groups of three. Using *Mathematica*, it is easy to observe this dependency. Figure 3(a) shows the default view of 6000 tuples generated from the RANDU generator. Based only on this view, the generator appears to be sufficiently random. By varying the orientation parameters, Figures 3(b) and 3(c) show a different view of the same cubic. In Figure 3(d), the dependency among the three-dimensional lattice structure is clearly evident. The code for the RANDU generator and for the plotting the different views is given in Appendix C.

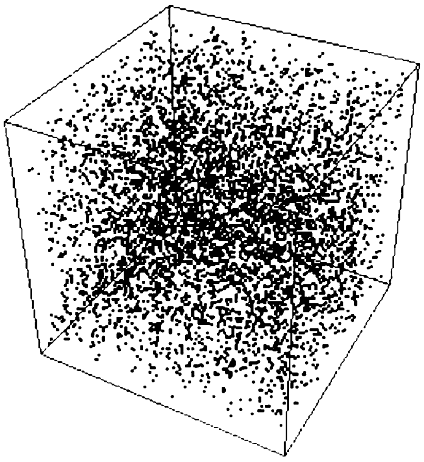

Figure 3(a): Default view of 6000 tuples from the RANDU generator.

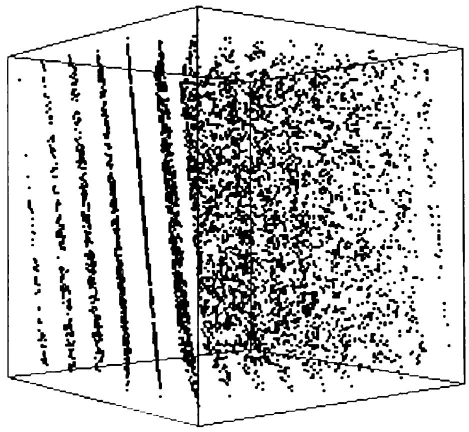

Figure 3(b): Different view of (a). Note the formation of the parallel planes.

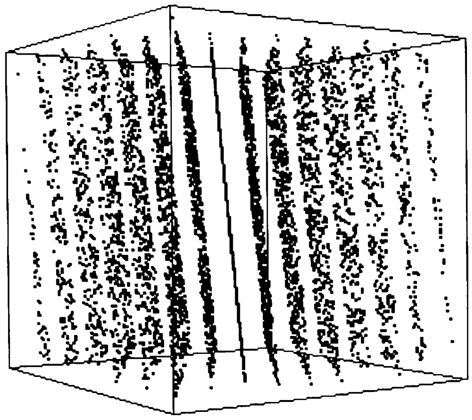
Figure 3(c): Different view of (a). The parallel planes are becoming more significant.

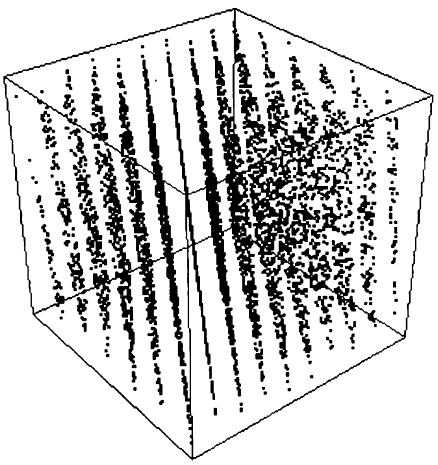
Figure 3(d): Different view of (a). Fifteen parallel planes are clearly evident.

3.2 Surface Plot

Experimental design permits evaluating alternative system designs by varying combinations of the decision variable settings. Simulation is primarily concerned with deciding how to develop a set of simulation runs that allow an analyst the ability to select which variables to measure and how to test if they significantly affect the output/response of the model. That is, perform optimization with simulation. *Mathematica* can aid in this process.

Figure 4(a) presents the response surface resulting from running 35 simulation models in which a number of expert and apprentice mechanics are varied from 0 to 5 (the 0,0 combination was not run) to estimate the cost of repairing a group of machine (Hoover and Perry 1989, Illustrative Problem 10.7). Each model was replicated five time for each combination of mechanics resulting in a total of 175 simulation runs. Figure 4(b) is an alternate view of the same response surface plot. Appendix D contains the data and code used to create the two plots.

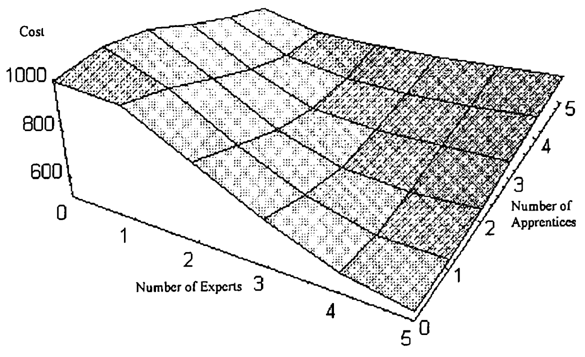

Figure 4(a): Response surface generated from plotting the repair cost for various levels of expert and apprentice mechanics. For instance, 5 experts and 1 apprentice mechanic has an approximate cost of $550.

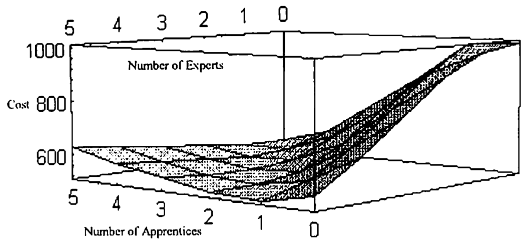

Figure 4(b): Another view of (a).

4 LEARNING MORE ABOUT *MATHEMATICA*

For more information on *Mathematica*, Wolfram Research, Inc. can be contacted at:

Phone: 1-800-441-MATH
E-Mail: info@wri.com
World Wide Web: http: //www.wri.com/

Wolfram Research also maintains MathSource, an electronic repository of *Mathematica* material. This can be accessed by e-mail, anonymous ftp, Gopher, or World Wide Web at mathsource.wri.com. In addition, *Mathematica* is a frequent topic of discussion on many Internet newsgroups; the sci.math.symbolic is one.

5 FINAL COMMENTS

Computers have brought about a fundamental change in the nature of research and in science and engineering education (Gaylord, Kamin, and Wellin 1993). One of these changes is *Mathematica,* a useful tool for those who do quantitative analysis, symbolic calculations and manipulations, and need to visualize functions or data. *Mathematica* has enormous power for aiding a simulation analysis. In addition to the examples presented in the paper, *Mathematica* provides functions for computing confidence intervals, performing hypothesis tests, estimating regression lines, solving linear programming problems, and assisting with matrix manipulations.

Mathematica is not without problems. Its key disadvantages is that it is slow. Based on my own experience, I estimate that a *Mathematica* program runs 50 to 100 times slower that a compiled version written in a traditional programming language. This time delay is primarily due to the interpretive nature of the software in that a user's input must be read and evaluated. An additional drawback is that all programs must be run within the *Mathematica* environment. A final hindrance is the lack of sophisticated user input and output routines. The current functions do not compare to those offered by traditional programming languages. These trade-offs are easily offset by *Mathematica* capability to perform tasks that would be impossible to program otherwise.

Mathematica does for simulation what robots do for manufacturing: carry out menial chores. If you need a flexible and extremely useful software for performing mathematical computations, *Mathematica* is a good choice.

ACKNOWLEDGMENTS

The author wishes to thank Ted McGlynn, a graduate student at the University of Nebraska for developing the KSTest[] and KSTable[] functions.

APPENDIX A: NORMALITY TEST

```
(* Kolmogorov-Smirnov test for whether data set is normally
   distributed *)
KSTest[]:=
 Module[{},
   (* Load statistics package *)
   Needs["Statistics`DescriptiveStatistics`"];

   (* Prompt for and read in the data *)
   datafile = InputString["Enter Data File to Test"];
   datalist=ReadList[datafile,Number];

   (* Sort data in ascending order *)
   data = Sort[datalist];

   (* Finds n, the number of data points *)
   leng=Length[data];
   (* Compute Fn(Yi) *)
   Y = Table[i/leng,{i,1,leng}];

   ksvalue = Input["Input alpha value to test at:\n
         1 = 0.20\n  2 = 0.10\n 3 = 0.05\n
         4 = 0.02\n  5 = 0.01"];
   (* Compute statistics *)
   ave=Mean[data];
   sd=StandardDeviation[data];

   (* cdf for normal distribution *)
   std = Table[N[((data[[i]]-ave)/sd),4],{i,1,leng}];
   X=Table[N[NIntegrate[.39894*Exp[-z^2/2],
         {z,5,std[[i]]}],4],{i,1,leng}];

   (* Compute *)
   Mm =Abs[Y-X];
   Nn = Table[Abs[X[[i]] - Y[[i-1]]],{i,2,leng}];
   Dplus = Max[Mm];
   Dneg = Max[Nn];
   Dval = Max[Dplus,Dneg];

   (* Look up KS critical table Value *)
   KSTable[];

   Print["Mean of data: ",ave];
   Print["SD of data: ",N[sd]];
   Print["Length of data: ",leng];
   Print["D+ : ",Dplus];
   Print["D- : ",Dneg];
   Print["D-Value = ",Dval];
   Print["KS critical value is:  " ,value];

   If[Dval>value,
     Print["The data does NOT follow the NORMAL
          distribution"],
     Print["The data does follows the NORMAL
          distribution"]
   ];  ]

KSTable[]:=
 Module[{},
   (* K-S Table *)
   ks =
   {{.900,.950,.975,.990,.995},{.684,.776,.842,.900,.929},
    {.565,.636,.708,.785,.829},{.493,.565,.624,.689,.734},
    {.447,.509,.563,.627,.669},{.410,.468,.519,.577,.617},
    {.381,.436,.483,.538,.576},{.358,.410,.454,.507,.542},
    {.339,.387,.430,.480,.513},{.323,.369,.409,.457,.489},
    {.308,.352,.391,.437,.468},{.296,.338,.375,.419,.449},
    {.285,.325,.361,.404,.432},{.275,.314,.349,.390,.418},
    {.266,.304,.338,.377,.404},{.258,.295,.327,.366,.392},
    {.250,.386,.318,.355,.381},{.244,.279,.309,.346,.371},
    {.237,.271,.301,.337,.361},{.232,.265,.294,.329,.352},
    {.226,.259,.287,.321,.344},{.221,.253,.281,.314,.337},
    {.216,.247,.275,.307,.330},{.212,.242,.269,.301,.323},
    {.208,.238,.264,.295,.317},{.204,.233,.259,.290,.311},
    {.200,.229,.254,.284,.305},{.197,.225,.250,.279,.300},
    {.193,.211,.246,.275,.295},{.190,.218,.243,.270,.290},
    {.187,.214,.238,.266,.285},{.184,.211,.234,.262,.281},
    {.182,.208,.231,.258,.277},{.179,.205,.227,.254,.273},
    {.177,.202,.224,.251,.269},{.174,.199,.221,.247,.265},
    {.172,.199,.221,.247,.265},{.170,.194,.215,.241,.258},
    {.168,.191,.213,.238,.255},{.165,.189,.210,.235,.252  };}
   (* Determine critical value *)
   If[leng<=40,value = ks[[leng,ksvalue]],
    If[ksvalue==1,value=(1.073/leng^.5),
    If[ksvalue==2,value=(1.2239/leng^.5),
    If[ksvalue==3,value=(1.3581/leng^.5),
    If[ksvalue==4,value=(1.5174/leng^.5),
        value=(1.6276/leng^.5)]
    ]]]; ]
```

APPENDIX B: M/M/S QUEUE

```
(* Summary statistics for M/M/S queue *)
Queue[]:=
    Module[{},
    L=Input["Input the Arrival Rate "];
    m=Input["Input the Service Rate "];
    s=Input["Input the Number of Servers "];
    (* rho *)
    r=L/(s*m);
    (* probability in state zero *)
    Po=1/((Sum[((L/m)^n)/Factorial[n],{n,0,(s-1)}])+
        (((L/m)^s)/(Factorial[s]*(1-(L/(s*m))))));
    (* average length of queu e*)
    Lq=(Po*((L/m)^s)*r)/(Factorial[s]*((1-r)^2));
    (* average length of system *)
    LS=Lq+(L/m);
    (* average wait time in queue *)
    Wq=Lq/L;
    (* average wait time in the system *)
    WS=Wq+(1/m);
    Print["Lambda = ",L,"  Mu = ",m,"  S = ",s];
    Print["rho = ",r];
    Print["W = ",WS,"  L = ",LS];
    Print["Wq  = ",Wq,"  Lq  = ",Lq];
]
```

APPENDIX C: RANDU GENERATOR

```
(* RANDU random number generator *)
Randu[]:=
  Module[ {},
    (* Initial seed *)
    z[0]=123456789;
    (* Generate uniform random numbers *)
    For[i=1,i<=6500,i++,
      z[i]=Mod[(65539*z[i-1]),(2^31)];
      U[i]=N[z[i]/(2^31)];
    ];
    (* Create 6000 overlapping tuples *)
    data=Table[Point[{U[i],U[i+1],U[i+2]}],{i,1,6000}];
    (* Figure 3-a *)
    view1=Show[Graphics3D[data]];
    (* Figure 3-b*)
    view2=Show[Graphics3D[data],ViewPoint->{-2,-2.5,0}];
    (* Figure 3-c*)
    view3=Show[Graphics3D[data],ViewPoint->{-2,-3,0}];
    (* Figure 3-d*)
    view4=Show[Graphics3D[data],ViewPoint->{-2,-2.5,2}];
```

APPENDIX D: RESPONSE SURFACE

```
(* Response surface for data set *)
(* average cost of the various mechanic combinations *)
data= {{1000,990,850,720,600,550},
      {980,890,740,620,550,550},
      {900,770,630,550,540,570},
      {780,650,550,540,560,590},
      {660,560,530,540,570,610},
      {560,520,530,560,590,630}};
(* Figure 4-a *)
w1=ListPlot3D[data,MeshRange->{{0,5},{0,5}},
        Boxed ->False];
(* Figure 4-b *)
w2=ListPlot3D[data,MeshRange->{{0,5},{0,5}},
        ViewPoint->{-3.5,4,.5}];
```

REFERENCES

Gaylord, R., S. Kamin, and P. Wallin. 1993. *Introduction to Programming with Mathematica.* New York: Springer-Verlag.

Hoover, S. and R. Perry. 1989. *Simulation - A Problem Solving Approach.* New York: Addison-Wesley.

Law, A. and D. Kelton. 1991. *Simulation Modeling and Analysis.* 2d ed. New York: McGraw Hill.

Swain, J. 1989. Review of MATHEMATICA. *OR/MS Today*, 16: 34-36.

Wolfram, S. 1991. *Mathematica: A System for Doing Mathematics by Computer.* 2d ed. New York: Addison-Wesley.

Wickham-Jones, T. 1994. *Mathematica Graphics.* New York: Springer-Verlag.

AUTHOR BIOGRAPHY

PAUL A. SAVORY is an Assistant professor in Industrial and Management Systems Engineering at the University of Nebraska - Lincoln (UNL). Prior to joining the faculty of UNL, he earned a Ph.D. in Industrial Engineering from Arizona State University, a Master's degree in Operations Research and a Bachelor's degree in Computer Science, both from Oregon State University. In addition to having taught university and industrial courses in simulation, operations research, and applied statistics, he has held positions as a software engineer and a quality control inspector.