CHAPTER SEVEN

TWO MODELING EXERCISES

This section will feature two detailed problems which are practical applications of mathematical modeling. The faithful reprint of these two clear examples was made possible through permission from Byte and our thanks are extended for their generous willingness to share them. Each article will be introduced by a commentary which reviews the key concepts that are necessary to understand the modeling procedure. The commentary also suggests appropriate enrichment activities for student research activities involving mathematical modeling. It is hoped that these articles will encourage you to attempt your own mathematical models, perhaps with teams of students attacking relevant problems and dividing responsibilities appropriately. For example, a team of three students could include a computer specialist, proficient mathematician, and effective writer. Of course, each could qualify in all three categories.

PROBLEM 1 - ANALYZING BUSINESS RISKS

Introduction

Mathematical modeling is a natural way to assess risk in starting any new business. This article uses ***Monte Carlo analysis***, which simply means that we utilize random numbers and assumptions that some "expert" makes connected with the probabilities of levels of sales.

The sales distribution table in the article makes use of probabilities (Chapter One) and requires the knowledge of the inverse transform method (Chapter Five) and the generation of random numbers (Chapter Four) for the purists who wish to improve on their computer's random number generation. The cumulative distribution is necessary in order for us to obtain a level of sales corresponding to each random number $x \in (0, 1)$.

Since it is unlikely that our random numbers will correspond exactly to the table values, the author uses linear interpolation for intermediate values. To illustrate, consider the projected sales X of a newly published book.

X	P(X)
1,000	1.00
10,000	.5
20,000	.7
100,000	0

We project with 100% confidence (or probability) sales of 1,000 or more. We are 70% confident of sales of 20,000 or more. Likewise, there is zero likelihood of sales exceeding 100,000. In order to determine sales for our Monte Carlo simulation, we start with a random number P between 0 and 1. If it falls between two table values P_1 and P_2 with corresponding sales volume V_1 and V_2, linear interpolation would calculate the volume corresponding to P by the given formula:

$$V = \frac{(P_1 - P)}{(P_1 - P_2)} \cdot (V_2 - V_1) + V_1$$

To illustrate, suppose our first random number was .65. To determine V for P = .65, let $V_1 = 10{,}000$, $P_1 = .5$, $V_2 = 20{,}000$, $P_2 = .7$. By substituting,

$$V = \frac{(-.15)}{(-.2)} (10{,}000) + 10{,}000 = 17{,}500$$

The author uses the same method to establish the discounted selling price (line 62040 and line 62050 of listing 1), the unit distribution cost (line 62070 and line 62080 of listing 1), and the fixed cost (line 62130 and line 62140 of listing 1).

Putting these into one equation gives us the author's main objective:

G(Gain) = SP • SV - UC • SV - FC

[Gain = selling price • sales volume - unit cost • sales volume - fixed cost]

The final point that the author makes is that taking ten runs of 200 samples gives different information from a single run of 2000 samples. For the 2000 samples the mean is 27,600, but the standard deviation is 13,400. If we take ten runs with 200 samples in each, we obtain a different result, namely $\bar{x}$ = 27,600 (the same) but the standard deviation is 1000.

If we use classical statistics and assume that σ = 13,400 from the result of our single run with 2000 samples, our sample of ten averages should have the same mean (which it does), but a sample standard deviation $\sigma_x = \sigma/\sqrt{n} = 13{,}440/\sqrt{200} = 947.5$.

Let us apply classical statistics to analyze the sample standard deviation of 1000 coming from the average of the ten averages. The standard formula for estimating the population variance σ^2 from the sample variance s^2 is:

$$\frac{(n-1)\,s^2}{\chi^2_{\alpha/2}} < \sigma^2 < \frac{(n-1)\,s^2}{\chi^2_{1-\alpha/2}}$$

Since we are taking ten runs to estimate σ^2, we let df = 10 - 1 = 9. Substituting, we obtain: (let α = .05)

$$\frac{9 \cdot 1000^2}{19.023} < \sigma^2 < \frac{9 \cdot 1000^2}{2.700}$$

$$473.111 < \sigma^2 < 3{,}333{,}333$$

$$687 < \sigma < 1825$$

Clearly our sample result of 1000 is in accord with our population σ (estimate) of 947.5. Of course, we are dealing with simulation data and have no exact way of determining population statistics.

The computation of standard deviation may also be affected by the fact that simulation outputs are always correlated. The random number generator for my most recent study of the behavior of dependent random variables had an autocorrelation of (-.03). The likely reason for this consistent result was the non-random character of our typical linear congruential random number generators.

But another question arises. How can we adjust to the correlatedness that arises in simulation output data? To illustrate this concept, let us try to make an adjustment of our statistical thinking to capture dependence and its influence on our estimates. Let us concentrate on analyzing the sample standard deviation of our simulation output data. Our objective is to estimate $\sigma^2 \cdot \overline{X}(n)$, adjusting to the inevitable correlatedness that is part of any simulation output data. We must assume that our random variables $x_1, x_2, \ldots, x_n$ are from a covariance stationary stochastic process. This means that

$\mu_1 = \mu$ for $i = 1, 2, \ldots,$ $\quad -\infty < \mu < \infty$

$\sigma_i^2 = \sigma^2$ for $i = 1, 2, \ldots,$ $\quad \sigma^2 < \infty$ and

$c_{i, i+j} = \text{COV}\ (x_i, x_{i+j})$ is independent of i for $j = 1, 2, \ldots$ We use the key formula:

$$\sigma^2 \cdot \overline{X}(n) = \sigma^2 \cdot (1 + 2 \sum_{j=1}^{n-1} (1 - j/n)\ P_j)/n$$

We must estimate P_j by the following formula:

$$P_j = \hat{P}_j = \frac{\hat{C}_j}{s^2(n)} \qquad \text{where}$$

$$\hat{C}_j = \sum_{i=1}^{n-j} \frac{[X_i - \bar{X}(n)]\,[X_{i+j} - \bar{X}(n)]}{n - j}$$

There are other estimates for P_j, but all are biased $[E(\hat{P}_j) \neq P_j]$, all have large variances, and all are correlated with other correlation estimates $[COV(\hat{P}_j, \hat{P}_k) \neq 0]$.*

Try programming to obtain an estimate $\sigma^2 \cdot \bar{X}(10)$ for the preceding simulation. Let us review the notation with a table.

Sample Means	Sample Standard Dev.	Sample Size
X_1	S_1	$n_1 = 200$
X_2	S_2	$n_2 = 200$
.	.	.
.	.	.
.	.	.
X_{10}	S_{10}	$n_{10} = 200$

Let $\sigma^2 = 13,400^2$, our population estimate. Let $\bar{X}(n)$ be the mean of our ten separate means. This process if computationally cumbersome, which is why we rely on the computer to estimate σ^2 $[\bar{X}(10)]$. We start by computing $\hat{P}_1, \hat{P}_2, \ldots, \hat{P}_q$. We then find

*Law, A. and Kelton, W. *Simulation Modeling and Analysis* (McGraw Hill: New York), pp. 146-147.

$$S^2(10) = \frac{\sum_{i=1}^{10} [X_i - \bar{X}(10)]^2}{n}$$, our $S^2(n)$. We next calculate $\hat{P}_j = \frac{\hat{C}_j}{S^2(n)}$.

This gives us estimates for $P_1, P_2, \ldots P_{n-1}$ in our equation of interest. We substitute these estimates into our equation:

$$\sigma^2 [\bar{X}(n)] = \sigma^2 \cdot (1 + 2 \sum_{j=1}^{n-1} (1 - j/n) P_j) /n$$

to obtain our desired result, an estimate $\sigma^2 [\bar{X}(n)]$. Finally, compare $\sigma^2 [\bar{X}(n)]$ with the results of our simulation $S^2 [\bar{X}(10)] = 1000^2$.

You have now sampled one of the biggest problems in statistics, how to adjust statistics that are based on independence assumptions to the real world situations that are nearly always dependent. Consider the following quote by a leading mathematician:

> "Thus one comes to perceive in the concept of independence, the first germ of the true nature of problems in probability theory." (Kolmogorov)

We will be looking more closely at dependence in one on the student research problems next chapter.

As a closing point that merits discussion and scrutiny, one should look at the two markedly different standard deviations of $S_2 = 1000$ and $S_1 = 13,400$. Which one would better serve the prospective venture analysts? Certainly if one did not have a guaranteed ten years to stay in the business, it would be misleading to use a statistic based on ten runs of 200 samples each. It would be sounder and preferable to use population statistics (their estimate based on a sample of 2000 runs).

For example, if you had one child and wanted to know what the chances were that the child's IQ was between 100 and 116, assuming $\sigma = 16$, you would use the well known population

statistic $\mu = 100$. The z value of 1.0 corresponds to a probability of .3413. Therefore, there is a 34.13% chance of the child having an IQ [on our woefully inadequate tests of IQ] between 100 and 116.

Similarly, if our venture analyst wanted to be 68% confident of his/her first year profit, we would use the same concept and calculate the result: $\bar{X} \pm 1$ [13,400]. Our confidence interval would be 27,600 ± 13,400 or (14,200, 41,000). Author Pat Macaluso prefers the smaller standard deviation that emerges from the average of the ten averages, it appears that the prospective venture analyst would be better served by the standard deviation of 13,400. After all, some ventures may be put out of business if their first year profit were only 14, 200.

* * * * * * *

A RISKY BUSINESS

AN INTRODUCTION TO MONTE CARLO VENTURE ANALYSIS

A simple method for analyzing business risks

by Pat Macaluso

[Byte, March 1984]

A business enterprise is aptly named a *venture*. It is a ship launched on a sea of uncertainty. The business of business is the taking of intelligent risks. Precious resources are committed to what can only be a hope of future gain. To reduce the risk, it would be helpful and profitable to have some insight as to possible future events.

It turns out that future prospects, elusive as they are, can be estimated in a way that is surprisingly useful for business purposes. The method involves four steps: (1) formulate a model

of the venture; (2) distribute appropriate data in the model; (3) sample from the model data; (4) analyze the sample.

The Monte Carlo Method

Aside from an investor's knowledge of a proposed venture, the Monte Carlo method requires nothing more than a personal computer and a program that is almost trivial in its simplicity. I'll use an example to illustrate how it works. We take at random a possible selling price, a possible sales volume, and so on. The selections are made from a range of possible values in each case according to the estimated probability of their occurrence. From this sample data, a corresponding outcome is calculated. This process is repeated for the entire range of possibilities. The resulting collection of outcomes is then arranged in sorted order. Examination of this distributed result yields information on the range of future outcomes and the relative chance of their occurrence. We'll see how this is done in detail later on, but the analysis might run something like this: in this business venture there is a 10 percent chance you will lose your shirt; a 65 percent chance you will achieve a 15 percent return on investment after taxes; a 5 percent chance you will really clean up, and so forth.

Such a formulation, even if stated less colorfully or dressed up in graphs and tables, may sound strange or even unsettling. Wouldn't it be simpler and more understandable to take the most likely selling price, sales volume, etc., and come up with the most likely result? Unfortunately that is not the case. Such an approach tends to underestimate the risks. It also throws away most of the information we have that bears on future possibilities.

In projecting sales figures for a product, sales managers can say that a realistic sales level will be 50,000 units. They can also say that there's little chance of 80,000 and no chance of more than 90,000 units being sold. Further, they might add that it is very likely that at least 15,000 units

and quite certain that 5000 units will be sold. The manager is expressing a wealth of hard information along with his uncertainty. He is weighing the size of the total market, the effect of competition, replacement rates, captive markets, limits on plant capacity, and so on. In other words, estimates by an informed person, though couched in uncertainty, contain valuable information that bears on future outcomes.

Faced with an investment decision, would you throw such information away, especially if it is easily expressed in a form suited to quantitative analysis? The most likely value or single-point methods do just that. They are quite inferior to the Monte Carlo sampling approach that allows us to use the extra information.

The most likely value method of risk analysis has tended to persist since calculations could be made by hand and managers felt they understood the result. It certainly seemed more definite and less threatening than a distribution that told of possible bad outcomes as well as the desired profitable ones. Times have changed. Many executives, aided by easier access to computers, have responded with increased sophistication as the safety of investments has become harder to gauge.

We can better understand the nature of the Monte Carlo method with the aid of a simple example. Suppose we wanted to determine the chance of getting "snake eyes," or two ones, in the roll of dice. We can calculate this precisely from probability theory as being one out of 36 tosses on the average. But what if we had no theoretical solution, as is the case with business ventures? There is another way to examine the chance of snake eyes. We can tally the result of thousands of rolls of the dice. Even better, we can simulate it on a computer. The result will, in general, not be exactly 1 in 36 but it will tend to approach it more and more closely as the size of the sample

increases. We will have performed a random-sampling experiment. It is easy, it works, and it is more than adequate when applied to business situations.

How Do I Get Started?

We present here three useful items for anyone who wishes to explore this method of risk analysis: (1) a simple technique for building a model of the venture; (2) a way to construct a sample from a distribution that is universal in its application; (3) a complete but elementary venture-analysis program to carry out the calculations. The program (see listing 1) can serve as a core upon which a more sophisticated or customized system can be built. More details are supplied in the author's book (see reference at the end of this article) but all the essentials are provided in this article.

A word of caution is in order. Compared to the sometimes mind-boggling complexity of actual business ventures, the model shown here will appear quite simplistic. Perhaps crude would be a better description. What good then is such an approach, aside from tutorial use? The answer may be somewhat surprising unless you are already well into this subject. Simple models work remarkably well to the extent that they embody the essentials of the enterprise they represent. There are advantages to stripping away nonessentials. At the very least, the act of analysis sharpens our understanding of the venture. It reveals what weaknesses may exist in the data, which factors are more critical, and so on.

The outcome of a simulation is a way to integrate the complexity of distributed values in a model. It is a tool that helps the entrepreneur make the actual decision. That decision will weigh factors that the model did not or could not include. The user must also decide exactly what the problem is and frame the model accordingly. For example, is a product to be made in new,

expanded, or shared facilities? If the latter, how will the effect of displaced products and production turnaround be handled in the model? Is the venture analyzed on its own merits or in comparison with other projects? It is clear that the real work is done both before and after the simulation. The program is just a convenient calculation tool.

Building a Model

The example we will use here is an estimate of the gain (or loss) to be expected in the production and sale of an item with a small-to-modest market. The model we will use is a simple one. Our purpose is to illustrate the technique without getting lost in the details. This will make it easier to highlight the possible weaknesses, as well as the strengths, of this approach.

As a starter, we need a model in the form of an equation that represents the venture. How do we develop such an equation? We can start at the top by noting that our objective, expected gain, can be taken as the difference between total income and total expenses before taxes. Thus: Gain = Income - Expense. We proceed with our top-down design by detailing income as: Income = Selling Price x Sales Volume, or SP•SV using BASIC notation. Likewise we can assume that: Expense = Fixed Cost + Variable Cost, or FC + VC. The latter can be expressed as: Variable Cost = Unit Cost x Sales Volume, or VC = UC•SV. Putting it all together, we have:

$$G = SP \cdot SV - UC \cdot SV - FC$$

Thus, the formidable phrase "formulate a model of the venture" requires nothing more than a simple equation. Some arbitrary decisions will have to be made on just variable costs versus fixed costs in a way that reflects the quantity produced rather than the quantity sold. Likewise, it may be necessary to use a fraction of the anticipated selling price to allow for discounts. Every venture has

its own scenario. The user needs to adjust either the model or the data to allow for the specific case. This can be done in stages by continuing the top-down expansion of the model.

Constructing a Distribution

Each value that may be assigned to a variable, for example, the sales volume, has a probability of occurrence associated with it. We have seen an example of this in the game of dice where a die has possible values of one through six, each with an equal chance of occurrence. The collection of values and their associated probabilities for a given variable is called a distribution. In this case, many tosses of a single die produce a uniform distribution, since each value has the same chance of occurrence.

A more common type of distribution is represented by the heights of people. We usually find many people in the five to six foot range, somewhat fewer in the four to five or the six to seven foot range, and many fewer at other heights. If we tabulate and plot the count of the heights, we get something like a bell-shaped curve. This is called a normal or Gaussian distribution. There are a rather large number of different kinds of formal distributions, each with a different shape. Some of them represent actual collections of specific things quite well. The problem with formal distributions is that they require various constants to be determined and specified. It would also be necessary to select a suitable distribution and perform special calculations or transformations to use them.

We present here a simple distribution that avoids all of these complications. It will approximately fit your data, whatever it may be. To illustrate its construction and use, we will consider the number of units of a product that might be sold from a total population of 15,000 units.

Numer Sold	Probability	Meaning
4,000	1.00	Certain to sell 4,000 or more
8,000	0.85	85 percent change of 8,000 or more sales
12,000	0.50	Even change of 12,000 or more sales
15,000	0.20	20 percent chance of selling all units
15,000	0.00	No chance of sales exceeding units produced

Several points of interest should be noted. First, we start and end with two certainties, namely 100 percent and 0 percent situations. This is not at all difficult for someone who knows the business being simulated. In our example, the user knows from experience that the established outlets will absorb at least 4,000 units. The other limit of 15,000 units is also quite certain. In this example, the user has decided to accept a 20 percent risk of loss of sales in excess of 15,000 units, perhaps counting on a second production run if all goes well. This illustrates an immediate advantage of this method of representing distributions. It naturally and easily takes care of cutoffs at both extremes, including special situations such as captive outlets and lost opportunity.

Another important factor is that the distribution is in cumulated form. This is a great advantage since other distributions must be converted to cumulative distributions before they can be used practically. A cumulative distribution in effect adds up all the chances on one side of any particular value. Instead of saying there is one chance in six of getting a four on the toss of a single die, we say there is a 50 percent chance of getting a four or higher. We can see why we need cumulative estimates in the case of a "continuous" distribution such as the number of units sold. It would be difficult to deal with the 1 out of 10,000 chance that we will sell exactly 8,000 units. Much the same applies to the 35 percent chance that between 8,000 and 12,000 units will be sold.

It's much easier to deal with the 85 percent change that sales will be greater than 8,000. This will become evident when we see how the actual calculations are carried out.

A close look at the estimated sales and their probabilities shows that they are not symmetrically distributed around the 50 percent (or even) chance point. This is often a problem with formal distribution functions since there are many varieties of skewed or nonsymmetric distributions. Again we have an advantage in that our estimate of the probable distribution is directly applied. Another feature is that the 50 percent estimate need not be one of the five cumulative points. The three middle estimates can be any that are suited to the data. It is not uncommon for the second and fourth estimates to be something like 95 percent and 5 percent or 90 and 10, etc. These correspond to easily visualized chances such as one out of twenty, one in ten, and so on. These might correspond to a pessimistic and optimistic estimate in addition to a more central or fifty-fifty estimate.

Sampling from the Distribution

We now have a distribution, that is, a quantitative expression of the uncertainties affecting a variable in our model of the business venture. The question is: how do we use it? In our example, the Monte Carlo method requires us to select a random one of the possible sales levels between 4,000 and 15,000. The random selection must conform to the distribution, which is not uniform and for which we have only five points. The simplest method is to assume that the distribution between two successive levels or points is uniform. This allows us to obtain intermediate values by simple or linear interpolation. A plot of our sales volume distribution in Figure 1 illustrates the process. If we compare the tabulated distribution with the plot in Figure 1 and with the formula V = (((P1-P)/(P1-p2))*(V2-V1)) + V1, we should be able to see how this works. Thus, using the BASIC

random number generator, a random probability P between P1 and P2 selects a sales volume V between V1 and V2.

There are several ways you could arrange such tables relating possible values to their probability of occurrence. The important thing is to be consistent so that your tables, formulas, program, interpretation, and logic all hang together. In the sample venture analysis we are developing here, we will use the format of our sales volume example. The specifications for a variable are: (1) use five levels with probabilities of 1 and 0 at the extremes; (2) estimate values in terms of "equal to or greater than"; (3) always start with the estimate for which the probability equals 1. That's almost all there is to it.

We need to develop similar tables for each of the variables in the equation representing the venture. These are shown as DATA statements at the end of listing 1. The program is now ready to sample them using a succession of random numbers. Each sampled value gets plugged into the equation. This yields one possible outcome. The program does this repeatedly, saving the results in a table of outcomes for further analysis.

Note how the fixed cost, which is a constant, is represented in the DATA statement on line 62140 of the program. This wastes random numbers and running time but it simplifies programming and is very flexible. With this arrangement any variable can be treated as either a constant or a distribution without reprogramming.

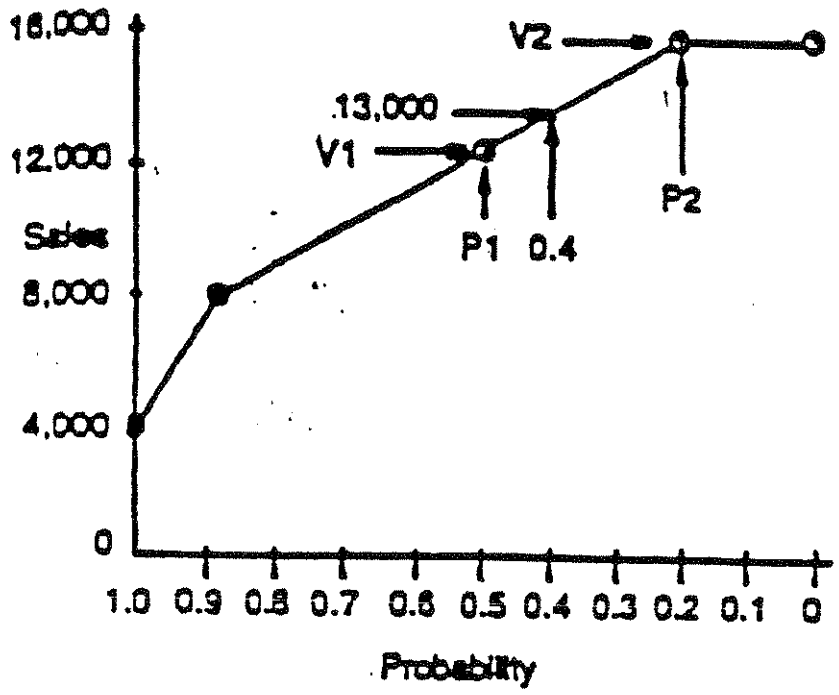

Figure 1: Sampling by interpolation. The probability of achieving a certain sales figure ranges from 1 (a certainty) to 0 (impossible). The approximate mid-range of 13,000 sales has a probability of 0.4.

Does It Really Work?

At this point you may be wondering whether we can really get away with fitting or representing the smooth curve of a distribution of expected values with straight line segments, and

only four of them at that. It turns out that, in almost all cases, the use of precise distributions, or of more points, makes little significant difference in the results. This has also been my own experience with venture analysis in the chemical industry over many years.

If we think about it, we can see why this is so. The most obvious consideration is that any estimate of future events is subject to error, however informed it may be. As Murphy, who by the workings of his own law must be counted an optimist, would put it, "The future is uncertain; you can count on it." There is a deeper reason, however, why this simplified approach works. It lies in the use of a distribution or spread of values. The mere fact that an informed estimator has approximate upper and lower limits adds far more information to the simulation than any refinements in the detailed form of the distribution.

It is the introduction of distributions that transforms the formerly popular (but wrong) single-point estimates into a sound and informative analysis. This does not mean that the forecast of outcomes is necessarily correct. Even when wrong, the method provides good information on which of the variables are most important. It may show that a doubling of promotional expenses will have very little effect, whereas a 10 percent increase in inventories can turn a profit into a loss. Such a use of venture analysis is called sensitivity analysis.

A Venture Analysis Program

Now we know how to sample a distribution by going in with a random probability P and coming out with a corresponding value V. The program shown in Listing 1 implements this and does a complete risk analysis. It embodies our specific model example. The program is easily modified for other venture analyses by simply replacing the equation, or model, in subroutine 31000. It is a no-frills program designed for ease of understanding.

The style of programming followed here consists of putting everything possible into subroutines. These are invoked by a short calling section at the start of the program.

It looks simple and that's the way it should be. We are looking at the main features of the program, avoiding all detail at this level. We see that the program starts with a specification section and ends with a sorted tabulation and display. The simulation is performed in three nested FOR...NEXT loops. The task carried out by each called subroutine is described in the remarks to the right.

Subroutines are used even if they are called only once. There are many advantages to this. Each subroutine performs a single task, which is described in its header. This makes it easy to follow the flow of the program. Another feature is that GOTOs, whether explicit or implied, never branch out of a subroutine. This is not a dodge since a subroutine always returns to the point immediately following its invocation. We could say that this programming style produces a bunch of grapes (GOSUBs) instead of a bowl of spaghetti (GOTOs). It is much easier to pick your way through a cluster of grapes than a tangle of spaghetti. GOTOs are used, of course, but they branch to points within their own routines. This makes it much safer to modify the program when necessary. It's about the closest approach we can make to structured programming in Microsoft BASIC.

As a further aid to understanding the program, the meaning of the program variables is shown in Table 1. The Monte Carlo sampling is carried out in subroutines 21000 through 23000 and is applied to the model equation in subroutine 31000. The outcomes are stored in the array variable OU. They are converted to a cumulative distribution by a Shellsort in subroutine 41000.

All that remains after this is to display the results. We could simply list our 200 outcomes in a table with their associated probabilities. Thus, the 20th outcome in the list would represent the minimum result of 90 percent of the trials. Such a table would not be very appealing. A better solution is to show probable outcomes in steps of 10 percent.

Table 1: Variables used in the Monte Carlo Venture Analysis program.

AD	Standard deviation over all samples.
AV	Average or mean.
CD	Array of cumulative distribution values.
FC	Fixed costs.
FD	Array of averages for selected cumulative distribution points.
NR	Number of runs.
NS	Number of samples per run.
NV	Number of variables in model equation.
OU	Array of outcomes for a single run.
P	Probability.
P1	Probability estimate point in a distribution.
P2	Probability estimate point in a distribution.
PA	Store of selected P1.
PB	Store of selected P2.
PF	Step increment for summary cumulative distribution.
RS	Random seed.
S#	Sum.
SD	Standard deviation.
SO	Single outcome calculated from the model equation.
SP	Selling price (effective).
SS#	Sum of squared deviations from the average.
ST	Array of run statistics.
SV	Sales volume.
UC	Unit distribution cost.
V	Sample value calculated from a distribution.
V1	Estimated value point in a distribution.
V2	Estimated value point in a distribution.
VA	Store of selected V1.
VB	Store of selected V2.
XS	Sink for dummy read, input, etc.

Some Results

With our program complete, our model specified, and our estimates in hand, we can now launch our venture thousands of times and see what the future promises. A typical run of 2000 samples gives the following projection. All values are rounded to the nearest $100.

Percent Chance	of Gain Exceeding
100	- 1,900
90	8,900
80	15,300
70	19,900
60	24,300
50	27,900
40	31,100
30	35,000
20	39,400
10	45,600
0	58,400

Not surprisingly, Monte Carlo simulation with its distributed inputs gives a corresponding spread of outcomes. The first thing we note is that there is a possibility of a small loss. If we were to plot the above results, we would find the chance of a loss is about 2 percent. The traditional single-point method would ignore the possibility of loss and come up with an overestimation of the gain as a most likely $30,400. This is very reassuring to people who like to keep their head in the sand.

We have achieved our objective of placing a probability estimate on a range of possible outcomes. Remember that we are dealing with essentially a one-shot proposition, and therefore the information in the two extremes is meaningful. If we were really dealing with the long run, then the extremes would hardly matter. We could be virtually certain of achieving something close to the long-term expectation or average of $27,600.

In spite of all that has been and can be said, many users still feel uncomfortable with this form of analysis. The reason is a basic one. This is the human predilection for twisting the facts of uncertainty into something that seems more certain. The analyst can help here by working up the results into a form the decision maker can relate to more easily. One type of analysis that is guaranteed to spark interest is a sensitivity analysis. Simply rerun the analysis with the same random seed but with a small percentage increase in sales. Do the same for each variable in turn and show which variable is most important in affecting the outcome. In models with more variables than in our example, such sensitivities are not always obvious.

Another possibility is to do some "what if?" simulations in which different possible estimates are used. The results can then be presented as a statement of conditions or scenarios required to avoid a loss or make a given profit. Presenting the results in the form of charts or curves can also help.

Have We Taken Enough Samples?

You may have wondered why the program carries out ten runs of 200 samples each. Why not one run of 2000 samples? By making several small runs instead of one large run we obtain information on the adequacy of our sample size. Recall that a Monte Carlo simulation is in the

nature of an experiment. It does not give a precise answer even when one is possible. By examining the results of several runs we can get a measure of how well we are zeroing in.

Here's how it works for our example: The following averages are for ten runs of 200 samples each. All the figures, including the standard deviations, are supplied by the program. The standard deviation is a statistical measure of the amount of variation or spread in the data represented by an average.

Run	Average	Standard Deviation
1	28,100	14,100
2	28,000	14,000
3	28,700	13,900
4	27,400	13,200
5	26,800	14,000
6	27,600	13,200
7	26,100	12,500
8	29,000	12,700
9	28,100	13,900
10	25,900	13,200

As expected, we find a fair amount of fluctuation. What about our overall average and standard deviation? For the 2000 samples we have:

27,600	Average
13,400	Standard Deviation

We appear to have gained nothing from our ten-part breakdown. But there is more information to be squeezed from this data. Suppose we take the average of the ten averages and, again courtesy of our program, take a standard deviation. This time it is for the average of the averages. We can do this since each average represents the same sample size. As expected, we get the same average but note the new standard deviation:

27,600	Average
1,000	Standard Deviation

Statistical theory tells us that the true average has a 68 percent chance of being within one standard deviation of $1,000 of our estimated average. If we had made only one run of 2000 samples, we would have little idea of how we were doing. The large standard deviation of $13,400 would have left us with a range of about $14,000 to $41,000 in which to expect the average in 2 out of 3 chances.

Note that we have given no hard criteria or explicit formula for determining an optimum or safe sample size. Experience shows that a venture analyst should have and does have a feel for what is acceptable. For example, if you had made several ten-run simulations with different sample sizes, you might have come up with:

Samples per run	Overall Average	Overall Standard Deviation	Standard Deviation of Ten Averages
20	28,100	13,700	3300
100	27,800	13,700	1700
200	27,600	13,400	1000

In view of the approximate nature of the estimates, a sample size of 200 appears adequate. Should your model have more variables, you might need to increase the sample size beyond 200. Fortunately, even fairly involved business-risk simulations need no more than five or six of their variables to be distributed. The other variables that are treatable as constants can be directly programmed as such into the model. This saves space and conserves the supply of non-repeating pseudorandom numbers. It allows a realistic but no-frills simulation to be run on a microcomputer.

Conclusions

We have demonstrated with the aid of a simple example the nature of a Monte Carlo simulation and how it can be applied to a business venture. In particular we have seen that:

- An expert's estimates of the uncertainties of his field - sales, production, marketing, costs, timings, and trends - is valuable information. It is the raw material for risk analysis by means of Monte Carlo simulation.
- There is a very simple, practically universal technique for expressing such estimates in the form of a distribution and taking random samples from it.
- A model can be built as each situation requires by a simple top-down design method that starts broadly and gets more detailed in stages.
- We can apply simple statistics through a program to get a handle on the adequacy of our experimental (simulated) probe into possible outcomes.

This article has touched on these subjects in an introductory way. Professional risk analyses will require familiarity with the venture to be modeled. Professional programs may need to provide for things like multiple years, time value of money, and various kinds of return on investment.

Listing 1: Monte Carlo Venture Analysis program, written in Microsoft BASIC, can be modified for your own use by programming your own model equation and changing the numbers in the DATA statements accordingly.

```
1000  'MONTE CARLO VENTURE ANALYSIS
1005  '
1010  CLEAR
1020  GOSUB 11000              'Specify runs, samples, etc.
1030  '
1040  FOR L=1 TO NR            'Carry out NR runs (10 max.)
1045  PRINT L
1050  FOR J=1 TO NS            'of NS samples each (200 max.)
1060  '
1070  FOR K=1 TO NV            'for a model with NV variables
1080  P=RND                    'Generate a random probability
1090  GOSUB 21000              'Find its posn. in a distrbn. of values
1100  GOSUB 22000              'Find correspg. posn. of variable value
1110  GOSUB 2300               'Calc a sample value  v  for a variance
1120  NEXT K
1130  '
1140  GOSUB 31000              'Calc a sample outcome from the model
1150  NEXT J
1160  '
1170  GOSUB 41000              'Sort outcomes in ascending order
1180  GOSUB 42000              'Calc statistics for each run
1190  GOSUB 43000              'Calc 11 cum distrbn points for each run
1200  NEXT L
1210  '
1220  GOSUB 44000              'Avg & std devn over all samples
1230  GOSUB 45000              'Std devn of avg of NR run averages
1240  GOSUB 46000              'Cum distn of outcomes over all samples
1250  '
1260  GOSUB 51000              'Display statistics for each run
1270  GOSUB 52000              'Display distn of outcomes over all samples
1280  '
1290  END
1300  '
11000 '--- Initiate the simulation
11010 '
11020 NV=4
11030 DIM OU(200), CD(10,10), ST(10,2), FD(10)
11035 PRINT"PATIENCE PLEASE. RUN NUMBER WILL DISPLAY WHILE COMPUTING."
11040 INPUT"NUMBER OF RUNS";NR
11050 INPUT"SAMPLES PER RUN";NS
```

```
11060 INPUT"ENTER RANDOM SEED (-32768 to 32767 ";RS
11070 RANDOMIZE RS
11080 IF NR 2 THEN NR=2:IF NR 10 THEN NR=10
11090 IF NS 10 THEN NS=10:IF NS 200 THEN NS=200
11100 RETURN
11110 '
12000 '--- Sum of squared deviations
12010 '
12020 SS#=o:FOR I=1 TO T
12030 SS#=SS#+(OU(I)-AV) 2:NEXT I:RETURN
12040 '
13000 '--- Standard Deviation
13010 '
13020 SD=(SS#/(T-1)) .5:RETURN
13030 '
21000 '--- Find the interval of P
21010 '
21020 FOR I=1 TO 5:READ P2
21030 IF P P2 and P =P1 THEN II=I:PA=P1:PB=P2
21040 P1=P2: NEXT I:READ X#:RETURN
21050 '
22000 '--- Find the corresponding value interval
22010 '
22020 FOR I=1 to 5:READ V2
22030 IF I=II THEN VA=V1:VB=V2
22040 V1=V2:NEXT I:REAL X$:RETURN
22050 '
23000 '--- Calc. a sample value V: assign to model variable
23010 '
23020 V=((PA-P)/(PA-PB))*(VB-VA)+VA
23030 IF K=1 THEN SP=V
23040 IF K=2 THEN UC=V
23050 IF K=3 THEN SV=V
23060 IF K=4 THEN FC=V
23070 RETURN
23080 '
31000 '--- Calc. a sample outcome and running sum for model
31010 '
31020 SD+((SP-UC)*SV)-FC
31030 OU(J)=SO:S#=S#+SO
31040 RESTORE:RETURN
31050 '
41000 '--- Sort sample outcomes in ascending order
41010 '
```

```
41020 D=NS
41030 IF D =1 THEN RETURN
41040 D+INT(D/2):R=NS-D:EX=0
41050 FOR I=1 to R:DI=D+I
41060 IF OU(I) +OU(DI) THEN GOTO 41080
41070 OT=OU(I):OU(I) =OU(DI):OU(DI)=OT:EX=1
41080 NEXT I
41090 IF EX=0 THEN GOTO 41030
41100 EX+0:GOTO 41050
41110 '
42000 '--- Avg. sum of sq devns, std devn for each run
42010 '
42020 T=NS:AV=S#/T:ST(L,O)=AV
42030 GOSUB 12000:ST(L,1)=SS#
42040 GOSUB 13000:ST(L,2)=SD
42050 S#=):RETURN
42060 '
43000 '--- Extract and store cum. distn in steps of 10%
43010 '
43020 PF=INT(NS/10+.5)
43030 FOR I=0 to 10:CD(L,I)=OU(I*PF)
43040 NEXT I:CD(L,O)=OU(I)
43050 RETURN
43060 '
44000 '--- Avg and std devn over all samples and runs
44010 '
44020 S#=0:SS#=0:FOR I=1 to NR
44030 S#=S#+ST(I,0):SS#=SS#+ST(I,1):NEXT I
44040 AV=S#/NR:AD=(SS#/(NS*NR-1___ .5
44050 RETURN
44060 '
45000 '--- Std devn of avg outcomes for NR runs
45010 '
45020 SS#=0:FOR I=1 TO NR
45030 SS#=SS#+(ST(I,0)-AV) 2:NEXT I
45040 SD=(SS#/(NR-1)) .5
45050 RETURN
45060 '
46000 '--- Final distribution in steps of 10%
46010 '
46020 FOR I=0 to 10:S#=0
46030 FOR M=1 to NR
46040 S#=S#+CD(M,I):NEXT M
46050 FD(I)=S#/NR:NEXT I
```

```
46060 RETURN
46070 '
51000 '--- Display statistics for NR runs
51010 '
51015 PRINT CHR#(26)
51020 PRINT"STATISTICS FOR";NR;"RUNS OF";NS;SAMPLES EACH.  SEED=";RS
51030 PRINT""
51040 PRINT"RUN          AVG OUTCOME              STD DEVN"
51050 PRINT"------       ---------------------    -------------"
51060 PRINT""
51070 FOR I=1 to NR
51080 PRINT I,ST(I,0),ST(I,2):NEXT I
51090 PRINT""
51100 PRINT"AVG=";AV,"STD DEVN=";SD
51110 PRINT""
51120 INPUT"PRESS RETURN TO CONTINUE";X#
51130 RETURN
51140 '
52000 '--- Display outcomes over all samples and runs
52010 '
52015 PRINT CHR#(26)
52020 PRINT"OUTCOMES FOR"INR*NS;"SAMPLES.   SEED=";RS
52030 PRINT""
52040 PRINT"% CHANCE          OUTCOME WILL EXCEED"
52050 PRINT"--------------    ----------------------------------"
52060 PRINT""
52070 FOR I=0 to 10
52080 PRINT 100-I*10,FD(I):NEXT I
52090 PRINT""
52100 PRINT"AVG=";AV,"STD DEVN=";AD
52110 PRINT""
52120 RETURN
52130 '
62000 '--- Data statements in order as per subr. 23000
62010 '    One pair (5 proby, 5 values) for each model var.
62020 '    Chance NV in subr 11000 if model in 31000 changes
62030 '
62040 DATA 1    , .90, .50, .10,0    ,   'SP proby; discount-
62050 DATA 3.00,3.40,4.20,5.00,5.40,  'ed selling price
62060 '
62070 DATA 1    , .85, .50, .15,0    ,   'UC proby.
62080 DATA 0.40,0.45,0.50,0.70,0.90,  'Unit distrn cost
62090 '
62100 DATA 1    , .85, .50,  .20, 0 ,   'SV proby.
```

```
62110 DATA  4000,8000,12000,15,000,15,000,   'Sales volume
62120 '
62130 DATA 1     ,  .80,  .50,   .20,    0      'FC proby.
62140 DATA 14000,14000,14000,14000,14000,  'Fixed cost
62150 '
62200 'FOR A DIFFERENT VENTURE, PROCEED AS FOLLOWS
62210 'REPLACE LINE 31020 WITH NEW MODEL EQUATION.
62220 'MODIFY LINES STARTING AT 23030 ACCORDINGLY.
62230 'MODIFY MV IN LINE 11020 IF NECESSARY.
62240 'EDIT DATA SECTION AT 62000 FOR NEW ESTIMATES.
62250 END
```

A personal computer makes it convenient to include such features incrementally. The simplicity of Monte Carlo sampling allows each programming addition to be small and modular. Fast execution times are not necessary. This allows interpreter languages to be used for each change in models and data. The availability of state-of-the-art implementations of BASIC and APL particularly on the IBM PC, is favorable to increased use of this form of simulation.

This article includes material from the author's book, *Learning Simulation Techniques on a Microcomputer Playing Blackjack and Other Monte Carlo Games*, published by Tab Books.*

* * * * * * * * *

*Macaluso, Pat. "A Risky Business," *Byte,* March 1984, pp. 179-191.

PROBLEM 2 - QUEUE SIMULATION

Introduction

We all experience queues when we are stuck in traffic or waiting in line at a checkout counter at a supermarket. This program analyzes the average waiting time and efficiency (utilization) of service stations.

The problem of modeling a queue can be divided into components as the article "Queue Simulation" explains. The basic components of the problem are:

1) Arrival time of a person (unit).

2) Interval between arrivals.

3) The number of servers.

4) The rate at which the server operates.

Before we can model the queue, we need to observe the real world situation and record data. To illustrate, suppose we want to develop a function that would model customer arrival into a queue, such as our arrival module that was written in Applesoft BASIC. Suppose we went to the business and spent a day (hopefully a typical one) and recorded when the customers entered. Let Table 1 represent the record of arrival times, restricting our model to business between 8 A.M. and 10 A.M. Our table indicates that the first customers entered at 8 A.M., our second customers entered at 8:10 A.M., and so on. In practice, sample size should be much larger than the 15 in our illustration.

Table 1 - Let X = Time

8:00 A.M.	8:42	9:10	9:24	9:45
8:10	8:45	9:15	9:26	9:48
8:24	9:00	9:18	9:32	9:51

We next need to convert the times into useable data. To achieve this, let 8 A.M. = 0, 10 A.M. = 2, 8:30 A.M. = .5, 9:15 = 1.25, etc. The converted data is listed below in Table 2.

Table 2 - Let Y = Converted Time

0	.7	1.2	1.4	1.8
.2	.8	1.3	1.4	1.8
.4	1.0	1.3	1.5	1.9

Suppose we wanted to test the hypothesis that customers arrive uniformly between 8 A.M. and 10 A.M. We simply divide the two hour interval into k intervals; arbitrarily let k = 4. We count the number of arrivals in the four intervals below:

Y	Observed Frequency
[0, .5]	3
[.5, 1]	2
[1, 1.5]	6
[1.5, 2]	4

We arbitrarily assign end points to the left hand of the interval, though more exact timing measures preclude arrival at a point.

We now compare the observed frequency with the expected frequency. Since we assume that our arrival times are coming from the uniform distribution, we can let the expected frequency in each of the four categories be equal to 15/4 = 3.75.

Next use the familiar chi-square formula:

$$X^2 = \sum \frac{(O - E)^2}{E} = \frac{(3 - 3.75)^2}{3.75} + \frac{(2 - 3.75)^2}{3.75} + \frac{(6 - 3.75)^2}{3.75} + \frac{(4 - 3.75)^2}{3.75}$$

$$\therefore \quad X^2 = 2.33$$

Compare this X^2 value with our critical X^2 .05, 1 df = 3.841. Since 2.33 < 3.841, we conclude that the uniform distribution is a good fit. Remember that the goodness of fit test accepts most distributions as good fits and only rejects those with significant discrepancies. Also there is no uniform way of deciding on the number of partitions, as we decided on four above.

We now can generate random numbers with the uniform distribution using our distribution function and the inverse transform method covered earlier.

To complete our illustration of how we generate our uniformly distributed arrival times, start with the uniform distribution:

$$f(x) = \frac{1}{b-a}, \quad x \in (a, b)$$

For our example, a = 0, b = 2. Therefore, f(x) = 1/2 is the appropriate density function. Next we find our distribution function.

$$F(x) = \int_0^x 1/2 \; dt = x/2$$

Letting y = x/2 we see that $F^{-1}(y) = x = 2y$.

We next rely on our computer to give us random numbers Y on the interval (0, 1) and substitute $F^{-1}(y) = 2y$ to generate arrival times that are uniformly distributed on our 8 A.M. to 10 A.M. interval.

People have observed that Poisson probability distributions, not uniform distributions, accurately describe patterns of arrival and service time.

As a recommended class activity, attempt to model arrival times at local businesses using the methods that we have discussed. Another enrichment activity could include student teams to model traffic patterns near the college. It would be a more difficult programming problem where one has to consider the complex arrival-departure patterns at intersections. One could vary the duration of red and green lights in the model to optimize traffic flow. Such considerations are important to the profitability of shopping malls that require an optimally smooth flow of traffic.

* * * * * * * * *

QUEUE SIMULATION

A Microcomputer Can Help You Manage Waiting Lines

by E. Hart Rasmussen [Byte, March, 1984]

When we wait at the supermarket checkout counter, are stuck in rush-hour traffic, or have trouble getting a telephone call through, we are in a queue. Queue is another word for "waiting line." If we could get a firm handle on how queues work, we would be able to manage them better and perhaps even eliminate them.

Simple waiting lines can be analyzed mathematically, but most queueing situations are so complex that they defy precise description. For these situations, a computer can help us.

Specifically, we can use a computer to model and simulate a queueing situation so that we can make predictions about it and learn how it behaves.

There are many sophisticated commercial queue-simulation programs available, but they are expensive and for large computers only. In this article, I present an Applesoft BASIC program that can simulate many queueing problems.

Know Your Ps and Qs (Probabilities and Queues)

My doctor's nurse knows that the average examination takes 17 minutes, so she schedules three patients an hour, one every 20 minutes. At first glance, it looks as if I should never have to wait for the doctor. In reality, he needs a sizeable waiting room. Why? Because examinations may take more time than expected, and patients don't always arrive on time.

A queue formation occurs when a unit that seeks a service must wait because the service facility is busy servicing another unit. To simulate a queue formation, we need to break it into its basic components.

The basic components are: the arrival of units seeking service, the interval between arrivals, the number of service facilities, and the rate at which the service facilities operate. Figure 1 illustrates some basic ways in which service facilities or stations and units may be combined.

We next need to make some assumptions about these components. For instance, we must assume that the overall capacity of the service facility exceeds the overall demand. (In queue terminology, we say that the mean service rate exceeds the mean arrival rate for a single channel.) If we didn't assume this, our queue would theoretically grow to infinite size. We also assume that the intervals between arrivals and service times are variable. To study the queue, we

must be able to describe these patterns of arrivals and service times, even if they seem to be unpredictable. Research has shown that the patterns of arrivals and service times often are completely random and can be described with the Poisson distribution. The formula for the Poisson distribution is shown in Figure 2. With this formula, we can easily program the random element that we need in our simulation.

	Variable		Equation	
Queue Component	Theory	Program	One S	Multiple S
Mean arrival rate	λ			
Mean service rate	μ			
Mean arrival interval	$\frac{1}{\lambda}$	Al		
Mean service time	$\frac{1}{\mu}$	Sl		
Number of stations	S	S%	1	S
Utilization factor	U	US(Z)	$\frac{\lambda}{\mu}$	$\frac{\lambda}{\mu}$
Average length of queue	L_c	OA	$\frac{U^2}{1-U}$	$\frac{U^{(s-1)}}{(S-1)!(S-U)^2}$ x $\frac{1}{\sum_{n=0}^{s-I} \frac{U^2}{n!} + \frac{U^2}{S!(1-U!S)}}$
Average waiting time in queue	W_c	O1	$\frac{L_c}{\lambda}$	$\frac{L_c}{\lambda}$

Table 1: Queueing theory variables and equations. S! and n! are the factorial values of these variables.

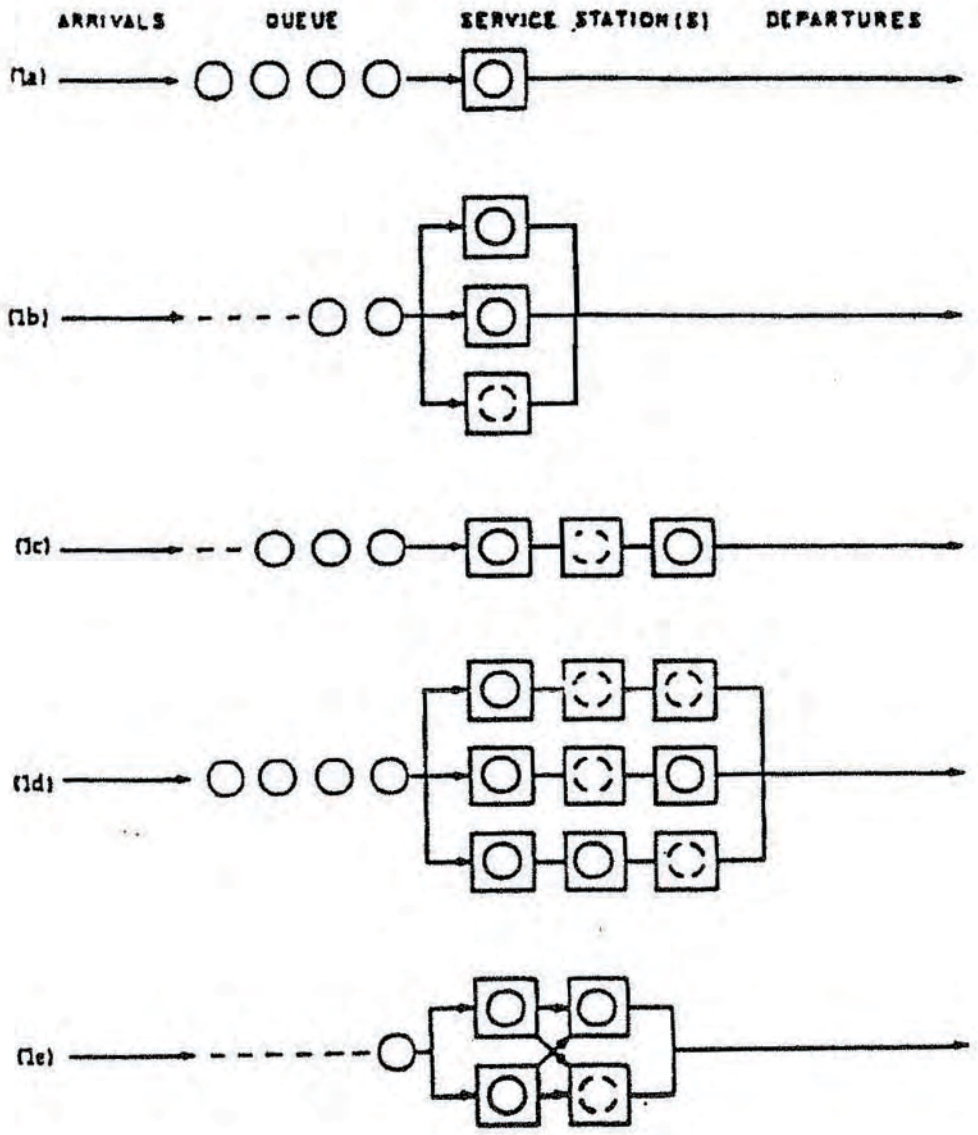

Figure 1: Fundamental queueing models: (1a) single-channel, single-phase service; (1b) multiple-channel, single-phase service; (1c) single-channel, multiple-phase service; (1d) multiple-channel, multiple-phase service; and (1e) channel switching. The first two models are simulated by the author's program. The broken circles indicate an open service station.

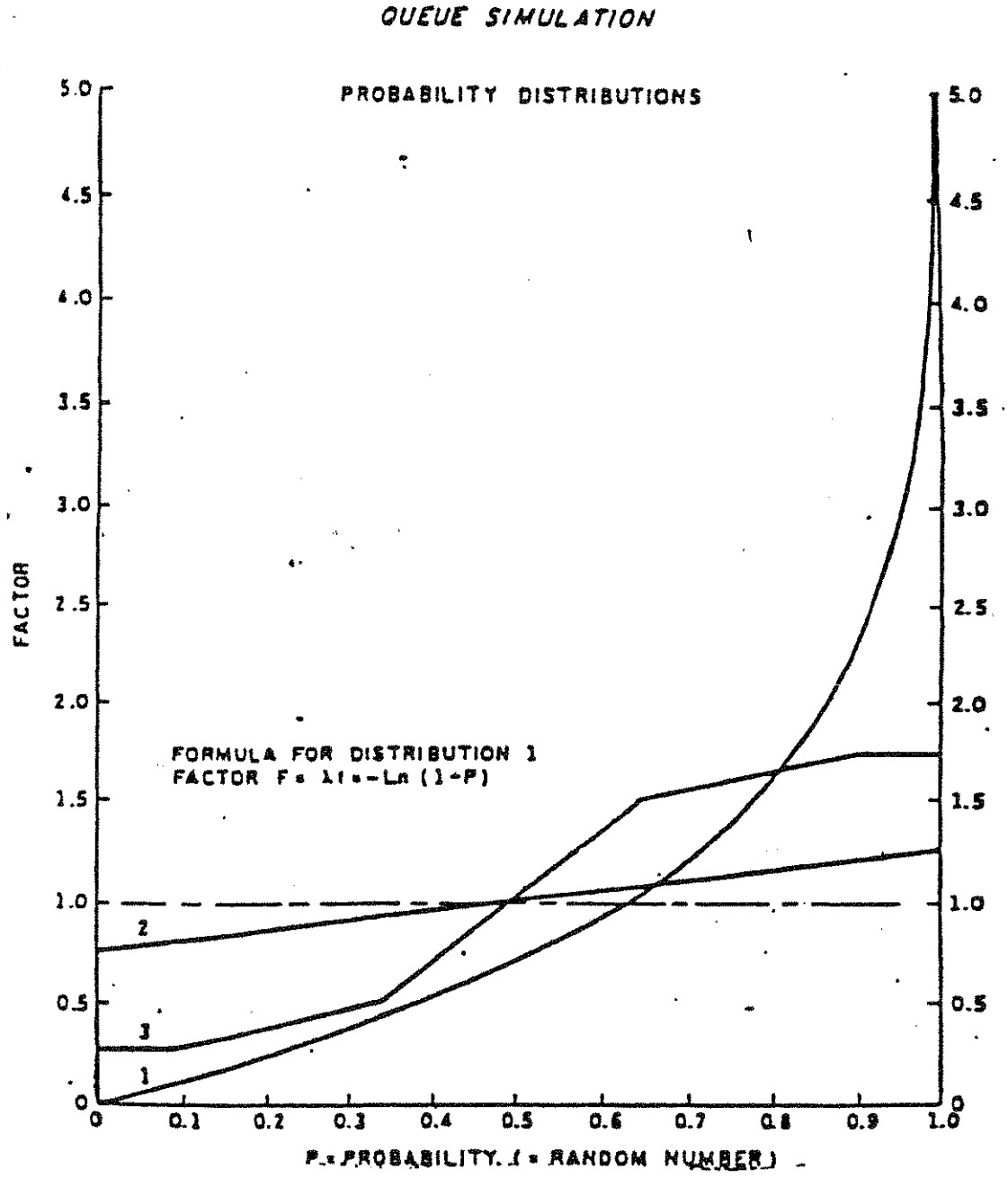

Figure 2: Probability distribution. Curve 1 shows the Poisson probability distribution. Curves 2 and 3 are arbitrary distributions that can replace the Poisson distribution.

The Program

The flowchart in Figure 3 shows the logic of a program that simulates a multi-channel, single-phase service problem. The Advance module provides the executive control that keeps track of time and events; it passes control to the appropriate action module as successive events are simulated. In the Arrival module, as one arrival occurs, the time for the next arrival is calculated in accordance with the specified algorithm. The arriving unit then joins the queue if all service stations are occupied, or it moves on to seize an available station. The Departure module frees a station when the service is completed and terminates the simulation if the sample size has been reached. Otherwise, it checks to see if any units are waiting in the queue and, if so, lets a unit leave the waiting line. The Seize module seizes an available service station and calculates the service time in accordance with the prescribed algorithm.

Listing 1 shows the Applesoft BASIC queue-simulation program. The listing is grouped in sections that correspond to the flowchart in Figure 3. All variable names used in the program are listed in Table 2.

I have defined the frequently used variables at the start of the program. I've dimensioned the arrays to allow 10 service stations, but they can easily be changed to accommodate a larger number. The opening screen and data input (lines 7000-7999) and the start of the simulation (lines 8000-8799) are at the end of the program so the sections of the program that are executed over and over can have the lowest possible line numbers.

The keyboard is used to input data. The only exception is the data for non-Poisson distributions, which is input through DATA statements.

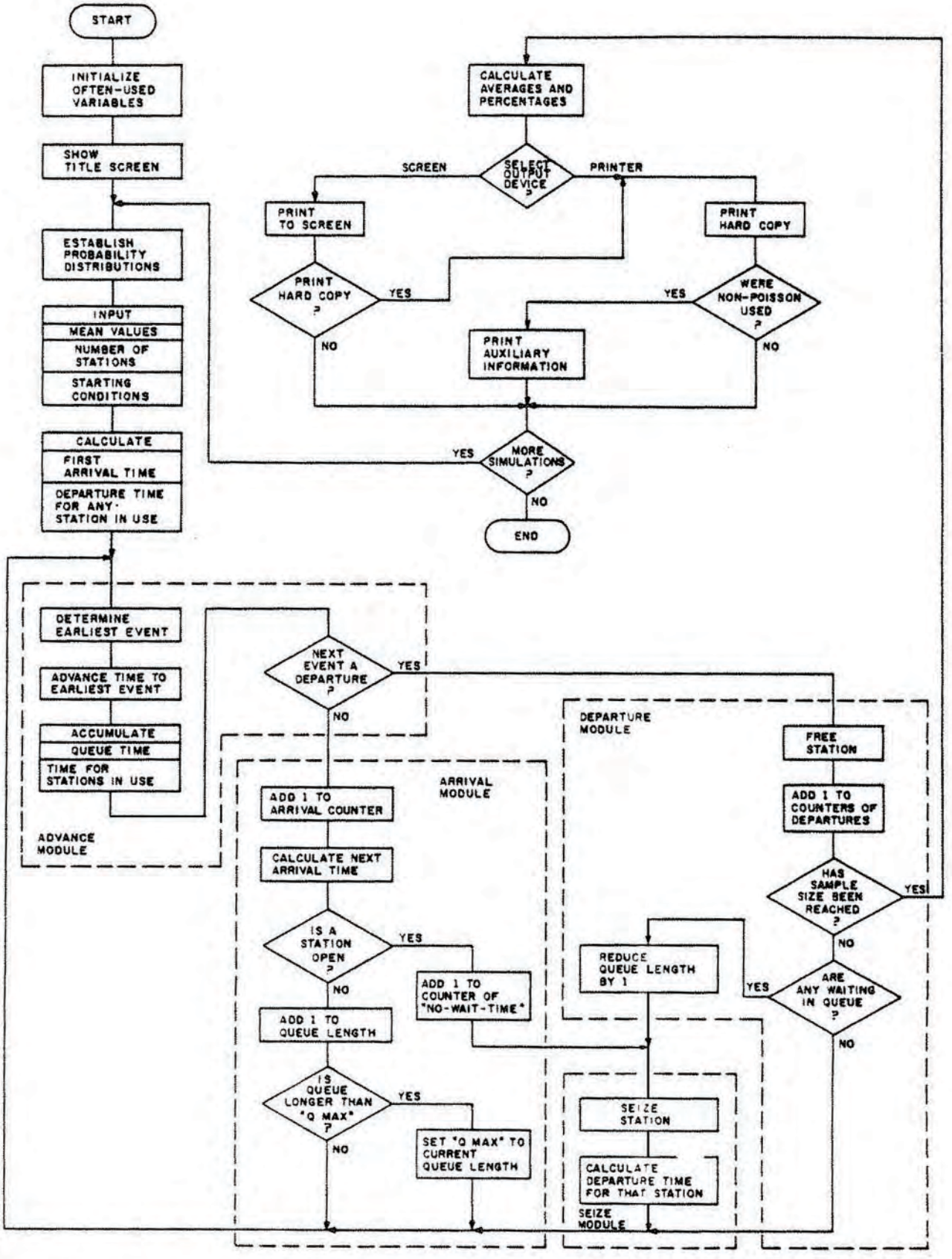

Figure 3: A flowchart for the author's queue-simulation program,

The times for the first arrival and the departure times for any units in a service station are calculated in lines 8000-8799. Depending upon the instructions given during data input, the program uses either a Poisson distribution or a user-defined probability distribution. Arrival intervals and service times can have different distributions, and the random-number generator is used to calculate the randomly varied event intervals in accordance with the specified distributions.

The Advance module determines the earliest event by first assuming that the event is an arrival (line 2000) and then checking if any departure occurs earlier (lines 2100-2130). The clock is then advanced to the earliest event (line 2300). Counters necessary to the calculation of queue statistics are incremented (lines 2310-2420), and control passes to the appropriate event module.

Variable	Description	Input	Output	
			Printer	Screen
A	Time advance			
A1	Mean arrival interval	yes	yes	yes
A$	Answer to yes/no question			
A$(2)	Station status at start	yes	yes	
C%	Sample size	yes	yes	yes
CD%	Number of arrivals with no wait time		yes	yes
C1%	Total number of arrivals		yes	yes
C2%	Total number of departures			

CS%(Z)	Number of departure from Station Z		yes	yes
D$	Date of report	yes	yes	
F	Simulation factor; a function of random			
FA(a,2)	Probability distribution of arrival intervals			
FS(b,2)	Probability distribution for service intervals			
I$	Project identification	yes	yes	
N	Switch for next event (0 = arrival, 1 = departure)			
P	Switch for type of probability distributions			
QO	Percent of arrivals with no wait time		yes	yes
Q1	Average wait time, all arrivals		yes	yes
Q2	Average wait time, arrivals entering queue		yes	yes
QA	Average length of queue		yes	yes
QL	Length of queue at start	yes	yes	yes
QL%	Length of queue		yes	yes
QM%	Maximum length of queue		yes	yes
QT	Cumulative queue time (i.e., SUM QL% • T)			
R	Random number = RND(1)			
S	Station number with earliest departure			

S%	Number of service stations	yes	yes	yes
S%(Z)	Station status: 0 = open, 1 = used			
S$(Z)	Station status		yes	yes
ST(Z)	Total time Station Z has been in use			
SI	Mean service time, all stations	yes	yes	yes
T	Time, cumulative from start		yes	yes
TA	Time for next arrival			
TD(Z)	Time for departure from Station Z			
TS(Z)	Average service time at Station Z		yes	yes
US(Z)	Percent of utilization of Station Z		yes	yes
X	General counter			
Z	Counter for stations			

Table 2: Queue simulation variables.

The Arrival module counts the total number of arrivals (line 3000) and calculates the time of the next arrival (lines 3050-3130). If no station is open, the queue length is incremented by line 3230 and the program checks to see if the new queue length exceeds the previous maximum (lines 3240).

The Departure module frees the station (line 4000), increments counters (lines 4010-4050), and checks to see if the sample size has been reached (line 4100). If the simulation has not been completed, a unit waiting in the queue (if any) is allowed to seize the free station. The Seize module sets the key that indicates a particular station is in use (line 4250) and then calculates the departure time from that station (lines 4300-4430).

When the sample size has been reached, the program leaves the simulation loop at line 4100 and passes to the section that calculates the statistics for the simulation (lines 5000-5199). After the calculations are completed, the user is given a choice (lines 5200-5399) of sending output to the screen or to a printer. If screen output (lines 5700-5999) is chosen, the user gets a second opportunity to get a printed report (lines 5400-5699). If any non-Poisson distributions have been used, that fact and the parameters for the distribution(s) are recorded on the report (lines 6500-6999). When the printing is completed, control is passed back to screen output (line 5690) and the user is given an opportunity to run additional simulations without restarting the program (lines 6000-6499).

The program length is about 7000 characters; it uses a total of about 8200 bytes of memory during execution. The run-time depends somewhat on the type of probability distribution (it runs faster when the formularized Poisson distribution is used). On my Franklin 1000, one simulation takes from 0.33 to 0.47 seconds, which means that a 2000 sample simulation takes 10 to 15 minutes.

User-Defined Probability Distributions

The program can evaluate queueing situations with unique, user-defined probability distributions. The arrival intervals and service intervals can have different distributions, independent of each other. The choice of the type of distribution to be used is made from the keyboard in lines 7200-7599. The program logic can most easily be explained by Table 3, which lists the values that the key P assumes for the various possible combinations.

Figure 2 shows the shape of the Poisson distribution curve (curve 1) and two arbitrary, user-defined curves. The distribution of curve 2 was used in the simulation reported in parts e and f of Figure 4. The user must input this non-standard distribution via DATA statements

starting at line 9000. The DATA statements provide the coordinates for the line segments that represent the special distribution. As an example, the distribution shown as curve 3 should be input as follows:

9000DATA0,.25,.1,.25,.35,.5,.65,1.5,.9,1.75,1,1.75

Arrival Interval	Service Interval	Value of P
Poisson	Poisson	1
Poisson	Non-Poisson	2
Non-Poisson	Poisson	3
Non-Poisson	Same Non-Poisson function as in Arrival Interval column	4
Non-Poisson	Different Non-Poisson function than in Arrival Interval column	5

Table 3: P values.

If arrival and service intervals have identical, non-Poisson distributions, we define the interval only once. If they have different, non-Poisson distributions, the definition of the arrival interval distribution precedes the definition of the service time distribution. For example,

9000DATA0,.25,.1,.25,.35,.5,.65,1.5,.9,1.75,1,1.75

9010DATA0,.75,1,1.25

assigns a distribution according to curve 3 to arrival intervals but distributes service times according to curve 2. Make sure that the defined distribution has an average value of 1.

Some Sample Programs

Parts a through f of Figure 4 show examples of printed output from simulations done by the program. Notice that all input data is repeated on the printed report. Parts e and f of the figure show simulations using non-Poisson distributions. This is noted at the bottom of the report; and the coordinates for the specified distribution curve(s) are shown. Part c of the figure shows a simulation of three parallel service stations. Notice that station 1 has been used most often. This is because of the way the program chooses open stations. The average utilization factor of 91.6 percent for all three stations is close to the theoretical, overall utilization factor of 90.9 percent.

The results of the simulation are summarized in Table 4 and compared with the theoretical values (where this is possible), which can be calculated using the formulas listed in Table 1. Notice that even with 5000 simulations there is up to 20 percent difference between the analytical and the simulated results. This does not indicate any flaw in the program but merely illustrates the possible differences between a finite and an infinite population.

A comparison of lines 1, 2, and 3 in Table 4 shows that multiple service channels with identical total capacity provide slightly improved service as more channels are used. A comparison of lines 1 and 4 shows the dramatic reduction in queue length and waiting time when a second service line is opened. (I hope the manager of my local supermarket reads this.)

Figure 4: Simulation results with (4a) one service station, (4b) two service stations, (4c) three service stations, (4d) doubled service capacity, (4e) service time has probability function from curve 2 of Figure 2, and (4f) both service and arrival times have probability function from curve 2 of Figure 2.

(4a) 1 (ONE) QUEUE SERVED BY 1 PARALLEL SERVICE STATIONS
AVERAGE ARRIVAL INTERVAL WAS SPECIFIED AS 11 TIME UNITS
AVERAGE SERVICE TIME WAS SPECIFIED AS 10 TIME UNITS
QUEUE LENGTH AT START OF SIMULATION WAS 8

STATUS OF SERVICE STATIONS AT START WAS:

STATION	STATUS
1	USED

SAMPLE SIZE IS 5000 DEPARTURES

TIME ELAPSED FOR SIMULATION IS 55470 TIME UNITS

STATUS OF SERVICE STATIONS AT END IS:

STATION	NUMBER OF DEPARTURES	UTILIZATION %	AVERAGE TIME PER SERVICE	STATUS
1	5000	91.98	10.2	OPEN

QUEUE CONTENT			ENTRIES		ZERO
CURRENT	MAXIMUM	AVERAGE	TOTAL	ZEROES	%
36	46	10.81	5027	370	7.36

AVERAGE WAIT TIME ALL ENTRIES	AVERAGE WAIT TIME UNITS ENTERING QUEUE
119.25	128.73

(4b) 1 (ONE) QUEUE SERVED BY 2 PARALLEL SERVICE STATIONS
AVERAGE ARRIVAL INTERVAL WAS SPECIFIED AS 11 TIME UNITS
AVERAGE SERVICE TIME WAS SPECIFIED AS 20 TIME UNITS
QUEUE LENGTH AT START OF SIMULATION WAS 8

STATUS OF SERVICE STATIONS AT START WAS:

STATION	STATUS
1	USED
2	USED

SAMPLE SIZE IS 5000 DEPARTURES

TIME ELAPSED FOR SIMULATION IS 55025 TIME UNITS

STATUS OF SERVICE STATIONS AT END IS:

STATION	NUMBER OF DEPARTURES	UTILIZATION %	AVERAGE TIME PER SERVICE	STATUS
1	2531	92.05	20.01	OPEN
2	2469	88.93	19.82	USED

QUEUE CONTENT			ENTRIES		ZERO
CURRENT	MAXIMUM	AVERAGE	TOTAL	ZEROES	%
2	38	7.12	4993	718	14.38

AVERAGE WAIT TIME ALL ENTRIES	AVERAGE WAIT TIME UNITS ENTERING QUEUE
78.42	91.59

(4c) 1 (ONE) QUEUE SERVED BY 3 PARALLEL SERVICE STATIONS
AVERAGE ARRIVAL INTERVAL WAS SPECIFIED AS 11 TIME UNITS
AVERAGE SERVICE TIME WAS SPECIFIED AS 30 TIME UNITS
QUEUE LENGTH AT START OF SIMULATION WAS 0

STATUS OF SERVICE STATIONS AT START WAS:

STATION	STATUS
1	USED
2	USED
3	OPEN

SAMPLE SIZE IS 5000 DEPARTURES

TIME ELAPSED FOR SIMULATION IS 55615 TIME UNITS

STATUS OF SERVICE STATIONS AT END IS:

STATION	NUMBER OF DEPARTURES	UTILIZATION %	AVERAGE TIME PER SERVICE	STATUS
1	1709	93.90	30.56	OPEN
2	1646	91.71	30.99	OPEN
3	1645	89.11	30.13	OPEN

QUEUE CONTENT			ENTRIES		ZERO
CURRENT	MAXIMUM	AVERAGE	TOTAL	ZEROES	%
0	33	6.06	4998	774	15.49

AVERAGE WAIT TIME ALL ENTRIES	AVERAGE WAIT TIME UNITS ENTERING QUEUE
67.47	79.83

(4d) 1 (ONE) QUEUE SERVED BY 2 PARALLEL SERVICE STATIONS
AVERAGE ARRIVAL INTERVAL WAS SPECIFIED AS 11 TIME UNITS
AVERAGE SERVICE TIME WAS SPECIFIED AS 10 TIME UNITS
QUEUE LENGTH AT START OF SIMULATION WAS 0

STATUS OF SERVICE STATIONS AT START WAS:

STATION	STATUS
1	USED
2	OPEN

SAMPLE SIZE IS 5000 DEPARTURES

TIME ELAPSED FOR SIMULATION IS 55796 TIME UNITS

STATUS OF SERVICE STATIONS AT END IS:

STATION	NUMBER OF DEPARTURES	UTILIZATION %	AVERAGE TIME PER SERVICE	STATUS
1	2973	54.60	10.25	USED
2	2027	36.27	9.98	OPEN

QUEUE CONTENT			ENTRIES		ZERO
CURRENT	MAXIMUM	AVERAGE	TOTAL	ZEROES	%
0	9	.24	5000	3556	71.12

AVERAGE WAIT TIME ALL ENTRIES	AVERAGE WAIT TIME UNITS ENTERING QUEUE
2.72	9.42

(4e) 1 (ONE) QUEUE SERVED BY 1 PARALLEL SERVICE STATIONS
AVERAGE ARRIVAL INTERVAL WAS SPECIFIED AS 11 TIME UNITS
AVERAGE SERVICE TIME WAS SPECIFIED AS 10 TIME UNITS
QUEUE LENGTH AT START OF SIMULATION WAS 5

STATUS OF SERVICE STATIONS AT START WAS:

STATION	STATUS
1	USED

SAMPLE SIZE IS 5000 DEPARTURES

TIME ELAPSED FOR SIMULATION IS 55556 TIME UNITS

STATUS OF SERVICE STATIONS AT END IS:

STATION	NUMBER OF DEPARTURES	UTILIZATION %	AVERAGE TIME PER SERVICE	STATUS
1	5000	89.66	9.96	OPEN

QUEUE CONTENT			ENTRIES		ZERO
CURRENT	MAXIMUM	AVERAGE	TOTAL	ZEROES	%
1	24	3.11	4995	533	10.67

AVERAGE WAIT TIME ALL ENTRIES	AVERAGE WAIT TIME UNITS ENTERING QUEUE
34.59	38.72

NOTE:
SERVICE TIME HAD PROBABILITY DISTRIBUTION:

0 .75 1 1.25

(4f) 1 (ONE) QUEUE SERVED BY 1 PARALLEL SERVICE STATIONS
AVERAGE ARRIVAL INTERVAL WAS SPECIFIED AS 11 TIME UNITS
AVERAGE SERVICE TIME WAS SPECIFIED AS 10 TIME UNITS
QUEUE LENGTH AT START OF SIMULATION WAS 0

STATUS OF SERVICE STATIONS AT START WAS:

STATION	STATUS
1	OPEN

SAMPLE SIZE IS 5000 DEPARTURES

TIME ELAPSED FOR SIMULATION IS 55109 TIME UNITS

STATUS OF SERVICE STATIONS AT END IS:

STATION	NUMBER OF DEPARTURES	UTILIZATION %	AVERAGE TIME PER SERVICE	STATUS
1	5000	90.64	9.99	OPEN

QUEUE CONTENT			ENTRIES		ZERO
CURRENT	MAXIMUM	AVERAGE	TOTAL	ZEROES	%
1	2	.12	5001	2496	49.91

AVERAGE WAIT TIME ALL ENTRIES	AVERAGE WAIT TIME UNITS ENTERING QUEUE
1.34	2.67

NOTE:
ARRIVAL SERVICE HAD IDENTICAL NON-POISSON DISTRIBUTION;

0 .75 1 1.25

The significance of the probability distribution is clearly demonstrated by comparing lines 1, 5, and 6. The average waiting time is reduced by 60 percent or more when the service interval follows a narrow, linear distribution rather than the Poisson distribution, and waiting is all but eliminated when both arrival and service intervals fall in a narrow range. This shows that good scheduling reduces wasteful waiting time without having to change the service capacity.

Mean Interval		Percent of Utilization		Average Queue Length		Average Waiting Time	
Arrival	Service	Analytical	Simulated	Analytical	Simulated	Analytical	Simulate
11	10	90.9	92.0	9.1	10.8	100.0	119.3
11	20	90.9	90.5	8.7	7.1	95.2	78.4
11	30	90.9	91.6	8.3	6.1	91.6	67.5
11	10	45.5	47.2	.24	.24	2.6	2.7
11	10^1	90.9	89.7	n.a.	3.1	n.a.	34.6
11^2	10^2	90.9	90.6	n.a.	.12	n.a.	1.3

Table 4: Analytical and simulated solutions of queue problems. The superscript 1 indicates that the service interval has linear probability distribution. The superscript 2 indicates that both the arrival and the service interval had linear probability distribution.

Conclusion

The program I have presented cannot compete with the very powerful, special-purpose simulation languages that are commercially available. But it does give you the ability to analyze queueing situations with your own probability distributions. My program only simulates single-phase service. But with the program's modular structure, you should be able to expand it to include other queueing models. A second Departure module can be added for simulation of multi-phased service. Or a switch can be inserted in the Arrival module to activate an additional service station whenever the queue length exceeds a specified value.

Other possible variations include putting a limitation on the queue length (limited parking space, for example) and counting the number of customers lost because they leave without joining the queue. You could use this information to justify more parking space. In short, the program is flexible; you should be able to adapt it to many queueing situations.

* * * * * * * * *

Our sincere appreciation is extended to the editors of *Byte* who generously permitted the re-publication of "A Risky Business - An Introduction to Monte Carlo Venture Analysis," by Pat Macaluso and "Queue Simulation," by E. Hart Rasmussen. These articles first appeared in Byte's March 1984 special issue on modeling and are clear and useful applications of the topics covered earlier in this manual.

Listing 1: A program, written in Applesoft BASIC, for queue simulation.

QUEUE SIMULATION

```
]LIST0,1999                    START

1000 DIM FA(25,2),FS(25,2),TD(10),S%(10),ST(10),CS%(10)
1010 R = 0: z = 0:A=0:S% = 0:T = 0:QL% = 0:CI% = 0:C2% = 0
1020 ONERR GOTO 8800
1100 GOTO 7000

]LIST2000,2999                 ADVANCE MODULE

2000 A = TA - T:N = 0
2100 FOR Z = 1 to S%
2110 IF S%(Z) = 0 THEN 2130
2120 IF A > = TD(Z) - T THEN A = TD(Z) - T:N = 1:S = Z
2130 NEXT
2300 T + T + A
2310 QT + QT + QL% * A
2400 FOR Z = 1 to 5%
2410 IF S%(Z) = 1 THEN ST(Z) = ST(Z) + A
2420 NEXT
2500 IF N = 1 THEN 4000

]LIST3000,3999                 ARRIVAL MODULE

3000 C1% = C1% + 1
3050 IF P < 3 THEN R = RND (1):F = - LOG (1 - R):GOTO 3130
3100 R + RND (1):X = 0
3110 IF R > FA(X,1) THEN X = X + 1:GOTO 3110
3120 F + FA(X - 1,2) + (R = FA(X - 1,1)) * (FA(X,2) - FA(X - 1,2))/ (FA(X,1) - FA(X - 1,1))
3130 TA = T + F * AI
3200 FOR Z = 1 to S%
3210 IF S%(Z) = 0 THEN S = Z:CO% = CO% + 1" GOTO 4250
3220 NEXT
3230 QL% = QL% + 1
3240 IF QM% < QL% THEN QM% = QL%
3250 GOTO 2000
```

```
]LIST4000,4249                    DEPARTURE MODULE
4000  S%(S) = 0
4010  CS%(S) = CS%(S) + 1
4050  C2% = C2% + 1
4100  IF C2% = > C% THEN 5000
4150  IF QL% = 0 THEN 2000
4200  QL% = QL% - 1

]LIST4250,4999                    SEIZE MODULE

4250  S%(S) = 1
4300  IF P = 2 THEN 4400
4310  IF P = 4 THEN 4400
4320  IF P = 5 THEN 4400
4340  R + RND (1):F = - LOG (1 - R): GOTO 4430
4400  R + RND (1):X = 0
4410  IF R > FS(X,1) THEN X = X + 1: GOTO 4430
4420  F + FS(X - 1,2) + (R - FS(X - 1,1) * (FS(X,2) - FS(X - 1,2))/(FS(X,1) - FS(X - 1,1)
4430  TD(S) = T + F * SI
4600  GOTO 2000

]LIST5000,5199                    CALCULATE AVERAGES AND PERCENTAGES

5000  FOR Z = 1 to S%: IF CS%(Z) = 0 THEN 5030
5010  TS(Z) = INT (100 * ST(Z) / CS%(Z) + .5) / 100
5020  US(Z) = INT (10000 * ST(Z) / T + .5) / 100
5030  NEXT
5040  Q0 = INT (10000 * C0% / C1% + .5) / 100
5050  Q1 = INT (100 * QT / C1% + .5) / 100
5060  IF C1% = C0% THEN Q2 = 0: GOTO 5080
5070  Q2 = INT (100 * QT / (C1% - C0%) + .5) / 100
5080  QA = INT (100 * QT / T + .5) / 100

]LIST5200,5399                    SELECT OUTPUT DEVICE

5200  HOME : PRINT CHR$ (7): PRINT CHR$ (7): VTAB (7)
5210  PRINT TAB ( 10)"SIMULATION COMPLETED"
5220  PRINT : PRINT TAB ( 13)"READY TO REPORT"
5230  PRINT : PRINT TAB ( 11)"Shall Report Go To"
5240  PRINT : PRINT TAB (  7)"SCREEN (S) or PRINTER (P) ?"
5250  PRINT : PRINT TAB (19)" ";: GET AS
5260  IF A$ = "S" THEN 5700
5270  IF A$ = "P" THEN 5400
5280  PRINT CHR$ (7): PRINT " Please answer 'S' or 'P'": GOTO 5250
]LIST5400,5699                    OUTPUT TO PRINTER
```

```
5400 HOME : PRINT : INPUT "What is Date of Report? ";D$
5410 PRINT : PRINT "What is Project Identification?"
5420 PRINT : INPUT I$
5430 PRINT : PRINT "Press RETURN when PRINTER is ready ";: GET AS
5440 PR# 1
5450 PRINT : PRINT TAB ( 12)D$: PRINT
5460 PRINT TAB( 40 - LEN (I$) / 2)I$
5470 PRINT TAB( 32)"QUEUE SIMULATION"
5480 PRINT : PRINT : PRINT
5490 PRINT TAB( 12)"1 (ONE) QUEUE SERVED BY ";S%;" PARALLEL SERVICE
     STATIONS
5500 PRINT TAB( 12)"AVERAGE ARRIVAL INTERVAL WAS SPECIFIED AS ";AI;"
     TIME UNITS"
5510 PRINT TAB( 12)"AVERAGE SERVICE TIME WAS SPECIFIED AS ";SI;"
     TIME UNITS
5520 PRINT TAB( 12)"QUEUE LENGTH AT START OF SIMULATION WAS ";QL
5530 PRINT : PRINT TAB( 12)"STATUS OF SERVICE STATIONS AT START WAS:"
5540 PRINT : PRINT TAB( 20)"STATION"; TAB( 36)"STATUS": PRINT
5550 FOR Z = 1 to S%: IF A$(Z) = "Y" THEN S$(Z) = "USED"
5560 IF A$(Z) < > "Y" THEN S$(Z) = "OPEN"
5570 PRINT TAB( 23)Z; TAB( 37)S$(Z): NEXT
5580 PRINT : PRINT TAB( 12)"SAMPLE SIZE IS ";C%;" DEPARTURES"
5590 PRINT : PRINT TAB( 12)"TIME ELAPSED FOR SIMULATION IS ";
     INT (T + .5);" TIME UNITS"
5600 PRINT : PRINT TAB( 12)"STATUS OF SERVICE STATIONS AT END IS:":
     PRINT : PRINT TAB( 12)"STATION NUMBER OF UTILIZATION AVERAGE
     TIME STATUS"
5610 PRINT TAB( 12)"      DEPARTURES      %      PER SERVICE": PRINT
5615 FOR Z = 1 to S%:  IF S%(Z) = 0 THEN S$(Z) = "OPEN"
5620 IF S%(Z) = 1 THEN S$(Z) = "USED"
5625 PRINT TAB( 15)Z; TAB( 24)CS%(Z); TAB( 35)US(Z); TAB( 49)TS(Z);
     TAB( 22)S$(Z): NEXT
5630 PRINT : PRINT : PRINT TAB( 18)"QUEUE CONTENT"; TAB( 42) "ENTRIES   ZERO"
5640 PRINT TAB( 12)"CURRENT MAXIMUM AVERAGE"; TAB( 40)"TOTAL ZEROS  %"
5650 PRINT TAB( 15)QL%;TAB( 24)QM%; TAB( 32)QA; TAB( 41)C1%; TAB(  8)C0%;
     TAB( 15)Q0
5660 PRINT : PRINT TAB( 12)"AVERAGE WAIT TIME"; TAB( 42)"AVERAGE WAIT
     TIME": PRINT TAB( 15)"ALL ENTRIES; TAB( 40)"UNITS ENTERING QUEUE"
5670 PRINT TAB( 18)Q1; TAB( 48)Q2
5680 IF P > 1 THEN 6500
5690 PR# ): GOTO 6000
```

```
]LIST5700,5999                    OUTPUT TO SCREEN

5700 HOME : PRINT
5710 PRINT "                QUEUE SIMULATION"
5720 PRINT: PRINT TAB( 2)"MEAN ARRV T"; TAB( 15)AI; TAB( 24)"MEAN
     SERV T"; TAB( 36) SI
5730 PRINT TAB( 2)"QUEUE START"; TAB( 15)QL; TAB( 24)"STATIONS"; TAB( 36) SI
5740 PRINT TAB( 2)"SAMPLE SIZE"; TAB( 15)C%; TAB( 24)"TOT TIME"; TAB( 35)
     INT (T + .5)
5750 PRINT : PRINT " STAT DEPART UTILIZ AVG TIME STATUS": PRINT
5760 FOR Z = 1 TO S%
5770 IF S%(Z) = 0 THEN S$(Z) = "OPEN"
5780 IF S%(Z) = 1 THEN S$(Z) = "USED"
5790 PRINT TAB( 4)Z; TAB( 9)CS%(Z); TAB( 17)US(Z); TAB( 25) TS(Z); TAB( 35)S$(Z)
5800 NEXT
5810 PRINT : PRINT "    QUEUE CONTENT      ENTRIES      %"
5820 PRINT " CURR  MAXI  AVERAGE  TOTAL  ZEROES  ZERO"
5830 PRINT TAB( 2)QL%; TAB( 8)QM%; TAB( 14)QA; TAB( 23)C1%; TAB( 29)C0%;
     TAB( 35) Q0
5840 PRINT TAB( 2)"AVG WAIT A"; TAB( 13)Q1; TAB( 22)"AVG WAIT Z"; TAB( 35)Q0
5850 PRINT : PRINT "Do you want printed copy?  Then press 'Y'"
5860 PRINT "Otherwise press RETURN when ready    ";: GET AS
5870 IF A$ = "Y" THEN 5400

]LIST6000,6499                    SELECT MODE SIMULATIONS OR END

6000 HOME : PRINT : PRINT
6010 PRINT "Want to do another simulation  (Y/N) ?"
6020 PRINT : PRINT TAB( 19)" ";: GET AS
6030 IF A$ + "Y" THEN CLEAR : DIM FA(25,2),FS(25,2): RESTORE : GOTO 7300
6040 IF A$ = "N" THEN END
6050 PRINT : PRINT CHR$ (7): PRINT "Please answer 'Y' or 'N'": GOTO 6020

]LIST 6500,6999                   AUXILIARY PRINTER OUTPUT

6500 PRINT : PRINT : PRINT TAB( 12)"NOTE:"
6510 ON P GOTO 5690,6520,6550,6580,6610
6520 PRINT TAB( 12)"SERVICE TIME HAD PROBABILITY DISTRIBUTION:": PRINT
6530 GOSUB 6750
6540 PRINT : GOTO 5690
6550 PRINT TAB( 12)"ARRIVAL TIME HAD PROBABILITY DISTRIBUTION:": PRINT
6560 GOSUB 6700
6570 PRINT : GOTO 5690
6580 PRINT TAB( 12)"ARRIVAL AND SERVICE HAD IDENTICAL NON-POISSON
     DISTRIBUTION:": PRINT
```

```
6590  GOSUB 6700
6600  PRINT : GOTO 5690
6610  PRINT TAB( 12)"ARRIVAL TIME HAD PROBABILITY DISTRIBUTION:": PRINT
6620  GOSUB 6700
6630  PRINT : PRINT
6640  PRINT TAB( 12)"SERVICE TIME HAD PROBABILITY DISTRIBUTION:": PRINT
6650  GOSUB 6750
6660  PRINT : GOTO 5690
6700  X = 0
6710  PRINT TAB( 12)FA(X,1);" ";FA(X,2);" ";
6720  IF FA(X,1) = 1 THEN RETURN
6730  X = X + 1: GOTO 6710
6750  X = 0
6760  PRINT TAB( 12)FS(X,1);" ";FS(X,2);" ";
6770  IF FS(X,1) = 1 THEN RETURN
6780  X = X + 1: GOTO 6760

]LIST7000,7199                    OPENING SCREEN

7000  HOME
7010  PRINT : PRINT : PRINT : PRINT
7020  PRINT "   ****************************"
7030  PRINT "   *                          *"
7040  PRINT "   *     QUEUE SIMULATION     *"
7050  PRINT "   *                          *"
7060  PRINT "   *           BY             *"
7070  PRINT "   *                          *"
7080  PRINT "   *     E. HART RASMUSSEN    *"
7090  PRINT "   *           PMS            *"
7100  PRINT "   *                          *"
7110  PRINT "   ****************************"
7120  PRINT : PRINT : PRINT
7130  PRINT"       Press RETURN to start"
7140  PRINT : PRINT TAB( 19): GET AS

]LIST7200,7599                    ESTABLISH PROBABILITY DISTRIBUTION

7200  HOME
7210  PRINT " Shall ARRIVAL and SERVICE have"
7220  PRINT : PRINT "  identical distributions  (Y/N)    ";: INPUT A%
7230  IF A$ = "Y" THEN P = 1: GOTO 7260
7240  IF A$ = "N" THEN P = 2: GOTO 7300
7250  PRINT CHR$ (7): PRINT "   Please answer 'Y' or 'N'     ";: INPUT A$: GOTO 7230
7260  PRINT : PRINT " Are they both Poisson   ";:  INPUT A$
7270  IF A$ = "Y" THEN 7600
```

```
7280 IF A$ = "N" THEN P = 4: GOTO 7400
7290 PRINT CHR$ (7): PRINT "  Please answer 'Y' or 'N'   ";:  INPUT A$: GOTO 7270
7300 PRINT : PRINT "   Does ARRIVAL have Poisson ";:  INPUT A$
7310 IF A$ = "Y" THEN 7400
7320 IF A$ = "N" THEN P = 3: GOTO 7360
7330 PRINT CHR$ (7): PRINT  "  Please answer 'Y' or 'N'  ";: INPUT A$: GOTO 7310
7360 PRINT : PRINT "  Does SERVICE have Poisson ";: INPUT A$
7370 IF A$ = "Y" THEN 7400
7380 IF A$ = "N" THEN P = 5: GOTO 7400
7390 PRINT CHR$ (7): PRINT  "   Please answer 'Y' or 'N' ;: INPUT A$: GOTO 7370
7400 ON P GOTO 7600,7410,7420,7430,7440
7410 GOSUB 7550: GOTO 7600
7420 GOSUB 7500: GOTO 7600
7430 GOSUB 7500: RESTORE : GOSUB 7550: GOTO 7600
7440 GOSUB 7500: GOSUB 7550: GOTO 7600
7500 X = -1
7510 X + X + 1:  READ FA(X,1),FA(X,2)
7520 IF FA(X,1)  <  1 THEN 7510
7530 RETURN
7550 X = -1
7560 X = X + 1: READ FS(X,1),FS(X,2)
7570 IF FS(X,1)  <  1 THEN 7560
7580 RETURN

]LIST7600,7999                          INPUT SIMULATION DATA

7600 HOME : PRINT
7610 PRINT : INPUT  "What is AVERAGE Arrival Intervals?   ";AI
7620 PRINT : INPUT  "What is AVERAGE Service Time?  ";SI
7630 PRINT : INPUT  "How many Service Stations are used?  ";S%
7640 PRINT " PRINT "Is there a waiting Queue at Start?   ";: INPUT A$
7650 IF A$ = "N" THEN PRINT : GOTO 7680
7660 IF A$ < > "Y" THEN PRINT CHR$ (7): PRINT "Please answer 'Y' or 'N'  ";:
     INPUT A$: GOTO 7650
7670 PRINT : INPUT "How many are waiting?  ";QL%:QL = QL%:QM% = QL%: PRINT
7680 PRINT "Are any Service Stations in Use  ";: INPUT A$
7690 IF A$ = "N" THEN 7900
7700 IF A$ < > "Y" THEN PRINT CHR$ (7): PRINT "Please answer 'Y' or 'N' ";:
     INPUT A$: GOTO 7690
7710 IF S% = 1 THEN A$(1) = "Y": GOTO 7900
7720 PRINT : PRINT "  Service Station #    In Use (Y/N)": PRINT
7730 FOR X = 1 to S%
7740 PRINT "         ";X;:  INPUT         ";A$(X)
7750 NEXT
7900 PRINT : PRINT : INPUT "Size of Simulation Sample:     ";C%
```

```
]LIST8000,8799              CALCULATE FIRST ARRIVAL AND DEPARTURE(S) TIMES

8000  HOME : PRINT : PRINT : PRINT : PRINT : PRINT TAB( 15) "PLEASE WAIT"
8010  PRINT : PRINT TAB( 10)"  ";: FLASH : PRINT "SIMULATION RUNNING"
8020  NORMAL
8090  IF P < 3 THEN R = RND (1):F = - LOG (1 - R): GOTO 8130
8100  R + RND (1):X = 0
8110  IF R > FA(X,1) THEN X = X + 1: GOTO 8110
8120  F + FA(X - 1,2) + (R - FA(X - 1,1)) * (FA(X,2) - FA(X - 1,2)) / (FA(X,1) - FA(X - 1,1))
8130  TA = F * AI
8150  FOR Z = 1 to S%
8160  IF A$(Z) = "Y" THEN S%(Z) = 1
8170  IF S%(Z) = 0 THEN TD(Z) = S% * 100 + AI: GOTO 8300
8180  IF P = 4 THEN 8240
8190  IF P = 5 THEN 8240
8200  R + RND (1):F = - LOG (1 - R): GOTO 8270
8240  R + RND (1):X = 0
8250  IF R > FS(X,1) THEN X = X + 1: GOTO 8250
8260  F = FS(X - 1,2) + (R - FS(X - 1,1)) * (FS(X,2) - FS(X - 1,2)) / (FS(X,1) - FS(X - 1,1))
8270  TD(Z) = F * SI
8300  NEXT
8500  GOTO 2000

]LIST8800,                          ERRORS IN DATA STATEMENTS

8800  HOME : PRINT : PRINT
8810  FOR X = 1 TO 3: PRINT CHR$ (7): NEXT
8820  PRINT "PLEASE CORRECT THE DATA STATEMENTS"
8830  PRINT : PRINT : PRINT : PRINT  "  NON-POISSON DISTRIBUTIONS"
8840  PRINT : PRINT "    SHOULD BE IN DATA STATEMENTS"
8850  PRINT : PRINT "       STARTING AT LINE 9000"
8860  PRINT : LIST 9000,
8870  END
```