The general nature of digital simulation of a system is discussed.

A machine-independent examination of the associated programming problem is conducted and illustrated by means of an example.

Finally, the nature and application of simulation languages are noted.

Systems simulation with digital computers

by K. Blake and G. Gordon

By *system,* we have in mind the meaning accorded by Webster, "a set of objects united by some form of regular interaction or interdependence." Change in a system is regarded as change in the properties of the objects or in the relationships between the objects. That is, when one considers a physical system or conceives of a hypothetical system, one perceives certain distinct objects such as machines, customers, messages, etc. Each object is seen to possess certain properties of interest that may be used to define it, machine availability, credit status, or length of a message.

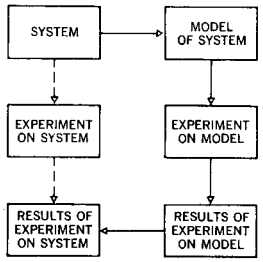
Most important, one perceives certain changes taking place as a result of events in the system. These changes, such as the loading of a machine, making a payment, or sending a message, describe relations among the objects and determine changes in the values of their properties.

Associated with the design of a system, two objectives are always present. We wish to:

1. Be able to predict performance before the system is built.
2. Have assurance that the system design selected is optimal in terms of the design criteria adopted.

Consequently, systems studies are frequently conducted with some representation of the system, called a *model.* By working with a properly constructed model, the systems engineer is able to make inferences about a proposed system from experiments conducted with the system model (Figure 1) and accomplish the

Figure 1 Direct and indirect experimentation



first design objective. We note that the use of models amounts to replacing "direct" by "indirect" experimentation. Either physical or mathematical models may be employed.

physical models

Physical models of a system represent the system to be studied in a concrete manner by establishing some other physical system that is equivalent to the system being studied. The alternative system may be a scale model which is more convenient to experiment with than the actual system, such as, for example, the use of scale models in wind tunnels for the design of aircraft. Other types of physical models rely upon an analogy between the system being studied and some physical system of a different nature which is easier to build or manipulate. Objects in the system being studied can be identified with elements of the model, and the relationships between the objects of the system can be reflected in the relationships between the elements of the model because of some underlying similarity in the physical laws that control the two systems. For example, electrical networks can be used to study problems of mechanical vibration because of the similarity between the equations governing the performance of electrical circuits and mechanical systems. A correspondence can be established between the inductance, capacitance, and resistance of electrical circuits and the inertia, resilience, and friction of mechanical systems.

The second design objective can be partially satisfied by constructing a number of different models, measuring the performance of each and, by means of a comparative examination of results, selecting the best design. However, building physical models is usually expensive and time consuming. Furthermore, there is no assurance that another design (other than those considered) would not be superior.

mathematical models

In contrast, a mathematical model of a system represents the system in a more flexible manner. It is constructed by identifying the objects and properties of the system with mathematical variables and representing the relationships of the system in the form of mathematical relations among these variables. *Analytic* techniques can then be applied. Other mathematical relations implicitly determined but not explicitly included in the given ones may be derived. For example, the model may include equations which can be solved to determine functional relationships between variables that are not given directly by the model description. In particular, it is often possible to determine relationships that may be evaluated in order to make the desired predictions relative to performance of the system. Sometimes this procedure can be carried out in terms of parameters and, by means of optimization techniques, values determined for assignment to the parameters which will correspond to an optimal system design.

With complex systems, however, analytic methods often become very difficult (or even impossible in terms of current mathematical knowledge).

digital simulation

If analytic techniques are not possible, mathematical models may be used to derive information about the system performance

by following the change of state of the system resulting from the succession of events affecting the system. This can be accomplished by using numerical techniques to follow the corresponding changes in the mathematical model, and the technique is called *digital simulation.*

Of course, a "succession of events" suggests presence of *time* as the single independent variable, so that we are now focusing attention on a particular type of system—but one including a wide variety of important instances.

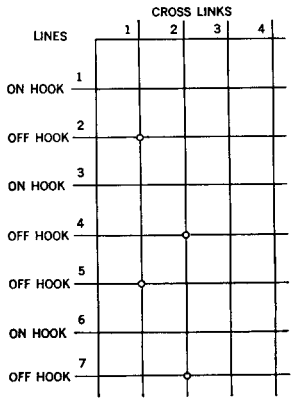
Model construction permits the use of any relationship that can be described satisfactorily by a mathematical or a logical statement. In particular, digital simulation is extremely valuable for studying systems in which model relationships are not deterministic and the behavior of some variables must be described by probability distributions. With digital simulation, it is relatively easy to accompany variables by probability distributions and, at each mention of a variable, to simulate random sampling. Where random processes are involved, simulation depends on the disciplines of statistical inference and experimental design for help in estimating probability distributions and in interpreting results.

Within the limits of model accuracy, digital simulation predicts the performance of a system before it is built. Finite experimentation never guarantees an optimal design, but the economy and speed of digital simulation permits more trial and improvement than is feasible with physical models.

simulation with digital computers

There are two principal tasks involved in preparing a digital computer to carry out a simulation. A model of the system to be simulated must be produced, and a program must be written to carry out the procedures involved in following the changes in the model. Essentially, a digital computer is a device for storing numbers and manipulating the numbers according to certain rules embodied in the computer program. To make a computer simulate a system, therefore, it is necessary to create an image of the system in the form of a set of numbers that represent the state of the system, and to write a computer program that embodies the relationships controlling the changes of state in the system.

Figure 2 Simple telephone exchange



To illustrate the principles involved in a digital simulation, consider the simple example of a telephone exchange illustrated in Figure 2. The system consists of a number of telephones, each connected to an exchange by its own line. The exchange contains a number of cross links which have the ability to connect any two lines together subject to the condition that only one connection at a time can be made to any given line. The current state of the system (Fig. 2), is that line 2 is connected to line 5 through cross link 1, and line 4 is connected to line 7 through cross link 2.

One way in which this system can be represented as a set of numbers is illustrated in Figure 3, which has three tables of numbers recording the state of the various system elements. One table shows the current status of each line. A *zero* means the

line is not connected, a *non-zero* number means the line is connected, and the value of the number represents the link used for the connection. A second table shows the status of the links. Here a *zero* means the link is not being used, and a *one* means that the link is being used. There is also a table of calls that are either in the system or waiting to enter the system. Several numbers are needed to record the information relevant to each call. Different numbers show the origin and destination of the call, the time it is due to begin and, for calls that are connected, the time the call will finish (this information being generated as explained below).

The sets of numbers in these three tables record the current state of the system and detail the potential events with their expected times of execution. They therefore constitute a model of the system which can be interpreted to determine the next state of the system. The simulation program must manipulate this model in a way that represents the succession of states followed by the real system. To do this with the system being described, it must have a number that represents the clock time existing in the real system. It must also have a procedure by which to introduce new calls to the system as the clock advances. A predetermined list of calls ordered by time of arrival and giving their duration may have been prepared, or a computational procedure can be employed to derive from probability distributions the interval between successive arrivals and the length of each call.

The set of numbers in Figure 3, for example, shows that at time 1027, the connections in the system are as shown in Figure 2, and it can be deduced that the next event will be that the call between lines 2 and 5 will be disconnected at time 1053. The simulation program determines that this is the next event, and proceeds to record the change of state that will occur at that time by modifying the numbers in Figure 3 to the form shown in Figure 4. The clock has been updated, the terminated call has been removed, and the fact that lines 2 and 5 and cross link 1 are now free is noted.

If a potential event cannot be implemented because of some condition in the system, the simulation program must determine this fact and take whatever action is dictated by the logic of the

Figure 3 System image before change

LINES

1	0
2	1
3	0
4	2
5	1
6	0
7	2

CROSS LINKS

1	1
2	1
3	0
4	0

CLOCK

1027

CALLS

BEGIN	FROM	TO	FINISH
1016	2	5	1053
1019	4	7	1064
1056	3	7	
1072	6	2	

Figure 4 System image after change

	LINES
1	0
2	0
3	0
4	2
5	0
6	0
7	2

	CROSS LINKS
1	0
2	1
3	0
4	0

CLOCK
1053

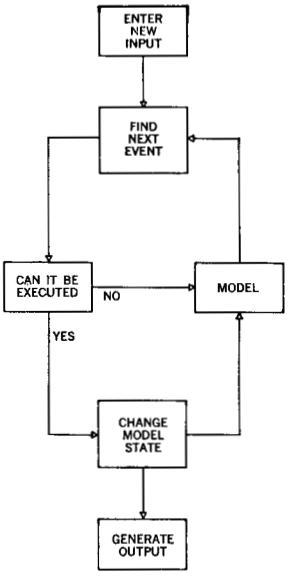
	CALLS			
	BEGIN	FROM	TO	FINISH
	1019	4	7	1064
	1056	3	7	
	1072	6	2	

system. For example, the next potential event after time 1053 will be that a new call from line 3 attempts to connect to line 7 at time 1056. However, line 7 will be busy at that time. If the logic of the system is that such a blocked call will be lost, the program must remove this call. If, however, a blocked call waits until the line being called becomes free, then the program must remember this fact and offer line 7 to line 2 as soon as the call between 4 and 7 is terminated.

By following the state of the system, step by step, in the manner that has been illustrated, a simulation can follow the behavior of the system. With appropriate programming, a great deal of valuable information can be derived about the performance of the system by accumulating statistics on such factors as queue lengths, equipment utilization and the time taken to complete transactions of the system.

There are many ways in which a simulation program could be organized, and a complete procedure for writing simulation programs cannot be given here. The over-all operations to be performed by a simulation program, however, can be illustrated as shown in Figure 5. Given a model of the system, the program must first find the next event that might occur in the system. Among the potential occurrences, it may be time to enter a new input into the system, and the program must arrange for this entry. Having found the next potential event, the program must study the relationships inherent in the system and determine whether the event can be executed. If so, the state of the model is changed. If not, the program proceeds to the next potential event, noting that the blocked event may re-appear as a future event when conditions in the system change.

Figure 5 General organization of simulation



At the time of changing the state of the model, it may be necessary to extract some statistics which will form part of the output, showing how the system performs. This cycle of operations is repeated as many times as is necessary to complete the simulation, and upon completion, there may be a final phase of computation for gathering statistics and reporting on the performance of the system.

The operations being performed in this process of simulation

are reflected in the programming tasks illustrated in Figure 6. The model representing the system must be converted into a set of tables such as that shown in Figure 3. Determining the next event of the system involves scanning the events recorded in tables, and introducing new inputs involves the generation of new data. Determining whether an event can be executed requires a number of logical tests. According to the results of these tests, the tables that constitute the model must be updated or the program begins another cycle of actions. Generating the output involves the computation of statistics and organizing these statistics in reports.

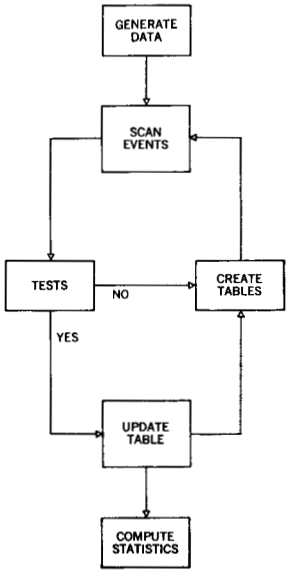
model construction

In constructing the model that represents the system, the main problems to be considered are the efficient use of computer memory space and the effect of the model organization on the simulation processing time. In some parts of the model, such as the line and cross link status tables of Figure 3, the number of elements is known and the required table size is fixed once the amount of information to be recorded about the elements has been determined. The program must constantly refer to this information to carry out the tests that determine whether an event can be executed or not. The main concern, therefore, is to organize such data compactly to save space and yet minimize the amount of time spent extracting fields and decoding information.

Other parts of the model, such as the records of calls in the system of Figure 3, represent elements that fluctuate in number and tend to arrive and depart in a random manner. Such records must be scanned to determine the sequence of events in the system. Careful consideration must therefore be given to the efficient organization of such records in order to preserve space while minimizing scanning time.

Two basic methods can be used to represent the flow of time in a simulation, and these methods influence the way in which a scan is organized. In some simulations, the clock is updated in uniform intervals of time; the scan is then aimed at finding any events coinciding with the new clock time. A second method is to update the clock to the next most imminent event. Usually one scan is needed to identify the next clock time, and a second scan is then needed to pick out the events that coincide with this time. In a system in which events can be expected to occur in a regular manner, the first method is usually more efficient. Where events occur unevenly in time, the second method is usually more efficient. Frequently the scan is, in fact, a number of scans arranged to consider the different categories of events separately. The next event in each category is determined first, and the next system event is chosen from among these.

Figure 6 Programming tasks performed in simulation



Following the general procedure that has been outlined, it is possible to write a special program for simulating each system to be studied. The maximum possible flexibility is usually incorporated in these programs by arranging an easy modification of the various system operation options that are under the designers'

control. The effects of these design options on system performance can then be checked with different simulation runs. The usual procedure is to arrange that such options are entered as input data for the simulation program used in initializing the model.

simulation languages

However, recent years have seen the development of many general purpose programs aimed at simplifying the tasks of performing simulations on a variety of models. In general, such programs provide a language with which to describe the model of the system and a set of routines capable of carrying out the simulation procedure. The task of simulating a system then becomes one of describing the model in the simulation language and of leaving the establishment of the program's tables and the process of simulation to be performed automatically by the general purpose simulation program.

The various general purpose simulation programs that are available differ in the extent to which they can be applied to different systems and in the degree to which they render the simulation process automatic. Some programs concentrate on a particular class of systems problems, such as job shop operation or inventory control. They are thereby able to employ a language specifically designed for that class of problem and are also able to make the simulation process highly automatic by reflecting the structure of the system in the model and scanning procedure. Other programs, such as GPSS II,[1] provide a language that can be applied generally to a broad class of systems while maintaining a relatively fixed set of procedures for carrying out the simulation automatically. Yet other programs, such as SIMSCRIPT,[2] provide a language suitable for a broad class of systems and allow for greater flexibility in organizing the simulation procedures.

The simulation program best suited for a particular simulation study depends upon the nature of the system and upon the programming skill of the individual conducting the study. As a general rule, an increase in the flexibility of a simulation program is obtained at the cost of requiring more understanding of programming procedures.

The future will undoubtedly see a steady improvement in general purpose simulation programs resulting from a better understanding of simulation languages evolving from applications of simulation and from the use of more skillful programming techniques.

CITED REFERENCES

1. R. Efron and G. Gordon, "A General Purpose Digital Simulator and Examples of its Application: Part I—Description of the Simulator," *IBM Systems Journal* **3**, No. 1, 22 (1964).
2. B. Dimsdale and H. M. Markowitz, "A Description of the SIMSCRIPT Language," *IBM Systems Journal* **3**, No. 1, 57 (1964).