THE SIMULATION LANGUAGE SIMAN ON MICROCOMPUTERS AND MAINFRAMES*

James Caristi
Department of Mathematics and Computer Science
Valparaiso University
Valparaiso, IN 46383 U.S.A

Computer simulation is often implemented in third generation languages such as Fortran or BASIC. But many disciplines related to engineering have been using increasingly sophisticated simulation languages such as GPSS, SIMAN, SIMSCRIPT, SLAM, and STELLA. These languages reduce the amount of coding needed to construct a model by including a variety of features pertinent to simulation. Most of the simulation languages have evolved during the last several years to facilitate any combination of discrete, continuous, deterministic, or stochastic modeling. Evidently, it is worth examining these types of software for their potential value in mathematical modeling.

Although some simulation languages are designed to handle continuous systems exclusively, SIMAN and the languages discussed here enable the modeling of both continuous and discrete systems. The software manages time automatically, causes discrete events to occur when necessary, and determines function values for differential equations with built-in numerical techniques (usually Runge-Kutta methods). For stochastic simulation, several random number streams are typically available, and variates can be chosen from any of a dozen distributions by simply specifying a function name. Statistical routines for the analysis of output are included.

The methodology used in the discrete component of a model involves abstract objects known as "entities," which traverse networks in time. Entities may be created or destroyed at any time and may be involved in completely disjoint networks. The entities may be subject to queueing processes or other delays and can affect other aspects of the simulation. Information specific to a particular entity is stored in "attributes" of that entity, while more general information is stored in global or state variables, which are accessible to any entity. Real-life examples which are modeled easily with this methodology include supermarket checkout lanes, traffic flow, assembly lines, and soaking pit furnaces in steel plants. Other scientific problems can be addressed in the framework of entities and

* Accepted August 1, 1988.

networks, particularly stochastic influences on continuous processes. An example involving bacterial growth is described later.

With the exception of STELLA, which is available on the MacIntosh, the simulation software runs on IBM PCs and compatibles, and also on most minicomputers and mainframes. On microcomputers, a math co-processor, an EGA card, and a color monitor are highly recommended. For people in academic institutions engaged in nonprofit activities, simulation languages vary in price from about \$50 to \$1000 (most cost about \$300). Site licenses are usually available.

SIMAN is a simulation language available from Systems Modeling Corporation (The Park Building, 504 Beaver Street, Sewickley, PA 15143). University faculty can obtain the complete system with teaching materials for \$600 (the cost for commercial institutions is \$16,000). Site licenses for small-scale versions are available to universities for \$400. SIMAN is very similar in style to GPSS and SLAM (indeed, the author of SIMAN was involved centrally in the development of SLAM as well). SIMAN was designed to model manufacturing systems; it contains special features for material handling such as "conveyors" and "transporters," which are not included in other simulation languages.

An important distinguishing feature of SIMAN is its separation of any simulation into two parts: a model and an experimental frame. The model describes structural aspects of a system, while the experimental frame supplies initial parameter values and details concerning the running of a simulation (such as the number of replications). The purpose of this separation is to facilitate testing and experimentation with a model. For example, sensitivity analysis can be carried out without making any changes in the model by changing only parts of the experimental frame. Models are also easier to understand when they are not burdened with such details as the exact means and variances to be used or what resetting of system values must be done between simulation runs.

Like many other simulation languages, SIMAN includes a capability for calling user-written subroutines composed in Fortran. Although this feature is slightly awkward to use (since the subroutines must be compiled and linked to the main system separately), it is possible to extend the capabilities of the simulation language. For example, since SIMAN does not include a mechanism for reading values from a data file, one can cause an

event to take place at the appropriate time which invokes a user subroutine for bringing in such data.

It is dangerous to focus attention on problematic aspects of a language, because flaws are often repaired in new revisions or ameliorated by the use of additional software. Nonetheless, version 3.1 of SIMAN has some significant disadvantages. Unless one is using a Tektronix graphics terminal, it is necessary to use the "batch" input mode, which has a difficult syntax and lacks a good user interface. It is almost impossible to predict whether a comma or a colon is required at any given time. There is no on-line help available, which makes it necessary to keep the user's manual always present. The conventions for naming variables are archaic; global variables are all called X(i) and entity attributes are A(i), for some expression i. This flaw can be remedied somewhat by the use of a SYNONYMS statement as shown below.

The following example describes a SIMAN program for simulating bacterial growth using a combined continuous and discrete approach. The continuous component of the model is contained in the Fortran subroutine STATE, which describes the differential equations for SIMAN to solve numerically. The discrete component is responsible for generating random weather conditions, making quantum changes in the bacterial populations, and selecting which differential equation is to govern the bacterial growth.

```
BEGIN;
SYNONYMS:      POPULATION=S(1): MOISTURE=X(1);
               CREATE;
WAITRAIN  DELAY:EX(1,1);           Wait for next rain
          ASSIGN:'MOISTURE'=1;      Rain on
          DELAY:EX(2,1)+0.08333;    Wait while it's raining
          ASSIGN:'MOISTURE'=0;      Rain off
;      Now reduce bacterial population 50% and cycle back
          ASSIGN:'POPULATION'='POPULATION'*0.5:NEXT(WAITRAIN);
END;
```

The expression EX(i,1) denotes a random sample taken from an exponential distribution with the parameter found in parameter set number i of the experiment, using random number stream 1.

```
BEGIN;
```

```

PROJECT,COMBINED CONT DISCR,CARISTI,8/8/88;
DISCRETE,1;
CONTINUOUS,1,0,0.01,0.03,0.0417;
CSTAT:1,S(1),TOTAL POPULATION,1;
INITIALIZE,S(1)=1000;
PARAMETERS:      1,7:
                  2,0.125;
REPLICATE,4,0,100;
END;

```

The CONTINUOUS statement indicates parameters for the Runge-Kutta method and tells that differential equations are contained in the subroutine STATE. The CSTAT statement calls for statistics to be collected for the variable S(1) (the bacterial population size), and hourly observations (indicated by the value 0.0417 in CSTAT) are to be recorded in an output file. Parameter set number one contains the value 7 (days), which is the mean of the exponential distribution used in determining the time between rainfalls. The value 0.125 in parameter set number two is used as the mean of an exponential distribution in determining the rainfall duration. The REPLICATE statement specifies that four simulations are to be run, each for 100 days.

```

SUBROUTINE STATE
COMMON/SIM/D(50),DL(50),S(50),SL(50),X(50),DTNOW,TNOW,
1      TFIN,J,NRUN
IF (X(1).EQ.1) THEN
    D(1)=8.3178*S(1)*(1-S(1)*1.0E-7)
ELSE
    D(1)=0
END IF
RETURN
END

```

The variable D(1) refers to the derivative of S(1). Thus the subroutine STATE causes SIMAN to use either a nonlinear logistic differential equation for bacterial growth, or zero growth, depending on the value of X(1), which represents moisture presence.

To obtain a better appreciation for the power of a simulation language, consider the sample model given below. The problem is to simulate the operation of two queues at a bank.

```

BEGIN;
      CREATE:EX(3,1):MARK(1);      New customers arrive
;      according to an exponential distribution
;      and their time of arrival is recorded
      PICKQ,SNQ,GETMAD:LINE1:LINE2;    Choose smaller line,
;      leave if queue is full
LINE1  QUEUE,1,4;      Get in line for teller 1
      SEIZE:TELLER1;
      DELAY:RN(1,1);      Normally distributed service time
      RELEASE:TELLER1:NEXT(STATS);    Teller freed to
;      serve new clients
LINE2  QUEUE,2,4;      Get in line for teller 2
      SEIZE:TELLER2;
      DELAY:RN(2,1);      Normally distributed service time
      RELEASE:TELLER2;
STATS  TALLY:1,INT(1):DISPOSE;      Record time in bank
GETMAD  COUNT:1:DISPOSE;      Record number of unhappy clients
END;

```

The SIMAN output processor is capable of analyzing the results of simulation runs in different ways. On microcomputers with EGA, bit-mapped color graphics are available with plots, bar charts, histograms, correlograms, various confidence interval plots, moving averages, and ANOVAs. The output processor can create data files which can be used with other software, such as statistical packages or spreadsheets. A separate software system called CINEMA is available for the purpose of displaying an animation of a simulation run.

SIMAN contains reasonable debugging features. A TRACE statement can be inserted in the experimental frame, which allows entities and selected variables to be monitored during a simulation run. One can also use the interactive debugger, which provides a good set of features including tracing, breakpoint creation, stepping, resetting of variable values, and listing of model source statements.

The microcomputer version of SIMAN is fully compatible with the mainframe version. The major limitation of the microcomputer version is that it can handle up to approximately one thousand entities simultaneously (depending upon how many attributes each entity contains). Mainframe versions are restricted only by the maximum size of an array in Fortran. Since most large computers now implement virtual arrays, there is almost no limit to the size of a model.

SIMAN is a powerful language for simulation, which can be recommended especially for experienced programmers. Others should wait for better user interfaces to be developed or investigate STELLA on the MacIntosh. SIMAN is not so different from other simulation languages. Thus it would be more important to acquire the same simulation language on one's microcomputer as is installed already on an accessible mainframe. The usual caveats related to numerical computing apply to SIMAN: Runge-Kutta methods can not handle "stiff" systems, and one must avoid subtractive cancellation and other numerical traps. With care and experience, SIMAN is a good system to use as an aid for investigating the behavior of systems involving discrete and continuous components.