CONDITIONS, CRITERIA, AND CAVEATS FOR COMPUTER SIMULATION WITH COBOL

by

Francis J. Brewerton
Middle Tennessee State University

R. Wayne Gober
Middle Tennessee State University

Elias R. Callahan
Middle Tennessee State University

INTRODUCTION

Simulation, as an emerging separate area of study, has enjoyed burgeoning popularity among scientists, engineers, and managers for some two decades, but particularly within the last ten years. The underlying cause of simulation's increasing popularity as a decision-making tool is based in the combined effect of uncertainty of events, dynamic interactions between decisions and subsequent events, the complex interdependency among variables in the simulation model, the sheer number of events included in the simulation, and the need to use finely divided time intervals.(1) To this extent, simulation may well be the "best" approach to problem solving, as it has a better general capability for handling the above complications which tend to render impractical or impossible more conventional mathematical or statistical solution approaches. However, little unanimity exists on this point; consequently, simulation is to some extent still employed only as a "last resort" technique. There are justifications for this point of view, and they reside in the nature of most simulation results and in the nature of the types of problems to which simulation is typically applied. In regard to the former, any results which are based on uncertainties must be interpreted as imprecise and, therefore, estimates; consequently, care must be exercised in forming conclusions based on these results. In the latter case, the very conditions which dictate the use of simulation as a problem solving technique also plague the decision maker in the design and analysis of the model used in the simulation exercise. Stated in a different fashion, the decision maker cannot avoid uncertainties, complexities, and dynamic interactions simply by adopting a simulation approach: these difficulties remain to plague him in his simulation efforts as well as in his efforts toward analytical solutions.

Simulation has been defined as a representation of reality through the use of a model or abstracting device which reacts in the same fashion as reality given a particular set of conditions.(2) Consequently, most simulations begin with the formulation of some type of model. This step is essentially the same in any decision making or operations research model and should be performed with a particular objective in mind in order to avoid excessive detail and elaboration. The second step in building a simulation is to design the experimental procedures, the measures of effectiveness, and the statistical tests that are to be employed. The third step is to develop the computer program that will be utilized, as simulation is typically performed internally by an electronic computer. Consequently, an appropriate language must be selected with which to communicate with the computer and direct it through the simulation experiments.

AVAILABLE COMPUTER LANGUAGES

A large number of computer languages exist, a detailed discussion of which is much too lengthy to undertake here. These languages range from widely used, highly popular languages to highly specialized, little used languages. For the purposes of this paper, however, computer languages may be broadly classified as simulation languages and non-simulation languages.

Simulation Languages

The most widely used simulation languages include SIMSCRIPT, GPSS, DYNAMO and GASP. These languages, like other simulation languages, are generally quite efficient for programming simulation studies. By design, simulation languages are intended to make simulation programming faster and less difficult than with non-simulation languages. If the model involved is simple and is a common application of simulation, then a program may already be available to which input information may be supplied. Moreover, some of the simulation languages require only the specification of the probability distribution functions, from which random events are automatically generated. Other languages collect statistics on operating characteristics of interest to the user and report the results in predesigned output form. Simulation languages also generally keep a time record in the model.

Non-Simulation Languages

Probably the most widely known non-simulation languages are FORTRAN, COBOL, PL/1 and ALGOL. These languages are of different design than simulation languages in that their basic purpose is to provide a general approach to computer programming of all types - not just the programming of simulation studies. Consequently, few if any sub-routines of any kind are built into these languages. The programmer must create every programming activity the user desires employing a very basic and rudimentary statement approach. For example, if a simulation requires the generation of random variables, the programmer must write a separate program component for each such variable; if statistics are to be collected, a program component must be written to accomplish this requirement; a program component must be written just to sequence the simulated events and to record time in the model.

Because of the generalized nature of non-simulation languages, they appear to be feasible but unattractive in programming simulation studies. FORTRAN and PL/1, for example, have been utilized in programming only moderately complex, infrequently used simulations; COBOL, by contrast, has received little attention or utilization in simulation work, probably because of its business data processing image. Despite its history of non-simulation applications, the COBOL language may be one of the most potentially attractive languages for use in simulation studies.

GENERAL ATTRACTIONS OF COBOL

As a general language, the Common Business Oriented Language has achieved widespread usage, particularly among business users for which it was designed. The attractiveness of the language in its general application forms the basis for its potential attractiveness as a simulation language.

Popularity

COBOL has become one of the most popular of all general languages, primarily as a result of its ease of application to business related computations. From an educational vantage, it is easily grasped by students, perhaps to a greater extent than FORTRAN. Its largest single user is the federal government in its many and varied activities. COBOL remains roday the only official programming language of the federal government, being utilized by eight out of ten federal agencies. Its popularity seems to be increasing as the language continues to develop and mature. A recently announced Federal Information Processing Standard made COBOL-74 the new standard programming language superceding COBOL-68, the version which had been standard since 1972.

Documentation

One of the most attractive characteristics associated with the COBOL language is its self-documenting aspect. Unlike the FORTRAN language in which self-documentation is virtually non-existent, COBOL is almost perfectly self-documenting. This feature derives from the fact that the language itself is essentially conversational English which utilizes only the most common wording. This results in programming statements which are seldom ambiguous and which are highly self-explanatory. Consequently, little supplemental documentation is required for programs written in COBOL. This self-documenting feature should be an extremely attractive element in applying COBOL to the programming of simulation studies, with their accompanying difficulty, complexity and scope. In such applications documentation is served at the expense of programming efficiency, a consideration which is discussed in a later section of this paper.

Believeability

An attractive feature of COBOL which emanates directly from the self-documenting feature is the concept of believeability. As stated earlier, COBOL is in widespread use by business managers whose programming talents (if they exist) tend to be in the COBOL language applications. Generally these managers do not possess the scientific background more closely associated with FORTRAN. Consequently when management people read COBOL programs they generally have some sense of reality and tend to be more confident that their programs will actually work. The sense of reality and accompanying confidence exist to a much lesser extent with programs written in FORTRAN. Even though COBOL may be less efficient from a programming point of view, its believeability causes it to be a highly regarded language among the business management practitioners; and at least on the current generation of computers, the continued development of the language has reduced the inefficiency gap between COBOL and other simulation and non-simulation languages.

The Triangular Distribution

One of the most attractive (but very indirect) features of COBOL as a simulation language is its efficiency in programming the triangular distribution. The triangular distribution is receiving widespread attention from business practitioners, researchers, and academicians largely because of its capability to approximate other statistical distributions, and because of its ease of parameter estimation.(3) Its complete density function is triangularly shaped and is determined by supplying values for its three parameters. The three parameters are the optimistic (O), pessimistic (P), and most likely (M) values of the random variable. The triangular distribution provides a simple and understandable means of portraying a manager's feelings of uncertainty regarding the variable he is estimating. This is particularly valid when the manager has a non-technical background, or when the manager is attempting to plan for the future. Managers seem to be able to identify with the optimistic, pessimistic, and most likely concepts to a much greater degree than with the parameters of other distributions. Consequently, they can readily provide these estimates based on experience, intuition, etc., without having to resort to extensive data collection and analysis.

Since simulations invariably involve uncertain factors, the factors in these simulations are treated as random variables; however, few if any, situations exist in which the actual probability distributions of these variables are known. Actual data must be approximated by some theoretical distribution. Despite the availability of statistical tests of goodness of fit, the decision maker in the end must still decide the goodness of fit issue on the basis of acceptable risks of being "right" or "wrong" in his judgement regarding the description of his actual data by a theoretical statistical distribution. The practical translation of the above is that the manager may not know or care about the particular theoretical distribution which best describes his actual data; he may not be able to provide reasonable estimates for the parameters μ and σ of the normal distribution; but he can always provide estimates (subjective, perhaps) of the optimistic, pessimistic, and most likely values for a particular business phenomenon. Because of this feature, the triangular distribution may provide the most direct, efficient, practical, and flexible vehicle for estimating the behavior of random variables.

The discussion above may seem to have little to do with COBOL as a simulation language. However, the opposite is the case. The managerial popularity of the distribution combined with the fact that it can be very easily programmed with COBOL provide a strong incentive for simlation application, an incentive which dissipates if other distributions are utilized. The triangular distribution is unique in that it is flexible in shape, has easily estimated parameters, can approximate continuous or discrete data, and can be COBOL programmed without having to write extensive subroutines. The key element here is not having to resort to writing subroutines. If distributions other than the triangular are used, this advantage is almost always lost. Furthermore, COBOL becomes very unattractive from an efficiency standpoint when other distributional forms are utilized in the simulation model, as other general purpose languages such as FORTRAN are more efficient for this purpose. The attractiveness of COBOL as a simulation language is thus conditional on the use of the triangular distribution to represent the distributions of the random variables contained in the simulation model.

CONDITIONS, CRITERIA, AND CAVEATS

With the general attractiveness of COBOL established, some attention should be given to a determination of those conditions under which COBOL is a likely candidate for use in simulation. Some attention should also be focused on the criteria to be used in determining the feasibility of simulation with COBOL, as well as those conditions under which its use should be discouraged.

The following general guidelines summarize the situations in which COBOL can be most favorably used for simulation studies programming.

-If the expertise of the programmer is limited, COBOL may be useful. This derives from the fact that simulation languages are special purpose languages and hence require special knowledge.

-If the computing equipment is limited in scope and capacity, then COBOL may be more appropriate. Not all simulation languages are available on all computers. With widening availability of service bureaus, this consideration is becoming less important.

-If the user is management oriented, the use of COBOL may be more appropriate. This derives from the manager's familiarity with COBOL, the managerial use of the triangular distribution, the self-documentation aspect, and the believeability of the programming statements.

-If programming effort efficiency is of primary importance, then COBOL is an attractive simulation language. By contrast, if running efficiency is of greatest concern, then a simulation language or machine based language is more appropriate. These statements are derived from the fact that machine based languages (and to some extent simulation languages) are very demanding on the programmer. Learning a simulation or machine based language well enough to use without gross inefficiencies and errors requires considerable time and effort. As a result prospective users must weigh programming effort efficiency against running efficiency and choose one at the expense of the other.

-If the simulation study is relatively simple or if simulation programming will be done infrequently, then COBOL is an attractive language candidate. By contrast, if the simulation is complex and is to be frequently run, a machine based language or a simulation language seems more appropriate.

AN EXAMPLE

A very simple cost-volume-profit example (income model) is useful in demonstrating some of the points discussed above. The model is typically expressed as

$$I = V(P-C) - F$$

in which

I = income
V = number of units sold
P = unit price
C = unit cost
F = fixed costs

In a non-simulation application of the model, single number estimates are supplied and a deterministic value is calculated for I. In a simulation application of the model, uncertainties are recognized in supplying estimates for each factor in the model. Applying the triangular distribution to each factor requires optimistic, pessimistic, and most likely parameter estimates for each factor, as hypothesized in Table 1.

Table 1

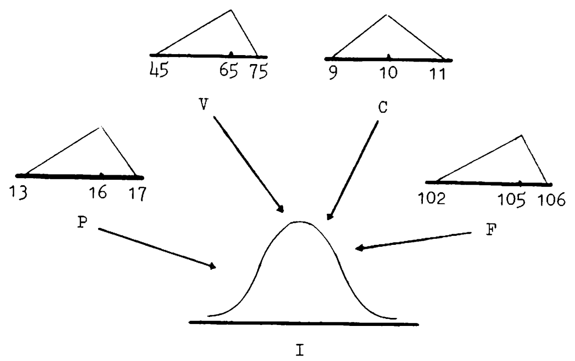
Parameter Estimates for Income Model Factors

	P	M	O
Volume	45	65	75
Unit Price	13	16	17
Unit Cost	9	10	11
Fixed Cost	102	105	106

The entire simulation model can be represented pictorially as shown in Figure 1.

Figure 1

Pictorial Representation of Income Model Simulation



Simulation Output

The actual simulation of the income model is simple conceptually and from a programming point of view. Figure 2 illustrates the key elements in programming the simulation model above using the COBOL language. The more elementary elements of the program are deleted and left for the reader to develop, along with the simulation output. The reader will further recognize the self-documenting, believeability, and simplistic features contained in the annotated COBOL simulation program in Figure 2.

SUMMARY

The experienced simulation study programmer should immediately recognize that the ideas expressed here are not intended to represent COBOL as a simulation language competitor under all programming conditions. To the contrary, the conditions under which COBOL is an attractive language alternative are seldomly experienced by a simulation programming specialist. Nevertheless, when these conditions do manifest themselves, COBOL should be considered an additional viable language alternative which may result in more efficient programming of the simulation model being studied.

Figure 2

Annotated COBOL Program for Income Model

```
WORKING-STORAGE SECTION.
77  SEED-NUMBER PICTURE 9(5) VALUE 97457.
77  RAND-NUMBER PICTURE 9(8).
77  I           PICTURE 9(5).
77  J           PICTURE 9(5).
77  RATIO       PICTURE 9(5).

Ø1  PARAMETER-TABLE.
    Ø5  PARAMETER-CARD OCCURS 4 TIMES.
        1Ø  PESSIMISTIC PICTURE 999V99.
        1Ø  MOST-LIKELY PICTURE 999V99.
        1Ø  OPTIMISTIC  PICTURE 999V99.
Ø1  INCOME-TABLE.
    Ø5  INCOME OCCURS 1ØØØ TIMES PICTURE 9(6)V99.
Ø1  TRIANGULAR-TABLE.
    Ø5  TRIANGULAR-NUMBERS OCCURS 4 TIMES
        PICTURE 9(3)V99.

PROCEDURE DIVISION.

1ØØ-BEGIN.
    PERFORM 2ØØ-READ
        VARYING I FROM 1 BY 1 UNTIL I > 4.
2ØØ-READ.
    READ PARAMETER-FILE INTO PARAMETER-CARD (I)
        AT END GO TO 3ØØ-INCOME-TABLE.
3ØØ-INCOME-TABLE.
    PERFORM 4ØØ-INCOME-MODEL
        VARYING I FROM 1 BY 1 UNTIL 1 > 1ØØØ.
    PERFORM 7ØØ-DATA-DESCRIPTION.


4ØØ-INCOME-MODEL.
    PERFORM 5ØØ-TRIANGULAR-GENERATOR
        VARYING J FROM 1 BY 1 UNTIL J > 4.
    COMPUTE INCOME (J) =

    TRIANGULAR-NUMBER (1) *
    (TRIANGULAR-NUMBER (2) -
        TRIANGULAR-NUMBER (3))
    - TRIANGULAR-NUMBER (4).

5ØØ-TRIANGULAR-GENERATOR.
    PERFORM 6ØØ-UNIFORM-GENERATOR.
    COMPUTE RATIO =

    (MOST-LIKELY (J) - PESSIMISTIC (J)) /
    (OPTIMISTIC (J) - PESSIMISTIC (J)).

    IF      SEED-NUMBER > RATIO

        COMPUTE TRIANGULAR-NUMBER (J) =
        OPTIMISTIC (J) -
        ((OPTIMISTIC (J) - MOST-LIKELY (J)) *
        (OPTIMISTIC (J) - PESSIMISTIC (J)) *
        (1 - SEED-NUMBER)) ** Ø.5.

    ELSE

        COMPUTE TRIANGULAR-NUMBER (J) =
        PESSIMISTIC (J) -
        ((MOST-LIKELY (J) - PESSIMISTIC (J)) *
        (OPTIMISTIC (J) - PESSIMISTIC (J)) *
        SEED-NUMBER) ** Ø.5.

6ØØ-UNIFORM-GENERATOR.
   COMPUTE RAND-NUMBER = SEED-NUMBER * 291.
   COMPUTE SEED-NUMBER = RAND-NUMBER / 1ØØØØØ.
   COMPUTE SEED-NUMBER =
       (RAND-NUMBER * 1ØØØØØ).

7ØØ-DATA-DESCRIPTION.
```

REFERENCES

1. Wagner, Harvey M., Principles of Operations Research, Prentice-Hall, 1969, p. 889.

2. Thiereuf, Robert J. and Grosse, Richard A., Decision Making Through Operations Research, Wiley and Sons, New York, 1970, p. 471.

3. Brewerton, F. J., and Gober, R. Wayne, "Management Science Applications of the Triangular Distribution: Some Pros and Cons," Proceedings Midwest AIDS Conference, Indianapolis, Indiana, April, 1975, pp. 428-431.