AN APL--BASED SIMULATION PACKAGE

Claude Boucher

Universite de Sherbrooke

ABSTRACT

The purpose of this paper is to introduce a package of 52 programs that can be used for the simulation of discrete event models. Those programs are written in the APL language which allows us to write and debug simulation models by taking advantage of a system that works in a conversational mode.

1. INTRODUCTION

The best known simulation languages all use the batch processing mode, although there has been a trend in recent years to introduce conversational versions of some of them. The inconveniences of that mode are well known. The slightest error or the smallest correction necessitates waiting for the next program run, which can occasion relatively important delays in a congested computer center. This means that the conception, writing and debugging of even moderately complex simulation models requires much time.

With conversational mode, however, errors can be detected and corrections made immediately. The model construction and debugging are shortened considerably.

The package of 52 functions presented here is mainly designed for the simulation of stochastic and discrete event models, although it can also be used in conjunction with deterministic models. It is written in the syntax of an enlarged version of APL which includes an execute operator.

II. STRUCTURE OF THE PACKAGE

In this section, we succinctly describe the roles accomplished by the most important functions of the package.

APLS plays a monitoring role in the simulation. It controls the simulation from the beginning to the end. It reads non-standard data, assumes the management of workspaces and calls at appropriate times the necessary functions.

BSIM prepares the simulation, with the aid of EDIT builds EVNT, the event functions selector, erases from the active workspace the functions which will not be used during the simulation, and defines groups of functions in such a manner that the active workspace is used in the most efficient way possible.

DATA converses with the user, receiving data and initializing certain variables and files and NWRN gives the possibility during ulterior runs to use DATA in the most economic way by considering only those variables that must be re-initialized for the run.

SMRY outputs the results of a simulation run as well as the content of the files at the end of the run. Under the name of OUTP, the author has the option to write a function which prints any other information that is not included in the regular task of SMRY.

Certain functions execute various auxiliary roles. For example, DBUG outputs required information and is useful as its name indicates, for debugging purposes. DUMP prints the content of the files. INTR is used for interpolation and extrapolation.

Other functions are related to the use of the files. FIND scans files in search of information with given characteristics. FILE enters the information into a file, taking into account the order in which this file must be classified. RMOV retrieves given data from a file. Those two functions will call UPDT which updates the file statistics.

The statistics collection for selected variables has been committed to three functions. CLCT is concerned with ordinary statistics, while TMST is concerned with time-integrated statistics and FREQ counts the occurrences in the classes of frequency distributions.

The problem of how are generated random variates diversely distributed has received much attention. Everybody knows the importance the generation of these variates has in the simulation of discrete event models. Not less than 29 functions are directly dedicated to this issue.

The author wishes to acknowledge the receipt of grants from the National Research Council of Canada and the Quebec Ministry of Education. The research has been conducted under the auspices of Le Centre de recherches en aménagement régional de l'Université de Sherbrooke.

We have considered three approaches to realize this generation of random variates. One makes use of algorithms to generate the variates. Despite its frequent use, this method has the disadvantage of consuming more time than the two others. Occasionally it happens, however, especially in the case of continuous distributions, that it is the only available method. But, for discrete distributions, we have other techniques at our disposal to generate those variates. One makes use of a table in which values are selected with frequencies proportional to the probabilities assigned to those values. This process is generally faster than the previous one. But the third process is even more rapid. It consists of storing values in a vector in proportion to the probabilities assigned to these values. To generate a random variate, it is then sufficient to select an entry in the vector at random.

The basic function in this group is RAND which generates a vector of random variates uniformly distributed between 0 and 1. UNFR and UNIN both generate uniformly distributed random variates; but for UNFR the distribution of the values is continuous, while for UNIN it generates integer values within an interval with integral bounds. DRAW generates randomly one of the values 1 and 0 with a probability proportional to the argument of the function.

In the group of the functions dedicated to the direct generation of a random variate distributed according to a given statistical law, we have BINO, ERLG, EXPO, FDIS, KISQ, LOGN, NORM, POIS, TDIS and WEIB which concern respectively binomial, Erlang, exponential, F-, Chi square, lognormal, normal, Poisson, t- and Weibull distributions.

BIBI, HGHG and POPO generate a vector whose entries are the probabilities attached to binomial, hypergeometric and Poisson distributions respectively. In conjunction with TBBI, TBHG and TBPO respectively they will be used to build tables for the generation of random variates extracted from a discrete distribution. The function ESTP will then allow us to choose a value in the table at random.

We can also use this process to generate random variates in relation to experimental data which do not follow a given statistical law. The table of values will then be built with the help of the function TBED. If these experimental data, instead of being discretely distributed, belong to a continuous distribution, the generation process will be obtained by the function ELIN in lieu of ESTP.

BIBI, HGHG and POPO are also employed in the construction of a vector of values whose occurrences are proportional to the probabilities they have been assigned. The construction of the vector will also depend on the functions GVBI, GVHG and GVPO respectively, each calling the function GENE to complete its tasks. In addition, we can also determine such a vector for experimental data that are discretely distributed. If so, the function GVED is needed. The function MARS is used for choosing a value at random in such vectors.

III. CONCLUSION

As we have already mentionned, with such a package building and debugging a simulation model can be done very rapidly and efficiently. But, there are some drawbacks. Since APL is an interpreted language, the exploitation of the built model on a massive scale can consume an important amount of CPU time. We are presently writing a parallel version of the package in FORTRAN. The model building phase can be done by using the APL version. When the model construction has been completed, the model is translated into a FORTRAN program and the exploitation is done in conjunction with the parallel version.