Proceedings of the 1986 Winter Simulation Conference
J. Wilson, J. Henriksen, S. Roberts (eds.)

Simulation and Analysis with IMSL Routines

James E. Gentle
IMSL, Inc.
2500 CityWest Boulevard
Houston, Texas 77042-3020

ABSTRACT

The IMSL Library is a unified collection of over 500 FORTRAN subroutines for applications in statistics, operations research, and other areas of applied mathematics. Routines are available for generating random numbers from eighteen univariate distributions, from three multivariate distributions, and from two time series models. In addition, several routines are provided for parameter estimation and for goodness-of-fit tests. This tutorial discusses the use of the IMSL Library in simulation and statistical analysis.

1. INTRODUCTION

While the general trend in simulation software has been toward greater degrees of 'packaging', resulting in high-level fourth-generation command languages, the researcher often needs specialized computing that cannot be met by any of the standard program packages. A subroutine library accessed from FORTRAN fills such needs and affords great flexibility. The user can customize a program by selecting those subroutine modules that perform the required computations. The results of the computations are available to the program itself, whereas most simulation packages merely print the results. Moreover, a subroutine library is immediately extensible; the library developer or the user can easily add new routines.

The main difference between a simulation software package and a FORTRAN subroutine library is the tradeoff between ease of use and flexibility. The package is probably easier to use, because it eliminates most of the bookkeeping and programming considerations not explicitly part of the problem of interest. This is an important convenience, since bookkeeping and the accumulation of simple statistics are major computational activities in many simulation problems. A packaged program for simulation, however, may not be able to handle certain nonstandard models; therefore the user may need the flexibility that a subroutine library affords. Sometimes it is possible to combine a simulation package with a FORTRAN library. This is particularly easy with a package such as GPSS-FORTRAN.

In choosing between a simulation language and a FORTRAN subroutine library, two minor additional considerations are the speed of execution and the availability of the system on different computers. It is likely that the FORTRAN library will execute more rapidly on a compute-bound problem because the program will be more tailored to the specific problem. Also, a FORTRAN library is much easier to port to a new computing environment than is a simulation package, so it is likely that the FORTRAN library is available on more machines.

The first edition of the IMSL Library, consisting of just over 200 subroutines, appeared in 1971. The current, ninth edition, with over 500 subroutines, retains the same general structure and conforms to similar design principles. The Library is available on over 30 different computer environments. The subroutines are organized into seventeen functionally related areas, which correspond to chapters in the user documentation. These areas include differential equations, eigenanalysis, and optimization. For the simulationist, the main areas of interest would be random number generation and statistical analysis.

2. UNIFORM RANDOM NUMBER GENERATORS

The random number generators in the IMSL Library use a multiplicative congruential method. The form of the generator is

$$x_i \equiv cx_{i-1} \bmod (2^{31} - 1).$$

Each x_i is then scaled into the unit interval (0,1). If the multiplier, c, is a primitive root modulo $2^{31} - 1$ (which is a prime), then the generator will have maximal period of $2^{31} - 2$. There are several other considerations, however. The lattice structure induced by congruential generators (see Marsaglia, 1968) can be assessed by the lattice test of Marsaglia (1972) or the spectral test of Coveyou and MacPherson (1969) (see also Knuth, 1981, pages 89-113). Also, empirical studies (e. g., Fishman and Moore 1982, 1986) indicate that different values of multipliers, all of which perform well under the lattice test and the spectral test, may yield quite different performances, where the criterion is similarity of samples generated to samples from a true uniform distribution.

The possible values for c in the IMSL generators are 16807 and 397204094. The choice of 16807 will result in the fastest execution time, but Fishman and Moore's studies would seem to indicate that the performance of 397204094 is better. The multiplier 16807 has been in use for some time (Lewis, Goodman, and Miller, 1969).

The generation of uniform (0,1) numbers is done by the routine `GGUBS`, `GGUBT`, or `GGUW`, or by the function analog of `GGUBS`, `GGUBSF`. These routines are *portable* in the sense that, given the same seed, they produce the same sequence in all computer/compiler environments. (See Gentle 1981 for further discussion of this issue.)

Shuffled Generator

The subroutine `GGUW` is a shuffled generator using a scheme due to Learmonth and Lewis (1973a). In this scheme, a table is filled with the first 128 uniform (0,1) numbers resulting from the simple multiplicative congruential generator. Then, for each x_i from the simple generator, the low-order bits of x_i are

used to select a random integer, j, from 1 to 128. The j-th entry in the table is then delivered as the random number and x_i, after being scaled into the unit interval, is inserted into the j-th position in the table. This scheme is similar to that of Bays and Durham (1976), and their analysis would be applicable to this scheme as well.

Customized Generators

All of the IMSL random number generators for nonuniform distributions use the basic uniform generator GGUBS. If for some reason a uniform random number generator other than GGUBS were preferred, the user can write a special subroutine called 'GGUBS' with the same calling sequence as the IMSL subroutine by that name, but which returns random numbers generated by the user's own algorithm. Consequently, when the user calls an IMSL subroutine to generate nonuniform random numbers, that subroutine will call the user's 'GGUBS'.

3. NONUNIFORM GENERATORS

The IMSL Library contains routines for generating random numbers from eighteen univariate distributions, from three multivariate distributions, and from two time series models. The nonuniform generators in the IMSL Library use a variety of transformation procedures. The most straightforward transformation is the *inverse CDF* technique, but it is often less efficient than others involving *acceptance/rejection* and *mixtures*. See Kennedy and Gentle (1980) for discussion of these and other techniques.

Many of the IMSL nonuniform generators use different algorithms depending on the values of the parameters of the distributions. This is particularly true of the generators for discrete distributions. Schmeiser (1983) gives an overview of techniques for generating deviates from discrete distributions.

Although, as noted above, the uniform generators yield the same sequences on different computers, because of rounding, the nonuniform generators that use acceptance/rejection may occasionally produce different sequences on different computer/compiler environments.

For most distributions there are choices of algorithms for generation of random numbers. The algorithms differ in speed, in accuracy, in storage requirements, and in complexity of coding. Some of the faster methods are approximate. The IMSL generators all use exact methods. Given the current cost of computing, the speed-up resulting from an approximation is not worth the accuracy loss. After accuracy, the next most important criterion is speed. There are two components to the speed of a random number generation algorithm: the set-up time and the generation time. In most cases the generation time is the more important component to optimize. Whenever the set-up time is significant, the IMSL subroutine preserves the variables initialized, so that if the subroutine is called again with the same parameters, the set-up step can be bypassed. In an extreme case of relatively expensive overhead, a second subroutine with less set-up time is provided. The other two criteria mentioned above, storage requirements and complexity of coding, are generally of no concern in selecting algorithms for IMSL random number generators.

Generators for Univariate Distributions

beta	GGBTR
binomial	GGBN
negative binomial	GGBNR
Cauchy	GGCAY
chi-squared	GGCHS
discrete uniform	GGUD
exponential	GGEXN
exponential mixture	GGEXT
gamma	GGAMR
geometric	GGEOT
hypergeometric	GGHPR
log-normal	GGNLG
normal	GGNML
Poisson	GGPOS
stable	GGSTA
triangular	GGTRA
von Mises	GGVMS
Weibull	GGWIB

Generators for Multivariate Distributions

multinomial	GGMTN
multivariate normal	GGNSM
uniform on sphere	GGSPH

Generators for Time Series

ARMA	FTGEN
nonhomogeneous Poisson process	GGNPP

Order Statistics and Antithetic Variates

Order statistics from a uniform distribution can be generated directly, using the known distribution of any specific order statistic or the known distribution of the spacing between specific order statistics. The IMSL subroutine GGUO generates any specified set of random order statistics from a uniform distribution. For example, the user can specify the largest five order statistics from a random sample of size 100, and GGUO generates these directly, rather than generating the complete sample and picking out the five largest. The subroutine GGNO perfoms a similar task for the normal distribution.

For those generators, such as GGCHY and GGNML, that use the inverse CDF technique, it is possible to generate any set of order statistics directly. This is done with a customized uniform generator, as discussed above, by generating order statistics in a custom 'GGUBS' or 'GGUBSF'. Some routines such as GGEXN and GGWIB that employ an inverse CDF technique use the uniform (0,1) deviate $1-u$, rather than directly using the uniform (0,1) deviate u from GGUBS. In such routines the i-th order statistic from the uniform will yield the $(n+1-i)$-th order statistic from the nonuniform distribution.

A similar technique can be used to get antithetic variates. For each uniform deviate u, a second deviate $1-u$ could be produced by a custom 'GGUBS' or 'GGUBSF'. As with order statistics, this technique would only be reasonable for routines that use the inverse CDF technique.

Random Samples, Permutations, and Tables

IMSL routines are provided for generating random samples without replacement from a finite population, for generating random permutations of sets, for generating frequency tables with fixed marginal totals, and for generating random correlation matrices. The routine for generating random samples can be used simply to get index numbers or actually to select the sample items from a data set.

Data-Based Random Number Generation

A common problem in simulation is to generate additional random samples from the distribution from which a given sample has been taken. Unless the user is willing to make an assumption about the form of the distribution, nonparametric techniques must be used. IMSL routines are available for sorting the data or for computing the empirical distribution function directly. With a distribution function or an empirical distribution function, the user can call a general continuous random number generator that first approximates the distribution function with quasi-cubic splines, and then generates random numbers from the approximation.

4. TESTING RANDOM NUMBER GENERATORS

Extensive empirical tests of some of the uniform random number generators available in GGUBS and GGUBSF are reported by Fishman and Moore (1982 and 1986). Results of tests on the generator using the multiplier 16807, with and without shuffling, are reported by Learmonth and Lewis (1973b). Several IMSL routines are provided for the user who wishes to perform additional tests. Often in Monte Carlo applications it is appropriate to construct an ad hoc test that is sensitive to departures that are important in the given application. For example, in using Monte Carlo methods to evaluate a one-dimensional integral, autocorrelations of order one may not be harmful, but they may be disastrous in evaluating a two-dimensional integral. The routines for generating random deviates from nonuniform distributions use exact methods; hence, their quality depends almost solely on the quality of the underlying uniform generator. Nevertheless, it is often advisable to employ an ad hoc goodness-of-fit test for the transformations that are to be applied.

The tests available in the IMSL Library include chi-squared goodness of fit, Kolmogorov-Smirov, runs, pairs-serial, triplets, d^2, and poker tests. In addition, several nonparametric tests are provided that may be adopted to test specific hypotheses.

5. STATISTICAL ANALYSIS OF OUTPUT

The IMSL Library provides many subroutines for statistical analysis. The routines in the basic statistics chapter compute frequency tables, sample moments, confidence intervals, and statistics for several hypothesis tests. There are routines for regression analysis, for time series analysis in both the time domain and the spectral domain, for nonparametric statistical procedures, and for goodness-of-fit tests. The IMSL graphics routines, which are only for line printers, provide scattergrams, histograms, stem-and-leaf plots, boxplots, probability plots, and plots to compare sample cumulative distribution functions with theoretical cumulative distribution functions.

6. ROLE OF MICROCOMPUTERS IN SIMULATION

Use of simulation and Monte Carlo methods will be even more common in the future as computing becomes less expensive. The researcher who 'turns on' a personal computer rather than 'logs on' to a mainframe is not intimidated with the thought of letting the computer run all weekend. Simulation will also play an important role in the solution of very large problems on supercomputers since it can often take advantage of parallel processing capabilities. Work on a problem involving simulation and Monte Carlo methods may progress through preliminary computations on a microcomputer to more extensive computations on a mainframe, perhaps to produce multi-way tables over several ranges of values of various parameters.

Microcomputers have brought an increased importance to portability of software. Portability is important for transfer of research and development efforts. Portability reduces the number of times the wheel is reinvented as well as the amount of the computer-knowledge overhead that burdens a researcher. The user may devote attention to the research problem rather than to the extraneous details of the computer tools used to address the problem.

Formerly, portability was a concern primarily for distributors of software, for users who may be switching jobs, or for computer installations changing or contemplating changing their hardware. With the widespread availability of personal computers, all computer users now are much more likely to use (or to attempt to use) the same program on more than one machine. There are both technical and tactical reasons for using a micro and a mainframe while working on the same program. The technical reasons include the differences in resources (memory, CPU speed, software) available on micros and mainframes. These differences likely will narrow as new and better micros are introduced and more software is developed for them. The availability of the different computers in different working environments such as home, lab, and office creates another advantage to using multiple computers on a single problem. These tactical reasons will persist, and it will become increasingly commonplace for a researcher to use more than one computer.

IMSL's MATH/LIBRARY and STAT/LIBRARY, available on the IBM PC, make it possible for the user to begin work on a personal computer and then, if necessary, later to move the work to the mainframe. The sequence of integers produced in the underlying multiplicative congruential generator is always exactly the same on the personal computer as on the mainframe so long as the same seed is used; hence, if work is done on both a personal computer and a mainframe, the studies can be combined because the next point in the pseudorandom sequence can be used as the starting point for the continuation. Furthermore, repetition of a small portion of the study will allow the user to determine that the other parts of the program are performing the same on both computers.

REFERENCES

Bays, Carter, and S. D. Durham (1976), Improving a poor random number generator, *ACM Transactions on Mathematical Software*, **2**, 59–64.

Coveyou, R. R., and R. D. MacPherson (1967), Fourier analysis of uniform random number generators, *Journal of the ACM*, **14**, 100–119.

Fishman, George F., and Louis R. Moore, III (1982), A statistical evaluation of multiplicative random number generators with modulus $2^{31} - 1$, *Journal on the American Statistical Association*, **77**, 129–136.

Fishman, George F., and Louis R. Moore, III (1986), An exhaustive analysis of multiplicative congruential random number generators with modulus $2^{31} - 1$, *SIAM Journal on Scientific and Statistical Computing*, **7**, 24–45.

Gentle, James E. (1981), Portability considerations for random number generators, in *Computer Science and Statistics: Proceedings of the 13th Symposium on the Interface*, (edited by William F. Eddy), Springer-Verlag, New York, 158–164.

Kennedy, William J., and James E. Gentle (1980), *Statistical Computing*, Marcel Dekker, Inc., New York.

Knuth, Donald E., (1981), *The Art of Computer Programming, Volume 2 / Seminumerical Algorithms*, second edition, Addison-Wesley, Reading, Massachesetts.

Learmonth, G. P., and P. A. W. Lewis (1973a), *Naval Postgraduate School Random Number Generator Package LLRANDOM, NPS55LW73061A*, Naval Postgraduate School, Monterey, California.

Learmonth, G. P., and P. A. W. Lewis (1973b), Statistical tests of some widely used and recently proposed uniform random number generators, in *Computer Science and Statistics: 7th Annual Symposium on the Interface*, (edited by William J. Kennedy), Statistical Laboratory, Iowa State University, Ames, Iowa, 163–171.

Lewis, P. A. W., A. S. Goodman, and J. M. Miller (1969), A pseudo-random number generator for the System/360, *IBM Systems Journal*, **8**, 136–146.

Marsaglia, G., (1968), Random numbers fall mainly in the planes, *Proceedings of the National Academy of Sciences*, **61**, 25–28.

Marsaglia, G., (1972), The structure of linear congruential sequences, in *Applications of Number Theory to Numerical Analysis*, (edited by S. K. Zaremba), Academic Press, New York, 249–286.

Schmeiser, Bruce, (1983), Recent advances in generation of observations from discrete random variates, in *Computer Science and Statistics: The Interface*, (edited by James E. Gentle), North-Holland Publishing Company, Amsterdam, 154–160.

AUTHOR'S BIOGRAPHY

JAMES E. GENTLE is a software designer with IMSL, Inc. He received a master's in computer science and a Ph.D. in statistics from Texas A&M Universtiy in 1973 and 1974 respectively. Prior to joining IMSL in 1979, he was Associate Professor of Statistics at Iowa State University. His current research interests include random number generation and robust techniques in statistics. He is a Fellow of the American Statistical Association and a member of the Board of Directors of AFIPS.