Discussion Paper No. 135

THE EFFECT OF DIFFERENT GPSS RANDOM NUMBER GENERATORS ON SIMULATION RESULTS

by

Jair M. Babad[*] and Edward A. Stohr[**]

March 1975

[*] Graduate School of Business, University of Chicago

[**] Graduate School of Management, Northwestern University

THE EFFECT OF DIFFERENT GPSS RANDOM NUMBER GENERATORS ON SIMULATION RESULTS

by

Jair M. Babad and Edward A. Stohr

1. INTRODUCTION

Many simulation programming languages have been developed to facilitate the translation of simulation models into a simulation program. Of these GPSS (General Purpose Simulation System) is the most common. An important aspect of a simulation language is that it provides the user with a source of random numbers - a "random number generator." In this paper we describe an experiment carried out using the random number generators from IBM's GPSS (GPSS/360 and GPSS V) and Control Data's GPSS V. The Control Data Corporation (CDC) random number generator is of the multiplicative congruential type which has generally been found to have satisfactory statistical properties (see e.g. Fishman [3] and Knuth [4]). However the IBM generator is a multiplicative "scrambling" generator which produces up to eight concurrent sequences of random numbers. The purpose of this research was to examine these generators from the user's point of view by determining the sensitivity of a simulation model to the choice of random number generator, random number seed, and "programming strategy" as defined below. The research follows the approach used in Babad, [1].

The IBM random number generator is described and analyzed in Section 2 and compared with the two CDC random number generators. The queuing models which were used in our tests, as well as the corresponding theoretical results, are described in Section 3. In Section 4 we compare the experimental outcomes with the theoretical expected values and in Section 5 we conclude the paper with some recommendations concerning the use of these generators.

2. THE GPSS RANDOM NUMBER GENERATORS

The eight sequences of random numbers produced by the IBM random number generator are identified as RN1 through RN8. Depending on the context in which it is used a random number may be either a decimal fraction in the range .000000 to .999999 (.abcdef) or an integer in the range 000 to 999 (def). Note that the integer is constructed from the last three digits of the decimal fraction. A fairly complete describtion of this generator is given in Felder [3], and the reader is referred to that source for details. However some points need further clarification as has previously been pointed out by Fishman [4].

Three integer arrays of eight elements each are used by the IBM generator: the INDEX table, the BASE number array, and the MULTIPLIER array. The BASE array does not change during the algorithm. It contains eight numbers: 37,584,381, 1,909,996,635, 1,964,463,183, 1,235,671,459, 1,480,745,561, 442,596,621, 340,029,185 and 2,030,226,625 (the first of these is called the "seed"). The contents of the MULTIPLIER array can be supplied by the user using the RMULT card; if this is not done the array is set to a default value (1 in GPSS/360 Version 1, and 37 in GPSS 360/Versions 2 and 3 and GPSS V). The effect of choosing a multiplier is to start the corresponding random number sequence at a different point - otherwise the sequences are identical.

When a random number from RNj is requested, the following steps are taken:

1. The jth word of the INDEX table which points to one of the eight numbers in the BASE array is selected. Since the INDEX array is initially zero, the first BASE number used will be the seed, independent of the RNj used.

2. The jth element of the MULTIPLIER array is multiplied by the BASE number that was chosen in the first step. The result is a 64 bit number (as each multiplicand consists of 32 bits, the first of which is a sign bit). For a clearer description of the following steps, we will number the bits from right to left by 1 through 64; bit 1 is then the lowest order bit of the result.
3. The low order 32 bits of the result correspond to an integer (as used on IBM SYS/360 and SYS/370 computers). If this integer is negative, it is replaced by its two's complement. Note that as a result of this step, bit 32 is always zero when this step is complete.
4. Bits 1 through 32 are stored in the jth word of the MULTIPLIER array, to be used the next time a number from RNj is needed. Note that the elements of MULTIPLIER are always positive, due to step 3.
5. Bits 49-51 of the result are stored in the jth word of the INDEX array, to be used the next time RNj is called.
6. Bits 17-48 of the result are used for the random number. If an integer random number is requested, the contents of these bits is divided by thousand, and the remainder becomes the random number; otherwise, the contents of these bits is divided by 1,000,000, and the remainder is converted to the desired decimal fraction.

Several points are of interest. First, the used of the INDEX table produces a scrambling effect which is intended to reduce the non-randomness that may otherwise enter the sequence of generated numbers. Second, in contrast to the congruential generators, the generated random number is not used directly in subsequent calls to RNj but rather a separate, although overlapping, segment of the 64 bit product is used. Finally, the effects of step 3 should be considered. When step 6 is reached, bit 32 is always zero.

This bit does not affect the generation of a random integer, since the remainder after division by one thousand is contained in bits 17-26. A random decimal fraction, on the other hand, is contained in bits 17-36, and is therefore affected by the zero in bit 32; in particular, some values are excluded from the first 3 decimal digits of the fraction.

The CDC GPSS V random number generator is of the multiplicative congruential type in which random numbers, X_1, X_2, ... are produced by successive application of the formula: $X_{n+1} = AX_n$ modulo M where the 'modulo', $M = 2^{48}$ and the 'multiplier', $A = 16777451$. The 'seeds', X_o, for each of the eight sequences R1 to R8 which can be produced by the generator are constructed from the values specified by the user on the RMULT card. This is done by a succession of arithmetic, Boolean and shifting operations which transform the user supplied number to a seed which is odd and in the range, $.5 < X_o < 1$. For an RMULT value of 1, $X_o = .5000152590$ while for an RMULT value of 37, $X_o = .5005645838$. The default value for RMULT is 37 which gives the generator the same external appearance as the IBM GPSS generator described above. The integer random numbers for the CDC generator are obtained from the higher order three digits of the floating point numbers rather than the lower order three digits as is the case with the IBM generator.

3. THE SIMULATED MODEL

The queuing models chosen for the simulation are the M/M/c Delay and the M/M/c Loss queuing systems. These systems are characterized by Poisson arrivals, Exponential service times and the availability of c servers. In both systems, when an arriving user finds a free server, he is immediately put into service. The systems differ with respect to a user who arrives when all the servers are busy: in a Loss system such a user balks, while in a Delay system he is put into a queue that is served in a First-In-First-Out order.

Let λ be the mean of the Poisson arrival stream, $1/\mu$ be the mean service time and $\rho = \lambda/c\mu$ be the traffic intensity of the system. If $\rho < 1$, the Delay system will eventually reach a steady state. Let P_n denote the steady state probability that there are n users in the system, either being served or waiting. P_n might also be interpreted as the proportion of the time, in the steady state, in which there are n users in the system. The distribution of the P_n's is well known (see, e.g. Saaty [6]). Specifically, for a Loss system we have

$$P_n = (\lambda/\mu)^n \Big/ \sum_{i=0}^{i=c} (\lambda/\mu)^i \qquad \text{for } 0 \leq n \leq c$$

while for a Delay system

$$P_n = \begin{cases} P_0 (\lambda/\mu)^n / n! & \text{for } 1 \leq n \leq c \\ P_0\, c^c \rho^n / c! = P_0\, (\lambda/\mu)^n / c^{n-c}\, c! & \text{for } c < n \end{cases}$$

and $$P_0 = 1 \Big/ \left[\sum_{i=0}^{i=c-1} (\lambda/\mu)^i / i! + (\lambda/\mu)^c / (1-\rho)\, c! \right].$$

Other characteristics of such systems, such as the expected number of users in the system, the probability distribution of the waiting time in the queue and the expected queue size for a Delay system, can easily be computed once the P_n's are known.

In the simulation experiments we used c = 10 servers and tested the systems for various traffic intensities, ranging from a light intensity of $\rho = .1$ to a moderate intensity of $\rho = .5$ (for higher intensities the execution time required to achieve steady state was too high and the storage for the queues was excessive).

As mentioned earlier, our point of departure was the user of the GPSS random number generator. Accordingly, we compared several strategies that might be used by various users. Specifically, we ran one group of tests (Group A) in which the same sequence of random numbers was used both for the interarrival times and the service times, and another group of tests (Group B) in which different sequences were used for the interarrival times and the service times. In the latter case, allowing for unsophisticated users, both sequences used the same RMULT values, and produced exactly the same sequence of random numbers. A-priori, we expect to get poor results for Group B due to the high correlation that this strategy introduces between the service and interarrival times. We also tested different strategies for the determination of the service times. In the first strategy, 'S', the service time for a user was determined together with his interarrival time; in the second strategy, 'A', the service time was determined on arrival of the user to the system, while in the third strategy, 'N', the service time was computed only when needed - i.e. when the user had to be served. A-priori, we expect to get similar results from the three strategies, assuming that the service times are based on a sequence of perfectly random numbers.

During the simulation experiments with both the IBM and CDC random number generators we tested the two default values that are used by GPSS for the MULTIPLIER table, namely the value 1 that was used by GPSS/360 Version 1, and the value 37 that is used by later versions of IBM's GPSS and by the CDC GPSS.

The tests were carried out for each random number generator (IBM and CDC), for each group of tests (Group A and Group B), for each type of system (L and D), for each value of ρ (.1, .3 and .5), for each strategy (S, A, and N), for each of the two default values for RMULT (1 and 37) and for both the integer ('I') and decimal ('D') sequences. For each of these cases we ran the simulation for five thousand arrivals and collected cumulative statistics after 1000, 2000, 3000, 4000 and 5000 arrivals. The statistics collected were for the distributions of the P_n's which were derived from the proportion of the time that each system contained a given number of users. We then computed the Chi-Square goodness-of-fit statistics which measure the deviations of the observed Pn's from their theoretical values. In addition we compared the first three moments of the observed Pn's with those of the theoretical Pn probability distributions.

In addition to the implicit tests of the generator as expressed by the queuing models we also carried out Chi-Square goodness of fit tests on the integer and decimal fraction random number sequences from the generators.

As is evident from the former discussion, many tests were conducted and many computations were required during the experiments. Due to the inherent inefficiencies of the GPSS simulator which result from its interpretive nature and from the manipulation of the Chains these experiments would have cost too much if they had been done in GPSS. Furthermore it was preferable for our experiments to generate the interarrival and service times from the exact exponential distributions rather than from a GPSS function which would have resulted in the use only of the decimal fraction random number sequence (unless a special FVARIABLE had been used with resultant increase in overhead). We therefore linked the GPSS random number generators to a FORTRAN program which simulated the queuing models and accumulated the needed sequences. The same program was used for the tests using the IBM generator (on an IBM 370 Model 168 computer) and for the tests using the CDC generator (on a CDC 6400 Computer).

4. EXPERIMENTAL RESULTS

The discussion of our results is divided into two parts. First, we present the outcomes of the tests of the random sequences and then we discuss the results of the queueing models. In each case we display the 'closeness' of the observed and theoretical distributions as measured by the Chi-Square statistics.

The tests of the random number sequences used in the experiments are summarized in Table 1. Specifically, we record the probability of achieving the observed (or higher) values of the NORMAL approximation to the attained Chi-Square statistics. NUMBER is the count of the random numbers tested. This count is cumulative so that our results reflect the behavior of the complete sequence rather than the behavior of the various subsequences. CELLS is the number of partitions of the range of the random numbers used to accumulate the frequency counts. The attained probabilities are displayed in the NORMAL columns where the suffices correspond to the user supplied value on the RMULT card; NORMAL1 corresponds to the value 1 and NORMAL37 to the value 37.

For the IBM generator the INTEGER columns refer to the sequence of random integers while the DECIMAL columns display the results for the sequence of the first three decimal digits of each fraction. It is seen that the integer results for NORMAL1 tend to become less and less reliable for higher values of NUMBER and it seems improbable that the sequence is (sufficiently) random. However this trend later reverses itself; we tested the first 100,000 numbers of this sequence at intervals of 10,000 numbers and found that the values of NORMAL1 for INTEGER fluctuated between .137 and .750 which are probable values for a random sequence. Similar probable behavior was observed for the DECIMAL of NORMAL1, which had values between .199 and .860. In other

TABLE 1

THE GPSS RANDOM SEQUENCES

NUMBER	CELLS	IBM				CDC	
		INTEGER		DECIMAL			
		NORMAL1	NORMAL37	NORMAL1	NORMAL37	NORMAL1	NORMAL37
1000	200	.236	.867	.671	.467	.674	.044
2000	334	.618	.327	.575	.608	<.001	<.001
3000	500	.628	.769	.190	.874	<.001	<.001
4000	500	.740	.747	.074	.576	<.001	<.001
5000	1000	.976	.653	.070	.991	<.001	<.001
6000	1000	.967	.691	.244	>.995	<.001	<.001

words, the value 1 for the MULTIPLIER table generates a sequence that passes (in general) the test for one-dimensional randomness (uniformity). However the user should be aware that the number of generated values might result in a (probably) nonrandom sequence, as is the case for NUMBER=5000 or NUMBER=6000. Similar observations pertain to the sequence of NORMAL37 for the IBM generator. In addition to the Chi-Square test, we also compared the first three moments of the attained random number sequences with those of the theoretical uniform distribution. Again, only minor deviations (of less than two percent) were recorded. These observations conform with the aforementioned results. We also counted the percentage of use of each of the BASE array elements, as a check for the additional randomness that is introduced by Step 5 of the GPSS algorithm; the results fit very well with the expected percentage of 12.5. Thus the element of a "second randomization," which is recommended by Knuth [5], seems to work pretty well.

For the CDC random number generator the Chi-Square statistics are the same for both the integer and decimal sequences (since the integer sequence is constructed from the high order three digits of the decimal numbers). Table 1 shows a relatively poor performance of the CDC generator when compared with IBM generator for sequences in the range 1000 to 6000 in length. The tests of the first three moments for these sequences show some deviations in excess of two percent from the theoretical values for a uniform distribution which were again inferior to the corresponding outcomes for the IBM random number generator.

The discussion concerning the IBM and CDC generators seems to indicate that the IBM generator is satisfactory while the CDC generator is suspect over the range of random numbers tested. The outcomes of the simulation experiments tend to confirm this observation. As originally guessed the experiments in Group B which used two different but identical sequences of random numbers gave very poor results. In all but six[1] of the cases tested the attained Chi-Square probabilities were either smaller than .01 or larger than .99. Since such outcomes should be rejected, this illustrates the dangers inherent in failing to specify different multiplier values for the RMULT instruction when more than one random sequence is being used.

We now discuss the outcomes of the experiments in Group A in which the same sequence of random numbers was used for both the interarrival and service times. The results for the IBM and CDC generators are summarized in Tables 2 and 3 respectively. In these tables the various cases tested are identified by a 3-character code. The first character is the type of system (L or D); The second character indicates whether an integer or decimal sequence was used (I or D), while the last character identifies the kind of strategy used for the determination of the service times (I, D or N). Thus, DIN, for example, is the model of the Delay system in which the interarrival and service times were drawn from the INTEGER sequence and service times were determined when needed, while LDS denotes the Loss model in which both interarrival and service times were scheduled at the same time using the DECIMAL sequence. When Loss systems are considered, we note that each arrival who does not balk is served immediately; as a result, for such systems the "service upon arrival"

[1]The six exceptions occurred for the CDC random number generator.

and the "service when needed" strategies are identical. We also found that for Delay systems both strategies resulted with almost identical outcomes; therefore we display in Tables 2 and 3 only the outcomes for "service when needed" as well as the outcomes for the strategy of concurrent allocation of interarrival and service times.

For 1000, 3000 and 5000 arrivals we present in Tables 2 and 3 the probabilities of achieving the attained (or higher) values of the Chi-Square statistics. These are headed PROB1 and PROB37 respectively for the RMULT values of 1 and 37. In our experiments we ran all models concurrently and were limited in the available storage; as a result some outcomes are missing as the queue "escaped too early".

Generally, outcomes with probabilities exceeding 0.99 or smaller than 0.01 should be rejected, and outcomes with probabilities exceeding 0.95 or less than 0.05 are highly suspect. In Table 2, for the IBM generator, it is clearly seen that such outcomes are much more common for an RMULT value of 1. than for 37; this is in agreement with Felder [3], who rejected the value 1. Further, the use of the DECIMAL sequence results in many rejected or highly suspect outcomes, in contrast to the INTEGER sequence. Even the acceptable outcomes of the DECIMAL sequence usually involve more extreme probabilities than their INTEGER counterparts. This outcome corresponds well with our initial guess about the poor behavior of the DECIMAL sequence. It is interesting to note that, with respect to the DECIMAL sequence, an RMULT value of 37 was worse than an RMULT value of 1.

TABLE 2

CHI-SQUARE OUTCOMES FOR QUEUEING MODELS USING THE IBM RANDOM NUMBER GENERATOR

TRAFFIC INTENSITY	MODEL	1000 OBSERVATIONS		3000 OBSERVATIONS		5000 OBSERVATIONS	
		PROB1	PROB37	PROB1	PROB37	PROB1	PROB37
0.1	DIS	.440	.202	.549	.404	.235	.884
	DDS	.809	.750	>.995	.936	.991	.919
	LIS	.440	.202	.549	.404	.235	.884
	LDS	.809	.750	>.995	.936	.991	.919
	DIN	.244	.719	.907	.931	.968	.573
	DDN	.847	.580	.784	.458	.855	.323
	LIN	.244	.719	.907	.931	.968	.573
	LDN	.847	.580	.784	.458	.855	.323
0.3	DIS	.798	.491	.218	.281	.200	.521
	DDS	.693	.758	>.995	.952	.941	<.001
	LIS	.801	.486	.223	.452	.209	.593
	LDS	.694	.754	>.995	.958	.946	.899
	DIN	.777	.571	<.001	.256	<.001	.099
	DDN	.971	.838	.975	.920	.939	<.001
	LIN	.794	.581	.661	.902	.047	.843
	LDN	.957	.841	.314	.985	.500	.907
0.5	DIS	.645	<.001	<.001**	<.001**	***	***
	DDS	<.001	<.001	<.001	<.001	<.001*	<.001*
	LIS	.943	.368	.493	.605	<.001*	.224*
	LDS	.732	.895	.911	.957	.937	***
	DIN	.217	.906	<.001**	<.001**	***	***
	DDN	<.001	<.001	<.001	<.001	<.001*	***
	LIN	.949	.469	.442**	.336	***	<.001
	LDN	.924	>.995	.978	>.995**	.980	***

* Outcomes based on 4000 observations
** Outcomes based on 2000 observations
*** Missing outcomes

In Table 3, for the CDC generator, it can be seen that more extreme probability values occur for an RMULT value of 37 than for an RMULT value of 1. In addition, for most of the models, and for all values of the traffic intensity, there is a pronounced tendency for the Chi-Square probabilities to become more extreme as the number of observations are increased. Since the opposite tendency might be expected these results suggest that something other than just random and/or non-steady state behavior might be affecting the results. It seems likely that the extreme probability values in Table 3 are caused by the poor performance of the CDC random number sequences in excess of 1000 observations as evidenced by the poor Chi-Square statistics shown in Table 1. Another indication of the sensitivity of the results to the properties of the random sequence is shown by the differences between the results for the DECIMAL and INTEGER sequences of the CDC generator. This difference is surprising since the integer random numbers are taken from the high order three digits of the decimal fractions in the CDC generator.

It seems clear from Tables 2 and 3 that the three strategies, S, A and N, for determining the service times are equally valid. However the outcomes of the strategies differ, sometimes quite markedly. This implies different P_n distributions for the strategies and is probably a result of an underlying serial correlation in the random number sequence. More experiments with the random number generators are required to judge the importance of this result and for the determination of the superior strategy.

During the conduct of the experiments we also compared the moments of the theoretical distributions with those of the observed distributions. As might be expected the means of the observed distributions are essentially unbiased. The differences between the observed and theoretical means were (usually) less than 3 percent for the experiments with the IBM generator

TABLE 3

CHI-SQUARE OUTCOMES FOR QUEUEING MODELS USING THE CDC RANDOM NUMBER GENERATOR

TRAFFIC INTENSITY	MODEL	1000 OBSERVATIONS		3000 OBSERVATIONS		5000 OBSERVATIONS	
		PROB1	PROB37	PROB1	PROB37	PROB1	PROB37
0.1	DIS	.668	.414	.019	.019	.002	<.001
	DDS	.281	.418	.002	.015	<.001	<.001
	LIS	.668	.414	.019	.019	<.001	<.001
	LDS	.280	.418	.002	.015	<.001	<.001
	DIN	.889	.727	.797	.085	.474	.327
	DDN	.139	.364	<.001	.168	<.001	.580
	LIN	.889	.727	.797	.085	.474	.327
	LDN	.139	.364	<.001	.168	<.001	.580
0.3	DIS	.964	.095	.348	<.001	.068	<.001
	DDS	.664	.115	.046	<.001	<.001	<.001
	LIS	.967	.141	.377	<.001	.069	<.001
	LDS	.671	.119	.050	<.001	<.001	<.001
	DIN	.963	.750	.211	.700	.008	<.001
	DDN	.523	.619	.004	.285	<.001	<.001
	LIN	.964	.761	.727	.791	.008	<.001
	LDN	.527	.632	.404	.286	<.001	<.001
0.5	DIS	<.001	.404	***	<.001	***	<.001
	DDS	***	.430	***	<.001	***	<.001
	LIS	.985	.510	***	<.001	***	<.001
	LDS	***	.462	***	<.001	***	<.001
	DIN	<.001	<.001	<.001	<.001	***	***
	DDN	<.001	<.001	***	***	***	***
	LIN	<.995	.891	.939**	.544	.728	***
	LDN	.992	.937	.903**	***	***	***

** Outcomes based on 2000 observations

*** Denotes missing values

and less than 6 percent for the experiments with the CDC random number generator.

On the other hand, the second moments are substantially underestimated; the underestimation fluctuated between 5 and 10 percent of the theoretical values for the experiments using the IBM random number generator and between 19 and 30 percent for the experiments using the CDC random number generator. The third moments were also severely underestimated--for the experiments with the IBM generator underestimation of between 25 and 30 percent was a common result while for the CDC generator the range was between 5 and 38 percent. These results strongly suggest an underlying positive correlation in the generated sequences which is a common phenomenon in queueing simulations (Fishman [4]). Since classical statistical analyses cannot be applied to the analysis of such results it would be useful if GPSS could supply the analyst with the means to estimate the autocovariances of the generalized sequences thereby allowing the precision of simulation results to be estimated with more precision.

It is evident from Tables 2 and 3 that the variability of the results increases with the traffic intensity. This sensitivity may be partly due to the transitional behavior of the simulated systems indicating that more observations are required to achieve a steady state. However, the inherent costs in longer runs are very high and many users would therefore be satisfied with shorter runs of the same order of magnitude as our experiments. If alternative policies are to be compared, it would be advisable to start all the simulation runs in a state approximating the steady state condition. As GPSS does not automatically provide for the restoration of the initial state unless it is an empty system, special measures to this end should be taken by the user even though this complicates the simulation and increases its cost.

5. CONCLUSION

In this paper we presented some results of tests using different GPSS random number generators. These results illustrate the sensitivity of the outcome of simulation experiments to the random number generator used and to the values chosen by the analyst for the RMULT instruction. They also show that the strategy chosen by the analyst for assigning random numbers to events can substantially effect the experimental results. In addition, the experiment with the CDC random number sequences suggests that the quality of the generated random number sequences (as measured by standard statistical tests) can have a significant effect both on the results and on the speed with which the simulation attains a steady state. These conclusions seem important enough to warrant much more extensive investigation and attention by simulation practitioners. In particular, much attention should be paid to choosing good random number generators and to avoiding the subsequences of a given generator which have poor statistical properties.

Of the two GPSS random number generators tested the IBM generator appears to give the more satisfactory performance. However the DECIMAL sequences in the IBM generator may not be as good as the INTEGER sequences. For this reason we would recommend a change in the random number generator algorithm: specifically, to precede Steps 3 and 4 by Step 6 and thus eliminate the effects of Step 3 (and the zero in bit 32) on the generated random decimal fraction. In addition, we would recommend that additional statistics, and in particular autocovariances, be incorporated (possibly as an option) into

GPSS. It would also be useful if GPSS were extended to include SYSTEMSAVE and SYSTEMRESTORE blocks, which would enable the user to save and restore a (steady state) initial status of the simulated system, its queues and its Chains. Finally, our results indicate that additional tests of the GPSS generator, which would supplement Felder's [3] tests, are required.

ACKNOWLEDGEMENT

We would like to thank Dr. Samuel Gorenstein and Ms. Mary Sortet of IBM, as well as Dr. Thomas Schriber and Mr. James Henriksen, who gave us the assembly code of IBM's GPSS random number generator. We also wish to thank Mr. Walter Gazdick and Mr. Barry Robinson of the Vogelback Computing Center, Northwestern University, for supplying us with the CDC GPSS random number generator.