A comparison of the Correlational Behavior of Random Number Generators for the IBM 360

JOHN RB WHITTLESEY
Mandrel Industries, Inc.* Houston, Texas

Hutchinson states that the "new" (prime modulo) multiplicative congruential pseudorandom generator, attributed to D. H. Lehmer, has passed the usual statistical tests for random number generators. It is here empirically shown that generators of this type can produce sequences whose autocorrelation functions up to lag 50 exhibit evidence of nonrandomness for many multiplicative constants. An alternative generator proposed by Tausworthe, which uses irreducible polynomials over the field of characteristic two, is shown to be free from this defect.

The applicability of these two generators to the IBM 360 is then discussed. Since computer word size can affect a generator's statistical behavior, the older mixed and simple congruential generators, although extensively tested on computers having 36 or more bits per word, may not be optimum generators for the IBM 360.

KEY WORDS AND PHRASES: random numbers, pseudorandom number generators, autocorrelation function, serial correlation, digital shift-register generators, linear recurrence modulo two, irreducible polynomials, primitive trinomials modulo two, congruential generators, prime numbers, statistical tests for randomness, IBM 360, 32-bit versus 36-bit word size
CR CATEGORIES: 5.5
Volume 11 / Number 9 / September, 1968

1. Correlations from the Lehmer Generator

Hutchinson [1], and earlier, Holz and Clark [2], Greenberger [3], and D. H. Lehmer [4], have suggested a method, attributed to Lehmer, for obtaining uniform pseudorandom numbers by means of the generators:

$$Y_{i+1} = AY_i \bmod (p), \tag{1}$$

where p is the largest prime less than some 2^s.

In this note we investigate certain empirical and theoretical properties of autocorrelation functions or "correlograms" calculated from sequences of numbers generated by several different pseudorandom generators.

The autocorrelation function is a measure widely used in studying stochastic processes, especially in applied fields such as brain research, geophysics, and oceanography [5]. If we

*Research Department, Ray Geophysical Division let X_i, $i = 1, 2, \ldots$ be a sequence from a zero mean stochastic process, then we may define the autocorrelation function of a sample of length N, from this sequence, as

$$R(t) = \frac{1}{N} \sum_{i=1}^{N} X_i X_{i+t}. \tag{2}$$

(Such a sequence can be generated by letting $X_i = Y_i - (p+1)/2$.) What we in fact employ in this note is the normalized autocorrelation function, defined by

$$R_{xx}(t) = \frac{R(t)}{R(0)}, \tag{3}$$

where the symbol $R_{xx}(t)$ is used to distinguish the autocorrelation function from the cross-correlation function $R_{xx}(t)$.

Values for $R_{xx}(t)$ were calculated with $N \approx 2500$ and lags, t, ranging from 1 to 50, for several thousands of sequences[1]obtained from generator (1) using $p = 2^{15} - 19 = 32749$ and $p = 2^{31} - 1$. The maximum value found in each of these autocorrelation functions, $\max|R_{xx}(t)|$, over the range $1 \leq t \leq 50$ was determined, and the lag, t, at which each of these maxima occurred was noted.

Using the large sample approximation[2] to the distribution of the serial correlation coefficient derived by Anderson [6] to calculate the expected frequencies of occurrence of various values for $\max|R_{xx}(t)|$, we tested a variety of integer multipliers A. For many of these multipliers[3], the single-precision Lehmer generator (using $p = 32749$) was found to give excessively large values for $\max|R_{xx}(t)|$ far more frequently than could be expected by chance. If A is either too small or too close to p, one would expect successive terms in the sequences generated by this method to be highly correlated [7]. (For small A, the expected value for $R_{xx}(1)$ is approximately $1/A$.) This prediction was confirmed by our empirical findings.

For $N \approx 2500$ and $1 \leq t \leq 50$, more than 99 percent of all sequences generated by truly random processes should give $\max|R_{xx}(t)|$'s in the range 0.03 to 0.08 with almost half of the values lying between 0.045 and 0.055.

Using equations from Greenberger [7], it is possible to predict (for $N = p$) that multipliers of the form $A = (p-1)/K$, where K is a small integer, will give Lehmer generated values

[1]IBM's high-speed peripheral "multiplier-summation processor" attachment for the 360 was used to calculate these correlations.

[2]Anderson states [6, p. 12] that "for $N > 75$, the large sample [Gaussian] approximations can be used." Having determined that $R_{xx}(1), R_{xx}(2), \ldots, R_{xx}(50)$ are independent and approximately Gaussian distributed (mean ≈ 0, variance $\approx 1/N$), one can then readily obtain histograms for the distribution of $\max|R_{xx}(1), R_{xx}(2), \ldots, R_{xx}(50)|$ by taking successive (two-sided) areas from this Gaussian probability density function, raising each of these areas to the 50th power, and then subtracting to obtain: $\Pr(a_i < \max|R_{xx}(\tau)| \leq a_{i+1})$.

[3]It should be noted that not all of the multipliers tested were primitive roots of p. However, all multipliers reported on in this note generated sequences whose period was long relative to N.

for $|R_{xx}(1)|$ approximately equal to

$$\frac{K}{p-1} + \frac{1}{K}. \tag{4}$$

Tests using both $p = 32749$ and $p = 2^{31} - 1$ showed that multipliers of this form did in fact produce correlations at lag $t = 1$ equal to the values predicted by expression (4) to within plus or minus two standard deviations (i.e., to within about ± 0.04). Thus, for example for $A = (p-1)/3$, we obtained $|R_{xx}(1)|$'s in the range 0.29 to 0.37 instead of $|R_{xx}(1)| \approx 0.02$ as would have been expected for truly random sequences. Thus, Greenberger's theory is empirically substantiated for N less than the above values for p.

Coveyou and Macpherson [8] suggest that the multipliers for congruential generators should not be chosen close to any simple rational multiple of the generator's period or the square root of that period. Empirically we observed that the following partial list of multipliers, lying between $p^{\frac{1}{3}}$ and $p^{\frac{1}{2}}$, gave abnormally large values for $\max|R_{xx}(t)|$ from Lehmer's generator using $p = 32749$:

$$A \approx p^{\frac{1}{3}}, \frac{1}{3}p^{\frac{1}{2}}, 2p^{\frac{1}{3}}, \frac{5}{12}p^{\frac{1}{3}}, \frac{10}{3}p^{\frac{1}{3}}, \frac{2}{3}p^{\frac{1}{2}}, 4p^{\frac{1}{3}}, \frac{4}{5}p^{\frac{1}{2}}, \frac{5}{6}^{\frac{1}{2}}, \text{ and } p^{\frac{1}{2}}$$

It was further noted that multipliers which produced $\max|R_{xx}(t)|$'s that exceeded 0.08 tended to produce these larger than normal $\max|R_{xx}(t)|$'s at fixed lags. For example, the multiplier $A = 106$ with $p = 32749$ produced $\max|R_{xx}(t)|$'s ranging in value between 0.18 and 0.23 consistently at lag $t = 21$, while the multiplier $A = 166$ produced $\max|R_{xx}(t)|$'s in the range 0.16 to 0.23 at a lag of 33. However, other multipliers in the range $p^{\frac{1}{3}} < A < p_{\frac{1}{2}}$ gave values for $\max|R_{xx}(t)|$ which were consistently too small.

The effect of scaling the sequence elements, X_i, was also studied. When each of the uniform pseudorandom numbers generated modulo $p = 32749$ was divided by 20 before calculating the autocorrelation functions, the results for $\max|R_{xx}(t)|$ remained essentially unchanged. But when this scaling factor was decreased from $\frac{1}{20}$th to $\frac{1}{80}$th, many of the sequences whose $\max|R_{xx}(t)|$'s had been in the range 0.04 to 0.06 increased their $\max|R_{xx}(t)|$'s to above 0.11. Thus both a decrease in the size of N, and a decrease in the number of bits used in obtaining the autocorrelation functions, were found to increase the values for $\max|R_{xx}(t)|$.

Finally we suggest that a pseudorandom number generator which fails to give appropriate values for $\max|R_{xx}(t)|$ is also likely to fail some of the other criteria recommended by authors such as MacLaren and Marsaglia [9] and Gorenstein [10] for testing such generators.

2. Tausworthe Generator

A generator of uniform pseudorandom numbers which is predicted to have consistently well-behaved autocorrelation functions has recently been proposed by Tausworthe [11]. This generator is based on irreducible polynomials over the field of characteristic two. In his paper Tausworthe calculates the mean and variance of the unnormalized autocorrelation function $R(t)$ for large samples of size $N < p$ obtained using his linear recurrence modulo two[4] generator with period p. Theoretical upper bounds for the mean value and variance of $R(t)$ from this generator for samples of size N are

$$|\text{mean } R(t)| < \frac{1}{p} \approx 0 \tag{5}$$

and

$$\text{var}[R(t)] = \frac{1}{9N} + \text{terms of order } (2^{-L}) \text{ and } \frac{1}{pN} \tag{6}$$

for nonzero integral values of $|t|$ less than $(p - L)/q$ (where q and L are parameters of his generator which are chosen to be $\ll p$; e.g., $p = 2^{31} - 1$, $q = L = 31$).

The mean value for the zero-lag of the unnormalized autocorrelation function from his generator is

$$\text{mean } R(0) \approx \frac{1}{3}. \tag{7}$$

The generator is predicted to give a distribution of well-behaved pseudorandom numbers which is uniform over the interval $(-1, 1)$, and is also uniform over the corresponding "unit" square, cube, etc. (corresponding to n-tuples of order 2, 3, etc.) for suitable choices of $L \leq q \leq n/2, n/3, etc.$

A version of the Tausworthe generator (personal communication) has been programmed in double precision for SDS 900 series computers by Dr. William B. Kendall of the California Institute of Technology's Jet Propulsion Laboratory. A single-precision algorithm for 32-bit-word computers based on Dr. Kendall's program and on a list[5] of primitive trinomials (mod 2) compiled by E. J. Watson of the University of Manchester is as follows:

> For n one less than the numbers of bits per word (or multiple word) and m less than $n/2$ (as chosen from the list of primitive trinomials), proceed as follows:
>
> 1. Let register A (or multiple-precision register, A) initially contain the previous random number y in bit positions 1 to n with zero in the sign-bit (position 0).
> 2. Copy register A into register B and then right-shift register B m places.
> 3. EXCLUSIVE-OR register A into register B and also store result back into register A. (Registers A and B now have bits for the new random number in positions $m + 1$ to n, but still contain bits from the old n-bit random number in positions 1 through m.)

[4]Tausworthe's linear recurrence modulo two generator is also known as a digital shift-register sequence generator.

[5]This list is available from the Jet Propulsion Laboratory. (A somewhat shorter list of these trinomials also appears in [12], Table 4-2 and in [13].)

4. Left-shift register B $(n-m)$ positions. (This places m bits for the new random number in positions 1 to m of register B and zero bits in positions $m+1$ through n.)
5. EXCLUSIVE-OR register B into register A and zero out register A's sign bit. (Register A now contains all n bits of the new random number y'.)

This algorithm will generate all possible $(2^n - 1)$ nonzero n-bit numbers before it repeats any of these numbers, if for $n = 31$, m is set equal to 3, 6, 7, or 13. Sequences of lengths up to $(p-L)/q \approx 67 * 10^6$ from this generator will then have desirable properties for $R_{xx}(t)$ for all values of t.

A program which employs the above algorithm has been written for the IBM 360. On the 360/50, the seven instructions which actually generate the 31-bit pseudorandom integers require 35 microseconds.

Using this program, and letting $x = (y - p/2)/(131 \cdot 10^4)$, an empirical distribution for the maximum autocorrelations, $\max |R_{xx}(t)|$, was obtained from a thousand nonoverlapping sequences of length $N = 2503$. This empirical distribution agreed almost exactly with theoretical predictions.[6]

A double-precision Tausworthe generator for the IBM 360 may be written by setting $n = 63$ and $m = 1, 5,$ or 31. Every four successive full-words from this generator can then be broken down into nine smaller pseudorandom numbers, y_k, whose length L and spacing $q (\leq L)$ can both equal 14 bits.[7]

The parameter q will now have become less than $n/4$ so that normalized N-tuples up to $(y_k, y_{k+1}, y_{k+2}, y_{k+3})$ from such a generator can be expected, by theory, to be uniform over the corresponding unit square, cube, and four-dimensional hypercube.

Both the single-precision and double-precision versions of the algorithm must be initialized by a suitable starting integer containing a random (or nearly random) arrangement of zeros and ones.

3. Discussion of Pseudorandom Number Generators for the IBM 360

The problem of finding satisfactory generators for the IBM 360 arises primarily from its reduced word size compared with earlier machines. While the older (mod 2^s) multiplicative congruential generators have been shown to be satisfactory on computers with word sizes of 35 bits or larger [10], a nonmixed congruential generator with $s = 31$ will allow a maximum period of only $2^s/4$. (Mixed congruential generators would allow periods of 2^{31}, but the instruction times on modern computers are now such that the earlier speed advantages of

[6]Details are available from the author.

[7]Alternatively, each 63-bit double-precision pseudorandom number can be broken down into three 21-bit numbers. For this partitioning, $q = L = n/3$.

these mixed generators is no longer as important [14].) Furthermore, it is only the first bit of these older congruential generators which has the full period. The other bit periods decrease with bit position until the last bit is always one [15, 16]. Thus, as pointed out by Coveyou and Macpherson [8], "unquestionably good statistical performance" may be difficult to achieve using the older congruential generators on computers with word sizes shorter than 35 bits, unless multiple precision arithmetic is employed.

It is for these reasons that studies are needed of other pseudorandom number generators. In this note we have compared two generators, the "new" Lehmer (prime number modulo) congruential generator, and the Tausworthe generator based on irreducible polynomials over the field of characteristic two. For the Tausworthe generator there are already a priori estimates and upper bounds concerning statistical properties, not only for the entire sequence of numbers generated by its algorithm but also for subsequences of lengths less than a full period [11, 17]. These subsequence estimates predict desirable properties with respect to its autocorrelations and with respect to the uniformity of the distribution of certain n-tuplets. The "new" Lehmer generator, on the other hand, is as yet largely untested on the IBM 360. Earlier articles on this generator [1, 2] claimed that it had passed some of the more "usual" random number tests, but only minimal clues [3] were given regarding the choice of suitable multipliers to be used with word lengths shorter than 2^{36}. More recently Coveyou and Macpherson [8] have suggested some new guidelines for choosing multipliers for any congruential generator. In particular they suggest (on the basis of a priori Fourier analyses of the full period of a number of such generators) that the multipliers should not be close to any simple rational multiple of the generator's period or the square root of that period. Our own empirical studies tend to substantiate and extend (for subsequences which are shorter than the full period) the rules suggested by Coveyou and Macpherson.

Other statistical properties have yet to be tested for the IBM 360 versions of these generators, for example, their run and serial properties [10, 18], their Fourier properties (e.g., see tests proposed by M. R. Schroeder in [16] and by Coveyou and Macpherson in [8]), their behavior as generators of two-dimensional random plots (see [16, 19]), their gap properties, their poker tests properties [20], etc.

Comparing times on the IBM 360/50, we find that a 63-bit double-precision version of the Tausworthe generator can be programmed to require only about 27 microseconds per random (16-bit) stored integer; that a load-multiply-divide-store sequence of instructions for the Lehmer generator[8] requires about 50 microseconds; and that mixed congruential generators with arbitrary multipliers might require on the order of 32 microseconds. Thus from several viewpoints the Tausworthe generator appears at present to be a preferable choice for computers of this word size.

[8]Greenberger [3] has devised a method for performing these Lehmer calculations using the following primitive roots of $p = 2^31 - 1$: $A = 2^{12} + 1$, $2^{14} - 1$, $2^{14} + 1$, $2^{17} + 1$, $2^{18} - 1$, $2^19 - 1$, $2^19 + 1$. His method requires no multiply or divide instructions.

Summary

Correlational properties of the Lehmer (modulo prime) congruential generator have been studied in detail using sequences of length $N \approx 2500$ and the primes $p = 32749$ and $p = 2^{31} - 1$. Some multipliers for this generator may give maximum autocorrelations and cross-correlations, for lags between 1 and 50, which are consistent with statistical estimates for independent samples of length N from truly random populations; but many multipliers definitely do not. Estimates for serial correlations for some of these unsatisfactory multipliers, based on theory developed by Greenberger, agree well, within the limits of expected statistical variability, with values obtained from actual correlated sequences.

The correlational properties of the Lehmer generator (for which only empirical data are currently available) are then compared with theoretical and empirical correlations for the "linear recurrence modulo two" generator proposed in 1965 by Tausworthe. Both single-precision and double-precision versions of this new Tausworthe generator have been coded in IBM 360 assembly language. The computation times for the double-precision version (whose period is approximately 10^{19}) are shorter than the times for various single-precision, simple and mixed, congruential generators (whose periods are all less than 10^{10}).

RECEIVED JANUARY, 1967 ; REVISED APRIL, 1968

References

[1] HUTCHINSON, D.W. A new uniform pseudorandom number generator. *Comm. ACM* 9, 6 (June 1966), 432–433.

[2] HOLZ, B. W., AND CLARK, C.E. Tests of randomness of the bits of a set of pseudorandom numbers. Published by Operations Research Office (ORO), Dec. 1958; reproduced by ASTIA as AD207553.

[3] ORCUTT, G. I., GREENBERGER, M., KORBEL, J., AND RIVLIN, A. M. Microanalysis of Socioeconomic Systems. Harper & Row, New York, 1961, App. to Pt. IV, pp. 356-370.

[4] LEHMER, D. H. Mathematical methods in large scale computing units. In Proceedings of a Second Symposium on Large-Scale Digital Calculating Machinery, Ann. Comput. Lab: Harvard U. 26 (1951), 141-146.

[5] WOLD, HERMAN O. A. (Ed.). Bibliography of Time Series and Stochastic Processes. M.I.T. Press, Cambridge, Mass., 1965.

[6] ANDERSON, R. L. Distribution of the serial correlation coefficient. *Ann. Math. Statist.* 13, 1 (Mar. 1942), 1-33.

[7] GREENBERGER, M. An a priori determination of serial correlation in computer generated random numbers. *Math. Comput.* 15 (1961), 383-389; corrigenda, *Math. Comput.* 16 (1962), 126, 406.

[8] COVEYOU, R. R., AND MACPHERSON, R.D. Fourier analysis of uniform random number generators. *J. ACM* 14, 1 (Jan. 1967), 100-119.

[9] MACLAREN, M. DONALD, AND MARSAGLIA, GEORGE. Uniform random number generators. *J. ACM* 12, 1 (Jan. 1965), 83-89.

[10] GORENSTEIN, SAMUEL. Testing a random number generator. *Comm. ACM* 10, 2 (Feb. 1967), 111-118.

[11] TAUSWORTHE, ROBERT C. Random numbers generated by linear recurrence modulo two. *Math. Comput.* 19 (1965), 201-209.

[12] KORN, GRANINO A. Random-Process Simulation and Measurement. McGraw-Hill, New York, 1966, p. 85.

[13] GOLOMB, S. W., WELCH, L. R., GOLDSTEIN, R.M., AND HALES, A. W. Shift Register Sequences. Holden-Day, San Francisco, 1967, p. 97.

[14] HULL, T. E., AND DOBELL, A.R. Mixed congruential random number generators for binary machines. *J. ACM* 11, 1 (Jan. 1964), 31-40.

[15] BARNETT, V. D. The behavior of pseudo-random sequences generated on computers by the multiplicative congruential method. *Math. Comput.* 16 (1962), 63-69.

[16] CHAMBERS, R. P. Random-number generation on digital computers. *IEEE Spectrum* 4, 2 (Feb. 1967), 48-56.

[17] CANAVOS, GEORGE C. A comparative analysis of two concepts in the generation of uniform pseudo-random numbers. Proc. 22nd Nat. Conf. ACM (ACM Pub. P-67), Thompson Book Co., Washington, D. C., pp. 485-501.

[18] DOWNHAM, D. Y., AND ROBERTS, F. D. X . Multiplicative congruential pseudo-random number generators. *Comput. J.* 10, 1 (May 1967), 74–77.

[19] GREENBERGER, M. Method in randomness. *Comm. ACM* 8, 3 (Mar. 1965), 177-179.

[20] ITZELSBERGER, G. Some experiences with the poker test for investigating pseudo-random numbers. In Hollingdale, S. H. (Ed.), Digital Simulation in Operational Research, American Elsevier, New York, 1967, pp. 64-68.