Multiplicative congruential pseudo-random number generators

By D. Y. Downham and F. D. K. Roberts*

Congruential random number generators are discussed. Such generators are both computer and compiler dependent: this is discussed in relation to high level languages on a binary machine. One of the statistical tests, which has only occasionally been used, is shown to be more "sensitive" than the other tests.

1. Introduction

In recent years, the advent of electronic computers has made it possible to use Monte Carlo and simulation techniques to solve many problems for which analytic or numerical solutions have been difficult or even impossible to find. These techniques require sequences of random numbers from various probability distributions to be readily available. Various methods (see, for example, Tocher, 1963) have been evolved for obtaining such sequences from a uniform random sequence in (0, 1). Such a sequence of uniform random numbers, which is generated within a computer in a deterministic manner, is often referred to as a *psuedo-random number sequence.*

The most frequently used generators take the form

$$x_{i+1} = kx_i + c \bmod(m),\ i = 0, 1, 2, \dots \quad (1)$$

where x_0, k, c, m are integers, x_0, k, $c < m$. The sequence $\{x_i/m\}$ is then taken to be the uniform random number sequence. Clearly the sequence must repeat itself at some stage but by suitable choice of k, c, m, x_0, a sufficiently long cycle can be generated. If $c = 0$, the generator is called *multiplicative*, otherwise it is said to be *mixed.* On binary computers, the modulus m is usually taken in the form $m = 2^N$, since the congruence can be more speedily evaluated, a direct division not being necessary.

For mixed generators with $m = 2^N$, necessary and sufficient conditions for a full length cycle are $k = 1 \bmod(4)$, c odd, x_0 arbitrary, $0 \leqslant x_0 \leqslant 2^N - 1$. With mixed generators of this type, k is often of the form $k = 2^a + 1, (a \geqslant 2)$ so that multiplication can be effected merely by a shift and add. The results of Hull and Dobell (1964), show that the statistical behaviour of this type of generator must be viewed with suspicion although certain values of k do appear satisfactory.

The maximum length cycle for multiplicative generators, (i.e. $c = 0$), when $m = 2^N$, is 2^{N-2} which is attained when $k = \pm 3 \bmod(8)$ and x_0 is odd, $1 \leqslant x_0 \leqslant 2^N - 1$.

Most of the literature already published (see, for example, Hull and Dobell (1964), Kuehn (1961), Pike and Hill (1965), Rotenberg (1960)) has dealt with the modulus $m = 2^N$. If access can easily be gained to the binary pattern within the machine, these generators will operate more speedily. However, we have been programming in ALGOL on KDF9 using the Kidsgrove compiler, and little advantage can be gained by using binary orientated parameters (see Section 4).

In this paper, we consider multiplicative generators of the type

$$x_{i+1} = kx_i \bmod(p), \quad i = 0, 1, 2, \dots$$

where the modulus p is a large prime and the multiplier k is a primitive root mod(p). This generates a full cycle which is a permutation of the integers $1, 2, \dots, p - 1$.

Many generators of this type were subjected to the statistical tests listed in Section 3, and a few of the results are included in **Table 1**.

2. Number Theory

Consider the generator

$$x_{i+1} = kx_i \bmod(p). \quad (2)$$

Then

$$x_i = k^i x_0 \bmod(p).$$

We will denote the highest common factor of a and b by $hcf(a, b)$. The length of the cycle is given by the minimum n satisfying

$$k^n = 1 \bmod(p). \quad (3)$$

The Fermat–Euler theorem states that if $hcf(k, m) = 1$ then $k^{\phi(m)} = 1 \bmod(m)$ where $\phi(m)$ is the number of integers less than and prime to m. In our case, $m = p =$ prime. Hence $\phi(p) = p - 1$. However $\phi(p)$ is not necessarily the smallest integer satisfying (3). It has been shown (see Tocher, 1963) that the smallest value of n satisfying (3) must divide $\phi(p)$. If the smallest value of n satisfying (3) is $\phi(p) = p - 1$, then k is called a *primitive root* mod(p) and the cycle of length $p - 1$ can be attained.

Hardy and Wright (1960), have shown that primitive roots exist for every prime p and that the number of primitive roots is equal to $\phi(p - 1)$. There is no general technique for finding a primitive root for a given prime. Usually a process of trial and error is used to find a small primitive root g. The remaining primitive roots

* *Department of Computational and Statistical Science, The University, Liverpool* 3.

Table 1*

Results of tests on 6 generators using a prime modulus p, and a primitive root k

Test	$k = 8192$ $p = 67101323$			$k = 8192$ $p = 67099547$			$k = 32768$ $p = 16775723$			$k = 54751$ $p = 99707$			$k = 8$ $p = 67100963$			$k = 32$ $p = 7999787$		
		run			run			run			run			run			run	
	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
Uniformity	50	92	39	34	23	82	82	30	43	51	96	13	75	15	28	38	35	76
Serial lag 1	18	54	90	61	12	46	3·9	73	14	4·9	12	22	ϵ	ϵ	ϵ	2·7	ϵ	1·3
Serial lag 2	7·9	82	60	7·0	54	94	95	56	36	29	20	25	24	24	3·3	40	70	53
Serial lag 3	84	98	94	20	82	55	59	23	99	ϵ	ϵ	ϵ	84	52	96	63	8·0	92
Serial lag 4	25	87	44	89	22	57	37	60	50	0·3	21	54	96	37	59	21	7·8	51
Serial lag 5	22	93	94	25	8·8	85	31	47	96	27	31	43	64	38	46	25	34	88
Serial lag 6	38	95	47	64	19	20	9·8	84	85	ϵ	ϵ	ϵ	42	93	46	50	12	99
d^2 test	42	79	12	97	45	47	91	3·4	29	20	0·04	0·27	16	9·9	2·8	87	33	22
Sum of 2	20	89	9·6	94	74	62	36	38	55	88	96	19	38	95	33	92	74	36
Sum of 3	76	58	89	69	30	12	14	95	75	56	66	19	26	3·6	37	7·0	54	9·6
Sum of 4	70	30	7	22	8	93	84	9·1	3·9	0·09	ϵ	ϵ	36	25	2·1	43	4·4	94
Sum of 5	34	92	25	3·6	43	18	54	53	89	ϵ	ϵ	ϵ	1·0	2·7	0·89	78	26	15
Runs/median	23	13	54	59	2·9	35	66	0·89	32	27	89	93	75	33	92	92	67	32
Runs up/down	48	67	98	93	19	44	90	82	60	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	0·30	0·26	1·5
Poker test	38	98	58	43	19	60	84	94	67	ϵ	0·03	ϵ	3·4	32	78	63	61	6·3

* The tabulated value is the probability, expressed as a percentage, that the appropriate Chi-square variate will exceed the computed value. Where ϵ occurs, it means the probability is less than 0·01 per cent. It should be noted that the χ^2 probabilities for a given run are not independent.

may then be generated by

$$k = g^a \bmod(p), hcf(a, p - 1) = 1.$$

However, sufficient conditions have been established for 2 to be a primitive root mod(p), namely:

(1) $(p - 1)/2$ prime
(2) $p = 3 \bmod(8)$.

The proof of this is given by Roberts (1966). For such a prime p, an appropriate primitive root k can be found by $k = 2^a \bmod(p), hcf(a, p - 1) = 1$.

Studies of the serial correlation between successive pseudo-random numbers by Coveyou (1960) and Greenberger (1961) have shown that small values of k produce unsatisfactory generators, and most of the generators we tested used a value of $k \doteq \sqrt{p}$. Larger values of k can produce good results, but this is not always the case. $p = 11$ satisfies the above conditions, hence 2 is a primitive root mod(11)

n	0	1	2	3	4	5	6	7	8	9	10
2^n	1	2	4	8	5	10	9	7	3	6	1

If $p = 67099547$, then the conditions are satisfied, and so 2 is a primitive root. An appropriate value of k is $2^{13} = 8192$. Hence the generator $x_{i+1} = 8192\, x_i \bmod(67099547)$ generates a full cycle, and the sequence $\{x_i/67099547\}$ can then be used as the uniform random number sequence. This generator has good statistical properties and is quite suitable for use on KDF9.

3. Statistical tests

Eight tests were used, each test being applied to the same sequence, and hence the results are not independent. Each generator was tested three times, the sequences being generated by different starting values, x_0. We give here a summary of six of the tests.

Test 1—*Uniformity.* The unit interval was divided into 100 equal intervals. For each run, a sequence of 2000 numbers was used. The occurrences in each interval were counted and the χ^2 statistic computed.

Test 2—*Serial.* The unit square was divided into 100 equal cells and a sequence of 2006 numbers was used. Each pair of random numbers (u_i, u_{i+l}), $i = 1(1)2000$, $l = 1(1)6$ was taken as the co-ordinates of a point in the unit square. The frequencies f_{ij} in cell (i, j) were computed. Good (1953) has shown that the statistic

$$S^2 = \frac{1}{20} \sum_{i,j=1}^{10} (f_{ij} - 20)^2 - \frac{1}{200} \sum_{i=1}^{10} (h_i - 200)^2$$

where $h_i = \sum_{j=1}^{10} f_{ij}$, has asymptotically a χ^2 distribution on 90 degrees of freedom. We computed this statistic for lags $l = 1(1)6$.

Test 3—*Gruenbergers d^2 test.* A sequence of 10,000 numbers was divided into 2,500 sets of 4. Each set of 4 was used to determine the co-ordinates of 2 points in the unit square and d^2, the square of the distance between them was computed. The range of d^2 is (0, 2). This was divided into 15 intervals 0(0·1) 1·4(0·6) 2·0, and the occurrences in each cell of the 2,500 values of d^2 were computed. The observed frequencies were tested against the expected values in each cell as evaluated by Gruenberger and Mark (1951), and the χ^2 statistic computed.

Test 4—*Sum of* $n(n = 2, 3, 4, 5)$. We tested to see if $F(u_1 + u_2 + \ldots u_n)$ was uniform using 100 equal intervals, where F is the distribution function of the sum of n random variables. Sequences of length $n \times 1000$ were used for $n = 2, 3, 4, 5$.

Test 5—*Runs above and below the median.* Herrman (1961) has evaluated the expected number of runs of length r and of length r or greater for a sequence of n random numbers, not differentiating between runs above and below the median. We used this test with sequences of length 10,000 counting the actual number of runs of lengths 1, 2, . . ., 9 and 10 or greater and evaluating the χ^2 statistic.

Test 6—*Runs up and down.* If a generator failed any other test, then it also failed this test. In this sense this test is the most sensitive. One can form sequences of numbers which would satisfy this test and fail other tests, but amongst the forty generators we tested, such a sequence did not occur. Since this test has only occasionally been applied, we give a fuller account of it.

A subsequence

$$x_{i-1}, x_i, x_{i+1}, \ldots x_{i+r-1}, x_{i+r}, x_{i+r+1}\ (2 \leqslant i \leqslant n-r-1)$$

of $r + 3$ consecutive numbers in a sequence of n random numbers is said to form an inside run "up" of length r if

$$x_{i-1} > x_i < x_{i+1} < \ldots < x_{i+r} > x_{i+r+1}.$$

An end run of length r is given by either of the two conditions

$$x_1 < x_2 < x_3 \ldots < x_{r+1} > x_{r+2}\ (1 \leqslant r \leqslant n-2)$$
$$x_{n-r-1} > x_{n-r} < x_{n-r+1} < \ldots < x_n\ (1 \leqslant r \leqslant n-2).$$

Runs down are defined by a reversal of the inequality signs.

Herrman (1961) has shown that the expected number of runs of length r, not differentiating between runs up and runs down, is given by

$$E(r) = 2 \times \left\{\frac{(r^2 + 3r + 1)n - (r^3 + 3r^2 - r - 4)}{(r+3)!}\right\}.$$

The expected number of runs of length r or greater is approximately given by

$$E'(r) = 2 \times \left\{\frac{(r+1)n - (r^2 + r - 1)}{(r+2)!}\right\}.$$

We used sequences of length 10,000 and counted the actual number of runs of length 1, 2, . . ., 5 and 6 or greater. These were compared with the expected values and the χ^2 statistic computed.

Test 7—*Poker test.* The poker test is used to examine the distribution of sets of five decimal digits. For each run, a sequence of 10,000 numbers was divided into 2,000 separate sets of five, the first digit of each number being used. The probabilities for the different arrangements of the digits are tabulated in Herrman (1961).

A test to examine the independence of the first three decimal digits in each number was also used and since it was satisfied even when many other tests were not, we have not included it in the table.

4. Computer considerations

Usually simulation programs are written in a high level language. Hence, in choosing the generators to be used in any problem, the peculiarities of the machine and compiler must be borne in mind. We have been programming in ALGOL on KDF9 using the Kidsgrove compiler. KDF9 is a 48-bit word machine: the bit patterns are not available to programmers using the ALGOL compiler unless special procedures are written in basic machine code and incorporated into ALGOL programs. Thus many of the advantages of using powers of 2 and multiplying by shifting are lost.

We have tried four different approaches:

(*a*) The algorithm of Pike and Hill (1965) was considered. This was written as an ALGOL procedure and used a value of $m = 2^{26}$. This generator was rejected because we could take no advantage of the binary form and the cycle is only one quarter the possible cycle length.

(*b*) Using basic machine code we were able to take advantage of the binary modulus. This proved unsatisfactory because the machine code had to be written as a procedure code body, and the access time to the procedure was prohibitive.

(*c*) We generated a full cycle using a prime modulus inserting in the program the three ALGOL instructions.

```
x:= k × x;
x:= x − (x ÷ p) × p;
random:= x/p;
```

where x, k, p are integers and *random* is a real variable. No time was wasted entering a procedure and the method proved three times faster than (*a*) and (*b*).

(*d*) Using the mixed generator
$x_{i+1} = (2^9 + 1)x_i + 29741096258473 \bmod(2^{47})$
suggested by Kuehn (1961), we generated an array of random numbers in a procedure code body. Thus we could take advantage of the binary parameters, and the procedure access time per random number was reduced. This generator satisfied the statistical tests but was only marginally faster than (*c*). Much of the time gained by using binary parameters was lost in locating array elements.

5. Results and conclusions

Table 1 gives the probability, expressed as a percentage, that a χ^2 variate will exceed the observed value. The first 3 generators appear quite satisfactory and are suitable for use on KDF9. The fourth generator uses a multiplier $k \gg \sqrt{p}$, which satisfies the equation $k^3 = 2 \bmod(p)$ where the modulus $p = 99707$. Hence we would expect this generator to fail the serial test with lags 3 and 6. Other large multipliers have also been tested and most of them appeared satisfactory. The last 2 generators in Table 1 use a small multiplier k, and as will be seen from the results of the serial test with lag 1 and Test 6—Runs up and down, these generators cannot be considered as acceptable. Test 6 was found to be a very useful and sensitive test and it is surprising that it has so rarely been used.

Acknowledgement

The second named author wishes to express his thanks to the Science Research Council for their financial support during the preparation of this paper.

References

Coveyou, R. R. (1960). "Serial Correlation in the generation of pseudo-random numbers", *J. Assoc. Comp. Mach.*, Vol. 7, p. 72.
Good, I. J. (1953). "The serial test for sampling numbers and other tests for randomness", *Proc. Camb. Phil. Soc.*, Vol. 49, p. 276.
Greenberger, M. (1961). "An *a priori* determination of serial correlation in computer generated random numbers", *Math. Comp.*, Vol. 15, p. 383.
Gruenberger, F., and Mark, A. M. "The d^2 test of random digits", *Math. Tables Other Aids Comp.*, Vol. 5, p. 109.
Hardy, G. H., and Wright, E. M. (1960). *The Theory of Numbers*, 4th ed. Oxford: Clarendon Press.
Herrman, R. G. (1961). "The statistical evaluation of random number generating sequences for digital computers", Washington D.C.: Office of Technical Services, U.S. Dept. of Commerce, APEX-635.
Hull, T. E., and Dobell, A. R. (1964). "Mixed congruential random number generators for binary machines", *J. Assoc. Comp. Mach.*, Vol. 11, p. 31.
Kuehn, H. G. (1961). "A 48-bit pseudo-random number generator", *Comm. A.C.M.*, Vol. 8, p. 350.
Pike, M. C., and Hill, I. D. (1965). "Pseudo-Random Numbers", *Comm. A.C.M.*, Vol. 8, p. 605.
Roberts, F. D. K. (1966). "Pseudo-random number generators for digital computers", M.Sc. thesis, Liverpool.
Rotenberg, A. (1960). "A new pseudo-random number generator", *J. Assoc. Comp. Mach.*, Vol. 7, p. 75.
Tocher, K. D. (1963). *The Art of Simulation*, London: English Universities Press.

Book Review

The Matrix Analysis of Vibration, by R. E. D. Bishop, G. M. L. Gladwell and S. Michaelson, 1965; 404 pages. (London: *Cambridge University Press*, 100s.)

The material in this book may be divided into two main parts. The first part gives an elementary exposition of matrix theory and its use in the formulation of vibrational problems. The second part deals with the solution of the fundamental problems of matrix algebra by computational methods.

In Chapter 1 the matrix concept is introduced, and matrix addition, multiplication and inversion and the determinant of a square matrix are defined. Chapter 2 discusses the vibration of a conservative system with a finite number of degrees of freedom while Chapter 3 covers the theory of linear equations and discusses such problems as rank and linear dependence. Chapter 4 takes up the theory of free vibration again, and covers change of co-ordinates and the effect of constraints, and gives a more rigorous treatment of natural frequencies and principal modes. In Chapter 5 the problem of damped systems is taken up, starting with a simple one-dimensional model and continuing with systems with many degrees of freedom. Chapter 6, easily the longest, concludes the first part of the book with a very comprehensive treatment of methods for reducing continuous systems to approximating systems having a finite number of degrees of freedom.

The second half consists of three chapters. Chapter 7 describes the solution of linear equations by a number of variations of triangular decomposition and includes a discussion of the effect of rounding errors. Chapters 8 and 9 deal with the solution of the algebraic eigenvalue problem, the first covering iterative methods and the second covering direct methods. Included in the last chapter is Householder's reduction of a symmetric matrix to tridiagonal form, and the calculation of the eigenvalues of the latter by the Sturm sequence methods and of the eigenvectors by inverse iteration. For unsymmetric matrices, Lanczos' method for the reduction to tridiagonal form is described, and the use of Muller's method for calculating the eigenvalues is discussed.

The book contains a very large amount of information and will undoubtedly be of great value to engineers who wish to make a serious study of vibrational problems. It includes a valuable collection of exercises. The main weaknesses are such as might be expected in a book written by three different authors with somewhat different backgrounds. The fundamental mathematical problem is the determination of values of λ for which $Ax = \lambda Bx$ has non-trivial solutions. This problem is treated several times during the course of the book by arguments of varying sophistication. It is possible that engineers prefer this piecemeal approach but I am not convinced of its effectiveness. Surely it would have been simpler to deal once and for all with the relevant canonical forms associated with $A-\lambda B$? I found the second half of the book much the more satisfying though unfortunately Chapters 8 and 9 were written at a time when new techniques for solving the eigenvalue problem were still advancing rapidly. At the present moment most unsymmetric matrix problems are solved either by reduction to Hessenberg form followed by the QR algorithm or Parlett's version of Laguerre's method, or by the application of Eberlein's method; but none of these techniques is mentioned.

The standard of production of the book is remarkably high and the price is very reasonable for such a handsome volume.

J. H. Wilkinson