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

EFFICIENT AND PORTABLE 32-BIT RANDOM VARIATE GENERATORS

Pierre L'Ecuyer
Département d'informatique
Université Laval
Ste-Foy, Qué., Canada, G1K 7P4.

ABSTRACT

This paper presents some Multiplicative Linear Congruential Generators (MLCGs) and one generator that combine two MLCGs. The individual MLCGs use short multipliers that are nearly optimal based on the spectral test. The shortness of the multipliers leads to fast and portable implementations. The combination method produces a generator whose period is the least common multiple of the individual periods. These generators can also be split into many "independent" generators (i.e. that produce disjoint subsequences) and it is easy to skip a long subsequence of numbers without doing the work of generating them all.

1. INTRODUCTION

Generating random variates between 0 and 1 is fundamental to stochastic simulation and has been the subject much theoretical research; see Bratley, Fox and Schrage (1983), Knuth (1981) and Niederreiter (1978). But still, the most commonly used generators on today's computers are far from perfect. Some are bluntly bad, others have a rather weak theoretical justification; see e.g. Modianos, Scott and Cornwell (1984).

All practical "random variate" generators on computers are actually deterministic finite state automata. They produce a periodic sequence of numbers which should look "apparently random". A generator is thus defined by a finite *state space* S, a function $f : S \to S$ and an initial state s_0 called the *seed*. The state of the generator evolves according to the recursion

$$s_i := f(s_{i-1}), \qquad i = 1, 2, 3, \ldots \tag{1}$$

and the *current state* s_i at stage i is usually transformed into a real value between 0 and 1, according to

$$U_i := g(s_i) \tag{2}$$

where $g : S \to (0, 1)$. The *period* of the generator is the smallest positive integer p such that for some integer $\nu \geq 0$,

$$U_{i+p} = U_i \qquad \text{for all } i > \nu. \tag{3}$$

The choice of f and g should be based on a firm theoretical ground. A good generator should be fast, use few computer memory words, be portable and above all, have good statistical behavior.

A rather popular kind of generator today is the *multiplicative linear congruential generator* (MLCG), for which

$$\begin{aligned} f(s) &= as \text{ MOD } m; \\ g(s) &= s/m; \end{aligned} \tag{4}$$

where the *modulus* m and the *multiplier* $a < m$ are positive integers. The maximal period of such a generator is $p = m - 1$; when m is prime, it is attained if a is a primitive element modulo m (see Knuth (1981), page 19).

MLCGs with too small modulus have too short periods to be used safely for serious applications, while MLCGs with large modulus are often tricky to implement, especially on smaller word size computers. Wichmann and Hill (1982) and Bratley, Fox and Schrage (1983) have proposed a very efficient way to implement a portable MLCG with modulus m using only integers between 0 and m, when a satisfies :

$$a^2 < m. \tag{5}$$

Fishman and Moore (1986) made an exhaustive search of all multipliers a for a MLCG with modulus $m = 2^{31} - 1$, to find the "optimal" ones, based on the spectral test in dimensions 2 to 6. Unfortunately, none of their 400 best multipliers satisfy the inequality (5). We have done a similar search for the best multipliers a, but among those which satisfy (5). The search has been made for a collection of values of m, namely the 50 largest primes smaller than 2^{31}. These multipliers are nearly optimal based on the spectral test and yield generators that are much easier to implement. A summary of the results appears in section 2 of this paper.

Often, the whole sequence of numbers produced by a generator must be split into disjoint subsequences, to produce so called "independent streams". This is usually necessary e.g. for variance reduction purposes. Sometimes, the disjoint subsequences must be split further to make independent replications. In a simulation software environment, all this splitting could be done beforehand, provided that (i) the period of the underlying generator is long enough and (ii) for any positive integer k, there is a fast way to "jump" from state s_i to state s_{i+k} (without generating all intermediate values, of course). The latter is easily done with a MLCG (see Marse and Roberts (1983) and L'Ecuyer (1986)). To obtain longer periods, one could combine two or more pseudo-random number generators. The combination is also an intuitively appealing heuristic, often supported by empirical tests, to improve the statistical behavior of the generators. A convenient combination method should keep property (ii) above. Some methods, like shuffling (Nance and Overstreet (1978)), don't.

Consider a family of l generators where for $j = 1, \ldots, l$, generator j is a MLCG with modulus m_j and multiplier a_j, and evolves according to :

$$s_{j,i} := f_j(s_{j,i-1}) = a_j s_{j,i-1} \text{ MOD } m_j. \qquad (6)$$

We may combine these individual MLCGs to obtain a generator whose state at stage i is denoted by s_i. If s_i is a function of only $s_{1,i}, \ldots, s_{l,i}$, then jumping from s_i to s_{i+k} can be done easily : it suffices to jump from $s_{j,i}$ to $s_{j,i+k}$ for $j = 1, \ldots, l$, i.e. for each individual MLCG, and then make the combination.

The combined 32-bit generator presented in section 3 of this paper is based on this approach, with $l = 2$. Its period exceeds 2.3×10^{18}. It has been submitted to a battery of statistical tests, with highly satisfactory results.

L'Ecuyer (1986) contains a formal presentation of the combination method used here, a more extensive discussion on implementation considerations and on the search for good multipliers, and the results of statistical tests.

2. GOOD MULTIPLIERS

It is well known (see Marsaglia (1968) or Knuth (1981)) that all the k-tuples $P_{i,k} = (U_{i+1}, \ldots, U_{i+k})$ of successive variates generated by a MLCG lie on a set of equidistant parallel hyperplanes in the k-dimensional hypercube $[0,1]^k$. The *spectral test* (see Knuth (1981)) computes the maximal distance $d_k(m,a)$ between any pair of adjacent parallel hyperplanes in dimension k. The smaller that maximum, the better is the generator, but there is a theoretical lower bound $d_k^*(m)$ on $d_k(m,a)$; see Knuth (1981), p. 105.

We applied the spectral test for the 50 largest primes m smaller than $2^{31} - 1$ to find, among all multipliers $a \leq \sqrt{m}$ that are primitive elements modulo m, those that perform well in every dimension k between 2 and 6. More specifically, we found those pairs (m,a) for which the normalized worst case measure

$$M_6(m,a) = \min_{2 \leq k \leq 6} \frac{d_k^*(m)}{d_k(m,a)} \qquad (7)$$

is the largest (the closest to unity). The three best are given in table 1. These are much better than the often recommended pair ($m = 2147483647, a = 16807$) (see Schrage (1979)), for which $M_6(m,a) = .3375$, and nearly as good as the best pair found by Fishman and Moore (1986), namely ($m = 2147483647, a = 742938285$), for which $M_6(m,a) = .8319$. Each of these three MLCGs is easy to implement in any high level language using the technique explained in Bratley, Fox and Schrage (1983), page 201.

Table 1. The three best (m,a) pairs.

m	a	$M_6(m,a)$
2147483399	40692	.8051
2147483563	40014	.7885
2147482811	41546	.7870

3. A COMBINED 32-BIT GENERATOR

We propose a combined generator based on the following recursion :

$$s_{1,i} := (40692 \times s_{1,i-1}) \text{ MOD } 2147483399;$$
$$s_{2,i} := (40014 \times s_{2,i-1}) \text{ MOD } 2147483563;$$
$$s_i := (s_{1,i} + s_{2,i} - 2) \text{ MOD } 2147483562;$$
$$U_i := (s_i + 1)/2147483563;$$

where $s_{1,0}$ and $s_{2,0}$ are appropriate initial seeds. Again, a portable implementation is easily done in any high level language, using the technique suggested above for each of

the two individual components. The validity of the combination follows from Lemma 1 in L'Ecuyer (1986). The two individual MLCGs have periods $p_1 = 2147483398$ and $p_2 = 2147483562$ respectively. The period of the combined generator is the least common multiple of the individual periods, namely $(p_1 \times p_2)/2 \approx 2.30584 \times 10^{18}$. A Pascal code for this generator, and the results of extensive statistical testing, appear in L'Ecuyer (1986). The tests raised no apparent defects.

ACKNOWLEDGMENTS

This work has been supported by NSERC-Canada grant # A5463 and FCAR-Quebec grant # EQ2831. The author thanks prof. Bennett L. Fox for many helpful suggestions.

REFERENCES

Bratley, P., Fox, B. L. and Schrage, L. E. (1983). *A Guide to Simulation*. Springer-Verlag, New York.

Fishman, G. S. and Moore III, L. S. (1986). An Exhaustive Analysis of Multiplicative Congruential Random Number Generators with Modulus $2^{31} - 1$. *SIAM Journal on Scientific and Statistical Computing* **7**, 1, 24–45.

Knuth, D. E. (1981) *The Art of Computer Programming : Seminumerical Algorithms*, vol. 2, second edition. Addison-Wesley.

L'Ecuyer, P. (1986). Efficient and portable combined pseudo-random number generators. Research report no. DIUL-RR-8612, Département d'informatique, Université Laval, Ste-Foy, Québec, Canada.

Marsaglia, G. (1968). Random Numbers Fall Mainly in the Planes. *Proceedings of the National Academy of Sciences of the United States of America* **60**, 25–28.

Marse, K. and Roberts, S. D. (1983). Implementing a Portable FORTRAN Uniform (0,1) Generator. *Simulation* **41**, 4, 135–139.

Modianos, D. T., Scott R. C. and Cornwell, L. W. (1984). Random Number Generation on Microcomputers. *Interfaces* **14**, 2, 81–87.

Nance, R. E. and Overstreet Jr., C. (1978). Some Experimental Observations on the Behavior of Composite Random Number Generators. *Operations Research* **26**, 5, 915–935.

Niederreiter, H. (1978). Quasi-Monte Carlo Methods and Pseudo-random Numbers. *Bulletin of the American Mathematical Society* **84**, 6, 957–1041.

Schrage, L. (1979). A More Portable Fortran Random Number Generator. *ACM Transactions on Mathematical Software* **5**, 2, 132–138.

Wichmann, B. A. and Hill, I. D. (1982). An Efficient and Portable Pseudo-random Number Generator. *Applied Statistics* **31**, 188–190.

AUTHOR'S BIOGRAPHY

PIERRE L'ECUYER is an Adjoint Professor in Computer Science at Laval University, Ste-Foy, Québec, Canada. He received the B.Sc. degree in mathematics in 1972, and was a college teacher in mathematics from 1973 to 1978. He then received the M.Sc. degree in operations research and the Ph.D. degree in computer science, in 1980 and 1983 respectively, both from the University of Montreal. From 1980 to 1983, he was also a research assistant at l'Ecole des Hautes Etudes Commerciales, in Montreal. His research interests are in Markov renewal decision processes, approximation methods in dynamic programming, discrete-event simulation and software engineering. He is a member of ACM, IEEE, ORSA and SCS.

Pierre L'Ecuyer
Département d'informatique
Pavillon Pouliot
Université Laval
Ste-Foy, Qué., Canada
G1K 7P4
(418) 656-3226