UNIFORM RANDOM NUMBER GENERATORS: A REVIEW

Pierre L'Ecuyer

Département d'IRO
Université de Montréal, C.P. 6128, Succ. Centre-Ville
Montréal, H3C 3J7, CANADA

ABSTRACT

This paper summarizes the current state-of-the-art on uniform random number generation for stochastic simulation. It recalls the basic ideas, discusses some linear methods and their theoretical analysis, and provides pointers to further details and to recommended implementations.

1. WHAT IS A GOOD RNG?

Without a good random number generator (RNG), simulation results are often meaningless. And questionable generators are still all over the place, so many experiments rest on shaky foundations. Why this problem was not solved long ago? Because it is not so easy. A so-called RNG actually produces a totally deterministic and periodic sequence of numbers, once its initial state (or seed) is chosen. This is in total contradiction with the assumption of a sequence of independent and identically distributed (i.i.d.) random variables, and there is no clean way to completely reconcile these two opposite aspects. Therefore, everything we do in this context is heuristic. This being said, the heuristic arguments lead to criteria that need theory to be analyzed.

A RNG has a *state* that evolves in a finite state space S, according to a recurrence of the form $s_n = f(s_{n-1})$, $n \geq 1$, where the initial state $s_0 \in S$ is called the *seed*, and $f : S \to S$ is the *transition function*. At step n, the generator outputs $u_n = g(s_n)$, where $g : S \to [0,1]$ is the *output function*. The output sequence of the RNG is thus $\{u_n,\, n \geq 0\}$. The output space could be more general, but we shall assume here that it is the real interval $[0,1]$. Since S is finite, the sequence must be periodic (possibly after some initial transient). Let ρ be the period length. Typically, one has ρ near $|S|$, that is, $\rho \approx 2^b$ if the state is represented over b bits, otherwise there is a waste of computer memory.

We now momentarily forget the deterministic nature of the RNG and view the u_n as random variables. This could make sense if, for example, the seed s_0 and/or some parameters of the functions f and g are chosen at random. Define the null hypothesis $\mathcal{H}_0$: "The u_n are i.i.d. $U(0,1)$ (i.e., uniform over the interval $(0,1)$) random variables". This hypothesis $\mathcal{H}_0$ means that for each n and t, the vector $\boldsymbol{u}_n = (u_n, \ldots, u_{n+t-1})$ is uniformly distributed over the t-dimensional unit hypercube $I^t = [0,1]^t$. Of course, this cannot hold, due to the finiteness of S. But for small t, a discretized version of it could hold, as follows.

Replace the interval $[0,1]$ by the finite set $\mathbb{Z}_m = \{0, 1/m, 2/m, \ldots, (m-1)/m\}$ for some large integer m, and suppose that s_0 is random, uniformly distributed over S, which is assumed to be of cardinality ρ. Consider the hypothesis $\mathcal{H}_0'$: "$\boldsymbol{u}_0$ (and therefore each $\boldsymbol{u}_n$) is uniformly distributed over the set $\mathbb{Z}_m^t$". The hypothesis $\mathcal{H}_0'$ can hold exactly only if ρ is a multiple of m^t. In practice, ρ is often slightly smaller than a multiple of m^t, and $\mathcal{H}_0'$ can hold approximately. For t such that $m^t \gg \rho$, the set $\Psi = \{\boldsymbol{u}_n, 1 \leq n \leq \rho\}$ is only a small fraction of $\mathbb{Z}_m^t$. In this case, which points of $\mathbb{Z}_m^t$ should be in Ψ? Our opinion is that the points of Ψ should be uniformly spread over $\mathbb{Z}_m^t$, for all t up to some reasonably large value determined by our analysis capabilities. This is our main criterion for discriminating among generators and for constructing what we call "good ones".

As a realistic illustration, let $m = 2^{30}$ and $\rho \approx 2^{210}$. Then, $\mathcal{H}_0'$ can hold approximately, if the generator is well designed, for $t \leq 7$. For $t > 7$, the fraction of $\mathbb{Z}_m^t$ that can be covered by Ψ is approximately $2^{-30(t-7)}$. As t increases, this fraction quickly becomes extremely tiny, and nothing can be done against this, aside from increasing ρ.

Note that our reasoning makes sense only if s_0 is random and if ρ is huge. One may argue that the points of Ψ should look like random points over $\mathbb{Z}_m^t$ instead of being too evenly distributed. But if Ψ is viewed as a sample space from which points are taken at random by the generator, without replacement, then a superuniform (i.e., very even) distribution of Ψ is justified.

In order to make this "uniformity of Ψ" criterion effective from a practical viewpoint, a good theoretical understanding of the structure of Ψ is required. This is (essentially) what is meant by *theoretical analysis* of RNGs. After a RNG has been designed and implemented, empirically-minded people will certainly want to apply empirical *statistical tests*. The number of such tests is unlimited and no amount of empirical testing can ever *prove* that a given generator is flawless. However, the fact that well-designed RNGs pass most reasonable statistical tests that are usually applied is somewhat reassuring. It makes less likely the possibility of an implementation error or that some major defect has been overlooked. The design and test of a RNG operates much like the design and test of a new car: Even if the car has been constructed based on the most solid theoretical analysis, there will always be people who will want to test it on the road, and for good reasons. Ideally, the statistical tests should be selected in close relation with the target application, that is, be based on a test statistic T that closely mimic the random variable of interest, and where a good approximation of the distribution of T under $\mathcal{H}_0$ is available. But this ideal is impractical, especially when designing and testing generators for general purpose software packages. In principle, for any RNG, since $\rho < \infty$, one can easily construct a test that this RNG will fail arbitrarily badly. But if ρ is large enough, such a test may be impossible to run in "practical" time.

A long period, good structure of the points, and passing statistical tests, are not the only required qualities. Simulations involving billions of random numbers are increasingly frequent, and the generator's speed is often a critical factor, regardless of the available computing power. The size of required memory may become important when virtual generators (or substreams) are maintained in parallel. This is required, for example, for proper implementation of certain variance reduction techniques. *Portability* means that the generator can be implemented easily in a standard high-level language, and produce the same sequence with a wide range of compilers and computers. *Repeatability*, i.e., being able to reproduce the same sequence all over again, is important for program verification and for variance reduction. This is a major advantage of RNGs over random numbers generated by physical devices. *Jumping ahead* means the ability to quickly compute, given the current state s_n, the state $s_{n+\nu}$ for any large ν. This is useful for breaking up the sequence into long disjoint substreams making up virtual generators. The packages of L'Ecuyer and Côté (1991) and L'Ecuyer and Andres (1997) implement tools to manipulate such substreams.

In the remainder of this paper, we survey some of the recent literature on RNGs. The coverage is certainly not exhaustive and is tainted by the author's own interests and opinions. I apologize in advance to those whose interesting papers have not been cited. Further pointers on other aspects or on other RNG classes can be found from the references given here. More extensive coverages of RNGs are given in Eichenauer-Herrmann (1995), Fishman (1996), Hellekalek (1995), Knuth (1981), L'Ecuyer (1990), L'Ecuyer (1994), L'Ecuyer (1997c), Niederreiter (1992), Niederreiter (1995b), Ripley (1990), and Tezuka (1995), among others. The next section describes RNGs based on linear recurrences in modular integer arithmetic. In Section 4, we survey some developments regarding the combination of such linear-type generators. Section 3 discusses their lattice structure and equidistribution properties. Section 5 mentions classes of nonlinear generators, Section 6 summarizes some work on empirical testing, and Section 7 points to recommended implementations.

2. LINEAR RECURRENCES

A multiple recursive generator (MRG) is defined by the recurrence

$$\begin{aligned} x_n &= (a_1 x_{n-1} + \cdots + a_k x_{n-k}) \bmod m; & (1)\\ u_n &= x_n/m. & (2) \end{aligned}$$

The *modulus* m and *order* k are positive integers, the *coefficients* a_i belong to $\mathbb{Z}_m = \{0, 1, \ldots, m-1\}$, and the *state* at step n is $s_n = (x_{n-k+1}, \ldots, x_n)$. For prime m and properly chosen a_i's, the sequence has (maximal) period length $\rho = m^k - 1$. This can be achieved with only two non-zero a_i's (Knuth 1981; L'Ecuyer 1997c), i.e.,

$$x_n = (a_r x_{n-r} + a_k x_{n-k}) \bmod m. \qquad (3)$$

This slim version makes the implementation faster. The classical linear congruential generator (LCG) corresponds to the case $k = 1$.

Taking $m = 2^e$ for $e > 1$ (a power-of-two modulus) also makes things easier from the implementation viewpoint, but leads to a much shorter period for $k > 1$ and to several important structural defects (L'Ecuyer 1990; L'Ecuyer 1997c). This should be avoided.

One approach for using a power-of-two modulus while keeping a long period and the potential for good properties is the linear recurrence with a carry

(Couture and L'Ecuyer 1995; Marsaglia 1994):

$$\begin{aligned} x_n &= (a_1 x_{n-1} + \cdots + a_k x_{n-k} + c_{n-1}) \bmod b, \\ c_n &= (a_1 x_{n-1} + \cdots + a_k x_{n-k} + c_{n-1}) \operatorname{div} b, \\ u_n &= x_n / b. \end{aligned}$$

where "div" is the integer division, b can be a power of two, and c_n is called the *carry* at step n. This so-called *Multiply-with-Carry* (MWC) generator turns out to be approximately equivalent (with a difference of less than $1/b$ on the u_n's) to an LCG with modulus $m = \sum_{\ell=0}^{k} a_\ell b^\ell$, where $a_0 = -1$, and multiplier a equal to the inverse of b modulo m. These MWC generators can thus be analyzed much in the same way as LCGs from the structural viewpoint, and offer a promising avenue for constructing fast and robust generators. Couture and L'Ecuyer (1997) give an in-depth theoretical analysis of their properties.

In (2), each number is a multiple of $1/m$. To reduce this discretization error, one may construct each output value from more than one term of the recurrence (1). For example,

$$u_n = \sum_{j=1}^{L} x_{ns+j-1} m^{-j}, \tag{4}$$

where s and $L \le k$ are positive integers. If (1) has period ρ and $\gcd(\rho, s) = 1$, then (4) also has period ρ. The digital expansion (4) permits one to take smaller values of m. An important special case is $m = 2$. Then, u_n is constructed from L successive bits of the binary sequence (1), with spacings of $s - L$ bits between the blocks. The resulting RNG is called a *linear feedback shift register* (LFSR) or *Tausworthe* generator (Knuth 1981; Niederreiter 1992; Tausworthe 1965). Its implementation is discussed by Bratley, Fox, and Schrage (1987), Fishman (1996), L'Ecuyer (1996b), and Tezuka (1995).

Another approach is to have L copies of the recurrence (1) running in parallel, with different initial values, and use one copy for each digit of the fractional expansion of u_n. If $\{x_{j,n}\}$ denotes the jth copy and if $x_{j,n} = x_{n+d_j}$ for all j and n, then

$$u_n = \sum_{j=1}^{L} x_{j,n} m^{-j} = \sum_{j=1}^{L} x_{n+d_j} m^{-j}. \tag{5}$$

If $d_j = (j-1)d$ for some integer d and if $\gcd(d, \rho) = 1$, then $n + d_j = n + (j-1)d = (ns + j - 1)d$, so (5) becomes equivalent to (4) if we replace $\{x_n\}$ by $\{y_n = x_{nd}\}$, which can be accomplished by changing the coefficients of (1) appropriately. When $m = 2$ and (3) is used, the generator (5) is called a *generalized feedback shift register* (GFSR) generator (Fushimi and Tezuka 1983; Fushimi 1989). Denoting $X_n = (x_{1,n}, \ldots, x_{k,n})$, we have

$$X_n = X_{n-r} \oplus X_{n-k}, \tag{6}$$

where $\oplus$ denotes the bitwise exclusive-or, and this provides for an extremely fast implementation of the GFSR.

A modification of the GFSR is the *lagged-Fibonacci* generator, where $\oplus$ can be replaced by any arithmetic or logical operation, such as $+$, $-$, etc. For example, the *additive* generator (Knuth 1981):

$$X_n = (X_{n-r} + X_{n-k}) \bmod m, \tag{7}$$

where $m = 2^L$, is proposed on many systems, including UNIX. It is a special case of the MRG with a power-of-two modulus. Slight variations of it are the *add-with-carry* (AWC) and *subtract-with-borrow* (SWB), proposed by Marsaglia and Zaman (1991), which are in fact special cases of the MWC generator with only two nonzero coefficients a_j, both equal to ± 1. The modification permits to increase the period length from 2^{k+L-1} up to approximately 2^{kL}. However, all these generators have gross structural defects. For example, for the additive one, all triples of the form $(u_n, u_{n+k-r}, u_{n+k})$, $n \ge 0$, lie in only two planes in the three-dimensional unit cube, and for the AWC/SWB generators, the same happens with the triples (u_n, u_{n+r}, u_{n+k}) (see L'Ecuyer 1997a).

Other (better) types of modifications of (6), which maintain the speed but increase the period length from $2^k - 1$ to $2^{kL} - 1$, are the *twisted GFSR* proposed by Matsumoto and Kurita (1994) and Matsumoto and Nishimura (1997). The *multiple recursive matrix method* of Niederreiter 1995a provides a general framework that encompasses many of these modifications and variants.

3. COMBINED GENERATORS

Combination is a way of increasing the period length and improving the statistical properties of generators (Knuth 1981; L'Ecuyer and Côté 1991; L'Ecuyer 1994; L'Ecuyer 1996b; L'Ecuyer 1996a; Marsaglia 1985; Tezuka 1995; Wang and Compagner 1993). However, blind combinations or mixtures give no guarantee of improvement. It is important to understand the structure of the resulting generator. Among the classes of combined generators that have been successfully analyzed theoretically, one finds the combined MRGs and the combined Tausworthe/GFSR generators.

Take J MRGs of order k, the form

$$x_{j,n} = (a_{j,1} x_{j,n-1} + \cdots + a_{j,k} x_{j,n-k}) \bmod m_j, \tag{8}$$

with distinct prime moduli and maximal period $\rho_j = m_j^k - 1$. Let $\delta_1, \ldots, \delta_J$ be arbitrary integers such that $\gcd(\delta_j, m_j) = 1$ for each j and define the combinations

$$z_n = \left(\sum_{j=1}^{J} \delta_j x_{j,n} \right) \bmod m_1; \qquad \tilde{u}_n = z_n / m_1 \quad (9)$$

and

$$w_n = \left(\sum_{j=1}^{J} \frac{\delta_j x_{j,n}}{m_j} \right) \bmod 1. \quad (10)$$

Let $m = \prod_{j=1}^{J} m_j$, n_j be the inverse of m/m_j modulo m_j, and

$$a_i = \left(\sum_{j=1}^{J} \frac{a_{j,i} n_j m}{m_j} \right) \bmod m$$

for $i = 1, \ldots, k$. Then, the periodic sequence $\{w_n\}$ is exactly the same as the sequence $\{u_n\}$ given by the MRG (1–2), and the sequence $\{\tilde{u}_n\}$ is almost the same in the sense that $|u_n - \tilde{u}_n|$ is uniformly very close to zero when the m_j's are close to each other. See L'Ecuyer and Tezuka (1991), L'Ecuyer (1996b) for details and proofs. Therefore, these combinations can be seen as convenient ways of implementing an MRG with a large composite modulus. The maximal period for the combination is $\rho = \mathrm{lcm}(\rho_1, \ldots, \rho_J) = 2^{1-J} \prod_{j=1}^{J} \rho_j$. The main advantages of these combinations are (a) an increased period length, while all computations are performed modulo relatively small integers m_j; (b) the fact that (1) can have many nonzero coefficients even if the recurrence (8) of each component has few of them.

LFSR and GFSR generators whose recurrence is based on a trinomial, like in (3), are fast but have important statistical defects (Matsumoto and Kurita 1994; Matsumoto and Kurita 1996; Compagner 1991). Again, this may be resolved by combination. Tezuka and L'Ecuyer (1991) and Wang and Compagner (1993) propose to combine J "easy-to-implement" LFSR generators, the jth producing a sequence $\{u_{j,n},\ n \geq 0\}$, by a bitwise exclusive-or of the $u_{j,n}$'s: $u_n = u_{1,n} \oplus \cdots \oplus u_{J,n}$. The resulting generator is equivalent to a LFSR generator whose recurrence has a reducible characteristic polynomial equal to the product of the characteristic polynomials of the individual components. The period could reach $\prod_{j=1}^{J}(2^{k_j} - 1)$, where k_j is the the degree of the characteristic polynomial of component j, provided that the k_j are pairwise coprime. GFSR and twisted GFSR generators can also be combined in a similar way. Such combinations can be viewed as practical ways of implementing recurrences with "good" characteristic polynomials, with many nonzero coefficients.

4. LATTICES, EQUIDISTRIBUTION, AND DISCREPANCY

The set

$$\begin{aligned} T_t \quad = \quad & \{\boldsymbol{u}_n = (u_n, \ldots, u_{n+t-1}) \mid n \geq 0, \\ & (x_0, \ldots, x_{k-1}) \in \mathbb{Z}_m^k\} \end{aligned}$$

of all t-tuples of successive values produced by (1–2), from all possible initial states, is the intersection of a lattice L_t with the unit hypercube $[0,1)^t$. This means that the points of T_t lie in a limited number of equidistant parallel hyperplanes (Knuth 1981). For the points to be evenly distributed over the entire period, the distance d_t between those successive hyperplanes should be small. One can define the figure of merit

$$M_T = \min_{t \leq T} \frac{d_t^*}{d_t}$$

for any positive integer T, where $d_t^* = \gamma_t m^{-k/t}$ is an absolute lower bound on the smallest possible d_t, and γ_t is the so-called Hermite constant. The constant γ_t is known exactly only for $t \leq 8$ but approximations of it are available for $t > 8$ (L'Ecuyer 1997d). An M_T close to 1 means a superuniform distribution over the full period. L'Ecuyer (1997d) has computed tables of LCGs with good figures of merit M_8, M_{16}, and M_{32}, for m equal to the largest prime less than 2^e, for several values of e from 8 to 64. L'Ecuyer (1997b) provides combined MRGs which are good with respect to these figures of merit, together with computer implementations in C.

Computing d_t, called the *spectral test*, amounts to solving a quadratic optimization problem with integer variables. The programs of L'Ecuyer and Couture (1997) compute d_t up to around 40 or more, and can handle large moduli m. Couture and L'Ecuyer (1996) explain how to analyze the (shifted) lattice associated with the set of recurrent states (only), for a given initial state, when that set generates a strict sublattice of L_t.

One may want to consider vectors of *non-successive* values produced by the generator: fix a set of nonnegative integers $I = \{i_1, i_2, \cdots, i_t\}$, put

$$\begin{aligned} T_t(I) \quad = \quad & \{(u_{i_1+n}, \ldots, u_{i_t+n}) \mid n \geq 0, \\ & s_0 = (x_0, \ldots, x_{k-1}) \in \mathbb{Z}_m^k\}, \end{aligned}$$

and let $d_t(I)$ be the distance between successive hyperplanes in the lattice generated by $T_t(I)$ and $\mathbb{Z}_m^k/m$. For $t > k$, one has the lower bound

$$d_t(I) \quad \geq \quad \left(\sum_{j=0}^{k} a_j^2 \right)^{-1/2}$$

if $\{j \geq 0 \mid a_{k-j} \neq 0\} \subseteq I$. Couture and L'Ecuyer (1994), Couture and L'Ecuyer (1996) and L'Ecuyer and Couture (1997) discuss how to compute $d_t(I)$ in different contexts. The fact that for the AWC/SWB and additive or subtractive lagged-Fibonacci generators, the set $T_3(I)$ (aside from the zero vector) is contained in only two planes, for a certain set I as mentioned previously, is a special case of their results. They also show, for example, that the two combined generators proposed in Marsaglia, Narasimhan, and Zaman (1990) and Marsaglia, Zaman, and Tsang (1990) can be approximated by linear congruential generators for which $d_6(I) \geq 1/\sqrt{6} \approx 0.408$ for certain sets I. For simulations dealing with random points in space, those bad structures could have a dramatic effect (Ferrenberg, Landau, and Wong 1992; L'Ecuyer 1992).

A different way of looking at uniformity is to analyze the equidistribution of the point set T_t produced by the generator, as follows (Couture, L'Ecuyer, and Tezuka 1993; L'Ecuyer 1994; L'Ecuyer 1996b; Tezuka 1995). This is typically applied to LFSR, GFSR, and other similar generators. Partition the hypercube $[0,1)^t$ into $2^{t\ell}$ cubic cells of equal size. If each cell contains the same number of points of T_t, the sequence (or the set T_t) is called (t,ℓ)-*equidistributed*. Assuming that T_t has cardinality 2^k, this is possible only for $\ell \leq \lfloor k/t \rfloor$. When T_t is $(t, \lfloor k/t \rfloor)$-equidistributed for $t = 1, \ldots, k$, it is called *maximally equidistributed* (ME). Several ME combined LFSR generators are listed in L'Ecuyer (1996b), L'Ecuyer (1997e), together with fast computer implementations.

Another measure of uniformity of a set of N points in t dimensions is through the notion of *discrepancy* (see Niederreiter 1992 for details). When the aim is to imitate i.i.d. uniform random variables, the set of points that are used during a simulation should have a discrepancy comparable to what is normally expected from a set of random points, which is roughly $O(N^{-1/2})$. Niederreiter (1992) gives general discrepancy bounds for several classes of generators, mostly for $N = \rho$. However, no efficient algorithm is available for computing the discrepancy exactly, except for certain special cases.

5. NONLINEAR GENERATORS

Some argue that since the structure of linear sequences is too regular, *nonlinear* generators should be used instead (Eichenauer-Herrmann 1995; Hellekalek 1995; Niederreiter 1995b; Niederreiter 1992). Nonlinearity can be introduced by either (a) using a linear transition function f with a nonlinear output function g, (b) using a nonlinear recurrence.

A simple example of (a) is the *explicit inversive generator* of Eichenauer-Herrmann (1993): let $x_n = an + c$, for $n \geq 0$, where $a \neq 0$ and c are in $\mathbb{Z}_m$, m prime, $z_n = x_n^{-1} = (an + c)^{m-2} \bmod m$, and $u_n = z_n/m$. The period is $\rho = m$ and it can be shown (Niederreiter 1994a) that every hyperplane in $\mathbb{R}^t$ contains at most t points from the set T_t. Other variants of inversive nonlinear generators can be found in Eichenauer-Herrmann (1992), Eichenauer-Herrmann (1995), L'Ecuyer (1997c), Niederreiter (1992), Niederreiter (1994b), Niederreiter (1995b), and the references given there. These nonlinear generators tend to avoid the planes and have the right asymptotic orders of magnitude for their discrepancies. Those based on power-of-two moduli, however, have defects similar to the LCGs and MRGs with power-of-two moduli (see, e.g., L'Ecuyer, Compagner, and Cordeau 1996).

Other nonlinear generators have also been proposed in the field of cryptology. For example, the BBS generator, proposed by Blum, Blum, and Schub (1986), evolves according to

$$x_n = x_{n-1}^2 \bmod m,$$

where m is the product of two distinct k-bit primes, both congruent to 3 modulo 4, and $\gcd(x_0, m) = 1$. At each step, the generator outputs the last ν bits of x_n, where ν is in $O(\log(k))$. Under the assumption that factoring is hard, and that m and x_0 are chosen "randomly" in a specific way, it is proven that no polynomial-time (in k) statistical test can distinguish (in some specific sense) the output of a BBS generator from a sequence of i.i.d. uniforms. This means that for large enough k, the generator should behave very nicely from a statistical point of view. But nobody knows for sure how large is large enough. For more about cryptographic-type generators, see Lagarias (1993) and L'Ecuyer and Proulx (1989). These generators are rather slow compared with those usually employed for simulation.

6. EMPIRICAL TESTS

Applying empirical statistical tests to RNGs is a highly heuristic affair. A test is defined by a statistic T, function of a finite number of output values, and whose distribution under $\mathcal{H}_0$ can be well approximated. The test tries to find empirical evidence against H_0. If such evidence can be found in reasonable time, then $\mathcal{H}_0$ is rejected. This happens when the p-value

$$p = P[T > t_1 \mid H_0],$$

where t_1 is the value taken by T, is too close to either 0 or 1. One can also construct a *two-levels* test, by comparing the empirical distribution of (say) N "independent" copies of T to the theoretical distribution of T.

When testing RNGs, an arbitrarily large sample size can be taken to increase the power of the test and come up with a clear decision when suspicious p-values occur. One can reject $\mathcal{H}_0$ is $p < 10^{-8}$, for example, and increase the sample size or repeat the test with other segments of the sequence when p is suspicious ($p = .005$, for example). In most cases, clear evidence against H_0 will quickly show up or suspicion will disappear. Note that when $\mathcal{H}_0$ is not rejected for a given test, this proves nothing. It may well be rejected by the next test. It may be a good idea to run important simulations twice using random number generators of totally different families (e.g., linear and nonlinear).

Knuth (1981) describes a set of tests considered as "standard" for testing RNGs. Marsaglia (1996) has proposed a battery of tests called DIEHARD, which he considers more stringent than the classical tests in Knuth (1981). A software package that contains most of the tests proposed so far, as well as several classes of generators implemented in generic form, is under development (L'Ecuyer 1996c). Examples of other statistical tests applied to random number generators can be found in Ferrenberg, Landau, and Wong (1992), L'Ecuyer (1992), L'Ecuyer (1994), L'Ecuyer, Cordeau, and Simard (1997), L'Ecuyer, Compagner, and Cordeau (1996), L'Ecuyer (1997f), Leeb and Wegenkittl (1997), Stephens (1986). After extensive experiments with these tests, the following (among other things) has been observed: (i) All RNGs with period lengths less than 2^{32} (say), especially the linear ones, fail several tests with relatively small sample sizes; (ii) RNGs with power-of-two moduli are worst-behaved than those with prime moduli, especially for their low-order bits; (iii) LFSRs and GFSRs based on primitive trinomials, or lagged-Fibonacci and AWC/SWB generators, whose structure is too simplistic, also fail some tests spectacularly; (iv) Combined generators with long periods and good or fair structural properties (lattice structure or equidistribution) pass the tests quite well; (v) When a large fraction of the period is used, nonlinear inversive generators with prime modulus do better than the LCGs.

7. IMPLEMENTATIONS

Computer implementations of generators with good theoretical support, well tested, and reasonably efficient, include those of the combined MRGs of L'Ecuyer (1996a), L'Ecuyer and Andres (1997) and L'Ecuyer (1997b), the combined Tausworthe generators of L'Ecuyer (1996b) and L'Ecuyer (1997e), the twisted GFSRs of Matsumoto and Kurita (1994) and Matsumoto and Nishimura (1997), and perhaps RANLUX in James (1994), with the highest luxury level. More references and implementations can be found from the URL pages: `http://www.iro.umontreal.ca/~lecuyer` and `http://random.mat.sbg.ac.at` on the internet. Of course, none of these RNGs is totally guaranteed against all possible defects. Such guarantee is impossible.

ACKNOWLEDGMENTS

This work has been supported by NSERC-Canada grants # ODGP0110050 and # SMF0169893, and FCAR-Québec grant # 93ER1654.

REFERENCES

Blum, L., M. Blum., and M. Schub. 1986. A simple unpredictable pseudo-random number generator. *SIAM Journal on Computing*, 15(2):364–383.

Bratley, P., B. L. Fox., and L. E. Schrage. 1987. *A Guide to Simulation*. Second ed. New York: Springer-Verlag.

Compagner, A. 1991. The hierarchy of correlations in random binary sequences. *Journal of Statistical Physics*, 63:883–896.

Couture, R., and P. L'Ecuyer. 1994. On the lattice structure of certain linear congruential sequences related to AWC/SWB generators. *Mathematics of Computation*, 62(206):798–808.

Couture, R., and P. L'Ecuyer. 1995. Linear recurrences with carry as random number generators. In *Proceedings of the 1995 Winter Simulation Conference*, 263–267.

Couture, R., and P. L'Ecuyer. 1996. Orbits and lattices for linear random number generators with composite moduli. *Mathematics of Computation*, 65(213):189–201.

Couture, R., and P. L'Ecuyer. 1997. Distribution properties of multiply-with-carry random number generators. *Mathematics of Computation*, 66(218): 591–607.

Couture, R., P. L'Ecuyer., and S. Tezuka. 1993. On the distribution of k-dimensional vectors for simple and combined Tausworthe sequences. *Mathematics of Computation*, 60(202):749–761, S11–S16.

Eichenauer-Herrmann, J. 1992. Inversive congruential pseudorandom numbers: A tutorial. *International Statistical Reviews*, 60:167–176.

Eichenauer-Herrmann, J. 1993. Statistical independence of a new class of inversive congruential pseudorandom numbers. *Mathematics of Computation*, 60:375–384.

Eichenauer-Herrmann, J. 1995. Pseudorandom number generation by nonlinear methods. *International Statistical Reviews*, 63:247–255.

Ferrenberg, A. M., D. P. Landau., and Y. J. Wong. 1992. Monte Carlo simulations: Hidden errors from "good" random number generators. *Physical Review Letters*, 69(23):3382–3384.

Fishman, G. S. 1996. *Monte Carlo: Concepts, Algorithms, and Applications.* Springer Series in Operations Research, New York: Springer-Verlag.

Fushimi, M. 1989. An equivalence relation between Tausworthe and GFSR sequences and applications. *Applied Mathematics Letters*, 2(2):135–137.

Fushimi, M., and S. Tezuka. 1983. The k-distribution of generalized feedback shift register pseudorandom numbers. *Communications of the ACM*, 26(7):516–523.

Hellekalek, P. 1995. Inversive pseudorandom number generators: Concepts, results, and links. In *Proceedings of the 1995 Winter Simulation Conference*, ed. C. Alexopoulos, K. Kang, W. R. Lilegdon, and D. Goldsman, 255–262. IEEE Press.

James, F. 1994. RANLUX: A Fortran implementation of the high-quality pseudorandom number generator of Lüscher. *Computer Physics Communications*, 79:111–114.

Knuth, D. E. 1981. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms.* Second ed. Reading, Mass.: Addison-Wesley.

Lagarias, J. C. 1993. Pseudorandom numbers. *Statistical Science*, 8(1):31–39.

L'Ecuyer, P. 1990. Random numbers for simulation. *Communications of the ACM*, 33(10):85–97.

L'Ecuyer, P. 1992. Testing random number generators. In *Proceedings of the 1992 Winter Simulation Conference*, 305–313. IEEE Press.

L'Ecuyer, P. 1994. Uniform random number generation. *Annals of Operations Research*, 53:77–120.

L'Ecuyer, P. 1996a. Combined multiple recursive generators. *Operations Research*, 44(5):816–822.

L'Ecuyer, P. 1996b. Maximally equidistributed combined Tausworthe generators. *Mathematics of Computation*, 65(213):203–213.

L'Ecuyer, P. 1996c. TestU01: Un logiciel pour appliquer des tests statistiques à des générateurs de valeurs aléatoires. In preparation.

L'Ecuyer, P. 1997a. Bad lattice structures for vectors of non-successive values produced by some linear recurrences. *INFORMS Journal on Computing*, 9(1):57–60.

L'Ecuyer, P. 1997b. Good parameter sets for combined multiple recursive random number generators. Manuscript.

L'Ecuyer, P. 1997c. Random number generation. In *Handbook on Simulation*, ed. J. Banks. Wiley. To appear. Also GERAD technical report number G-96-38.

L'Ecuyer, P. 1997d. A table of linear congruential generators of different sizes and good lattice structure. Manuscript.

L'Ecuyer, P. 1997e. Tables of maximally equidistributed combined LFSR generators. Manuscript.

L'Ecuyer, P. 1997f. Tests based on sum-functions of spacings for uniform random numbers. *Journal of Statistical Computation and Simulation.* To appear.

L'Ecuyer, P., and T. H. Andres. 1997. A random number generator based on the combination of four LCGs. *Mathematics and Computers in Simulation.* To appear.

L'Ecuyer, P., A. Compagner., and J.-F. Cordeau. 1996. Entropy-based tests for random number generators. Submitted.

L'Ecuyer, P., J.-F. Cordeau., and R. Simard. 1997. Close-point spatial tests for random number generators. Submitted.

L'Ecuyer, P., and S. Côté. 1991. Implementing a random number package with splitting facilities. *ACM Transactions on Mathematical Software*, 17(1):98–111.

L'Ecuyer, P., and R. Couture. 1997. An implementation of the lattice and spectral tests for multiple recursive linear random number generators. *INFORMS Journal on Computing*, 9(2). To appear.

L'Ecuyer, P., and R. Proulx. 1989. About polynomial-time "unpredictable" generators. In *Proceedings of the 1989 Winter Simulation Conference*, 467–476. IEEE Press.

L'Ecuyer, P., and S. Tezuka. 1991. Structural properties for two classes of combined random number generators. *Mathematics of Computation*, 57(196):735–746.

Leeb, H., and S. Wegenkittl. 1997. Inversive and linear congruential pseudorandom number generators in selected empirical tests. *ACM Transactions on Modeling and Computer Simulation.* To appear.

Marsaglia, G. 1985. A current view of random number generators. In *Computer Science and Statistics, Sixteenth Symposium on the Interface*, 3–10, North-Holland, Amsterdam. Elsevier Science Publishers.

Marsaglia, G. 1994. Yet another rng. Posted to the electronic billboard `sci.stat.math`, August 1.

Marsaglia, G. 1996. Diehard: A battery of tests of randomness. Available on the internet at `http://stat.fsu.edu/~geo/diehard.html`.

Marsaglia, G., B. Narasimhan., and A. Zaman. 1990. A random number generator for PC's. *Computer Physics Communications*, 60:345–349.

Marsaglia, G., and A. Zaman. 1991. A new class of random number generators. *The Annals of*

Applied Probability, 1:462–480.

Marsaglia, G., A. Zaman., and W. W. Tsang. 1990. Towards a universal random number generator. *Statistics and Probability Letters*, 8:35–39.

Matsumoto, M., and Y. Kurita. 1994. Twisted GFSR generators II. *ACM Transactions on Modeling and Computer Simulation*, 4(3):254–266.

Matsumoto, M., and Y. Kurita. 1996. Strong deviations from randomness in m-sequences based on trinomials. *ACM Transactions on Modeling and Computer Simulation*, 6(2):99–106.

Matsumoto, M., and T. Nishimura. 1997. Mersenne twister: A 623-dimensionally equidistributed uniform pseudorandom number generator. Submitted.

Niederreiter, H. 1992. *Random Number Generation and Quasi-Monte Carlo Methods*. volume 63 of *SIAM CBMS-NSF Regional Conference Series in Applied Mathematics*. Philadelphia: SIAM.

Niederreiter, H. 1994a. On a new class of pseudorandom numbers for simulation methods. *Journal of Computational and Applied Mathematics*, 56:159–167.

Niederreiter, H. 1994b. Pseudorandom vector generation by the inversive method. *ACM Transactions on Modeling and Computer Simulation*, 4(2):191–212.

Niederreiter, H. 1995a. The multiple-recursive matrix method for pseudorandom number generation. *Finite Fields and their Applications*, 1:3–30.

Niederreiter, H. 1995b. New developments in uniform pseudorandom number and vector generation. In *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*, ed. H. Niederreiter and P. J.-S. Shiue, number 106 in Lecture Notes in Statistics, 87–120. Springer-Verlag.

Ripley, B. D. 1990. Thoughts on pseudorandom number generators. *Journal of Computational and Applied Mathematics*, 31:153–163.

Stephens, M. S. 1986. Tests for the uniform distribution. In *Goodness-of-Fit Techniques*, ed. R. B. D'Agostino and M. S. Stephens, 331–366. Marcel Dekker, New York and Basel.

Tausworthe, R. C. 1965. Random numbers generated by linear recurrence modulo two. *Mathematics of Computation*, 19:201–209.

Tezuka, S. 1995. *Uniform Random Numbers: Theory and Practice*. Norwell, Mass.: Kluwer Academic Publishers.

Tezuka, S., and P. L'Ecuyer. 1991. Efficient and portable combined Tausworthe random number generators. *ACM Transactions on Modeling and Computer Simulation*, 1(2):99–112.

Wang, D., and A. Compagner. 1993. On the use of reducible polynomials as random number generators. *Mathematics of Computation*, 60:363–374.

AUTHOR BIOGRAPHY

PIERRE L'ECUYER is a professor in the "Département d'Informatique et de Recherche Opérationnelle", at the University of Montreal. He received a Ph.D. in operations research in 1983, from the University of Montréal. From 1983 to 1990, he was with the Computer Science Department, at Laval University, Quebec. He obtained the *E. W. R. Steacie* grant from the Natural Sciences and Engineering Research Council of Canada for the period 1995–97. His main research interests are random number generation, efficiency improvement via variance reduction, sensitivity analysis and optimization of discrete-event stochastic systems, and discrete-event simulation in general. He is the Departmental Editor for the Simulation Department of *Management Science* and an Area Editor for the *ACM Transactions on Modeling and Computer Simulation*. Check his web page at: `http://www.iro.umontreal.ca/~lecuyer`.