TESTING RANDOM NUMBER GENERATORS

Pierre L'Ecuyer

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

ABSTRACT

So-called Random number generators on computers are deterministic functions producing a sequence of numbers which should *mimic* a sample of i.i.d. $U(0,1)$ random variables. Two classes of tests are commonly applied to such generators. Firstly, the theoretical tests, which look at the intrinsic structure of the generator to derive behavioral properties of the sequence of points, usually over the whole period. These theoretical tests are specific to each class of generators and include the lattice and spectral tests, discrepancy bounds calculations, period length calculations, etc.. Secondly, the empirical goodness-of-fit tests, which try finding statistical evidence against the null hypothesis: "the sequence is a sample from i.i.d. $U(0,1)$ random variables". In this paper, we survey the main techniques from both classes, discuss their philosophy, and look at some of the most recent developments in that area.

1. INTRODUCTION

Testing randomness, when speaking about (pseudo)-random number generators on computers, is a rather fuzzy concept. We already know, indeed, that none of these generators is truly random in the classical sense. According to Kolmogorov, an infinite sequence of bits is *random* if it cannot be described by a sequence shorter than itself. Obviously, efficiency considerations preclude using that kind of sequence for simulation applications. For a thorough discussion of "what is randomness", with appropriate references, see §3.5 of Knuth (1981).

The *random number generators* that are used in practice produce in fact a deterministic, periodic, sequence of numbers which should *look* sufficiently random for the target application. Typically, they produce numbers between 0 and 1, which should look as if they were a sample of i.i.d. variates from the $U(0,1)$ distribution (the uniform distribution between 0 and 1).

To fix ideas, let us adopt the following definition, as in L'Ecuyer (1990). A *generator* is defined by a finite set of *states* S, a *transition* function $f : S \to S$, a probability distribution μ over S for the initial state, a set U of *output* symbols, and an output function $g : S \to U$. The initial state s_0 of the generator, called the *seed*, is selected randomly from S according to μ. Often, μ is degenerate, which means that even s_0 is chosen deterministically. The generator evolves according to $s_n := f(s_{n-1})$ and output symbols are produced at each step by $u_n := g(s_n)$. Since S is finite, U will also be finite. If the aim of the generator is to produce (approximations of) $U(0,1)$ random variates, then U will be a finite set of real numbers between 0 and 1. The smallest positive integer ρ such that $s_{\rho+n} = s_n$ for all $n \geq \nu$ for some $\nu \geq 0$ is called the *period* of the sequence (or of the generator).

How do we decide whether the numbers are sufficiently random-looking ? One can apply a set of empirical *statistical tests*. If the generator passes all the tests, that proves nothing formally, but improves confidence in the simulation results that could be obtained by using that generator. Some "standard" statistical tests for random number generators are described in Dudewicz and Ralley (1981), Knuth (1981), and Marsaglia (1985). Besides the empirical tests, most generators can also be analyzed *theoretically*. For example, one can in some cases compute bounds on the serial correlation, bounds on the discrepancy, or characterize the geometrical behavior of the set of all t-dimensional vectors formed by taking t successive values produced by the generator, over its full period.

In this paper, we survey part of the developments in techniques for testing random number generators. Section 2 discusses theoretical tests, while Section 3 discusses the empirical statistical tests. In Section 4, we apply a set of statistical tests to some well known or recently proposed generators. Practical implications are discussed in the Conclusion.

2. THEORETICAL TESTS

2.1. Period Length

Clearly, the period of a generator cannot exceed the cardinality of its state space S. For optimal memory use, it should be close to that cardinality. So, if b bits are required to represent the state, the period will be close to 2^b. Maximal period linear congruential generators (LCGs), in scalar or matrix form, as well as Tausworthe generators, inversive non-linear generators, many kinds of combined generators, etc., have periods equal (or very close) to 2^b for a b-bit state, if the parameters are chosen appropriately (Knuth 1981, L'Ecuyer 1990). Generalized Feedback Shift Register (GFSR) generators (Lewis and Payne 1973, Fushimi and Tezuka 1983) do not have that property and can be seen as wasteful of memory. But Matsumoto and Kurita (1992) have recently proposed a new variant, called Twisted GFSR (TGFSR), which is practically as fast as GFSR while making optimal use of their memory.

Modern computer simulations, because of faster computers, are getting increasingly ambitious, and require more and more random numbers. Any generator must therefore have a very long period before deserving any further consideration for general use. In my view, standard LCGs with modulo (and period) near 2^{31}, which are still recommended in most simulation books, should be discarded because their period is too short (one can exhaust the period in a few minutes with a good computer). I would say that anything less than 2^{50} for the period is too low. In fact, with the latest developments in random number generation, there is no reason for not taking much longer periods than that, e.g., over 2^{200}. Of course, no generator should be used for any serious purpose if its period (or, at least, a lower bound on it) is unknown.

2.2. Lattice and Spectral Tests

It is well known that the vectors of successive values produced by a LCG, in any given dimension, have a lattice structure (Knuth 1981, Grothe 1988, L'Ecuyer 1990, Ripley 1983). This was first observed by Coveyou and MacPherson (1967), who also gave an algorithm for computing the distance between hyperplanes and called it the *spectral test*. Marsaglia (1968) later discussed the geometrical interpretation more explicitly. LCG's in matrix form, including multiple recursive generators, also possess that lattice property.

More precisely, let m be a positive integer (the *modulus*), A be a $k \times k$ matrix (the *multiplier*), while C and X_n, $n \geq 0$, are k-dimensional (column) vectors with components in $\{0, \ldots, m-1\}$. Let $S = \{0, \ldots, m-1\}^k$ and $b \in \mathbb{Z}^k$ be any k-dimensional vector of integers. For any $X_0 \in S$, define the sequence

$$X_n \quad := \quad (AX_{n-1} + C) \bmod m, \tag{1}$$

$$u_n \quad := \quad (b' X_n \bmod m)/m, \tag{2}$$

for $n \geq 1$, where b' denotes the transpose of b, and the "modulo m" operation is performed elementwise. For fixed m, A, C, b, and $t \geq 0$, let

$$T_t = \{(u_n, \ldots, u_{n+t-1}) \mid n \geq 0,\ X_0 \in S\}, \tag{3}$$

the set of all possible overlapping t-tuples of successive values produced by (1–2), from all possible initial seeds. The set T_t turns out to be the intersection of a lattice L_t with the t-dimensional unit hypercube $[0,1)^t$. Recall that a d-dimensional *lattice* in $\mathbb{R}^t$ (for $d \leq t$) can be defined as a set of the form

$$L = \left\{ V = \sum_{j=1}^{d} z_j V_j \mid \text{ each } z_j \in \mathbb{Z} \right\},$$

where $V_1, \ldots, V_d$ is a set of independent vectors called a basis. In our case, we want the lattice L_t to be t-dimensional, otherwise all the points of T_t will be contained in one hyperplane. For example, one can take m prime, $C = 0$, $b' = (0, \ldots, 0, 1)$, and A as the companion matrix of a polynomial $f(z) = a_1 z + \cdots + a_k z^k$, which is primitive modulo m (Knuth 1981, L'Ecuyer 1990). This is equivalent to a *multiple recursive generator* (MRG), of the form

$$x_n \quad := \quad (a_1 x_{n-1} + \cdots + a_k x_{n-k}) \bmod m, \tag{4}$$

$$u_n \quad := \quad x_n / m, \tag{5}$$

where each $x_n \in \{1, \ldots, m-1\}$. The period of such a generator is then $\rho = m^k - 1$ (assuming that $x_0 \neq 0$), and a basis for the lattice L_t, as well as for its dual lattice, can be constructed as explained in Grothe (1988), L'Ecuyer and Blouin (1988), and L'Ecuyer and Couture (1992).

If the generator does not have full period and if one considers only the cycle that corresponds to a given initial seed, then, in general, the points do not form a lattice, but are still a subset of the lattice defined above, and typically also *generate* that same lattice. There are cases, however, where the points over one cycle generate a *sublattice* of L_t (e.g., if $k = 1$, $C = 0$, m is a power of two, and $A \bmod 8 = 5$; see Atkinson 1980). In the latter case, one should analyze the appropriate sublattice instead of L_t, which is the union of four such sublattices.

The fact that the points of T_t belong to a lattice means that they lie on a set of equidistant parallel hyperplanes. The shorter the distance d_t between those hyperplanes, the better, because this means thinner empty (without points) slices of space. Knuth (1981) gives an algorithm for computing d_t (called the spectral test). That algorithm is not very efficient and becomes impracticable in dimensions, say, larger than 10. Fincke and Pohst (1985) give a more efficient algorithm, which can cope with dimensions up to 25 or so.

A slightly different way of measuring the "quality" of the lattice is by the Beyer quotient, defined as follows. Geometrically, the lattice L_t "partitions" the space $\mathbb{R}^t$ into a juxtaposition of identical t-dimensional parallelepipeds, whose faces intersect, whose vertices are points of the lattice, and which contain no other lattice points except for their vertices. Those are called the *unit cells* of the lattice. All the edges connected to a given vertex of a unit cell form a set of linearly independent vectors which is a *Minkowski-reduced lattice basis* (MRLB) of L_t. The *Beyer quotient* q_t is the ratio of the length of the shortest vector over the length of the longest vector in a MRLB. A ratio q_t near one means that the unit cells are more cubic-like, while a ratio near 0 means the opposite. In contrast to d_t, q_t is a normalized measure (always between 0 and 1). As a figure of merit to rank generators, one can take for example the worst-case measure

$$Q_T = \min_{t \le T} q_t \tag{6}$$

for some some (fixed) large T. However, q_t is much more costly to compute than d_t. An algorithm to compute a MRLB and the Beyer quotient is given in Afflerbach and Grothe (1985) and Grothe (1988). The results of a search to find good multiple recursive generators based on the criterion Q_{20} are given in L'Ecuyer, Blouin, and Couture (1992), for orders k up to 7 and prime moduli up to near 2^{63}. As observed in L'Ecuyer (1990), comparing Beyer quotients makes sense only for generators having the same number of lattice points in the unit hypercube. A full period MRG has m^k such points, i.e., unit cells of volume m^{-k}, in all dimensions. For $t \le k$, the lattice is a perfect square grid of size $1/m$, and the Beyer quotient is 1. Increasing m or k gives smaller unit cells. If a generator G_1 has smaller Beyer quotient than another generator G_2, then G_1 might still be better than G_2 if it has smaller unit cells. In such a situation, as a bottomline criterion, one can turn back to the distance d_t between hyperplanes. If G_1 has a smaller d_t than G_2 for all t, then we say that G_1 *dominates* G_2 in terms of the spectral test, and claim that G_1 has a better lattice structure. L'Ecuyer, Blouin, and Couture (1992), for instance, propose MRG's with very small Beyer quotients, but which dominate all LCG's with the same modulus, while being competitive in terms of speed. There exists other MRG's of the same order and with much better Beyer quotients, but those are harder to implement and much slower.

Many pitfalls await those wanting to implement the Beyer and spectral tests. The current best algorithms are basically of "branch-and-bound" type. The bounds in the branch-and-bound are computed via a Choleski decomposition of the matrix of scalar products of the vectors of the lattice basis (Fincke and Pohst 1985, Grothe 1988). With larger moduli and in higher dimensions, numerical instabilities are commonplace. According to our experience, with 64-bit floating-point computations, standard Choleski decomposition algorithms often fail in high dimensions: at some point, one has to extract the square root of a negative value! Even when that does not happen, the results could be unreliable. Since it is used for computing bounds for the branch-and-bound, any small numerical error in the Choleski decomposition may give wrong bounds, which in turn could eliminate better solutions, and yield wrong results for the Beyer or spectral test. Further, for that branch-and-bound procedure to find the optimal solution in reasonable time, the basis should be reasonably pre-reduced (i.e., close to a MRB). A number of heuristics can be used for that purpose. L'Ecuyer and Couture (1992) have implemented algorithms for performing the Beyer and spectral tests in high dimensions (up to around 40 or more, depending on the generator), and for large moduli (say, a few hundred bits). Numerical errors have been carefully taken into account to obtain true bounds and "guaranteed" exact results.

To improve the statistical properties and perhaps get rid of the lattice structure, different kinds of *combined* generators have been proposed. Some of them have been analyzed successfully and turn out to be equivalent, or approximately equivalent, to LCG's with large moduli (L'Ecuyer and Tezuka 1991). Such combined generators can be viewed as efficient ways of implementing LCG's with very large moduli (sometimes with added "noise"), and can also be analyzed with the Beyer and spectral tests. Other classes of combined generators are not (yet) well understood theoretically. See L'Ecuyer (1990) and the references given there.

The highly efficient add-with-carry and subtract-with-borrow generators, recently proposed by Marsaglia and Zaman (1991), also turn out to be equivalent to LCG's with huge moduli, with a truncated fractional expansion (see Tezuka and L'Ecuyer 1992). So, again, those generators have an approximate lattice structure (in high dimensions).

Tausworthe and GFSR generators also have a lattice structure. Tezuka and L'Ecuyer (1991) and Couture, L'Ecuyer, and Tezuka (1992) show that simple or combined Tausworthe generators are equivalent to linear congruential generators defined over a field of formal Laurent series. By analyzing the lattice structure of such generators, they are able to characterize their t-dimensional behavior. For a given generator, for any partition of the t-dimensional unit hypercube into 2^{kl} identical subcubes, they can quickly compute a table giving the exact number of subcubes that contain n points, for each integer n. Tausworthe generators can be viewed as a special case of GFSR generators. Indeed, a Tausworthe generator can be implemented as a GFSR with a special kind of seed, and any GFSR with that kind of seed is equivalent to a Tausworthe generator (Fushimi 1989). So, the above theory also characterizes the behavior of GFSR generators with that kind of seed. To deal with more general GFSR generators, one needs some extensions of the notion of lattice, as pointed out by Tezuka (1990).

2.3. Discrepancy and Other Measures

The notion of *discrepancy* has been the subject of a lot of papers on random number generators. Here, we just give it a quick look. For more details, see the many references given in Niederreiter (1991).

Suppose we generate N t-dimensional points $P_n = (u_n, \ldots, u_{n+t-1})$, $0 \le n \le N-1$, formed by successive overlapping subsequences of length t. For any hyper-rectangular box aligned with the axes, of the form $R = \prod_{j=1}^{t}[\alpha_j,\ \beta_j)$, with $0 \le \alpha_j < \beta_j \le 1$, let $I(R)$ be the number of points P_n falling into R, and $V(R) = \prod_{j=1}^{t}(\beta_j - \alpha_j)$ be the volume of R. Let R be the set of all such regions R, and

$$D_N^{(t)} = \max_{R \in \mathcal{R}} |V(R) - I(R)/N|.$$

The latter is the t-dimensional *discrepancy* for the sequence of points $P_0, \ldots, P_{N-1}$. (There is also an alternative definition of discrepancy which imposes $\alpha_j = 0$ for all j.)

Points whose distribution is far from uniform will have high discrepancy, while points which are too evenly distributed will tend to have a discrepancy that is too low. A well behaved generator should have its discrepancy in the same order as that of a uniform sequence. Discrepancy is interesting and useful because one can obtain error bounds for (Monte Carlo) numerical integration or random search procedures in terms of $D_N^{(t)}$. In that context, the smaller the discrepancy, the better (because the aim is to minimize the numerical error, not really to imitate i.i.d. $U(0,1)$ random variables). Finally, generators of different types (e.g., linear vs nonlinear) can be compared in terms of the order of magnitude of their discrepancies. This cannot be done with the spectral test. The major difficulty with discrepancy, though, is that it can be computed exactly only for a few very special cases (e.g., for a LCG, for $t = 2$). However, for many classes of generators, bounds on $D_N^{(t)}$ are available (Niederreiter 1991). Another drawback is that discrepancy measures are dependent on the orientation of the axes, in contrast to the Beyer or spectral tests.

A number of other "statistical properties" can be computed exactly (or bounds for them can be computed) for specific classes of generators. That includes computing bounds on the serial correlation (Knuth 1981), computing the results of the "run" test applied over the whole sequence of a Tausworthe generator (Tootill, Robinson, and Adams 1971), computing the nearest pair of points over the whole period, or the minimal number of hyperplanes that cover all the points, etc.

3. EMPIRICAL STATISTICAL TESTS

An unlimited number of empirical tests can be designed for random number generators. The null hypothesis is H_0: "The sequence is a sample of i.i.d. $U(0,1)$ random variables", and a statistical test tries to find empirical evidence against H_0. Any function of a finite number of $U(0,1)$ random variables, whose (sometimes approximate) distribution under H_0 is known, can be used as a statistic T which defines a test for H_0.

3.1. Multilevel Tests

To increase the power, a given test can be replicated N times, on disjoint parts of the sequence, yielding values $T_1, \ldots, T_N$ of the statistic. The empirical distribution of those N values can then be compared to the theoretical distribution of T under H_0, via standard univariate goodness-of-fit tests, like Kolmogorov-Smirnov (KS), Anderson-Darling, or Cramer-von Mises (see Durbin 1973, Stephens 1986a, b). We call this a *two-level* test.

For the examples of the next section, we computed the value d of the KS statistic D_N, and the descriptive level δ_2 of the two-level test, defined as

$$\delta_2 = P[D_N > d \mid H_0]. \tag{7}$$

Under H_0, δ_2 should be $U(0,1)$. A very small value of δ_2 (say, $\delta_2 < .01$) provides evidence against H_0. In case of doubt, the whole procedure can be repeated (independently), and if small values of δ_2 are produced consistently, H_0 should be rejected, which means that the generator *fails* the test. If δ_2 is not too small, that improves confidence in the generator. It should be clear, however, that statistical tests never *prove* that a generator is foolproof.

For further increase in power, one can also perform a *three-level* test: replicate the two-level test R times, and compare the empirical distribution of the R values of δ_2 with the $U(0,1)$ distribution, using again a goodness-of-fit test, and yielding a descriptive level δ_3. Reject if δ_3 is too small. One can also go up to fourth level, fifth level, and so on. However, one major problem showing up with higher-level tests is that in most cases, the exact distribution of the first-level statistic T under H_0 is not available, but only an approximation of it is (e.g., a chi-squared distribution). Often, that approximation is also the asymptotic distribution as the (first-level) sample size increases to infinity. What could happen then is that the higher-level test detects the lack-of-fit of that approximation, leading to rejection even if the generator is good. The higher the level, the more this is likely to happen. Perhaps this problem could be alleviated in some cases by finding better statistics or better approximations, but usually not eliminated. Similarly, for a three-level (or more) test, if a KS statistic is used at the second level, the N descriptive level values δ_2 will usually be computed using the asymptotic (as $N \to \infty$) KS distribution. Again, if N is too small, the test will detect the fact that the asymptotic is not yet a good enough approximation. So, for higher-level tests, one must take much larger sample sizes at the lower levels. This quickly becomes time-wise prohibitive.

If higher-level tests are problematic, why not just use one level ? One-level tests do not test the local behavior of generators as well as higher-level tests. Some sequences have good properties when we take the average over the whole sequence, but not when we look at very short subsequences. As an illustration, consider the (extreme) example of a generator producing the values $i/2^{31}$, $i = 1, 2, \ldots, 2^{31} - 1$, in that order. A uniformity test over the whole sequence will give a perfect adjustment. In fact, the adjustment will be too good, giving what is called *super-uniformity* (Stephens 1986b). On the other hand, uniformity tests over disjoint shorter subsequences will give terribly bad adjustments. So, one-level tests are not always appropriate, and two-level tests seem to offer a good compromise.

3.2. Standard and More Stringent Tests

Knuth (1981) describes a set of tests which have been considered for a while as "the standard tests" for testing random number generators. Arguing that those so-called standard tests were not sufficiently discriminatory, i.e., that many "bad" generators passed most of the tests, Marsaglia (1985) proposed a new set of more stringent tests. Indeed, sophisticated applications like probabilistic computational geometry, probabilistic combinatorial algorithms, design of statistical tests, and so on, often require generators with excellent high-dimensional structures. Marsaglia argued that for such classes of applications, simple generators (e.g., LCG, Tausworthe, GFSR, etc.) were not good enough, and advocated combining generators with different (incompatible) algebraic structures.

3.3. The Nearest Pair

We now give an example of a test, based on a suggestion of Ripley (1983), which most LCG's currently in use would certainly fail. We call it the *nearest-pair* test. It goes as follows. Generate n points in the t-dimensional hypercube $(0,1)^t$, each one using t successive values produced by the generator. Among those n points, find the nearest pair of points and let D be the (Euclidean) distance between them. For large n, under H_0, the random variable $T = n^2 D^t / 2$ is approximately exponential with mean $1/V_t$, where V_t is the volume of a t-dimensional unit sphere. For a two-level test, one generates N independent values of T and compare their empirical distribution with that of an exponential. Note that the larger N is, the larger n must be, because the exponential distribution is only the asymptotic law as $n \to \infty$, and a larger N permits one to detect the lack of fit in that approximation. Also, the approximation gets worse as t increases, which means that one must then decrease N or increase n. Finally, increasing n gets very costly in high dimensions because finding the nearest pair is more time-consuming. Numerical results for that test are given in the next section.

3.4. A Vicious Circle

Bickel and Breiman (1983) have proposed a goodness-of-fit test for multidimensional densities, which could be applied here to test t-dimensional uniformity. It is somewhat related to the nearest pair test and goes as follows. Generate n points in $(0,1)^t$ as above, and for each point i, compute the distance R_i to the nearest other point and let $W_i = \exp(-nV_i)$, where V_i is the volume of a t-dimensional sphere of radius R_i. Construct the empirical distribution of $W_1, \ldots, W_n$, and compare it to the appropriate theoretical distribution under H_0. That test looks more powerful than the nearest pair test, because it uses more information. For the theoretical distribution under H_0, one can show that for each i, W_i is approximately (for large n) $U(0,1)$. The difficulty, however, is that all those W_i's are correlated. A measure of deviation of the W_i's from uniformity is

$$T = \sum_{i=1}^{n} (W_{(i)} - i/n)^2,$$

where $W_{(1)}, \ldots, W_{(n)}$ are the W_i's sorted in increasing order. Bickel and Breiman show that as $n \to \infty$, T converges in law to $\int_0^1 Z^2(t)dt$, where Z is a Gaussian process with mean 0 and a complicated covariance function (see their Equation 5.13). Computing the distribution function of T under H_0 is a challenging task and it seems that currently, the best way for estimating that theoretical distribution is by Monte Carlo simulation. But which generator should we use for that ? We get into a vicious circle, because what we want to test is precisely whether the random number generators are able to reproduce the right distribution function for T.

Marsaglia (1985) also proposes tests for which the theoretical distribution is unknown (e.g., the parking lot test) and suggests comparing the empirical distribution of a generator to be tested with that of a "good" generator. Again, we are caught into the same vicious circle: which one is the good one ? In practice, a reasonable (heuristic) compromise could be to estimate the theoretical distribution with many different types of (supposedly good) ran-

Table 1: The selected statistical tests.

T1.	Poker ($N = 1000,\ n = 10^4,\ d = 16,\ k = 12$).
T2.	Runs Up ($N = 1000,\ n = 10^5$).
T3.	OPSO ($N = 100, n = 2^{21},\ r = 0,\ b = 10$).
T4.	OPSO ($N = 100, n = 2^{21},\ r = 15,\ b = 10$).
T5.	OPSO ($N = 100, n = 2^{22},\ r = 0,\ b = 11$).
T6.	OPSO ($N = 100, n = 2^{22},\ r = 15,\ b = 11$).
T7.	Birthday spacings ($N = 100,\ n = 100,\ m = 5000,\ d = 2 \times 10^9$).
T8.	Nearest Pair ($N = 100,\ n = 10^5,\ t = 4$).
T9.	Nearest Pair ($N = 20,\ n = 10^5,\ t = 6$).
T10.	Nearest Pair ($N = 20,\ n = 50000,\ t = 9$).

Table 2: The selected generators.

G1.	LCG with $m = 2^{31} - 1$ and $a = 16807$.
G2.	LCG with $m = 2^{31} - 1$ and $a = 630360016$.
G3.	LCG with $m = 2^{31} - 1$ and $a = 742938285$.
G4.	CSD generator of Sherif and Dear (1990).
G5.	Combined generator in Fig. 3 of L'Ecuyer (1988).
G6.	Combined Tausworthe generator G1 of Tezuka and L'Ecuyer (1991).
G7.	Twisted GFSR with $(r, s, p) = (25, 7, 32)$.
G8.	Subtract-with-borrow generator with $(b, r, s, L) = (2^{32} - 5, 43, 22, 1)$.

dom number generators. If the results agree, it certainly improves our confidence that this is the right distribution.

3.5. Other tests

Besides the tests described in Knuth (1981) and in Marsaglia (1985), other statistical tests for random number generators are proposed in Yuen (1977), Solomon and Stephens (1983), Marsaglia and Tsay (1985), Stephens (1986b), Maurer (1992), Ugrin-Sparac (1991), Matsumoto and Kurita (1992), and Dalle Molle, Hinich, and Morrice (1992). The latter paper proposes some tests based on the Fourier transform of high order cumulant functions, and preliminary investigations suggest that these tests are very powerful.

4. SOME STATISTICAL TEST RESULTS

As an illustration, we have selected 10 statistical tests, and applied them to 8 popular or recently proposed random number generators. All tests are two-level, with a KS test at the second level. For each test, n, N, and δ_2 are as defined in the previous section. We now describe briefly the tests that we have selected. For the (simplified) *Poker test* (Knuth 1981), generate k integers uniformly from $\{0, \ldots, d-1\}$, and compute the number r of distinct values. Repeat n times and compare the distribution of those n values of r with the theoretical distribution via a chi-squared test. For the *runs up* test, generate a sequence of n values in (0,1), compute the number of increasing subsequences of each length within that sequence, and compare that with the theoretical distribution using a chi-squared (Knuth 1981). The OPSO (Overlapping Pairs Sparse Occupancy) test is described in Marsaglia (1985). Take n overlapping pairs of successive $U(0,1)$ variates, and for each pair, juxtapose the bits $r+1$ to $r+b$ of the binary expansion of each variate, to obtain a $2b$-bit number. Compute the number of distinct $2b$-bit numbers thus obtained. The *birthday spacings* test is also from Marsaglia (1985): generate m integers uniformly from the set $\{1, \ldots, d\}$, sort them in increasing order, compute the spacings $I_{j+1} - I_j$ between the successive (sorted) values, and the number Y of spacing values which appear more than once. Compute n independent values of Y and compare their distribution with the Poisson distribution via a chi-squared. Finally, the *Nearest pair* test was described in the previous section.

We selected the 10 tests defined in Table 1, and applied them to the 8 generators described in Table 2. The generators G1 to G3 are recommended respectively by Bratley, Fox, and Schrage (1987), Law and Kelton (1991), and Fishman and Moore (1986). G7 is proposed by Matsumoto and Kurita (1992), while G8 is proposed by Marsaglia and Zaman (1991) and further recommended by James (1990).

Table 3 gives the descriptive level δ_2 for each test–generator pair. For each generator, the different tests were

Table 3: Descriptive Levels δ_2

	G1	G2	G3	G4	G5	G6	G7	G8
T1	.3072	.7752	.2208	.5122	.1317	.0051	.1719	.0038
T2	.4650	.9565	.9763	.0750	.5754	.8456	.9913	.6251
T3	.0096	5.9E-5	7.7E-6	< E-15	.8730	.9717	.2072	.0298
T4	2.0E-5	3.0E-5	6.1E-6	< E-15	.9420	.0329	.1625	.1877
T5	7.5E-5	< E-15	< E-15	< E-15	.3355	.1165	.5780	.9341
T6	1.2E-4	< E-15	< E-15	< E-15	.4025	.3666	.3974	.9101
T7	< E-15	< E-15	7.8E-16	< E-15	.2264	.7144	.9018	< E-15
T8	< E-15	< E-15	< E-15	< E-15	.6955	.0470	2.4E-8	.2075
T9	6.1E-15	1.5E-11	< E-15	.5745	.3463	.6440	.0944	.4833
T10	1.1E-7	2.1E-13	1.1E-6	.5177	.4931	.9526	.4045	.2768

performed using disjoint ("independent") subsequences. The results should be shocking to naive LCG users: those generators fail not only the nearest pair test, but most of the tests. CSD also fails miserably. The other generators, G5 to G8, behave reasonably well, except for the recently recommended SWB generator G8, which fails the birthday spacings test, and the twisted GFSR, which fails one of the nearest pair tests. G6 also has a few suspicious values. Overall, the combined generators (G5 and G6) are the best performers.

The fact that all the LCG's fail the nearest pair tests is not surprising. Indeed, because of their lattice structure, the distance between the nearest points is bounded below by the length of the shortest vector in the lattice. In t dimensions, if the unit cell volume is $1/m$ and the Beyer quotient is near one, the length of that shortest vector is approximately $(1/m)^{1/t}$. Then, for large n and t, the statistic T defined in Section 3.3 cannot take values as small as expected. This was pointed out by Ripley (1983). One interesting aspect of this is that generators with very small Beyer quotient can (possibly) do better with respect to this test, because they have a smaller lower bound. So, having a small Beyer quotient is not necessarily bad for all situations. There are applications for which it could actually be better! To a certain extent, this goes against some well incrusted folklore.

5. CONCLUSION AND PRACTICAL IMPLICATIONS

We have seen that some previously "recommended" and heavily used random number generators fail severely some statistical tests. Is that really disastrous ? Of course, that depends on the target application. People often ask for examples of simulation models where using one of those generators would produce wrong results. Most likely, the commonly used LCG's with moduli near 2^{31} perform well enough when simulating typical queueing network models. However, one can easily construct examples, based on the tests that the generators fail, for which the generators actually produce completely wrong results. For example, if our aim is to estimate the distribution function of the statistic T in the test of Bickel and Breiman (Section 3.4), a standard LCG is certainly not appropriate.

Statistical tests are far from being clean cut testing tools. Because any generator has finite period, almost any good test, if run long enough, will eventually detect regularity and reject the generator. So, how can we be satisfied with empirical test results ? A reasonable practical view here is to restrict ourselves to tests that we can practically run on a computer. For example, if a test needs 10 million years of CPU time on a Cray-2, we don't care about its eventual results. However, we would like the generator to pass all known tests which can run in, say, less than a few weeks, assuming that the generator's structure is unknown to the test builder and only the output values u_n are observed. But even this is not easy to achieve with efficient generators. If the generator is a LCG, for example, there exists efficient algorithms which can find out the modulus and multiplier only from the output values, and guess the following values (Boyar 1989). From that, it is easy to design a test that the LCG will fail. Perhaps asking a generator to pass all such tests is asking too much ? Well, that depends on the application. In cryptology, for example, one needs generators which are (practically) *unpredictable* in a specific sense (L'Ecuyer and Proulx 1989). Such generators should pass all reasonable statistical tests, but their current limitation is that they are not fast enough for simulation applications. Research is still under way.

ACKNOWLEDGMENTS

Richard Simard helped doing the computations for the numerical examples. This work has been supported by NSERC-Canada grant # OGP0110050 and FCAR-

Québec grant # 93ER1654.

REFERENCES

Afflerbach, L. and H. Grothe. 1985. Calculation of Minkowski-Reduced Lattice Bases. *Computing*, **35**: 269–276.

Atkinson, A. C. 1980. Tests of Pseudo-Random Numbers. *Applied Statistics*, **29**, 2: 164–171.

Bickel, P. J. and L. Breiman. 1983. Sums of Functions of Nearest Neighbor Distances, Moment Bounds, Limit Theorems and a Goodness of Fit Test. *The Annals of Probability*, **11**, 1:185–214.

Boyar, J. 1989. Inferring Sequences Produced by a Linear Congruential Generator Missing Low-Order Bits. *Journal of Cryptology*, **1**:177–184.

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

Couture, R., P. L'Ecuyer, and S. Tezuka 1992. On the Distribution of k-Dimensional Vectors for Simple and Combined Tausworthe Sequences. To appear in *Mathematics of Computation*.

Coveyou, R. R. and R. D. MacPherson. 1967. Fourier Analysis of Uniform Random Number Generators. *Journal of the ACM*, **14**:100–119.

Dalle Molle, J. W., M. J. Hinich, and D. J. Morrice. 1992. Higher-Order Cumulant Spectral Based Statistical Tests of Pseudo Random Variate Generators. In these Proceedings.

Dudewicz, E. J. and T. G. Ralley. 1981. *The Handbook of Random Number Generation and Testing with TESTRAND Computer Code*. American Sciences Press, Columbus, Ohio.

Durbin, J. 1973. *Distribution Theory for Tests Based on the Sample Distribution Function*. SIAM Regional Conference Series in Applied Mathematics, vol. 9, SIAM, Philadelphia.

Fincke, U. and M. Pohst. 1985. Improved Methods for Calculating Vectors of Short Length in a Lattice, Including a Complexity Analysis. *Mathematics of Computation*, **44**, 170:463–471.

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

Fushimi, M. 1989. An Equivalence Relation Between Tausworthe and GFSR Sequences and Applications. *Applied Math. 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.

Grothe, H. 1988. Matrixgeneratoren zur Erzeugung gleichverteilter Pseudozufallsvektoren (in german). Dissertation (thesis), Tech. Hochschule Darmstadt, Germany.

James, F. 1990. A review of pseudorandom number generators. *Computer Physics Communications*, **60**:329-344.

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

Law, A. M. and W. D. Kelton. 1991. *Simulation Modeling and Analysis*. Second edition, McGraw-Hill.

L'Ecuyer, P. 1988. Efficient and Portable Combined Random Number Generators. *Communications of the ACM*, **31**, 6:742–749 and 774.

L'Ecuyer, P. 1990. Random Numbers for Simulation. *Communications of the ACM*, **33**, 10:85–97.

L'Ecuyer, P. and F. Blouin. 1988. Linear Congruential Generators of Order $k > 1$. *Proceedings of the 1988 Winter Simulation Conference*, IEEE Press, 432–439.

L'Ecuyer, P., F. Blouin, and R. Couture. 1992. A Search for Good Multiple Recursive Generators. Submitted for publication.

L'Ecuyer, P. and R. Couture. 1992. An Implementation of the Lattice and Spectral Tests for Linear Congruential and Multiple Recursive Generators. In preparation.

L'Ecuyer, P. and R. Proulx. 1989. About Polynomial-Time "Unpredictable" Generators. *Proceedings of the 1989 Winter Simulation Conference*, IEEE Press, 467–476.

L'Ecuyer, P. and S. Tezuka. 1991. Structural Properties for Two Classes of Combined Random Number Generators. *Mathematics of Computation*, **57**, 196:735–746.

Lewis, T. G. and W. H. Payne. 1973. Generalized Feedback Shift Register Pseudorandom Number Algorithm. *J. of the ACM*, **20**, 3:456–468.

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.

Marsaglia, G. 1985. A Current View of Random Number Generation. *Computer Science and Statistics, Proceedings of the Sixteenth Symposium on the Interface*, Atlanta, march 1984. Elsevier Science Publ. (North-Holland), 3–10.

Marsaglia, G. and L.-H. Tsay. 1985. Matrices and the Structure of Random Number Sequences. *Linear Algebra and its Applications*, **67**:147–156.

Marsaglia, G. and A. Zaman. 1991. A New Class of Random Number Generators. *The Annals of Applied Probability*, **1**, 3:462–480.

Matsumoto, M. and Y. Kurita. 1992. Twisted GFSR Generators. Submitted for publication.

Maurer, U. M. 1992. A Universal Statistical Test for Random Bit Generators. *Journal of Cryptology*, **5**, 2:89–105.

Moore, D. S. 1986. Tests of Chi-Squared Type. In *Goodness-of-Fit Techniques*, Edited by R. B. D'Agostino and M. S. Stephens, Marcel Dekker, New York and Basel.

Niederreiter, H. 1991. Recent Trends in Random Number and Random Vector Generation. *Annals of Operations Research*, **31**:323–345.

Niederreiter, H. 1992. New Methods for Pseudorandom Number and Pseudorandom Vector Generation. In these Proceedings.

Read, T. R. C. and N. A. C. Cressie. 1988. *Goodness-of-Fit Statistics for Discrete Multivariate Data*. Springer-Verlag.

Ripley, B. D. 1983. The Lattice Structure of Pseudorandom Number Generators. *Proc. Roy. Soc. London, Series A*, **389**:197–204.

Ripley, B. D. 1990. Thoughts on Pseudorandom Number Generators. *J. of Computational and Applied Mathematics*, **31**:153–163.

Sherif, Y. S. et R. G. Dear. 1990. Development of a New Composite Pseudo-random Number Generator. *Microelectronics and Reliability*, **30**, 3:545–553.

Solomon, H. and M. A. Stephens. 1983. On Neyman's Statistic for Testing Uniformity. *Communications in Stat. Simul. Comput.*, **12**, 2:127–134.

Stephens, M. S. 1986a. Tests Based on EDF Statistics. In *Goodness-of-Fit Techniques*, Edited by R. B. D'Agostino and M. S. Stephens, Marcel Dekker, New York and Basel.

Stephens, M. S. 1986b. Tests for the Uniform Distribution. In *Goodness-of-Fit Techniques*, Edited by R. B. D'Agostino and M. S. Stephens, Marcel Dekker, New York and Basel.

Tezuka, S. 1990. Lattice Structure of Pseudorandom Sequences From Shift-Register Generators. *Proceedings of the 1990 Winter Simulation Conference*, IEEE Press, 266–269.

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.

Tezuka, S. and P. L'Ecuyer. 1992. Analysis of Add-with-Carry and Subtract-with-Borrow Generators. In these Proceedings.

Tootill, J. P. R., W. D. Robinson, and A. G. Adams. 1971. The Runs Up-and-Down Performance of Tausworthe Pseudorandom Number Generators. *Journal of the ACM*, **18**:381–399.

Ugrin-Sparac, G. 1991. Stochastic Investigations of Pseudo-Random Number Generators. *Computing*, **46**:53–65.

Yuen, C.-K. 1977. Testing Random Number Generators by Walsh Transform. *IEEE Transactions on Computers*, **C-26**, 4:329–333.

AUTHOR BIOGRAPHY

PIERRE L'ECUYER is a full professor in the department of "Informatique et Recherche Opérationnelle" (IRO), at the University of Montreal. He received a Ph.D. in operations research in 1983, from the University of Montreal. From 1983 to 1990, he was with the computer science department, at Laval University, Québec. His research interests are in Markov renewal decision processes, sensitivity analysis and optimization of discrete-event stochastic systems, random number generation, and discrete-event simulation in general. He is an Associate Editor for *Management Science* and for the *ACM Transactions on Modeling and Computer Simulation*. He is also a member of ACM, IEEE, ORSA and SCS.

This paper appeared in the *Proceedings of the 1992 Winter Simulation Conference*, IEEE Press, Dec. 1992, pp. 305–313.