UNIFORM RANDOM NUMBER GENERATORS

Pierre L'Ecuyer

Département d'Informatique et de Recherche Opérationnelle
Université de Montréal, C.P. 6128, Succ. Centre-Ville
Montréal, H3C 3J7, CANADA

ABSTRACT

We briefly overview the design principles, implementation techniques, and empirical testing of uniform random number generators for simulation. We first discuss some philosophical issues and quality criteria. Then we explain a few concrete families of generators and give appropriate references to further details and to recommended implementations.

1. WHAT ARE WE LOOKING FOR?

1.1. Definition

What we call a *random number generator* (RNG) is actually a program that produces, once its initial state is chosen, a deterministic and periodic sequence of numbers. An RNG has a *state* that evolves in a finite state space $\mathcal{S}$, according to a recurrence of the form $s_n = f(s_{n-1})$, $n \geq 1$, where the initial state $s_0 \in \mathcal{S}$ is called the *seed*, and $f : \mathcal{S} \to \mathcal{S}$ is the *transition function*. At step n, the generator outputs $u_n = g(s_n)$, where $g : \mathcal{S} \to [0, 1)$ is the *output function*. The output sequence is thus $\{u_n, n \geq 0\}$. The output space could be more general, but we assume here that it is a subset of the real interval $[0, 1)$. Since $\mathcal{S}$ is finite, the output sequence is periodic (possibly after some initial transient), say with period length ρ. A well-designed RNG normally has ρ near $|\mathcal{S}|$, i.e., $\rho \approx 2^e$ if the state is represented over e bits.

Formally, this deterministic construction contradicts the idea of a sequence of independent and identically distributed (i.i.d.) random variables. But from a practical viewpoint, experience indicates that this works fine. We give some heuristic explanations of why in what follows. These heuristics lead to theoretical quality criteria that need advanced mathematical tools to be assessed.

1.2. Equidistribution

The idealized mathematical abstraction that we want to imitate corresponds to the null hypothesis $\mathcal{H}_0$: "The u_n are i.i.d. $U(0, 1)$" (i.e., independent random variables uniformly distributed over the interval $(0, 1)$). This hypothesis means that for each n and t, the vector $\mathbf{u}_{n,t} = (u_n, \ldots, u_{n+t-1})$ is uniformly distributed over the t-dimensional unit hypercube $[0, 1)^t$. We know *a priori* that $\mathcal{H}_0$ is false. But can we still assume it *for practical purposes*?

To better illustrate the ideas, suppose that the RNG has period length $\rho = |\mathcal{S}|$, that the output space is $\mathbb{Z}_m/m = \{0, 1/m, 2/m, \ldots, (m-1)/m\}$ for some positive integer m, and that the seed s_0 is random, uniformly distributed over $\mathcal{S}$. Such an RNG is called *t-distributed in base m* if all of the m^t possible t-dimensional output vectors appear exactly the same number of times in the set

$$\Psi_t \stackrel{\text{def}}{=} \{\mathbf{u}_{0,t} : s_0 \in \mathcal{S}\} = \{\mathbf{u}_{n,t}, 0 \leq n < \rho\}.$$

In this case, the hypothesis $\mathcal{H}'_0(m, t)$: "Each $\mathbf{u}_{n,t}$ is uniformly distributed over the set $\mathbb{Z}^t_m/m$" holds. Note that this is possible only if m^t divides ρ. This hypothesis is weaker than $\mathcal{H}_0$: We have uniformity only over a discretization of $[0, 1)^t$ and we do not have independence. The effect of this can be negligible *only if* ρ is huge; i.e., a good RNG *must* have a very long period (but this condition is not sufficient). In practice, ρ is often slightly smaller than a multiple of m, but we can settle for an approximation of $\mathcal{H}'_0(m, t)$.

For a given value of $|\mathcal{S}|$, one can choose the m above in different ways. For example, suppose $|\mathcal{S}| = 2^e$ for some large e. Taking the first ℓ bits of each output value (i.e., ℓ bits of resolution) gives $m = 2^\ell$. A smaller ℓ means the possibility of t-distribution for a larger t, up to $t = \lfloor e/\ell \rfloor$. If the RNG is $\lfloor e/\ell \rfloor$-distributed with ℓ bits of resolution for $1 \leq \ell \leq \min(e, w)$, it is called *asymptotically random* or *maximally equidistributed* (ME) for the word size w (see

L'Ecuyer 1996b; Tezuka 1995). Then, for a partition of $[0,1)^t$ into $2^{t\ell}$ cubic boxes of equal size, for $\ell \le w$ and $t\ell \le e$, the point set Ψ_t has the best possible equidistribution into the boxes. A stronger property than ME is that of a (t,m,s)-net, where one requires equidistribution for a more general class of partitions of $[0,1)^t$ into rectangular boxes (not only cubic boxes). See Niederreiter (1992b) and Owen (1998) for details and references. Explicit constructions and implementations of ME RNGs are available (L'Ecuyer 1996b; L'Ecuyer 1998e). The construction of large-period RNGs whose point sets are (t,m,s)-nets is still a matter of current investigation.

1.3. Figures of Merit in Large Dimensions

For finite point sets, t-distribution is an interesting concept only in relatively small dimensions. For t such that $m^t \gg \rho$, Ψ_t can cover only a tiny fraction of $\mathbb{Z}_m^t/m$. When the seed s_0 is random, Ψ_t can be viewed in a way as the *sample space* from which the output vectors $\mathbf{u}_{n,t}$ are taken, without replacement. It thus makes sense to require that Ψ_t be "evenly" or "uniformly" distributed over the unit hypercube $[0,1)^t$, in some sense. There are several ways of measuring this uniformity by *figures of merit*. Perhaps the single most important factor in the choice of a figure of merit is the ability to compute it efficiently. As a result, different families of RNGs are analyzed in practice using different figures of merit. Some may argue that the points of Ψ_t should look like random points over $\mathbb{Z}_m^t/m$ instead of being too evenly distributed. But if Ψ_t is viewed as a (huge) sample space from which points are taken at random without replacement, a superuniform (i.e., very even) distribution of Ψ_t seems justified.

1.4. Discrepancy

The *discrepancy* of a point set $\Psi_t \subset [0,1)^t$ refers to a measure of departure between the empirical distribution of Ψ_t and the uniform distribution. There is an infinite number of ways of defining such a measure. For example, for each rectangular box $B \subseteq [0,1]^t$ with one corner at the origin, compute the absolute difference between the fraction of Ψ_t falling in B and the volume of B, and take the supremum over the set of all such boxes B. This is the "standard" *star discrepancy*. More generally, if one replaces the supremum by the $\mathcal{L}_p$-average over all boxes, where the upper corner of the box is uniformly distributed in $[0,1]^t$, this gives the $\mathcal{L}_p$ *star discrepancy* (the standard case corresponds to $p=\infty$). By removing the condition that the lower corner is at the origin and taking the average over all boxes $B \subseteq [0,1]^t$, one obtains the *unanchored $\mathcal{L}_p$ discrepancy*. More general regions B can be considered, such as all convex sets, and so on. We refer to Hickernell (1998) and Niederreiter (1992a) for more details. A major problem with most of these definitions is that no efficient algorithm is available to compute the value of the discrepancy for the large point sets Ψ_t that are required (we believe) for good RNGs. Here, we are thinking of period lengths of 2^{100} or more, so even an algorithm working in time $O(n)$ for n points is not good enough.

1.5. Spectral Analysis

A general approach for comparing a given density f_1 to the uniform density f_0 over $[0,1)^t$ goes as follows. Choose an orthonormal basis for a space $\mathcal{F}$ of functions $f : [0,1)^t \to \mathbb{R}$ (or perhaps $f : [0,1)^t \to \mathbb{C}$, the complex numbers), such that both f_0 and f_1 are in $\mathcal{F}$. Then expand f_0 and f_1 in terms of this orthonormal basis and compare the coefficients. This is the general idea of Fourier analysis (e.g., Folland 1992).

As an illustration, consider the Fourier basis $\mathcal{B} = \{\psi_{\mathbf{h}},\ \mathbf{h} \in \mathbb{Z}^t\}$, where $\psi_{\mathbf{h}}(\mathbf{u}) = e(\mathbf{h}\cdot\mathbf{u})$, $e : \mathbb{R} \to \mathbb{C}$ is the complex trigonometric function $e(x) = \exp(2\pi\iota x)$, and $\iota = \sqrt{-1}$. This $\mathcal{B}$ is a basis for the space of square-integrable functions over $[0,1)^t$. The *Fourier expansion* of f in terms of $\mathcal{B}$ is

$$f(\mathbf{u}) = \sum_{\mathbf{h}\in\mathbf{Z}^t} \hat{f}(\mathbf{h})\psi_{\mathbf{h}}(\mathbf{u}), \tag{1}$$

with *Fourier coefficients* $\hat{f}(\mathbf{h}) = \int_{[0,1)^t} f(\mathbf{u})\psi_{\mathbf{h}}(-\mathbf{u})d\mathbf{u}$. A probability density over $[0,1)^t$ always has $\hat{f}(\mathbf{0}) = 1$, and the uniform density f_0 has Fourier coefficients $\hat{f}_0(\mathbf{h}) = 0$ for all $\mathbf{h} \ne \mathbf{0}$. Therefore, one way to test whether f_1 is close to uniform is to test its Fourier coefficients $\hat{f}_1(\mathbf{h})$ for $\mathbf{h} \ne \mathbf{0}$. Of course, this can also be done with other bases; see, e.g., Hellekalek (1998b).

For a given point set $\Psi_t = \{\mathbf{u}_{0,t}, \ldots, \mathbf{u}_{n-1,t}\}$ in $[0,1)^t$, one can estimate the coefficients $\hat{f}(-\mathbf{h})$ by the *exponential (Weyl) sums*

$$S_n(\mathbf{h}) = \frac{1}{n}\sum_{j=0}^{n-1} e(\mathbf{h}\cdot\mathbf{u}_{j,t}). \tag{2}$$

If we want Ψ_t to be uniform, the $S_n(\mathbf{h})$ must be close to zero for $\mathbf{h} \ne \mathbf{0}$. Since the high-amplitude low-frequency variations are usually more damaging than the high-frequency variations, it is customary to give more weight to the former, i.e., to the small vectors $\mathbf{h}$. This motivates *measures of discrepancy* that are weighted sums (or supremums) of increasing functions of the $|S_n(\mathbf{h})|$, as explained, e.g., by

Hellekalek (1998b, 1999), Hellekalek and Niederreiter (1998), Hickernell (1998), and Leeb and Hellekalek (1998).

A special case of this is

$$\sup_{0 \neq \mathbf{h} \in \mathbf{Z}^t} = |S_n(\mathbf{h})| / \|\mathbf{h}\|_2, \tag{3}$$

where $\|\cdot\|_2$ is the Euclidean norm, and it corresponds to the *spectral test* originally proposed by Coveyou and MacPherson (1967) and commonly applied to linear congruential and multiple recursive generators. For these generators, this spectral test can be applied even for astronomical sizes of Ψ_t (e.g., over 2^{1000}) by exploiting the lattice structure (see Section 4).

1.6. Statistical Tests

After an RNG has been designed and implemented, empirically-minded people want to "test it on the road" by applying *statistical tests*.

Optimists may be looking for the *ultimate* RNG, which runs fast and passes *all statistical tests*. But such an object does not exist. To understand why, suppose that the output space is finite, with cardinality $|U|$, and consider the tests that look at n successive output values, for some fixed n. For a fixed α, $0 < \alpha < 1$, a *test* of level α is any function $T_n : U^n \to \{0, 1\}$ such that $|T_n^{-1}(0)| = \alpha |U|^n$ (assumed to be an integer, to simplify). The test rejects $\mathcal{H}_0$ when T_n maps the observed sequence to 0. The number of such tests is the number of ways of choosing $\alpha|U|^n$ objects among $|U|^n$. A key observation here is that *every* sequence of n elements from U is mapped to 0 by the same number of tests T_n. In other words, all sequences (or generators) fail exactly the same number of tests. See Leeb 1995 and Wegenkittl (1995) for more about this.

Statistical testing of RNGs is thus meaningless unless the tests are not all considered on equal footing. For large n, the number of tests is in fact incredibly huge and most of them are so complicated that they cannot be run on a computer in our lifetime. Then we can say that a *bad* RNG is one that fails *simple* tests, and a *good* RNG is one that fails only complicated tests that are hard to find.

In practice, batteries of more or less natural tests are applied to RNGs, and systematic failures reveal *defects* in the structure of the RNG. No amount of testing can *prove* that a given RNG is flawless. It only improves our confidence to a certain extent.

Ideally, the statistical tests should be selected in close relation with the target application, i.e., be based on a test statistic T that closely mimics the random variable of interest. But this is usually impractical, especially when designing and testing generators for general purpose software packages. For a sensitive application, it is highly recommended that the user tests the RNG specifically for his (or her) problem, or tries RNGs from totally different classes and compares the results.

1.7. Additional Requirements

A long period, good structure of Ψ_t, and passing reasonable statistical tests, are not the only requirements. For simulations involving billions of random numbers, the generator's speed can be critical. The size of required memory may also be important when virtual generators (or substreams) are maintained in parallel. This is required, for example, for the 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. The packages of L'Ecuyer and Côté (1991) and L'Ecuyer and Andres (1997) offer software tools to manipulate such substreams.

2. LINEAR RECURRENCES

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

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

$$u_n = x_n / m. \tag{5}$$

The *modulus* m and the *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 1997; L'Ecuyer 1998b), i.e.,

$$x_n = (a_r x_{n-r} + a_k x_{n-k}) \bmod m. \tag{6}$$

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

A key issue for the computer implementation of the MRG is how to compute efficiently the products $ax \bmod m$ when m is large. We mention three of the most useful approaches for this. In general, when searching for MRGs with good theoretical behavior, one would search only in areas where the MRG coefficients a_i satisfy the conditions for one of these implementation techniques. A first approach uses integer arithmetic and we call it *approximate factoring*; see Bratley, Fox, and Schrage (1987), L'Ecuyer and Côté (1991) for details. It works if $a^2 < m$ or if $a = \lfloor m/i \rfloor$ where $i^2 < m$, and if all integers between $-m$ and m are well represented on the computer. A second approach computes the product and the division (for the mod operation) directly in *floating-point* arithmetic. On computers that follow the IEEE 64-bit floating-point standard (most computers nowadays), all integers up to 2^{53} are represented *exactly* in floating point, and the floating-point implementation works if $am < 2^{53}$. See L'Ecuyer (1998a) for details and examples. A third approach, recently proposed by Wu (1997), assumes that a is a sum or a difference of a few (say, 2 or 3) powers of 2. The product ax can then be decomposed into a sum of products by powers of 2, which are implemented as left shifts on the computer. A little additional gymnastic takes care of the modulo operation. Wu assumes $m = 2^{31} - 1$, but his method can be generalized to other values of m as well. Construction of "good" generators of this type is under way. The second and third approach appear to be the most efficient on today's computers.

Taking m equal to a power of 2 in the MRG simplifies the implementation, but leads to a much shorter period than a prime m (for $k > 1$) and to major deficiencies (L'Ecuyer 1990, 1998b). This is a bad idea. But a modification of the MRG, with a *carry* or a *borrow*, permits one to use a power-of-2 modulus while keeping a long period and the potential for good properties (Couture and L'Ecuyer 1995, 1997; Marsaglia 1994). The resulting *Multiply-with-Carry* (MWC) generator turns out to be approximately equivalent to an LCG with a large modulus and can be analyzed much in the same way as LCGs from the structural viewpoint.

In (5), each output value is a multiple of $1/m$. To reduce the discretization error, one may construct each u_n from several x_j's, i.e.,

$$u_n = \sum_{j=1}^{L} x_{ns+j-1} m^{-j}, \tag{7}$$

where s and $L \le k$ are positive integers. If (4) has period ρ and $\gcd(\rho, s) = 1$, (7) has period ρ as well. The digital expansion (7) allows smaller m, even $m = 2$. For $m = 2$, u_n is constructed from L successive bits of the binary sequence (4), with spacings of $s - L$ bits between the blocks, and the resulting RNG is called a *linear feedback shift register* (LFSR) or *Tausworthe* generator (Knuth 1997; Niederreiter 1992b; Tausworthe 1965). Its implementation is discussed by Bratley, Fox, and Schrage (1987), Fishman (1996), L'Ecuyer (1996b), and Tezuka (1995).

One can also use L copies of (4) in parallel, with different seeds, and use one copy for each digit of the fractional expansion of u_n. If $\{x_{j,n}\}$ is the jth copy and if $x_{j,n} = x_{n+d_j}$ for all j and n, then

$$u_n = \sum_{j=1}^{L} x_{j,n} m^{-j} = \sum_{j=1}^{L} x_{n+d_j} m^{-j}. \tag{8}$$

If $d_j = (j-1)d$ for some integer d and if $\gcd(d, \rho) = 1$, then $n + d_j = n + (j-1)d = (ns + j - 1)d$ and (8) becomes equivalent to (7) if we replace $\{x_n\}$ by $\{y_n = x_{nd}\}$, which is accomplished by changing the coefficients of (4) appropriately. When (6) and (8) are used with $m = 2$, we obtain the *generalized feedback shift register* (GFSR) generator (Fushimi and Tezuka 1983; Fushimi 1989), usually expressed as

$$X_n = X_{n-r} \oplus X_{n-k}, \tag{9}$$

where $X_n = (x_{1,n}, \ldots, x_{L,n})$ and where $\oplus$ denotes the bitwise exclusive-or.

A generalization of (9) is the *lagged-Fibonacci* generator, where the bitwise $\oplus$ can be replaced by an arbitrary arithmetic or logical operation, such as $+$, $-$, etc., not necessarily bitwise. A popular one is the *additive* generator (Knuth 1997):

$$X_n = (X_{n-r} + X_{n-k}) \bmod m, \tag{10}$$

where $m = 2^L$. 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) of Marsaglia and Zaman (1991), which are also special cases of the MWC. However, the additive, AWC, and SWB generators share an important deficiency: All triples of the form $(u_n, u_{n+k-r}, u_{n+k})$ for the additive generator, and (u_n, u_{n+r}, u_{n+k}) for the AWC/SWB, for $n \ge 0$, lie in only two planes in the three-dimensional unit cube (see L'Ecuyer 1997).

In a series of papers, M. Matsumoto and his coauthors have proposed a nice class of modifications of the GFSR, which they call the *twisted GFSR*. Their modifications increase the period length of the GFSR from $2^k - 1$ to $2^{kL} - 1$, when using kL bits for the state, and improves the structural properties in a significant way. See Matsumoto and Kurita (1994) and Matsumoto and Nishimura (1998) for details.

The *multiple recursive matrix method* of Niederreiter (1995a) provides a general framework that encompasses many of these modifications and variants as special cases.

3. COMBINED GENERATORS

Combining different recurrences can increase the period length and improve the structural properties of generators (Knuth 1997; L'Ecuyer 1994; Marsaglia 1985; Tezuka 1995; Wang and Compagner 1993). But it can also (conceivably) make things worse. So it is important to understand what we are doing from the theoretical viewpoint when we combine generators. Combined MRGs and combined LFSR generators are two classes which have been well analyzed in recent years.

To combine LFSR generators, one can take several full-period LFSR components whose period lengths are relatively prime to each other, and add their outputs bitwise modulo 2 (i.e., by exclusive or). The result is another LFSR generator whose period length is the product of the periods of its components, and whose structure can be analyzed theoretically just like that of a single LFSR generator. GFSR and twisted GFSR generators can be combined in a similar way. Several combined LFSR generators which have the ME property (defined in section 1) are listed in L'Ecuyer (1996b, 1998e), together with fast computer codes.

MRGs can be combined in a similar way by adding their outputs modulo 1, or using a slightly different combination variant (L'Ecuyer 1996a). Again, the combined generator is another MRG with very large period (up to the product of the periods of the components, divided by 2^{J-1} if there are J components) and large modulus (the product of the individual moduli).

In those two cases, combination can be seen as a way of implementing efficiently some RNGs with huge period lengths. Another important advantage is that the RNG can be designed so that the individual components are implemented very efficiently (e.g., have several zero coefficients), whereas the combination has a complicated recurrence and excellent structural properties.

4. LATTICE STRUCTURE

For the MRG (4–5), the set Ψ_t turns out to be equal to the intersection of a lattice L_t with $[0,1)^t$. This means that L_t is the set of all integer linear combinations of t independent vectors in $\mathbb{R}^t$. This also implies that L_t lies on a limited number of equidistant parallel hyperplanes, at a distance d_t apart (Knuth 1997). For Ψ_t to be evenly distributed over $[0,1)^t$, d_t should be small.

One can choose a constant $t_1 > k$ and define the figure of merit

$$M_{t_1} = \min_{t \le t_1} d_t^* / d_t,$$

where d_t^* is an absolute lower bound on d_t, given k and t (see L'Ecuyer 1998d). We seek an M_{t_1} close to 1. L'Ecuyer (1998d) has computed tables of "good" LCGs based on M_8, M_{16}, and M_{32}, for a wide range of values of m. L'Ecuyer (1998a) provides combined MRGs selected via these figures of merit, together with computer implementations.

One may also consider vectors of *non-successive* output values of an RNG. For a fixed set of non-negative integers $I = \{i_1, i_2, \cdots, i_t\}$, put

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

and let $d_t(I)$ be the distance between successive hyperplanes in the lattice generated by $\Psi_t(I)$ and $\mathbb{Z}_m^k/m$. Couture and L'Ecuyer (1994, 1996) and L'Ecuyer and Couture (1997) discuss how to compute $d_t(I)$ in general, and more efficiently for special classes of generators. Their results imply, for instance, the bad structure (which we already mentioned) of certain triples from the AWC/SWB, and additive or subtractive lagged-Fibonacci generators. Entacher (1998) exhibits bad values of $d_t(I)$ for some popular LCGs.

Computing d_t is called the *spectral test*. Why? Getting back to the spectral analysis of Section 1.5, it turns out that if $\Psi_t = L_t \cap [0,1)^t$, the Weyl sums become $S_n(\mathbf{h}) = 1$ if $\mathbf{h} \in L_t^*$ and $S_n(\mathbf{h}) = 0$ otherwise, where $L_t^* = \{\mathbf{h} \in \mathbb{R}^t : \mathbf{h} \cdot \mathbf{u} \in \mathbb{Z} \text{ for all } \mathbf{u} \in L_t\}$ is the *dual lattice* to L_t. It is also known (Dieter 1975; Knuth 1997) that $d_t = \max\{1/\|\mathbf{h}\|_2 : \mathbf{0} \ne \mathbf{h} \in L_t^*\}$, which is (3). Computing d_t amounts to solving a quadratic integer optimization problem. An implementation is described in L'Ecuyer and Couture (1997).

5. NONLINEAR GENERATORS

Arguing that the point structure produced by linear sequences is too regular, some prefer *nonlinear* generators (Eichenauer-Herrmann 1995; Eichenauer-Herrmann and Herrmann 1997; Hellekalek 1995; Niederreiter 1992b; Niederreiter 1995b). Nonlinearity can be introduced by either using a linear transition function f with a nonlinear output function g, or using a nonlinear recurrence. Nonlinear generators are used (and essentially *required*) in the area of cryptology. See, e.g., Blum, Blum, and Schub (1986), Knuth (1997), Lagarias (1993), L'Ecuyer and Proulx (1989). Under reasonable assumptions, they are *provably* better than the linear ones, but only in an asymptotic sense (as the size of the state space grows to infinity).

For equal period lengths, the nonlinear generators actually do much better than the linear ones with respect to the usual statistical tests (Hellekalek 1995; L'Ecuyer 1998c; L'Ecuyer and Hellekalek 1998; Leeb and Wegenkittl 1997). But they are also much slower. At a running speed comparable to that of an acceptable nonlinear RNG, one can find linear generators with period lengths well over 2^{200}, and which pass all the standard empirical tests.

6. STATISTICAL TESTS

Traditionally, applying statistical tests to an RNG has been much like a fishing expedition: Try a number of tests with arbitrary parameter values, and "take a picture" when the RNG fails a test badly. Recently, the author undertook a project where we try to better understand the interaction between specific RNG families and certain empirical tests. Preliminary results are presented in L'Ecuyer and Hellekalek (1998).

As a simple illustration, consider the classical *collision test* (Knuth 1997): Cut the interval $[0,1)$ into d equal segments. This partitions $[0,1)^t$ into $k = d^t$ cubic boxes. Generate n points, independently, in $[0,1)^t$ and let C be the number of times a point falls in a box that already had a point in it. For large k, under $\mathcal{H}_0$, C follows approximately the Poisson distribution with mean $n^2/(2k)$. Now, take an RNG with period ρ, and choose $k \approx \rho$. If Ψ_t is very regular, one may expect C to be much too small. In the worst case, $C = 0$, and then the left p-value of the test would be $p^- = P[C \le 0] \approx e^{-n^2/(2k)}$. So we need *at least* $n = O(\sqrt{k}) = O(\sqrt{\rho})$ points for rejection and then p^- will decrease exponentially fast in n^2. A similar argument shows if Ψ_t is concentrated on a small fraction of the boxes (e.g., half of them), C should be too large, we would also need $n = O(\sqrt{\rho})$ for rejection, and the right p-value would then decrease exponentially fast with n^2. Empirical experiments with LCGs and LFSR generators confirm that this is actually what happens for these generators. With $t = 2$, the best LCGs according to the spectral test all start to fail significantly with $n \approx 8\sqrt{\rho}$.

7. IMPLEMENTATIONS

No RNG can be fully guaranteed against all possible defects. Such a guarantee is impossible. Nevertheless, RNGs designed based on sound theoretical arguments, reasonably well-tested, and fast enough, are available. Among those, we recommend the combined MRGs of L'Ecuyer (1998a), the combined LCGs of L'Ecuyer and Andres (1997), the combined LFSR generators of L'Ecuyer (1996b, 1998e), and the twisted GFSR of Matsumoto and Nishimura (1998). This short list is admitedly biased towards the RNGs that we know best. Additional references and implementations can be found at the URL pages:

`www.iro.umontreal.ca/~lecuyer`

and

`random.mat.sbg.ac.at`

on the internet.

7.1. TO PROBE FURTHER

Our coverage in this paper is only partial. Several important papers are not cited. For more detailed coverages and additional references, we refer the reader to Eichenauer-Herrmann (1995), Eichenauer-Herrmann, Herrmann, and Wegenkittl (1997), Fishman (1996), Hellekalek (1995, 1998a, 1998b), Knuth (1997), L'Ecuyer (1990, 1994, 1998b), L'Ecuyer and Hellekalek (1998) Niederreiter (1992b, 1995b), and Tezuka (1995).

ACKNOWLEDGMENTS

This work has been supported by NSERC-Canada grants # ODGP0110050 and # SMF0169893. The paper was written while the author was visiting the Department of Industrial Engineering at North Carolina State University; thanks to James R. Wilson and Steve Roberts for their help. Thanks also to Peter Hellekalek and Christiane Lemieux for pointing out several mistakes in earlier drafts.

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.

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.

Coveyou, R. R., and R. D. MacPherson. 1967. Fourier analysis of uniform random number generators. *Journal of the ACM*, 14:100–119.

Dieter, U. 1975. How to calculate shortest vectors in a lattice. *Mathematics of Computation*, 29(131):827–833.

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

Eichenauer-Herrmann, J., and E. Herrmann. 1997. Compound cubic congruential pseudorandom numbers. *Computing*, 59:85–90.

Eichenauer-Herrmann, J., E. Herrmann., and S. Wegenkittl. 1997. A survey of quadratic and inversive congruential pseudorandom numbers. In *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*, volume 127 of *Lecture Notes in Statistics*, 66–97. Springer.

Entacher, K. 1998. Bad subsequences of well-known linear congruential pseudorandom number generators. *ACM Transactions on Modeling and Computer Simulation*, 8(1):61–70.

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

Folland, G. B. 1992. *Fourier Analysis and its Applications.* Pacific Grove, California: Wadsworth and Brooks.

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*, 255–262. IEEE Press.

Hellekalek, P. 1998a. Don't trust parallel Monte Carlo! In *Twelfth Workshop on Parallel and Distributed Simulation, May 1998, Banff, Canada*, 82–89, Los Alamitos, California. IEEE Computer Society.

Hellekalek, P. 1998b. On correlation analysis of pseudorandom numbers. In *Monte Carlo and Quasi-Monte Carlo Methods 1996*, volume 127 of *Lecture Notes in Statistics*, 251–265. Springer-Verlag, New York.

Hellekalek, P. 1998c. On the assessment of random and quasi-random point sets. In *Random and Quasi-Random Point Sets*, Lecture Notes in Statistics 138, New York. Springer. To appear.

Hellekalek, P., and H. Niederreiter. 1998. The weighted spectral test: Diaphony. *ACM Transactions on Modeling and Computer Simulation*, 8(1):43–60.

Hickernell, F. J. 1998. Lattice rules: How well do they measure up? In *Random and Quasi-Random Point Sets*, Lecture Notes in Statistics. New York: Springer. To appear.

Knuth, D. E. 1997. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms.* Third 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. 1994. Uniform random number generation. *Annals of Operations Research*, 53:77–120.

L'Ecuyer, P. 1996a. Combined multiple recursive random number 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. 1997. 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. 1998a. Good parameters and implementations for combined multiple recursive random number generators. *Operations Research.* To appear.

L'Ecuyer, P. 1998b. Random number generation. In *The Handbook of Simulation*, ed. J. Banks. Wiley. To appear in Aug. 1998. Also GERAD technical report number G-96-38.

L'Ecuyer, P. 1998c. Random number generators and empirical tests. In *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*, volume 127 of *Lecture Notes in Statistics*, 124–138. New York: Springer.

L'Ecuyer, P. 1998d. A table of linear congruential generators of different sizes and good lattice structure. *Mathematics of Computation.* To appear.

L'Ecuyer, P. 1998e. Tables of maximally equidistributed combined LFSR generators. *Mathematics of Computation.* 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*, 44:99–107.

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):206–217.

L'Ecuyer, P., and P. Hellekalek. 1998. Random number generators: Selection criteria and testing. 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.

Leeb, H. 1995. Random numbers for computer simulation. Master's thesis, University of Salzburg.

Leeb, H., and P. Hellekalek. 1998. Strong and weak laws for the spectral test and related quantities. Submitted.

Leeb, H., and S. Wegenkittl. 1997. Inversive and linear congruential pseudorandom number generators in empirical tests. *ACM Transactions on Modeling and Computer Simulation*, 7(2):272–286.

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., and A. Zaman. 1991. A new class of random number generators. *The Annals of Applied Probability*, 1:462–480.

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

Matsumoto, M., and T. Nishimura. 1998. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, 8(1):3–30.

Niederreiter, H. 1992a. New methods for pseudorandom number and pseudorandom vector generation. In *Proceedings of the 1992 Winter Simulation Conference*, 264–269. IEEE Press.

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

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*, number 106 in Lecture Notes in Statistics, 87–120. New York: Springer-Verlag.

Owen, A. B. 1998. Latin supercube sampling for very high-dimensional simulations. *ACM Transactions of Modeling and Computer Simulation*, 8(1):71–102.

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.

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

Wegenkittl, S. 1995. Empirical testing of pseudorandom number generators. Master's thesis, University of Salzburg.

Wu, P.-C. 1997. Multiplicative, congruential random number generators with multiplier $\pm 2^{k_1} \pm 2^{k_2}$ and modulus $2^p - 1$. *ACM Transactions on Mathematical Software*, 23(2):255–265.

AUTHOR'S BIOGRAPHY

PIERRE L'ECUYER received a Ph.D. in operations research in 1983, from the University of Montréal. He was then a professor with the Computer Science Department at Laval University, Quebec, until 1990, and since then he is at the University of Montreal. He obtained the *E. W. R. Steacie* grant from the Natural Sciences and Engineering Research Council of Canada for 1995–97. His main research topics are uniform random number generation, efficiency improvement, sensitivity analysis, and optimization via simulation. He is an Area Editor for the *ACM Transactions on Modeling and Computer Simulation* and was the Departmental Editor for the Simulation Department of *Management Science* for 1993–98. You can find more at: `www.iro.umontreal.ca/~lecuyer`.

To appear in the *Proceedings of the 1998 Winter Simulation Conference. This version: September 9, 1998.*