Proceedings of the 2005 Winter Simulation Conference
M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, eds.

FAST RANDOM NUMBER GENERATORS BASED ON LINEAR RECURRENCES MODULO 2: OVERVIEW AND COMPARISON

Pierre L'Ecuyer and François Panneton

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

ABSTRACT

Random number generators based on linear recurrences modulo 2 are among the fastest long-period generators currently available. The uniformity and independence of the points they produce, over their entire period length, can be measured by theoretical figures of merit that are easy to compute, and those having good values for these figures of merit are statistically reliable in general. Some of these generators can also provide disjoint streams and substreams efficiently. In this paper, we review the most interesting construction methods for these generators, examine their theoretical and empirical properties, and make comparisons.

1 INTRODUCTION

Given that computers work in binary arithmetic, if we want fast uniform random number generators (RNGs), it seems natural to define them so that they can be implemented in the computer by few elementary operations on bit strings, such as shifts, rotations, exclusive-or's (xor's), and bit masks. Very fast RNGs with huge period length can indeed be constructed this way. Among them, we find the Tausworthe or linear feedback shift register (LFSR), generalized feedback shift register (GFSR), twisted GFSR (TGFSR), Mersenne twister, the WELL, and xorshift generators (Tezuka 1995, L'Ecuyer 1996, Matsumoto and Nishimura 1998, L'Ecuyer and Panneton 2002, L'Ecuyer 2004b, Panneton, L'Ecuyer, and Matsumoto 2005, Panneton and L'Ecuyer 2004a, Panneton 2004). They will be described later in the paper. An important property of all these generators is that they are special cases of a general class of generators whose state evolves according to a (matrix) linear recurrence modulo 2 and the bits that form the output are also determined by a linear transformation modulo 2 applied to the state. Since doing arithmetic modulo 2 can be interpreted as working in the finite field $\mathbb{F}_2$ of cardinality 2, with elements $\{0, 1\}$, we shall refer to this general class as $\mathbb{F}_2$-*linear generators*.

Several widely-used RNGs of this form are *not* statistically reliable, but some well-designed ones are good, reliable, and fast. Which ones? What defects do the others hide? What mathematical tools can be used to analyze and practically assess their quality from a theoretical viewpoint? Is it easy to jump ahead quickly in their sequence in order to split their period into multiple streams and substreams? (Such multiple streams and substreams are now commonly available in the best simulation software and are very convenient, e.g., to obtain parallel RNGs and to support the implementation of variance reduction methods; see Law and Kelton 2000, L'Ecuyer, Simard, Chen, and Kelton 2002.)

These questions are answered in the remainder of this paper, where we summarize the recent developments (over the past 10 years or so) in that area. In the next section, we give a general framework that covers all these $\mathbb{F}_2$-linear generators. We also provide a simple way to jump ahead with these generators and explain how different $\mathbb{F}_2$-linear generators can be combined (via a bitwise xor) to construct larger (and often better-behaved) $\mathbb{F}_2$-linear generators. In Section 3, we discuss the theoretical measures of uniformity and independence that are typically used in practice as figures of merit to assess their quality. These RNGs turn out to have a lattice structure in spaces of polynomials and formal series over $\mathbb{F}_2$. There are counterparts in those spaces of the spectral test, and other lattice-based tests and properties that have been developed for linear congruential generators. Interestingly, these tests are strongly linked with computing the measures of uniformity of $\mathbb{F}_2$-linear generators. Section 4 briefly outlines this theory. In Section 5, we describe several types of $\mathbb{F}_2$-linear generators proposed over the years, show how they fit the general framework, and summarize what we know about their strengths and weaknesses. In Section 6, we compare specific implementations in terms of their speed and (theoretical) figures of merit, and discuss their behavior in empirical statistical tests. Compared with the most widely used RNG that offers multiple streams and substreams in simulation software, the best $\mathbb{F}_2$-linear RNGs are faster by a factor of 2 to 3. Section 7 concludes the paper.

2 $\mathbb{F}_2$-Linear Generators

2.1 General Framework

Consider an RNG defined by a matrix linear recurrence over the finite field $\mathbb{F}_2$, as follows:

$$\mathbf{x}_n = \mathbf{A}\mathbf{x}_{n-1}, \quad (1)$$

$$\mathbf{y}_n = \mathbf{B}\mathbf{x}_n, \quad (2)$$

$$u_n = \sum_{\ell=1}^{w} y_{n,\ell-1} 2^{-\ell} = .y_{n,0}\, y_{n,1}\, y_{n,2} \cdots, \quad (3)$$

where $\mathbf{x}_n = (x_{n,0}, \ldots, x_{n,k-1})^{\mathsf{t}} \in \mathbb{F}_2^k$ is the k-bit *state vector* at step n, $\mathbf{y}_n = (y_{n,0}, \ldots, y_{n,w-1})^{\mathsf{t}} \in \mathbb{F}_2^w$ is the w-bit *output vector* at step n, k and w are positive integers, $\mathbf{A}$ is a $k \times k$ *transition matrix* with elements in $\mathbb{F}_2$, $\mathbf{B}$ is a $w \times k$ *output transformation matrix* with elements in $\mathbb{F}_2$, and $u_i \in [0, 1)$ is the *output* at step n. All operations in (1) and (2) are performed in $\mathbb{F}_2$, i.e., modulo 2. This setting was adopted in L'Ecuyer and Panneton (2002).

The *characteristic polynomial* of the matrix $\mathbf{A}$ is

$$P(z) = \det(z\mathbf{I} - \mathbf{A}) = z^k - \alpha_1 z^{k-1} - \cdots - \alpha_{k-1} z - \alpha_k,$$

where $\mathbf{I}$ is the identity matrix and each α_j is in $\mathbb{F}_2$. This $P(z)$ is also the characteristic polynomial of the linear recurrence

$$x_{n,j} = (\alpha_1 x_{n-1,j} + \cdots + \alpha_k x_{n-k,j}) \mod 2 \quad (4)$$

and it is well-known (Niederreiter 1992, L'Ecuyer 1994) that when the $\mathbf{x}_n$'s obey (1), the sequence $\{x_{n,j},\, i \geq 0\}$ follows the linear recurrence (4) for each j. The sequences $\{y_{n,j},\, n \geq 0\}$, for $0 \leq j < w$, also obey the same recurrence (although they may follow recurrences of shorter order in certain situations, depending on $\mathbf{B}$). In this paper, we assume that $\alpha_k = 1$, so that the recurrence (4) has *order* k and is purely periodic. Its period length is $2^k - 1$ (i.e., maximal) if and only if $P(z)$ is a primitive polynomial over $\mathbb{F}_2$ (Niederreiter 1992, Knuth 1998).

Several popular classes of RNGs fit this framework as special cases, by appropriate choices of the matrices $\mathbf{A}$ and $\mathbf{B}$. Many will be described in Section 5.

2.2 Jumping Ahead

Jumping ahead directly from $\mathbf{x}_n$ to $\mathbf{x}_{n+\nu}$ for a very large integer ν is easy in principle with this type of generator. It suffices to precompute the matrix $\mathbf{A}^\nu \bmod 2$ (this can be done in $O(k^3 \log \nu)$ operations by a standard method) and then multiply $\mathbf{x}_n$ by this binary matrix, modulo 2. The latter step requires $O(k^2)$ operations in general. This approach works fine for relatively small values of k (e.g., up to 100 or so), but becomes excessively slow when k is large. For example, the Mersenne twister of Matsumoto and Nishimura (1998) has $k = 19937$ and the above method is impractical in that case.

For this reason, it is not a good idea in our opinion to construct $\mathbb{F}_2$-linear generator with such (excessively) large periods and values of k. One way to make the jumping-ahead easier is to adopt a combined generator (see Subsection 2.3), for which the ν-step jumping-ahead is done separately for each component.

2.3 Combined $\mathbb{F}_2$-Linear Generators

A very simple way of combining $\mathbb{F}_2$-linear generators is as follows. For some integer $C > 1$, consider C distinct recurrences of the form (1)–(2), where the cth recurrence has parameters $(k, w, \mathbf{A}, \mathbf{B}) = (k_c, w, \mathbf{A}_c, \mathbf{B}_c)$ and state $\mathbf{x}_{c,n}$ at step n, for $c = 1, \ldots, C$. The output of the combined generator at step n is defined by

$$\mathbf{y}_n = \mathbf{B}_1\mathbf{x}_{1,n} \oplus \cdots \oplus \mathbf{B}_C\mathbf{x}_{C,n},$$

$$u_n = \sum_{\ell=1}^{w} y_{n,\ell-1} 2^{-\ell},$$

where $\oplus$ denotes the bitwise exclusive-or (xor) operation. One can show (Tezuka and L'Ecuyer 1991, Tezuka 1995) that the period length ρ of this combined generator is the least common multiple of the period lengths ρ_c of its components and that this combined generator is equivalent to the generator (1)–(3) with $k = k_1 + \cdots + k_C$, $\mathbf{A} = \mathrm{diag}(\mathbf{A}_1, \ldots, \mathbf{A}_C)$, and $\mathbf{B} = (\mathbf{B}_1, \ldots, \mathbf{B}_C)$.

With this method, by selecting the parameters carefully, the combination of $\mathbb{F}_2$-linear generators with characteristic polynomials $P_1(z), \ldots, P_C(z)$ gives yet another $\mathbb{F}_2$-linear generator with characteristic polynomial $P(z) = P_1(z) \cdots P_C(z)$ and period length equal to the product of the period lengths of the components (Tezuka and L'Ecuyer 1991, Wang and Compagner 1993, L'Ecuyer 1996, Tezuka 1995).

So why would we want to combine these generators? We already gave one good reason in the previous subsection: efficient jumping-ahead is easier for a combined generator of order k having several components than for a non-combined generator with the same k. Another important reason is that matrices $\mathbf{A}$ that give very fast implementations typically lead (unfortunately) to poor quality RNGs from the statistical viewpoint, because of a too simplistic structure. Combined generators provide a way out of this dilemma: select simple components that allow very fast implementations and such that the corresponding combined generator has a more complicated structure, good figures of merit from the theoretical viewpoint, and good statistical properties. Many of the best $\mathbb{F}_2$-linear generators are defined via such combinations. We will give specific examples later.

3 QUALITY CRITERIA

In general, good RNGs must have a long period ρ (say, $\rho \approx 2^{200}$ or more), must run fast, should not waste memory (the state should be represented in no more than roughly $\log_2 \rho$ bits of memory), be repeatable and portable (able to reproduce exactly the same sequence in different software/hardware environments), and allow efficient jumping-ahead in order to obtain multiple streams and substreams. But these properties do not suffice to imitate independent random numbers.

Recall that a sequence of random variables $U_0, U_1, U_2, \ldots$ are i.i.d. $U[0,1)$ if and only if for all integers $i \geq 0$ and $t > 0$, the vector $(U_i, \ldots, U_{i+t-1})$ is uniformly distributed over the t-dimensional unit hypercube $[0,1)^t$. Of course, this cannot hold for algorithmic RNGs that have a finite period length. For RNGs that fit our $\mathbb{F}_2$-linear framework, any vector of t successive output values of the generator belongs to the finite set

$$\Psi_t = \{(u_0, \ldots, u_{t-1}) : \mathbf{x}_0 \in \mathbb{F}_2^k\},$$

i.e., the set of output points obtained when the initial state runs over all possible k-bit vectors.

This set Ψ_t always has cardinality 2^k when viewed as a multiset (i.e., if the points are counted as many times as they appear).

If $\mathbf{x}_0$ is drawn at random from the set of k-bit vectors $\mathbb{F}_2^k$, with probability 2^{-k} for each bit vector, then $(u_0, \ldots, u_{t-1})$ is a random vector having the uniform distribution over Ψ_t. Thus, to approximate well the uniform distribution over $[0,1)^t$, Ψ_t must cover the hypercube $[0,1)^t$ very uniformly (L'Ecuyer 1994, L'Ecuyer 2004b). More generally, we may also want to measure the uniformity of sets of the form

$$\Psi_I = \{(u_{i_1}, \ldots, u_{i_t}) \mid \mathbf{x}_0 \in \mathbb{F}_2^k\},$$

where $I = \{i_1, \cdots, i_t\}$ is a fixed set of non-negative integers such that $0 \leq i_1 < \cdots < i_t$. For $I = \{0, \ldots, t-1\}$, we recover $\Psi_t = \Psi_I$

The uniformity of Ψ_I is usually assessed by measures of *discrepancy* between the empirical distribution of its points and the uniform distribution over $[0,1)^t$ (Hellekalek and Larcher 1998, L'Ecuyer and Lemieux 2002, Niederreiter 1992). These measures can be defined in many ways and they are in fact equivalent to goodness-of-fit tests for the multivariate uniform distribution. They must be computable *without enumerating the points*, because the cardinality of Ψ_t makes the enumeration practically infeasible when the period is large enough. For this reason, the uniformity measures are usually tailored to the general structure of the RNG. Measures that are commonly used for $\mathbb{F}_2$-linear RNGs will be described in a moment. The selected discrepancy measure can be computed for each set I in some predefined class $\mathcal{J}$, then these values can be weighted or normalized by factors that may depend on I, and the worst-case (or average) over $\mathcal{J}$ can be adopted as a *figure of merit* used to rank RNGs. The choice of $\mathcal{J}$ and of the weights are arbitrary. Typically, $\mathcal{J}$ would contain sets I such that t and $i_t - i_1$ are rather small. We generally try to optimize this figure of merit when searching (by computer) for concrete RNG parameters.

For $\mathbb{F}_2$-linear generators, the uniformity of the point sets Ψ_I is typically assessed by measures of equidistribution defined as follows (L'Ecuyer 1996, L'Ecuyer and Panneton 2002, L'Ecuyer 2004a, Tezuka 1995). For an arbitrary vector $\mathbf{q} = (q_1, \ldots, q_t)$ of non-negative integers, partition the unit hypercube $[0,1)^t$ into 2^{q_j} intervals of the same length along axis j, for each j. This determines a partition of $[0,1)^t$ into $2^{q_1+\cdots+q_t}$ rectangular boxes of the same size and shape. If a given set Ψ_I has exactly 2^q points in each box of this partition, for an integer q that must satisfy $k - q = q_1 + \cdots + q_t$, we say that Ψ_I is $\mathbf{q}$-*equidistributed*. This means that among the 2^k points $(u_{i_1}, \ldots, u_{i_t})$ of Ψ_I, if we consider all $(k-q)$-bit vectors formed by the q_j most significant bits of u_{i_j} for $j = 1, \ldots, t$, each of the 2^{k-q} possibilities occurs exactly the same number of times. Of course, this is possible only if $q \leq k$.

If Ψ_I is $(\ell, \ldots, \ell)$-equidistributed for some $\ell \geq 1$, it is called (t, ℓ)-*equidistributed* (L'Ecuyer 1996). The largest value of ℓ for which this holds is called the *resolution* of the set Ψ_I and is denoted by ℓ_I. It cannot exceed $\ell_t^* = \min(\lfloor k/t \rfloor, w)$. We define the *resolution gap* of Ψ_I as $\delta_I = \ell_t^* - \ell_I$. Possible figures of merit can then be defined by

$$\Delta_{\mathcal{J},\infty} = \max_{I \in \mathcal{J}} \delta_I \quad \text{and} \quad \Delta_{\mathcal{J},1} = \sum_{I \in \mathcal{J}} \delta_I,$$

where $\mathcal{J}$ is a preselected class of index sets I.

We also denote by t_ℓ the largest dimension t for which Ψ_t is (t, ℓ)-equidistributed, and define the *dimension gap* for ℓ bits of resolution as

$$\tilde{\delta}_\ell = t_\ell^* - t_\ell,$$

where $t_\ell^* = \lfloor k/\ell \rfloor$ is an upper bound on t_ℓ. We may then consider the *worst-case dimension gap* and the *sum of dimension gaps*, defined as

$$\tilde{\Delta}_\infty = \max_{1 \leq \ell \leq w} \tilde{\delta}_\ell \quad \text{and} \quad \tilde{\Delta}_1 = \sum_{\ell=1}^{w} \tilde{\delta}_\ell,$$

as alternative figures of merit for our generators.

When $\tilde{\Delta}_\infty = \tilde{\Delta}_1 = 0$, the RNG is said to be *maximally equidistributed* (ME) or *asymptotically random* for the word size w (L'Ecuyer 1996, Tezuka 1995, Tootill, Robinson, and Eagle 1973). This property ensures perfect equidistribution of all sets Ψ_t, for any partition of the unit hypercube into subcubes of equal sizes, as long as $\ell \leq w$ and the number of subcubes does not exceed the number of points in Ψ_t.

Large-period ME (or almost ME) generators can be found in L'Ecuyer (1999b), L'Ecuyer and Panneton (2002), Panneton and L'Ecuyer (2004b), and Panneton, L'Ecuyer, and Matsumoto (2005), for example.

The $(k-q)$-bit vectors involved in assessing the $\mathbf{q}$-equidistribution of Ψ_I can be expressed as a linear function of the k-bit initial state $\mathbf{x}_0$, that is, as $\mathbf{M_q x}_0$ for some $(k-q)\times k$ binary matrix $\mathbf{M_q}$. Clearly, Ψ_I is $\mathbf{q}$-equidistributed if and only if $\mathbf{M_q}$ has full rank $k-q$. Thus, $\mathbf{q}$-equidistribution can easily be verified by constructing this matrix $\mathbf{M_q}$ and checking its rank via (binary) Gaussian elimination (Fushimi 1983, L'Ecuyer 1996, Tezuka 1995). This is a major motivation for adopting this measure of uniformity.

For very large values of k, the matrix $\mathbf{M_q}$ is expensive to construct and reduce, but a more efficient method based on the computation of the shortest nonzero vector in a lattice of formal series (see Section 4), as explained in Couture and L'Ecuyer (2000), can be used in that case to verify (t,ℓ)-equidistribution.

The figures of merit defined above look at the *most significant bits* of the output values, but give little importance to the least significant bits. We could of course extend them so that they also measure the equidistribution of the least significant bits, simply by using different bits to construct the output values and computing the corresponding $\mathbf{q}$-equidistributions. But this becomes quite cumbersome and expensive to compute in general because there are too many ways of selecting which bits are to be considered. Certain classes of $\mathbb{F}_2$-linear generators (e.g., the Tausworthe/LFSR RNGs defined in Subsection 5.1) have the interesting property that if all output values are multiplied by a given power of two, modulo 1, all equidistribution properties remain unchanged. In other words, they enjoy the nice property that their least significant bits have the same equidistribution as the most significant ones. We call such generators *resolution-stationary*.

Aside from excellent equidistribution, good $\mathbb{F}_2$-linear generators are also required to have characteristic polynomials $P(z)$ whose number of nonzero coefficients is not too far from half the degree, i.e., near $k/2$ (Compagner 1991, Wang and Compagner 1993). In particular, generators for which $P(z)$ is a trinomial or a pentanomial, which have been widely used in the past, must be avoided. They fail rather simple statistical tests (Lindholm 1968, Matsumoto and Kurita 1996). So the fraction of nonzero coefficients in $P(z)$ can be used as a secondary figure of merit.

Other measures of uniformity are popular in the context where k is small and the entire point set Ψ_t is used for quasi-Monte Carlo integration (Niederreiter 1992, Hellekalek and Larcher 1998, L'Ecuyer and Lemieux 2002); for example the smallest q for which Ψ_t is a (q,k,t)-net, the P_α measure and its weighted versions, the diaphony, etc. However, no one knows how to compute these measures efficiently when $k > 50$ (say), which is always the case for good $\mathbb{F}_2$-linear RNGs.

4 LATTICE STRUCTURE IN SPACE OF FORMAL SERIES

The lattice structure of linear congruential generators (LCGs) is well-known in the simulation community (Law and Kelton 2000, Knuth 1998). $\mathbb{F}_2$-linear RNGs do not have a lattice structure in the real space, but they do have a similar form of lattice structure in a space of formal series (Couture and L'Ecuyer 2000, L'Ecuyer 2004a, Lemieux and L'Ecuyer 2003, Tezuka 1995), which we now outline. In comparison with the lattices of LCGs, the real space $\mathbb{R}$ is replaced by the space $\mathbb{L}_2$ of formal power series with coefficients in $\mathbb{F}_2$, of the form $\sum_{\ell=\omega}^{\infty} x_\ell z^{-\ell}$ for some integer ω, and the integers are replaced by polynomials over $\mathbb{F}_2$.

The sequence of values taken by the jth bit of the *output* has *generating function*

$$G_j(z) = y_{0,j}z^{-1} + y_{1,j}z^{-2} + \cdots = \sum_{n=1}^{\infty} y_{n-1,j}z^{-n}.$$

When multiplying this formal series by $P(z)$, we obtain the polynomial $g_j(z) = G_j(z)P(z)$ in $\mathbb{F}_2[z]/P(z)$ (the space of polynomials of degree less than k, with coefficients in $\mathbb{F}_2$), because the successive terms of the series satisfy a recurrence with this characteristic polynomial. For $\ell = 1,\ldots,w$, let $\mathbf{G}^{(\ell)}(z) = (G_0(z),\ldots,G_{\ell-1}(z))$.

If $P(z)$ is an irreducible polynomial and $G_0(z) \neq 0$, then $g_0(z)$ has an inverse modulo $P(z)$. In this case, there is an initial state of the RNG that corresponds to the vector

$$\begin{aligned}\bar{\mathbf{G}}^{(\ell)}(z) &= g_0^{-1}(z)\mathbf{G}_\ell(z) \\ &= (1, g_0^{-1}(z)g_1(z),\ldots,g_0^{-1}(z)g_{\ell-1}(z))/P(z).\end{aligned}$$

When the latter holds, we have the following.

Let $\mathbb{L}_2 = \mathbb{F}_2((z^{-1}))$ the space of formal series of the form $\sum_{n=i}^{\infty} d_{n-1}z^{-n}$ where $i \in \mathbb{Z}$ and $d_{n-1} \in \mathbb{F}_2$ for each n. Let $\mathbb{L}_{2,0}$ those series for which $i \geq 1$. Suppose that the first ℓ lines of the matrix $\mathbf{B}$ are linearly independent. Then the vectors $\mathbf{v}_1(z) = \bar{\mathbf{G}}^{(\ell)}(z)$, $\mathbf{v}_2(z) = \mathbf{e}_2(z),\ldots,\mathbf{v}_\ell(z) = \mathbf{e}_\ell(z)$ form a basis of a *lattice* $\mathcal{L}_\ell$ in $\mathbb{L}_2$, defined by

$$\mathcal{L}_\ell = \left\{ \mathbf{v}(z) = \sum_{j=1}^{\ell} h_j(z)\mathbf{v}_j(z) \text{ such that } h_j(z) \in \mathbb{F}_2[z] \right\}.$$

This lattice is called the ℓ-bit *resolution-wise lattice* associated with the RNG. The matrix $\mathbf{V}$ whose lines are the $\mathbf{v}_j$'s has an inverse $\mathbf{W} = \mathbf{V}^{-1}$ whose columns

$$\begin{aligned}\mathbf{w}_1(z) &= (P(z),0,\ldots,0)^{\mathsf{t}}, \\ \mathbf{w}_2(z) &= (-g_1(z),1,\ldots,0)^{\mathsf{t}}, \\ &\cdots \\ \mathbf{w}_\ell(z) &= (-g_{\ell-1}(z),0,\ldots,1)^{\mathsf{t}}\end{aligned}$$

form a basis of the *dual lattice*

$$\mathcal{L}_\ell^* = \{\mathbf{h}(z) \in \mathbb{L}_b^\ell : \mathbf{h}(z) \cdot \mathbf{v}(z) \in \mathbb{F}_2[z] \text{ for all } \mathbf{v}(z) \in \mathcal{L}_\ell\},$$

where $\mathbf{h}(z) \cdot \mathbf{v}(z) = \sum_{j=1}^{s} h_j(z) v_j(z)$ (the scalar product). This resolution-wise lattice fully describes all the possible output sequences of the RNG via the following:

Theorem (Couture and L'Ecuyer 2000). We have

$$\begin{aligned} & \mathcal{L}_\ell \cap \mathbb{L}_{2,0} \\ = \; & \{(g_0(z), \ldots, g_{\ell-1}(z))/P(z) : g_0(z) \in \mathbb{F}_2[z]/(P(z))\}. \end{aligned}$$

For any $\mathbf{h}(z) = (h_1(z), \ldots, h_\ell(z)) \in (\mathbb{F}_2[z])^\ell$, we may define the *length* of $\mathbf{h}(z)$ by $\|\mathbf{0}\| = 0$ and

$$\log_2 \|\mathbf{h}(z)\| = \max_{1 \le j \le \ell} \deg h_j(z) \quad \text{for } \mathbf{h}(z) \neq \mathbf{0}.$$

Theorem (see Tezuka 1995, Couture and L'Ecuyer 2000, L'Ecuyer 2004a). Ψ_t is (t, ℓ)-equidistributed if and only if

$$\min_{\mathbf{0} \neq \mathbf{h}(z) \in \mathcal{L}_\ell^*} \log_2 \|\mathbf{h}(z)\| > \ell.$$

This theorem shows that checking equidistribution amounts to computing a shortest nonzero vector in the dual lattice $\mathcal{L}_\ell^*$, just like the spectral test commonly applied to LCGs but with a different lattice. As it turns out, very similar algorithms can be used to compute the shortest vector in both cases (Couture and L'Ecuyer 2000). This approach is more efficient than applying Gaussian elimination to the matrix $\mathbf{M_q}$ (see Subsection 3) when t is large.

Some $\mathbb{F}_2$-linear RNGs (e.g., the LFSR generators) also have a *dimension-wise* lattice structure where the lattice contains vectors of t-dimensional formal series, whose coordinate j is the generating function for the binary expansion of the jth output value, for a given initial state (Tezuka and L'Ecuyer 1991, L'Ecuyer 1994, Tezuka 1995, Lemieux and L'Ecuyer 2003). This lattice can also be used to study equidistribution. However, it only applies to a subclass of $\mathbb{F}_2$-linear RNGs.

5 SPECIFIC CLASSES OF GENERATORS

5.1 The LFSR Generator

The *Tausworthe* or *linear feedback shift register* (LFSR) generator (Tausworthe 1965, L'Ecuyer 1996, Tezuka 1995) is defined by a linear recurrence modulo 2, from which a block of w bits is taken every s steps, for some positive integers w and s:

$$x_n = a_1 x_{n-1} + \cdots + a_k x_{n-k} \mod 2, \tag{5}$$

$$u_n = \sum_{\ell=1}^{w} x_{ns+\ell-1} 2^{-\ell}. \tag{6}$$

where $a_1, \ldots, a_k$ are in $\mathbb{F}_2$ and $a_k = 1$. This fits our framework by taking $\mathbf{A} = (\mathbf{A}_0)^s$ (in $\mathbb{F}_2$) where

$$\mathbf{A}_0 = \begin{pmatrix} & 1 & & \\ & & \ddots & \\ & & & 1 \\ a_k & a_{k-1} & \ldots & a_1 \end{pmatrix}, \tag{7}$$

and blank entries in this matrix are zeros. The matrix $\mathbf{B}$ contains the first w lines of the $k \times k$ identity matrix, assuming that $w \le k$.

Note that $P(z)$ is the characteristic polynomial of the matrix $\mathbf{A} = (\mathbf{A}_0)^s$, not that of the recurrence (5), and the choice of s is important for determining the quality of this generator. A frequently encountered case is when a single a_j is nonzero in addition to a_k; then, $P(z)$ is a trinomial and we say that we have a *trinomial-based* LFSR generator. Typically, s is small to make the implementation efficient. These trinomial-based generators are known to have important statistical weaknesses (Matsumoto and Kurita 1996, Tezuka 1995) but they can be used a components of combined RNGs (Tezuka and L'Ecuyer 1991, Wang and Compagner 1993, L'Ecuyer 1996). They also enjoy the important properties of being resolution-stationary.

Tables of specific parameters for maximally equidistributed combined LFSR generators, together with concrete implementations for 32-bit and 64-bit computers, can be found in L'Ecuyer (1999b). These generators are among the fastest ones currently available.

5.2 The GFSR, Twisted GFSR, and Mersenne Twister

Here we take $\mathbf{A}$ as the $pq \times pq$ matrix

$$\mathbf{A} = \begin{pmatrix} & & \mathbf{I}_p & & \mathbf{S} \\ \mathbf{I}_p & & & & \\ & \mathbf{I}_p & & & \\ & & \ddots & & \\ & & & \mathbf{I}_p & \end{pmatrix}$$

for some positive integers p and q, where $\mathbf{I}_p$ is the $p \times p$ identity matrix, $\mathbf{S}$ is a $p \times p$ matrix, and the matrix $\mathbf{I}_p$ on the first line is in columns $(r-1)p + 1$ to rp for some positive integer r. Often, $w = p$ and $\mathbf{B}$ contains the first w lines of the $pq \times pq$ identity matrix. If $\mathbf{S}$ is also the identity matrix, this generator is the trinomial-based *generalized feedback shift register* (GFSR), for which $\mathbf{x}_n$ is obtained by a bitwise exclusive-or of $\mathbf{x}_{n-r}$ and $\mathbf{x}_{n-q}$ (Lewis and Payne 1973). This provides an extremely fast RNG. However, its period length cannot exceed $2^q - 1$, because each bit of $\mathbf{x}_n$ follows the same binary recurrence of order $k = q$, with characteristic polynomial $P(z) = z^q - z^{q-r} - 1$.

More generally, we can define $\mathbf{x}_n$ as the bitwise exclusive-or of $\mathbf{x}_{n-r_1}, \mathbf{x}_{n-r_2}, \ldots, \mathbf{x}_{n-r_d}$ where $r_d = q$, so

that each bit of $\mathbf{x}_n$ follows a recurrence in $\mathbb{F}_2$ whose characteristic polynomial $P(z)$ has $d+1$ nonzero terms. However, the period length is still bounded by 2^q-1, whereas considering the pq-bit state, we should expect a period length close to 2^{pq}. This was the main motivation for the *twisted GFSR* (TGFSR) generator. In the original version introduced by Matsumoto and Kurita (1992), $w=p$ and the matrix $\mathbf{S}$ is defined as the transpose of $\mathbf{A}_0$ in (7), with k replaced by p. The characteristic polynomial of $\mathbf{A}$ is then $P(z)=P_S(z^q+z^m)$, where $P_S(\zeta)=\zeta^p-a_p\zeta^{p-1}-\cdots-a_1$ is the characteristic polynomial of $\mathbf{S}$, and its degree is $k=pq$. If the parameters are selected so that $P(z)$ is primitive over $\mathbb{F}_2$, then the TGFSR has period length 2^k-1. Matsumoto and Kurita (1994) pointed out important weaknesses of the original TGFSR and proposed an improved version that uses a well-chosen matrix $\mathbf{B}$ whose lines differ from those of the identity. The operations implemented by this matrix are called *tempering* and their purpose is to improve the uniformity of the points produced by the RNG.

The *Mersenne twister* (Matsumoto and Nishimura 1998, Nishimura 2000) is a variant of the TGFSR where k is slightly less than pq and can be a prime number. A specific instance proposed by Matsumoto and Nishimura (1998) and named `MT19937` is fast, has the huge period length of $2^{19937}-1$, and has become quite popular. A weakness of this RNG is underlined and illustrated in Panneton, L'Ecuyer, and Matsumoto (2005): if the generator starts in (or reaches) a state that has very few ones, it may take up to several hundred thousands steps before the ratio of ones in the output and/or the average output value are approximately 1/2. For example, for `MT19937`, if we average the output values at steps $n+1$ to $n+100$ (a moving average) and average this over all 19937 initial states $\mathbf{x}_0$ that have a single bit at one, then we need at least $n>700,000$ before the average gets close to 1/2 as it should be (this is graphically illustrated in Panneton, L'Ecuyer, and Matsumoto 2005). Likewise, if two states differ by a single bit, or by only a few bits, a very large number of steps are required on average before the states or the outputs differ by about half of their bits. The source of the problem is that this RNG has a (huge) 19937-bit state and few of these bits are modified from one step to the next. In the terminology of cryptologists, the recurrence has *low diffusion capacity*. This may be linked to the fact that its characteristic polynomial has only 135 nonzero coefficients out of 19938. Moreover, the figure of merit $\tilde{\Delta}_1$ takes the large value 6750 for this generator.

It has been proved that the TGFSR and Mersenne twister construction methods used in Matsumoto and Kurita (1994), Matsumoto and Nishimura (1998) *cannot* provide ME generators in general. They typically have large equidistribution gaps. But combining them via a bitwise xor *can* yield generators with the ME property. Concrete examples of ME combined TGFSR generators with period lengths around 2^{466} and 2^{1250} are given in L'Ecuyer and Panneton (2002). These generators have the additional property that the resolution gaps δ_I are also zero for a class of index sets I of small cardinality and whose elements are not too far apart. These RNGs are of course somewhat slower than their original (uncombined) counterpart.

5.3 The WELL RNGs

These RNGs were developed by Panneton (2004) and are described by Panneton, L'Ecuyer, and Matsumoto (2005). The idea was to "sprinkle" a small number of very simple operations such as xor, shift, bit mask, etc., into the matrix $\mathbf{A}$ in a way that the resulting RNG has maximal period and runs about as fast as the Mersenne twister, but also has (under these constraints) the best possible equidistribution properties, and a characteristic polynomial with around 50% nonzero coefficients.

The state $\mathbf{x}_n=(\mathbf{v}_{n,0}^{\mathsf{t}},\ldots,\mathbf{v}_{n,r-1}^{\mathsf{t}})^{\mathsf{t}}$ is comprised of r blocks of $w=32$ bits $\mathbf{v}_{n,j}$, and the recurrence is defined by a set of linear transformations that apply to these blocks, as described in Panneton, L'Ecuyer, and Matsumoto (2005). Essentially, the transformations modify $\mathbf{v}_{n,0}$ and $\mathbf{v}_{n,1}$ by using several of the other blocks. They are selected so that $P(z)$, a polynomial of degree $k=rw-p$, is primitive over $\mathbb{F}_2$. The output is defined by $\mathbf{y}_n=\mathbf{v}_{n,0}$.

The authors list specific parameters for WELL generators with period lengths ranging from $2^{512}-1$ to $2^{44497}-1$. Many of them are ME and the others are nearly ME. Their characteristic polynomials have nearly 50% coefficients equal to 1. These RNGs have much better diffusion capacity than the Mersenne twister and have comparable speed.

5.4 Xorshift Generators

Marsaglia (2003) has proposed a class of very fast RNGs whose recurrence can be implemented by a small number of xorshift operations only, where a *xorshift operation* consists in replacing a w-bit block in the state by a (left or right) shifted version of itself (by a position, where $0<a<w$) xored with the original block. The constant w is the computer's word size (usually 32 or 64). The specific generators he proposed in his paper use three xorshift operations at each step. As it turns out, xorshifts are linear operations so these generators fit our $\mathbb{F}_2$-linear setting.

Panneton and L'Ecuyer (2004a) analyzed the theoretical properties of a general class of xorshift generators that contains those proposed by Marsaglia. They studied maximal-period conditions, limits on the equidistribution, and submitted xorshift generators to empirical statistical testing. They concluded that three-xorshift generators are unsafe and came up with generators based on 7 and 13 xorshifts, whose speed is only 20% slower than those with three xorshifts to generate $U(0,1)$ real numbers. Aside from the tests that detect $\mathbb{F}_2$-linearity, these RNGs pass other standard statistical tests.

5.5 Linear Recurrences in $\mathbb{F}_{2^w}$

Fix a positive integer w (e.g., $w = 32$) and let $q = 2^w$. Panneton (2004) and Panneton and L'Ecuyer (2004b) consider fast RNGs based on recurrences in the finite field $\mathbb{F}_q$, which can be written as

$$m_n \quad = \quad b_1 m_{n-1} + \cdots + b_r m_{n-r}$$

for some integer r, where the arithmetic is performed in $\mathbb{F}_q$. The maximal period $\rho = 2^{rw} - 1$ is reached if and only if $\tilde{P}(z) = z^r - b_1 z^{r-1} - \cdots - b_{r-1} z - b_r$ is a primitive polynomial over $\mathbb{F}_q$.

To implement this recurrence, these authors select an algebraic element ζ of $\mathbb{F}_q$, take $\{1, \zeta, \ldots, \zeta^{r-1}\}$ as a basis of $\mathbb{F}_q$ over $\mathbb{F}_2$, and represent the elements $m_n = v_{n,0} + v_{n,1}\zeta + \cdots + v_{n,w-1}\zeta^{w-1}$ of $\mathbb{F}_q$ by the bit vectors $\mathbf{v}_n = (v_{n,0}, v_{n,1}, \ldots, v_{n,w-1})^{\mathsf{t}}$. The state of the RNG is thus represented by a rw-bit vector and the output is constructed as in (3), from the bits of $\mathbf{v}_n$. This construction fits our $\mathbb{F}_2$-linear framework (1–3) and generalizes the TGFSR generators. Panneton and L'Ecuyer (2004b) call them *LFSR generators in* $\mathbb{F}_{2^w}$.

The same authors also propose a slightly different construction called *polynomial LCG in* $\mathbb{F}_{2^w}$, and based on the recurrence

$$q_n(z) \quad = \quad z q_{n-1}(z) \mod \tilde{P}(z)$$

in $\mathbb{F}_q[z]$ (the ring of polynomials with coefficients in $\mathbb{F}_q$), where $\tilde{P}(z) \in \mathbb{F}_q[z]$ is a primitive polynomial. To implement this, each coefficient of $q_n(z)$ is represented by a w-bit vector just as for m_n and the output is defined in a similar way. Again, this fits the $\mathbb{F}_2$-linear framework (1–3).

Panneton (2004) (see also Panneton and L'Ecuyer 2005) goes further by proving certain properties of the equidistribution of these RNGs. For instance, he shows that if $\tilde{P}(z)$ is irreducible over $\mathbb{F}_q$ and can be written as

$$\tilde{P}(z) = p_0(z) + \zeta p_1(z) + \cdots + \zeta^{\gamma} p_{\gamma}(z)$$

where each $p_i(z)$ is in $\mathbb{F}_2[z]$, then the RNG cannot be (t, ℓ)-equidistributed if $t > r$ and $\ell > \gamma$. As a special case, since the TGFSR has $\tilde{P}(z) = p_0(z) + \zeta p_1(z)$, it cannot be equidistributed with more than a single bit of resolution in any dimension $t > r$. He also shows that if $\tilde{P}(z)$ is irreducible over $\mathbb{F}_q$ and has at least three nonzero coefficients, then among the $2^{rw} - 1$ two-dimensional point sets $\Psi_{\{0,j\}}$ where $1 \le j < 2^{kw}$, exactly $2^w - 1$ are not $(2, w)$-equidistributed. For example, if $w = 32$ and $r = 25$ (so $k = 800$), only one two-dimensional projection out of 2^{768} is not equidistributed!

Panneton (2004) and Panneton and L'Ecuyer (2004b) propose tables of good parameters for LFSRs and polynomial LCGs in $\mathbb{F}_q$. These parameters were found by computer searches based on the figure of merit $\tilde{\Delta}_1$. They also provide concrete implementations in the C language. These implementations are fast, comparable to the Mersenne twister for instance, but one drawback is that they use precomputed multiplication tables that require a non-negligible amount of memory. (In the case of multiple streams, a single copy of the tables is shared by all the streams.) The output transformation by a non-trivial matrix $\mathbf{B}$ is integrated into these multiplication tables to improve the efficiency.

Table 1: CPU time (sec) to generate 10^9 random numbers, and CPU time to jump ahead 10^6 times, with some RNGs available in SSJ

RNG	$\rho \approx$	gen. time	jump time
LFSR113	2^{113}	31	0.1
LFSR258	2^{258}	35	0.2
WELL512	2^{512}	33	234
WELL1024	2^{1024}	34	917
MT19937	2^{19937}	36	—
MRG31k3p	2^{185}	51	0.9
MRG32k3a	2^{191}	70	1.1

6 PERFORMANCE

Table 1 reports the speed of some RNGs available in the SSJ simulation package (L'Ecuyer and Buist 2005). These timings are for the Java implementations, running on a 2.4 GHz 64-bit computer with SUN's JDK 1.5. The first and second columns give the generator's name and its approximate period length. Column 3 gives the CPU time (sec) to generate 10^9 random numbers and add them up, whereas column 4 gives the CPU time needed to jump ahead 10^6 times by a very large number of steps (to get a new stream). The first five RNGs are $\mathbb{F}_2$-linear and the last two are combined multiple recursive generators (MRGs). The first two are combined LFSRs proposed by L'Ecuyer (1999b) for 32-bit and 64-bit computers, with four and five components, respectively. The two WELL RNGs are proposed in Panneton, L'Ecuyer, and Matsumoto (2005). Other WELL generators with much longer periods (up to nearly 2^{44497}) proposed in that paper have approximately the same speed as those given here to generate random numbers, but are slower for jumping ahead because of their larger value of k. For the Mersenne twister `MT19937`, proposed by Matsumoto and Nishimura (1998), jumping ahead is just too slow to be useful. All these $\mathbb{F}_2$-linear RNGs have roughly the same speed for generating random numbers. Other good ones with about the same speed are also proposed by Panneton and L'Ecuyer (2004b) and Matsumoto and Kurita (1994), e.g., with period lengths near 2^{800}.

The timings of the two MRGs in the table are reported

for comparison. The first one (`MRG31k3p`) was proposed by L'Ecuyer and Touzin (2000) while the second one (`MRG32k3a`) was proposed by L'Ecuyer (1999a) and is used in several simulation packages to provide multiple streams and substreams. This latter RNG has been heavily tested over the years and is very robust. On the other hand, the $\mathbb{F}_2$-linear generators are definitely faster.

Other timings are reported in Panneton, L'Ecuyer, and Matsumoto (2005), this time on a 2.8 GHz 32-bit computer using the C language. In that setting, the $\mathbb{F}_2$-linear RNGs have roughly the same speed (they all require between 30 and 43 CPU seconds to generate 10^9 random numbers) whereas `MRG32k3a` needs nearly 100 seconds.

All the RNGs discussed above have been submitted to empirical statistical testing using the batteries Smallcrush, Crush, and Bigcrush of the `TestU01` package (L'Ecuyer and Simard 2002). They passed all the tests in these batteries with the following notable exceptions: All $\mathbb{F}_2$-linear generators fail the tests that look for linear relationships in the sequences of bits they produce, namely the matrix-rank test (Marsaglia 1985) for huge binary matrices and the linear complexity tests (Erdmann 1992). The reason for this general failure is obvious: We know from their definitions that these generators produce bit sequences that obey linear recurrences, so they cannot have the linear complexity of a truly random sequence. Should this be viewed as a strong limitation of these RNGs for simulation? In my opinion, this is very unlikely to cause a problem in practice, unless the system we simulate has a lot to do with linear dependencies among bits. To make these RNGs safer for such applications without slowing them down too much, we could either combine them with a generator from another class (such as an MRG, for instance), or combine them with a small nonlinear RNG implemented via precomputed tables as suggested by L'Ecuyer and Granger-Piché (2003), or add a nonlinear output transformation that is fast to compute.

7 CONCLUSION

$\mathbb{F}_2$-linear RNGs are convenient for simulation because they are fast and the high-dimensional uniformity of their point sets can be measured by theoretical figures of merit that can be computed efficiently. Combined $\mathbb{F}_2$-linear generators with relatively small components have the important advantage of faster jumping-ahead, because the (smaller) components can be dealt with separately. Some $\mathbb{F}_2$-linear generators proposed in the literature have huge period lengths, but it is not always true that larger is better. A huge state has the disadvantages that it uses more memory (this can be important when there is a large number of streams in a simulation), makes jumping ahead much slower, and requires more operations to modify a large fraction of the bits in the state. Of course, very long bit sequences produced by these generators will always fail statistical tests that measure their linear complexity. This can be viewed as a weak limitation, which could be overcome by adding a nonlinear output transformation or combining the $\mathbb{F}_2$-linear RNG with a generator from another class.

ACKNOWLEDGMENTS

This work has been supported by NSERC-Canada grant No. ODGP0110050 and a Canada Research Chair to the first author. The paper was written while the first author was enjoying the hospitality of Peter Hellekalek at the University of Salzburg, Austria. Much of it was actually written in late evening at Riedel Weinbar, where everything from water to Beerenauslese is served in the world-famous Riedel glasses!

REFERENCES

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

Couture, R., and P. L'Ecuyer. 2000. Lattice computations for random numbers. *Mathematics of Computation* 69 (230): 757–765.

Erdmann, E. D. 1992. Empirical tests of binary keystreams. Master's thesis, Department of Mathematics, Royal Holloway and Bedford New College, University of London.

Fushimi, M. 1983. Increasing the orders of equidistribution of the leading bits of the Tausworthe sequence. *Information Processing Letters* 16:189–192.

Hellekalek, P., and G. Larcher. (Eds.) 1998. *Random and quasi-random point sets*, Volume 138 of *Lecture Notes in Statistics*. New York: Springer.

Knuth, D. E. 1998. *The art of computer programming, volume 2: Seminumerical algorithms*. Third ed. Reading, Mass.: Addison-Wesley.

Law, A. M., and W. D. Kelton. 2000. *Simulation modeling and analysis*. Third ed. New York: McGraw-Hill.

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

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

L'Ecuyer, P. 1999a. Good parameters and implementations for combined multiple recursive random number generators. *Operations Research* 47 (1): 159–164.

L'Ecuyer, P. 1999b. Tables of maximally equidistributed combined LFSR generators. *Mathematics of Computation* 68 (225): 261–269.

L'Ecuyer, P. 2004a. Polynomial integration lattices. In *Monte Carlo and Quasi-Monte Carlo Methods 2002*, ed. H. Niederreiter, 73–98. Berlin: Springer-Verlag.

L'Ecuyer, P. circa 2004b. Uniform random number generation. In *Stochastic Simulation*, ed. S. G. Henderson and B. L. Nelson, Handbooks of Operations Research and

Management Science. Elsevier Science. chapter 3, to appear.

L'Ecuyer, P., and E. Buist. 2005. Simulation in Java with SSJ. In *Proceedings of the 2005 Winter Simulation Conference*. submitted.

L'Ecuyer, P., and J. Granger-Piché. 2003. Combined generators with components from different families. *Mathematics and Computers in Simulation* 62:395–404.

L'Ecuyer, P., and C. Lemieux. 2002. Recent advances in randomized quasi-Monte Carlo methods. In *Modeling Uncertainty: An Examination of Stochastic Theory, Methods, and Applications*, ed. M. Dror, P. L'Ecuyer, and F. Szidarovszky, 419–474. Boston: Kluwer Academic Publishers.

L'Ecuyer, P., and F. Panneton. 2002. Construction of equidistributed generators based on linear recurrences modulo 2. In *Monte Carlo and Quasi-Monte Carlo Methods 2000*, ed. K.-T. Fang, F. J. Hickernell, and H. Niederreiter, 318–330. Berlin: Springer-Verlag.

L'Ecuyer, P., and R. Simard. 2002. *TestU01: A software library in ANSI C for empirical testing of random number generators*. Software user's guide. Available at ⟨http://www.iro.umontreal.ca/~lecuyer⟩.

L'Ecuyer, P., R. Simard, E. J. Chen, and W. D. Kelton. 2002. An object-oriented random-number package with many long streams and substreams. *Operations Research* 50 (6): 1073–1075.

L'Ecuyer, P., and R. Touzin. 2000. Fast combined multiple recursive generators with multipliers of the form $a = \pm 2^q \pm 2^r$. In *Proceedings of the 2000 Winter Simulation Conference*, ed. J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, 683–689. Pistacaway, NJ: IEEE Press.

Lemieux, C., and P. L'Ecuyer. 2003. Randomized polynomial lattice rules for multivariate integration and simulation. *SIAM Journal on Scientific Computing* 24 (5): 1768–1789.

Lewis, T. G., and W. H. Payne. 1973. Generalized feedback shift register pseudorandom number algorithm. *Journal of the ACM* 20 (3): 456–468.

Lindholm, J. H. 1968. An analysis of the pseudo-randomness properties of subsequences of long m-sequences. *IEEE Transactions on Information Theory* IT-14 (4): 569–576.

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. 2003. Xorshift RNGs. *Journal of Statistical Software* 8 (14): 1–6. See ⟨http://www.jstatsoft.org/v08/i14/xorshift.pdf⟩.

Matsumoto, M., and Y. Kurita. 1992. Twisted GFSR generators. *ACM Transactions on Modeling and Computer Simulation* 2 (3): 179–194.

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. 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. 1992. *Random number generation and quasi-Monte Carlo methods*, Volume 63 of *SIAM CBMS-NSF Regional Conference Series in Applied Mathematics*. Philadelphia: SIAM.

Nishimura, T. 2000. Tables of 64-bit Mersenne twisters. *ACM Transactions on Modeling and Computer Simulation* 10 (4): 348–357.

Panneton, F. 2004, August. *Construction d'ensembles de points basée sur des récurrences linéaires dans un corps fini de caractéristique 2 pour la simulation Monte Carlo et l'intégration quasi-Monte Carlo*. Ph. D. thesis, Département d'informatique et de recherche opérationnelle, Université de Montréal, Canada.

Panneton, F., and P. L'Ecuyer. 2004a. On the xorshift random number generators. Manuscript.

Panneton, F., and P. L'Ecuyer. 2004b. Random number generators based on linear recurrences in F_{2^w}. In *Monte Carlo and Quasi-Monte Carlo Methods 2002*, ed. H. Niederreiter, 367–378. Berlin: Springer-Verlag.

Panneton, F., and P. L'Ecuyer. 2005. Infinite-dimensional highly-uniform point sets defined via linear recurrences in F_{2^w}. In *Monte Carlo and Quasi-Monte Carlo Methods 2004*, ed. H. Niederreiter and D. Talay. Submitted.

Panneton, F., P. L'Ecuyer, and M. Matsumoto. 2005. Improved long-period generators based on linear recurrences modulo 2. *ACM Transactions on Mathematical Software*. to appear.

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.

Tootill, J. P. R., W. D. Robinson, and D. J. Eagle. 1973. An asymptotically random Tausworthe sequence. *Journal of the ACM* 20:469–481.

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

AUTHOR'S BIOGRAPHIES

PIERRE L'ECUYER is Professor in the Département d'Informatique et de Recherche Opérationnelle, at the Université de Montréal, Canada. He holds the Canada Research Chair in Stochastic Simulation and Optimization. His main research interests are random number generation, quasi-Monte Carlo methods, efficiency improvement via variance reduction, sensitivity analysis and optimization of discrete-event stochastic systems, and discrete-event simulation in general. He is an Area/Associate Editor for *ACM TOMACS*, *ACM TOMS*, and *Statistics and Computing*. He obtained the prestigious *E. W. R. Steacie* fellowship in 1995-97 and a *Killam* fellowship in 2001-03. His recent research articles are available on-line from his web page: ⟨http://www.iro.umontreal.ca/~lecuyer⟩.

FRANÇOIS PANNETON received his PhD from the the Université de Montréal in 2004. A large part of his thesis was on the topics discussed in this paper. His main research interests are random number generation, the construction of highly-uniform point sets for quasi-Monte Carlo, and computational finance. He now works at the Caisse Centrale Desjardins, a major financial institution in Montreal.