Construction of Equidistributed Generators Based on Linear Recurrences Modulo 2

Pierre L'Ecuyer and François Panneton

Département d'informatique et de recherche opérationnelle
Université de Montréal
C.P. 6128, Succ. Centre-Ville,
Montréal (Québec), H3C 3J7, CANADA
e-mail: lecuyer@iro.umontreal.ca and panneton@iro.umontreal.ca

Abstract. Random number generators based on linear recurrences modulo 2 are widely used and appear in different forms, such as the simple and combined Tausworthe generators, the GFSR, and the twisted GFSR generators. Low-discrepancy point sets for quasi-Monte Carlo integration can also be constructed based on these linear recurrences. The quality of these generators or point sets is usually measured by certain equidistribution criteria. Combining two or more recurrences and adding linear output transformations can be used to improve the equidistribution properties.

In this paper, we explore the combination of recurrences of different types, together with different kinds of linear output transformations. We have developed flexible software tools to make computer searches for good generators with respect to equidistribution criteria that consider selected low-dimensional projections. We provide several examples of search results.

1 Introduction: Generators Based on Linear Recurrences in $\mathbb{F}_2$

We consider (pseudo)random number generators based on a matrix linear recurrence over $\mathbb{F}_2$ (the finite field with the two elements 0 and 1), defined by

$$\mathbf{x}_n = A\mathbf{x}_{n-1}\,, \tag{1a}$$

$$\mathbf{y}_n = B\mathbf{x}_n\,, \tag{1b}$$

$$u_n = \sum_{i=1}^{w} y_{n,i-1}2^{-i}\,, \tag{1c}$$

where $\mathbf{x}_n = (x_{n,0},\ldots,x_{n,k-1})^T \in \mathbb{F}_2^k$ is the k-bit *state vector* at step n, $\mathbf{y}_n = (y_{n,0},\ldots,y_{n,w-1})^T \in \mathbb{F}_2^w$ is the w-bit *output vector* at step n, k and w are positive integers, A is a $k\times k$ matrix called the *transition matrix*, B is a $w\times k$ matrix called the *output transformation matrix*, and $u_n \in [0,1)$ is the output at step n of the recurrence. The operations in (1a) and (1b) are performed in $\mathbb{F}_2$, which is equivalent to performing the arithmetic modulo 2.

It is well-known [15,3] that each coordinate of $\mathbf{x}_n$ follows a linear recurrence of *order* k, of the form

$$x_{n,i} = (\alpha_1 x_{n-1,i} + \cdots + \alpha_k x_{n-k,i}), \tag{2}$$

whose *characteristic polynomial*

$$P(z) = z^k - \alpha_1 z^{k-1} - \cdots - \alpha_{k-1} z - \alpha_k \tag{3}$$

is the characteristic polynomial of the matrix A, i.e., $P(z) = \det(A - zI)$, and the coefficients α_i are elements of $\mathbb{F}_2$. Each coordinate of $\mathbf{y}_n$ also follows the same recurrence.

If we assume that $\alpha_k = 1$, then the recurrence (2) has *order* k and is purely periodic (i.e., has no transient state). Its period length equals $2^k - 1$ (the largest possible value) if and only if $P(z)$ is a primitive polynomial over $\mathbb{F}_2$ [15].

It is customary (e.g., [5,7,6]) to assess the quality of a generator such as (1a)–(1c) by measuring the uniformity of the point set

$$\Psi_t = \{\mathbf{u}_{0,t} = (u_0, \ldots, u_{t-1}) : \mathbf{x}_0 \in \mathbb{F}_2^k\}, \tag{4}$$

which contains all vectors of t successive output values produced by the generator, from all of the 2^k possible initial states. This set is examined for all values of t up to a pre-selected limit. Certain low-dimensional projections over non-successive coordinates can also be examined [10,11]. Here, we measure the uniformity by a standard *equidistribution* criterion detailed in Section 3, and we use a figure of merit defined in terms of the equidistribution properties of all the sets and projections that are considered.

We have developed a flexible and powerful software package for finding generators with a good figure of merit, within various subclasses that satisfy the recurrence (1a)–(1c). The aim of this paper is to illustrate what this package can do and to give examples of its findings, for both small and large values of k.

In Section 2, we mention certain types of generators that are special cases of (1a)–(1c), namely the Tausworthe generator, the polynomial LCG, the twisted GFSR (TGFSR), and combinations of these. Our selection criteria are explained in Sections 3 and 4. Specific linear output transformations are presented in Section 5. Section 6 briefly describes the software package that we have developed. Section 7 gives several examples of generators found with this package. We give examples of small generators whose point sets Ψ_t could be used for quasi-Monte Carlo (QMC) integration, and of large generators that could be used for Monte Carlo simulation. Among the latter, we give three examples of maximally equidistributed combined TGFSR generators. This settles an open question: It has been proved that no simple TGFSR generator can be maximally equidistributed, but it was still unknown whether this was possible for combined TGFSR generators. Finally, we give an example showing the ability of our software to find combined generators with components of different types, with a different linear output transformation applied to each component. This demonstrates the power and the flexibility of the software package.

Our aim here is not to recommend specific generators or QMC point sets, but to explore new territory regarding their potential constructions and selection criteria. Combining generators and tempering the output (when well done) usually improves the statistical qualities of the generator, but at the expense of reducing its speed. So the final choice of a specific generator is a question of compromise.

2 Special cases

In this section, we examine three special cases of the linear recurrence (1a) that can be implemented efficiently. To each of them one can apply any type of output transformation (1b). Further details about the implementation of these generators, their analysis, and the choice of their parameters, can be found in [4,6,14,16,18].

2.1 Tausworthe Generator

Suppose we take $A = A_0^s$ (the sth power of A_0 in $\mathbb{F}_2$) for a given positive integer s, where

$$A_0 = \begin{pmatrix} & 1 & & \\ & & \ddots & \\ & & & 1 \\ a_k & a_{k-1} & \dots & a_1 \end{pmatrix}, \tag{5}$$

$a_1, \dots, a_k$ are in $\mathbb{F}_2$ and $a_k = 1$. The blank entries in the matrix are zeros. If $B = I$ (the identity), this gives a *Tausworthe* generator [17,4,18], which can also be rewritten by $x_n = a_1 x_{n-1} + \cdots + a_k x_{n-k}$ and $u_n = \sum_{i=1}^{w} x_{ns+i-1} 2^{-i}$. The parameters to be selected in this case are k, s, and $a_1, \dots, a_{k-1}$. In the frequent cases where only one [resp., three] of the these a_i is nonzero in addition to a_k, so the characteristic polynomial of A_0 is a trinomial [resp., a pentanomial], we call the Tausworthe generator *trinomial-based* [resp., pentanomial-based].

2.2 Polynomial LCG

Here we take the matrix

$$A = \begin{pmatrix} a_1 & 1 & & \\ \vdots & & \ddots & \\ a_{k-1} & & & 1 \\ a_k & & & \end{pmatrix}. \tag{6}$$

where $a_1, \dots, a_k$ are in $\mathbb{F}_2$ and $a_k = 1$. The characteristic polynomial of this matrix is given by (3) with $\alpha_i = a_i$. With $B = I$, this gives the *polynomial LCG* described in [9,16]. To implement the multiplication by A in this case, it suffices to shift $\mathbf{x}_{n-1}$ to the left by one bit, and make a bitwise exclusive-or with $\mathbf{a} = (a_1, \dots, a_k)^T$ if the original leftmost bit of $\mathbf{x}_{n-1}$ was 1.

2.3 Twisted GFSR

Suppose we take A as the $pq \times pq$ matrix

$$A = \begin{pmatrix} & & I_p & & S \\ I_p & & & & \\ & I_p & & & \\ & & \ddots & & \\ & & & I_p & \end{pmatrix} \tag{7}$$

where p and q are positive integers, I_p is the $p \times p$ identity matrix, S is a $p \times p$ matrix, and the matrix I_p on the first line is in the $(m-1)$th block of $p \times p$ matrices from the right (i.e., its entries are from $(1, (m-1)p+1)$ to (p, mp)). Usually, for ease of implementation, the matrix S is defined as

$$S = \begin{pmatrix} & & a_1 \\ 1 & & a_2 \\ & \ddots & \vdots \\ & 1 & a_p \end{pmatrix} \tag{8}$$

where $a_1, \dots, a_p$ are in $\mathbb{F}_2$ and $a_1 = 1$. We shall assume that it is the case for the remainder of this paper. Then, $k = pq$ and the characteristic polynomial of A is $P(z) = P_S(z^q + z^m)$ where $P_S(z) = z^p - a_p z^{p-1} - \cdots - a_1$ is the characteristic polynomial of S. The parameters to be selected are p, q, m, and $\mathbf{a} = (a_1, \dots, a_p)$. With $w = p$ and B made from the first p lines of the $pq \times pq$ identity matrix, this gives the original TGFSR generator introduced in [13].

2.4 Combined generators

One can combine J distinct recurrences of the form (1a)–(1b) as follows. For $j = 1, \dots, J$, let A_j be the $k_j \times k_j$ transition matrix, B_j the $w \times k_j$ output transformation matrix, and $\mathbf{x}_{j,n}$ the k_j-bit state vector at step n, for generator j. We define the output of the combined generator at step n by

$$\mathbf{y}_n = B_1 \mathbf{x}_{1,n} \oplus \cdots \oplus B_J \mathbf{x}_{J,n}, \tag{9}$$

$$u_n = \sum_{i=1}^{w} y_{n,i-1} 2^{-i} \tag{10}$$

where $\oplus$ denotes the bitwise exclusive-or operation. One can show (see [18]) that the period length ρ of the combined generator is the least common multiple of the period lengths ρ_j of its individual components, i.e., $\rho = \mathrm{lcm}(\rho_1, \ldots, \rho_J)$. It is also easily seen that the combined generator (9)–(10) is equivalent to (1a)–(1c) with $k = \sum_{j=1}^{J} k_j$, $A = \mathrm{diag}(A_1, \ldots, A_J)$, and $B = (B_1, \ldots, B_J)$.

3 Equidistribution

For a given integer $\ell \geq 0$, if we partition each axis of the unit hypercube $[0,1)^t$ into 2^ℓ equal parts, this determines a partition of the hypercube into $2^{\ell t}$ small cubes of equal volume. The point set Ψ_t (and the corresponding RNG) is called (t,ℓ)*-equidistributed*, or t*-distributed with* ℓ *bits of accuracy*, if each of these small cubes contains exactly $2^{k-\ell t}$ points from Ψ_t. This means that if we consider the ℓ most significant bits of the t coordinates of $\mathbf{u}_{0,t}$, the $2^{\ell t}$ different bit vectors that can be constructed appear exactly the same number of times in Ψ_t. For a given dimension t, the largest value of ℓ for which Ψ_t is (t,ℓ)-equidistributed is called the *resolution in dimension* t and is denoted by ℓ_t. This value has the upper-bound $\ell_t^* = \min(\lfloor k/t \rfloor, w)$.

The *resolution gap in* t *dimensions*, defined as

$$\delta_t = \ell_t^* - \ell_t, \tag{11}$$

gives the difference between the best possible resolution in t dimensions and the one that is achieved. If $\delta_t = 0$ (i.e., Ψ_t is t-distributed with ℓ_t^* bits of accuracy) for $1 \leq t \leq k$, the RNG is called *asymptotically random* or *maximally equidistributed* (ME) for the word size w (see [4,18,20]). An ME generator has the best possible equidistribution for Ψ_t, for partitions of the unit hypercube $[0,1)^t$ into cubic boxes of equal size, for all $\ell \leq w$ and $t\ell \leq k$. In particular, a generator for which A has full rank k and $B = I$ always has $\delta_1 = 0$.

To verify the equidistribution of Ψ_t in general, one can write a system of linear equations that expresses the ℓt bits of interest as a linear transformation of the binary vector $\mathbf{x}_0$. One has t-distribution to ℓ bits of accuracy if and only if the matrix of this linear transformation has full rank [4,6].

4 Equidistribution of the Projections and a Selection Criterion

The measure of equidistribution of the previous section can be also applied to point sets formed by non-successive output values, as follows. For arbitrary integers $0 \leq i_1 < \ldots < i_t$, let $I = \{i_1, i_2, \ldots, i_t\}$ and ℓ_I denote the resolution of the set

$$\Psi_I = \{(u_{i_1}, u_{i_2}, \ldots, u_{i_t}) : \mathbf{x}_0 \in \mathbb{F}_2^k\}. \tag{12}$$

This set is the projection of Ψ_{i_t+1} over the t-dimensional subspace determined by the coordinates in I. In particular, for $I = \{0, 1, \ldots, t-1\}$, we have $\ell_I = \ell_t$. The largest possible value of ℓ_I is again ℓ_t^*, and we define the *resolution gap* as $\delta_I = \ell_t^* - \ell_I$. We also define

$$\begin{aligned} S(1, s) &= \{\{0, 1, \ldots, i\} : i < s\}, \\ S(t, s) &= \{\{i_1, \ldots, i_t\} : 0 = i_1 < \ldots < i_t < s\} \quad \text{for } t \geq 2, \end{aligned}$$

and

$$\tilde{\delta}(t, s) = \max_{I \in S(t,s)} \delta_I \quad \text{for } t \geq 1.$$

Our search for good generators will be based on the following criterion (13), introduced and motivated in [10] (see also [8,11,12]). For selected positive integers $s_1, \ldots, s_d$, define

$$\Delta(s_1, \ldots, s_d) = \max_{1 \leq t \leq d} \tilde{\delta}(t, s_t) = \max_{1 \leq t \leq d} \max_{I \in S(t, s_t)} \delta_I. \tag{13}$$

In this criterion, $\tilde{\delta}(1, s_1)$ measures the uniformity of the successive output values up to s_1 dimensions, whereas $\tilde{\delta}(t, s_t)$, for $t \geq 2$, measures the uniformity of the t-dimensional projections over coordinates that are all in the range from 0 to $s_t - 1$. A value of 0 means that the points have the best possible equidistribution for all the projections that are considered. Using $d = 1$ means that we look only at the successive output values, as in Section 3. In particular, a generator is ME if and only if $\Delta(k) = 0$. Using a larger d and larger values of the parameters $s_1, \ldots, s_d$ means that we examine a larger number of projections. Then, the computations become more costly and it also becomes more difficult (sometimes impossible) to find generators with $\Delta(s_1, \ldots, s_d)$ equal or close to 0, because the maximum is taken over too many projections. The choice of parameters $d, s_1, \ldots, s_d$ is therefore a question of compromise and is arbitrary.

5 Linear Output Transformations

This section gives two examples for the choice of the matrix B for which a very fast implementation is available. Many others can be defined, e.g., by using arbitrary combinations of shifts, bitwise AND and XOR operations, and bit masks. The major role of the output transformation by the matrix B is to improve the equidistribution, i.e., reduce the value of (13). Further details about the transformations presented here can be found in [9,14,16].

5.1 MK-style Tempering

Matsumoto and Kurita [14] have proposed a slight variant of the linear output transformation defined by the operations:

$$\tilde{\mathbf{x}}_n = \mathbf{x}_n \ \& \ \mathbf{m}_v;$$

$$\mathbf{z}_n = \tilde{\mathbf{x}}_n \oplus ((\tilde{\mathbf{x}}_n \ll s_1) \ \& \ \mathbf{b}_1);$$
$$\tilde{\mathbf{y}}_n = \mathbf{z}_n \oplus ((\mathbf{z}_n \ll s_2) \ \& \ \mathbf{b}_2);$$
$$\mathbf{y}_n = \tilde{\mathbf{y}}_n \ \& \ \mathbf{m}_w$$

where $\ll s$ means a bitwise shift by s_1 bits to the left, & $\mathbf{b}$ means a bitwise AND with $\mathbf{b}$, s_1 and s_2 are positive integers, $\mathbf{b}_1$ and $\mathbf{b}_2$ are arbitrary bit masks, and $\mathbf{m}_v$ [resp. $\mathbf{m}_w$] is a bit mask that keeps only the first v [resp. w] bits. In the original version of [14], $v = w$. The variant given here applies the tempering to the first v bits of the state $\mathbf{x}_n$ instead of to the first w bits only. Matsumoto and Kurita [14] suggest taking s_2 near $v/2$ and s_1 near $s_2/2$. The parameters s_1, s_2, $\mathbf{b}_1$, and $\mathbf{b}_2$ are to be chosen in a way that the equidistribution be as good as possible. In our software, the bit masks $\mathbf{b}_i$ are constructed adaptively via a decision tree, in order to reduce the gaps δ_t.

5.2 Permutation

This transformation obtains $\mathbf{y}_n$ simply by applying a fixed permutation π to the coordinates of $\mathbf{x}_n$, as suggested in [1,2,9]. The permutation is any one-to-one mapping $\pi : \mathbb{Z}_k \to \mathbb{Z}_k$ and the transformation puts $y_{n,i} = x_{n,\pi(i)}$ for $i = 0, \dots, k$, assuming that $w = k$. The matrix B that corresponds to this permutation is the square matrix obtained by applying the same permutation π to the rows of the identity matrix I. This matrix is invertible and satisfies $BB^T = I$, so the recurrence can be written directly in terms of the $\mathbf{y}_n$'s:

$$\mathbf{y}_n = (BAB^T)\mathbf{y}_{n-1}. \tag{14}$$

In practice, one would choose the parameters so that the recurrence (14) can be implemented directly in an efficient way. The case where $w < k$ can be dealt with easily by applying a bit mask to $\mathbf{y}_n$. In our software and in Section 7.4, we use permutations of the form $\pi(i) = (\nu i + \mu) \bmod k$. Fushimi [1,2] proposed using a permutation of the bits to improve the equidistribution of a GFSR generator; in that case, the permutation changes nothing to the generator's recurrence, but changes its initialization. He also suggested optimization algorithms to find good permutations.

6 The Software Package

We have developed a software package, written in the C language, to analyze the equidistribution of single and combined generators of the form (1a)–(1c), and search for good generators of that form in terms of the criterion (13). The package lets us combine generators of different types (including all those mentioned in Section 2), and apply different linear ouput transformations (including those of Section 5) to each component. The software has a modular design and it is easy to add new types of generators and output transformations.

For any given generator, the program computes the gaps δ_I that appear in the criterion (13), for values of d and $s_1, \ldots, s_d$ specified by the user. For combined generators, the user specifies only the individual components and the output transformations applied to them. The software can also analyze in a single run a whole family of generators, specified by giving the types of the components and output transformations and by putting constraints on their parameters. It then provides a list of the best parameters with respect to the selected criterion $\Delta(s_1, \ldots, s_d)$, or a list of those whose criterion value, or sum of gaps, does not exceed a given constant, or more generally a list of those whose gaps satisfy a given set of constraints. For large families, a random search is made for good generators within the family. In certain cases, the program can also construct the output transformations intelligently in an adaptive manner, in order to find the good generators more rapidly.

7 Good Generators Found With the Software Package

In this section, we give examples of generators found by our programs and which perform well against the criterion $\Delta(32, 24, 16, 8)$. The choice of parameters in this criterion is of course arbitrary, but reasonable (see [8,11]). We start with combined Tausworthe-type generators, first with small ones that could eventually be used for QMC integration, then larger ones (i.e., with longer period). We then give examples of combined TGFSR generators for which $\Delta(32, 24, 16, 8) = 0$. Finally, in the last subsection, we give an example of a hybrid combined generator.

7.1 Tausworthe-Type Generators for QMC

Tables 1 and 2 give combined Tausworthe generators with 2 or 3 components, with values of k from 11 to 23. In all cases, the cardinality of Ψ_t is 2^k and the period length of the main cycle is $\rho = \prod_{j=1}^{J}(2^{k_j} - 1)$, which is close to 2^k.

In Table 1, for each component generator j, the characteristic polynomial of the matrix A_0 in (5) is a trinomial of the form $P_j(z) = z^{k_j} + z^{q_j} + 1$ where $k_j > q_j > 0$. For the generators of Table 2, we have $J = 2$, and the characteristic polynomial of A_0 is a trinomial for the first component and a pentanomial of the form $P_j(z) = z^{k_j} + z^{q_{j,1}} + z^{q_{j,2}} + z^{q_{j,3}} + 1$ where $k_j > q_{j,1} > q_{j,2} > q_{j,3} > 0$ for the second component. We denote by s_j the value of s for the jth component. In both tables, the column marked "gaps" gives the values of $\tilde{\delta}(1, 32)$, $\tilde{\delta}(2, 24)$, $\tilde{\delta}(3, 16)$, and $\tilde{\delta}(4, 8)$. For $t \geq 2$, the smaller the value of the $\tilde{\delta}(t, s_t)$, the better the generator behaves with respect to the t-dimensional projections. The gap never exceeds 1 for all the generators in the two tables, and all the generators of Table 2 are ME.

Table 1. Trinomial-based Tausworthe generators

J	k	k_j	q_j	s_j	gaps	$\Delta(32, 24, 16, 8)$
2	11	5	2	3	0,1,1,1	1
		6	1	5		
2	17	7	1	5	1,1,1,1	1
		10	3	7		
		7	3	2		
3	23	5	2	1	1,1,1,1	1
		11	2	7		

Table 2. Trinomial-pentanomial-based Tausworthe generators

J	k	k_j	$q_{j,1}$	$q_{j,2}$	$q_{j,3}$	s_j	gaps	$\Delta(32, 24, 16, 8)$
2	11	5	2			3	0,1,1,0	1
		6	4	3	1	1		
2	17	6	1			1	0,1,1,1	1
		11	4	2	1	4		
2	19	6	1			4	0,1,1,1	1
		13	5	2	1	5		
2	23	4	1			2	0,1,1,1	1
		19	6	4	1	7		

7.2 Larger Tausworthe-Type Generators

Several combined trinomial-based Tausworthe generators have been suggested in references [19,4,6], based on the criterion $\Delta(k)$ only. Some of these generators are given in Table 3, where we also give the value of $\Delta(32, 24, 16, 8)$ and the gaps for the corresponding projections. We see that some of these gaps are quite large for the three generators of [19]. This means that for certain low-dimensional projections, the resolution $\ell_t(I)$ is rather small. For example, the third generator has $\tilde{\delta}(3, 16) = 6$ and $\ell_3^* = k/3 = 20$, which means that at least one of its 3-dimensional projections, over coordinates $(0, i_2, i_3)$ with $0 < i_2 < i_3 < 16$, has 14 bits of resolution instead of the 20 bits given by the upper bound. The generator of [4] is ME but has a gap of 3 in three dimensions, whereas the two generators taken from [6] are excellent with respect to all the selected projections: no gap exceeds 1.

To see how one could improve on this, we made new searches for good generators of the same form with respect to the criterion $\Delta(32, 24, 16, 8)$. Part of the results are in Table 4. We have improved upon the previously proposed generators for k around 60 and around 88. For $k = 113$, on the other hand, we found no improvement upon the (already excellent) generators of [6] given in Table 3, even when we tried 4-component pentanomial-based generators.

Table 3. Generators Proposed in [19], [4] and [6].

J	k	k_j	q_j	s_j	gaps	$\Delta(32, 24, 16, 8)$
proposed in [19]:						
2	60	29	2	20	1,3,2,3	3
		31	13	1		
2	60	29	2	17	1,2,4,3	4
		31	13	12		
2	60	29	2	17	1,2,6,3	6
		31	3	21		
proposed in [4]:						
		29	2	4		
3	88	28	3	17	0,0,3,2	3
		31	13	12		
proposed in [6]:						
		31	6	18		
4	113	29	2	2	0,0,0,1	1
		28	13	7		
		25	3	13		
		31	6	24		
4	113	29	2	3	0,0,0,1	1
		28	13	11		
		25	3	12		

Table 4. New Combined Tausworthe Generators

J	k	k_j	q_j	s_j	gaps	$\Delta(32, 24, 16, 8)$
2	59	28	9	16	1,1,1,1	1
		31	6	18		
		29	2	7		
3	83	23	5	16	0,0,1,1	1
		31	6	24		
		29	2	21		
3	88	28	9	16	1,0,1,1	1
		31	3	28		

7.3 Combined TGFSR Generators

For a TGFSR generator without tempering, the 2-dimensional resolution ℓ_2 cannot exceed p, which is quite small compared with the upper bound $\ell_2^* = \lfloor pq/2 \rfloor$. With the MK-tempering proposed in [14], one can have $\ell_2 = \ell_2^*$, but it has been proved that ℓ_t cannot reach ℓ_t^* for certain values of t larger than 2, so it is still impossible for the generator to be ME.

Table 5. 2 and 3-Components TGFSR with MK-tempering

J	k	$\mathbf{a}_j$	p_j	q_j	m_j	$s_{j,1}$	$s_{j,2}$	$\mathbf{b}_{j,1}$	$\mathbf{b}_{j,2}$
2	718	9b6bf432	31	11	2	6	14	0568a302	6bf50008
		9a911d68	29	13	2	7	14	21452808	4e2a0000
3	466	cfae8af3	32	7	6	7	15	26ba6501	3a818006
		f94aba8e	31	5	3	7	15	19382200	73b60000
		fea4abc8	29	3	2	7	14	02541800	16540000
3	1250	d84be803	32	13	7	7	15	26a68400	432a8000
		9b6bf432	31	11	2	7	15	5c941200	194f0006
		bdfee2f8	29	17	13	7	14	50280008	2aaa0000

Here we show, by giving concrete examples, that it is possible to construct ME combined generators where each component is a TGFSR as in Section 2.3 with MK-tempering as in Section 5.1. Examples of such combined generators are given in Table 5. Each line gives the parameters of one component. For the MK-tempering, we use $v_j = p_j$ and $w = \min_{1 \le j \le J} p_j$. The bit vectors in Table 5 are given in hexadecimal notation. For the first generator, with $J = 2$ and $k = 718$, one has $\Delta(32, 24, 16, 8) = 1$ and the gaps are $1, 0, 0, 0$. The other two, with $J = 3$, have $\Delta(32, 24, 16, 8) = 0$, i.e., perfect equidistribution for all the projections considered!

In our search for good parameters, to construct the best possible bit vectors $\mathbf{b}_{j,1}$ and $\mathbf{b}_{j,2}$ for the tempering of the individual components, given their values of $s_{j,1}$, $s_{j,2}$, and w, we used a version of the algorithm proposed in [14], adapted to combined generators.

7.4 A Hybrid Generator

Table 6. MK-temperings for the hybrid generator

Components	$s_{j,1}$	$s_{j,2}$	$\mathbf{b}_{j,1}$	$\mathbf{b}_{j,2}$	v_j	w_j
Polynomial	7	15	32660000	330f8000	32	32
Tausworthe	7	14	13892008	4b8e0000	29	29
TGFSR	7	15	3a049102	6cc60000	31	31

To illustrate the flexibility of our software, we now give the results of a search for a hybrid combined generator with three components of different types, and where a different kind of tempering is applied to each component. The first component is specified as a polynomial LCG to which a permutation of the coordinates and an MK-tempering are applied successively. The second component is a trinomial-based Tausworthe generator with MK-tempering.

The third component is a TGFSR with MK-tempering. Their orders are fixed to $k_1 = 32$, $k_2 = 29$, and $k_3 = 93$, respectively, so $k = 154$. The program searched for generators of this form with respect to the criterion $\Delta(32, 24, 16, 8)$ and found several ones with maximal gap of 1. One of them, whose gap values $\tilde{\delta}(t, s_t)$ are $0, 0, 0, 1$, is defined as follows. Its first component is a polynomial LCG with characteristic polynomial $P(z) = z^{32} + z^{30} + z^{25} + z^{24} + z^{22} + z^{15} + z^6 + z^2 + 1$, with the permutation $\pi(i) = 21i + 19$. Its second component is a Tausworthe generator with $k_j = 29$, $q_j = 2$, and $s_j = 1$. Its third component is a TGFSR with $\mathbf{a}_j = \texttt{c39bde7a}$, $p_j = 31$, $q_j = 3$, and $m_j = 1$. Table 6 gives the MK-temperings applied to these three components. The bit vectors are in hexadecimal notation in this table.

8 Acknowledgments

This work has been supported by NSERC-Canada grant No. ODGP0110050 and FCAR-Québec Grant No. 00ER3218 to the first author, and by NSERC-Canada and FCAR-Québec scholarships to the second author.

References

1. M. Fushimi. Increasing the orders of equidistribution of the leading bits of the Tausworthe sequence. *Information Processing Letters*, 16:189–192, 1983.
2. M. Fushimi. Designing a uniform random number generator whose subsequences are k-distributed. *SIAM Journal on Computing*, 17(1):89–99, 1988.
3. P. L'Ecuyer. Uniform random number generation. *Annals of Operations Research*, 53:77–120, 1994.
4. P. L'Ecuyer. Maximally equidistributed combined Tausworthe generators. *Mathematics of Computation*, 65(213):203–213, 1996.
5. P. L'Ecuyer. Uniform random number generators. In *Proceedings of the 1998 Winter Simulation Conference*, pages 97–104. IEEE Press, 1998.
6. P. L'Ecuyer. Tables of maximally equidistributed combined LFSR generators. *Mathematics of Computation*, 68(225):261–269, 1999.
7. P. L'Ecuyer and P. Hellekalek. Random number generators: Selection criteria and testing. In P. Hellekalek and G. Larcher, editors, *Random and Quasi-Random Point Sets*, volume 138 of *Lecture Notes in Statistics*, pages 223–265. Springer, New York, 1998.
8. P. L'Ecuyer and C. Lemieux. Variance reduction via lattice rules. *Management Science*, 46(9):1214–1235, 2000.
9. P. L'Ecuyer and F. Panneton. A new class of linear feedback shift register generators. In J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, editors, *Proceedings of the 2000 Winter Simulation Conference*, pages 690–696, Pistacaway, NJ, 2000. IEEE Press.
10. C. Lemieux. *L'utilisation de règles de réseau en simulation comme technique de réduction de la variance*. PhD thesis, Université de Montréal, May 2000.
11. C. Lemieux and P. L'Ecuyer. Randomized polynomial lattice rules for multivariate integration and simulation. submitted for publication, 2001.

12. C. Lemieux and P. L'Ecuyer. Selection criteria for lattice rules and other low-discrepancy point sets. *Mathematics and Computers in Simulation*, 55(1–3):139–148, 2001.
13. M. Matsumoto and Y. Kurita. Twisted GFSR generators. *ACM Transactions on Modeling and Computer Simulation*, 2(3):179–194, 1992.
14. M. Matsumoto and Y. Kurita. Twisted GFSR generators II. *ACM Transactions on Modeling and Computer Simulation*, 4(3):254–266, 1994.
15. H. Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods*, volume 63 of *SIAM CBMS-NSF Regional Conference Series in Applied Mathematics*. SIAM, Philadelphia, 1992.
16. F. Panneton. Générateurs de nombres aléatoires utilisant des récurrences linéaires modulo 2. Master's thesis, Département d'informatique et de recherche opérationnelle, Université de Montréal, 2000.
17. R. C. Tausworthe. Random numbers generated by linear recurrence modulo two. *Mathematics of Computation*, 19:201–209, 1965.
18. S. Tezuka. *Uniform Random Numbers: Theory and Practice*. Kluwer Academic Publishers, Norwell, Mass., 1995.
19. S. Tezuka and P. L'Ecuyer. Efficient and portable combined Tausworthe random number generators. *ACM Transactions on Modeling and Computer Simulation*, 1(2):99–112, 1991.
20. J. P. R. Tootill, W. D. Robinson, and D. J. Eagle. An asymptotically random Tausworthe sequence. *Journal of the ACM*, 20:469–481, 1973.