MAXIMALLY EQUIDISTRIBUTED COMBINED TAUSWORTHE GENERATORS

PIERRE L'ECUYER

ABSTRACT. Tausworthe random number generators based on a primitive trinomial allow an easy and fast implementation when their parameters obey certain restrictions. However, such generators, with those restrictions, have bad statistical properties unless we combine them. A generator is called maximally equidistributed if its vectors of successive values have the best possible equidistribution in all dimensions. This paper shows how to find maximally equidistributed combinations in an efficient manner, and gives a list of generators with that property. Such generators have a strong theoretical support and lend themselves to very fast software implementations.

1. Introduction

Theoretical and empirical investigations suggest that random number generators based on a too simple linear recurrence, such as simple linear congruential generators or trinomial-based Tausworthe or GFSR generators, should be avoided, and that combined generators should be used instead [1, 5, 7, 13]. In this paper, we examine a class of combined Tausworthe generators first introduced by Tezuka [10] and propose specific instances which have the best possible equidistribution properties (relative to their period length) in all dimensions.

More specifically, suppose that we partition the t-dimensional unit hypercube into $2^{t\ell}$ cubic cells of equal sizes, and place into these cells the t-dimensional points (vectors of successive values) produced by the generator over its entire period. The largest value of ℓ for which each cell contains the same number of points is called the *resolution* in dimension t, and is denoted by ℓ_t. The generator is called *maximally equidistributed* (ME) if ℓ_t reaches its largest possible value in all dimensions, that is, $\ell_t = \min(L, \lfloor k/t \rfloor)$ for $t = 1, \ldots, k$, where L is a constant that represents the number of bits of resolution that are retained on the target computer. A ME generator is also called *collision-free* (CF) if it has the additional property that for $L \geq \ell > \ell_t$ and all t, none of the $2^{t\ell}$ cells contains more than a single point. (All of this is explained in further details in the next section.) The values of ℓ_t can be computed as explained in [2, 10], by exploiting the fact that they are related to the lengths of the shortest vectors in certain lattices. However, the computations described in [2] provide more information than just ℓ_t, and the aim of this paper is to describe a much more efficient algorithm for computing the resolution ℓ_t for all t and checking for maximal equidistribution.

In Section 2, we give some background on simple and combined Tausworthe generators and their implementation. We explain the notions of (t, ℓ)-equidistribution, resolution, and ME and CF sequences. We also show how to compute ℓ_t, and how

1991 *Mathematics Subject Classification.* 65C10.

Key words and phrases. Random number generation, equidistribution, combined generators.

to check if a generator is ME or ME-CF, in an efficient manner. In Section 3, we give the results of a computer search for near-ME, ME, and ME-CF combined Tausworthe generators, within certain classes of parameters that give rise to a fast implementation. A search for combined generators of that sort had already been performed within a small subclass of two-component combinations [11] with period length of approximately 2^{60}. A value of $L = 60$ was used (implicitly). No ME generator was found in that subclass, but three "almost-ME" generators were reported and proposed, after passing successfully a battery of empirical statistical tests. Software implementations of those combined generators are slightly faster than those of combined linear congruential generators of comparable period lengths. It appears that a period length of 2^{60} is rather short, in the sense that it imposes rather small upper bounds on ℓ_t for moderate values of t. Arguments based on the order of the discrepancy bounds for Tausworthe generators [8] also suggest that one should never use more than a negligible fraction of the period of linear-type generators, which implies that practical general purpose generators should have huge period lengths. Combinations with three components or more should then be considered and we provide a list of ME combinations of that kind. We used $L = 32$, having in mind 32-bit computers.

2. Tausworthe Generators and (t, ℓ)-Equidistribution

2.1. **Definition and Implementation.**

Tausworthe generators [6, 8, 9, 11] produce pseudorandom numbers by generating a sequence of bits from a linear recurrence modulo 2, and forming fractional numbers by taking blocks of successive bits. More precisely, let $\mathbf{F}_2$ denote the finite field with two elements (say, 0 and 1). Let $P(z) = z^k - a_1 z^{k-1} - \cdots - a_k$ be a polynomial with coefficients in $\mathbf{F}_2$, and consider the recurrence

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

whose characteristic polynomial is $P(z)$. It should be understood that in (1) and all along the paper, all computations are performed in $\mathbf{F}_2$ (this can be identified with working in integer arithmetic modulo 2, and we will omit writing "mod 2" explicitly). Suppose that $\boldsymbol{s}_0 = (x_0, \ldots, x_{k-1}) \in \{0,1\}^k$ is fixed and define

$$u_n = \sum_{i=1}^{L} x_{ns+i-1} 2^{-i}, \tag{2}$$

where s and L are positive integers. If P is primitive, $\boldsymbol{s}_0 \neq 0$, and $\rho = 2^k - 1$ is coprime to s, then the sequences (1) and (2) are both purely periodic with period ρ.

In principle, computing u_n from u_{n-1} involves performing s steps of the recurrence (1), which could be really slow in general. However, fast software implementations are available for certain special cases, and one such case is when the following condition is statisfied.

Condition 1. Suppose that $P(z)$ is a primitive trinomial of the form $P(z) = z^k - z^q - 1$, with $0 < 2q < k$, $0 < s \leq k - q < k \leq L$, $\gcd(s, 2^k - 1) = 1$, and L is equal to the computer's word size.

We now describe an algorithm which, under that condition, quickly computes $\tilde{\boldsymbol{s}}_n = (x_{ns}, \ldots, x_{ns+L-1})$ from $\tilde{\boldsymbol{s}}_{n-1} = (x_{(n-1)s}, \ldots, x_{(n-1)s+L-1})$. This is a slight generalization of the algorithm given in [11], where $L = k$ was implicitly assumed. The vectors $\tilde{\boldsymbol{s}}_n$ are maintained as unsigned (L-bit) integers, which are then multiplied by the normalization constant 2^{-L} to produce u_n. That algorithm is fast and easy to program in any computer language which supports shifting and bitwise logical operations.

Define $r = k - q$. Let A, B, and C be bit vectors of size L and suppose that A initially contains $\tilde{\boldsymbol{s}}_{n-1}$, while C is a "mask" comprised of k ones followed by $L - k$ zeros. The symbols & and $\oplus$ denote the logical (bitwise) "and" and "exclusive-or" operators, respectively. After step 6, A contains $\tilde{\boldsymbol{s}}_n$. To describe the behavior of the algorithm, take $n = 1$. Initially, A contains $\tilde{\boldsymbol{s}}_0 = (x_0, \ldots, x_{L-1})$. After step 2, B contains $x_k, \ldots, x_{r+L-1}$ (because $x_{k+i} = x_i \oplus x_{i+q}$), followed by $x_{L-q}, \ldots, x_{L-1}$ (the latter q bits unused afterwards). After step 5, A contains $x_s, \ldots, x_{k-1}$ followed by $s + L - k$ zeros, while B contains $k - s$ zeros followed by $x_k, \ldots, x_{s+L-1}$. Then, after step 6, A contains $\tilde{\boldsymbol{s}}_1$.

Algorithm QuickTaus:

1. $B \leftarrow q$-bit left-shift of A;
2. $B \leftarrow A \oplus B$;
3. $B \leftarrow (k - s)$-bit right-shift of B;
4. $A \leftarrow A \& C$;
5. $A \leftarrow s$-bit left-shift of A;
6. $A \leftarrow A \oplus B$.

For this algorithm to work properly, A must be initialized correctly with a valid initial $\tilde{\boldsymbol{s}}_0$; that is, which agrees with the recurrence. The first k bits, $\boldsymbol{s}_0 = (x_0, \ldots, x_{k-1})$, can be chosen arbitrarily (with $\boldsymbol{s}_0 \neq 0$); but when $L > k$, the other bits, $x_k, \ldots, x_{L-1}$, are uniquely determined by $\boldsymbol{s}_0$ and the recurrence (1). The following procedure initializes A properly, assuming that $k+r \geq L$ (which covers most cases of practical interest): set the first k bits of A to an arbitrary initial state $\boldsymbol{s}_0$, followed by $L - k$ zeros, then do:

1. $B \leftarrow q$-bit left-shift of A;
2. $B \leftarrow A \oplus B$;
3. $B \leftarrow k$-bit right-shift of B;
4. $A \leftarrow A \oplus B$.

After step 4, A contains $\tilde{\boldsymbol{s}}_0$. This can also be expanded to a more general initialization procedure (not given here but implemented in our programs) which does not require that $k + r \geq L$. If the additional condition $L - k \leq r - s$ is satisfied, then it can be easily verified that after the first pass through the 6 steps of QuickTaus, A will necessarily contain a valid state, even if the initial state $\tilde{\boldsymbol{s}}_0$ was not valid. In that case, the above initialization procedure is not necessary for running the generator; just skip the first value. On the other hand, the initialization procedure is needed for constructing the binary matrices required for the equidistribution analysis, as we shall discuss in Section 3.

2.2. **Combined Generators.** Unfortunately, Tausworthe generators based on polynomials of the above special form are unacceptable for two major reasons. Firstly, recurrences based on polynomials with too few non-zero coefficients are known to have important statistical defects [1, 7]. Secondly, for a b-bit computer, the period length of the generator cannot exceed 2^b, which is much too short for many applications on current 32-bit computers.

One way to address those drawbacks without slowing down the generator too much, suggested by Tezuka [10] and Wang and Compagner [13], is to combine two or more trinomial-based generators of the special form, as follows. For $j = 1, \dots, J$, consider a Tausworthe generator with primitive characteristic trinomial P_j of degree k_j, with $s = s_j$ such that $\gcd(s_j, 2^{k_j} - 1) = 1$, whose corresponding sequence is denoted by $x_{j,n}$, and whose output at step n is $u_{j,n} = \sum_{i=1}^{L} x_{j,ns_j+i-1} 2^{-i}$. Define the output u_n of the combined generator as the bitwise exclusive-or of $u_{1,n}, \dots, u_{J,n}$. If the polynomials $P_j(z)$ are pairwise relatively prime, then the period of the combined generator is $\rho = \text{lcm}(2^{k_1} - 1, \dots, 2^{k_J} - 1) = (2^{k_1} - 1) \times \cdots \times (2^{k_J} - 1)$. Further, as shown in [11], the combined generator is equivalent to a Tausworthe generator with (reducible) characteristic polynomial $P(z) = P_1(z) \cdots P_J(z)$ and s such that

$$z^s \equiv \sum_{j=1}^{J} z^{s_j} \eta_j(z)(P(z)/P_j(z)) \pmod{P(z)}, \tag{3}$$

where $\eta_j(z)$ is the inverse of $P(z)/P_j(z)$ modulo $P_j(z)$. See [6, 11, 13] for further details. The main interest of that kind of combination is that not only large periods can be achieved, but also $P(z)$ typically has many non-zero coefficients even when the individual $P_j(z)$ are all trinomials. In other words, this combination approach is just an efficient way of implementing a Tausworthe generator with a "good" (reducible) characteristic polynomial.

2.3. **Equidistribution and Resolution.** Consider the set P_t of all t-dimensional vectors (or points) formed by successive values of (2), from all possible initial states $\boldsymbol{s}_0$:

$$P_t = \left\{ \boldsymbol{u}_n = (u_n, \dots, u_{n+t-1}) \mid n \geq 0, \boldsymbol{s}_0 \in \{0,1\}^k \right\}.$$

We are interested in knowing whether those points are well distributed in the unit hypercube $I^t = [0,1)^t$. In the following analysis, we must assume that $\ell \leq L$. Suppose we partition I^t into $2^{t\ell}$ cubic cells of equal size. We say that P_t (and the sequence $\{u_n\}$) is (t, ℓ)-*equidistributed* if all the cells contain exactly the same number of points, namely $2^{k-t\ell}$ points each. Of course, this is possible only if $t\ell \leq k$, because the cardinality of P_t is at most 2^k. The largest value of $\ell \leq L$ for which the sequence is (t,ℓ)-equidistributed is called the *resolution* in dimension t, and denoted by ℓ_t. A dual set of numbers are t_ℓ, $1 \leq \ell \leq L$, where t_ℓ is the largest value of t for which the sequence is (t,ℓ)-equidistributed. One has the upper bounds

$$\ell_t \quad \leq \quad \ell_t^* \stackrel{\text{def}}{=} \min(L,\ \lfloor k/t \rfloor)$$

for each t and

$$t_\ell \quad \leq \quad t_\ell^* \stackrel{\text{def}}{=} \lfloor k/\ell \rfloor$$

for each $\ell \leq L$. We denote the *resolution gap* in dimension t by $\Lambda_t \stackrel{\text{def}}{=} \ell_t^* - \ell_t$ and the *dimension gap* for resolution ℓ by $\Delta_\ell \stackrel{\text{def}}{=} t_\ell^* - t_\ell$. A generator is said to have *maximal resolution* in dimension t if $\Lambda_t = 0$, and *maximal dimension* for resolution ℓ if $\Delta_\ell = 0$. Adopting a definition similar to [12] (in [12], $L \geq k$ was implicitly assumed), we call the sequence *maximally equidistributed* (ME) if $\Lambda_t = 0$ for $t = 1, \dots, k$ or, equivalently, if $\Delta_\ell = 0$ for $\ell = 1, \dots, L$. The equivalence is easily seen by observing that $\ell_t < \ell_t^*$ for some t implies that $\Delta_{\ell_t^*} > 0$, whereas $t_\ell < t_\ell^*$ implies $\Lambda_{t_\ell^*} > 0$. These definitions apply not only to generators based on irreducible polynomials, but also to generators based on reducible polynomials, i.e., combined generators.

Suppose now that we have a ME sequence. For the values of t that divide k, each of the 2^k cells in dimension t and resolution $\ell_t = k/t < L$ contains exactly one point of P_t. But when t does not divide k and $\ell_t < L$, each cell in dimension t and resolution ℓ_t contains exactly 2^{δ_t} points, where $\delta_t = k - t\ell_t$. Now, if $\ell_t + 1 \leq L$ and if we partition I_t into $2^{t(\ell_t+1)}$ cubic cells, then some of the cells will be empty while others will contain one or more points. If all non-empty cells contain exactly one point, we say that the ME sequence is *collision-free in dimension* t, or CF(t). If that holds in all dimensions t such that $\ell_t < L$, we just call it *collision-free* (CF). A ME-CF sequence is one for which the points P_t are as evenly distributed as possible in all dimensions, in terms of a partition of I^t into cubic cells as described above.

3. Computing the Resolution, and Finding ME and ME-CF Generators

3.1. **General Results.** We will now examine how the resolution ℓ_t can be computed, and how one can check whether a sequence is maximally equidistributed and collision-free. For a given Tausworthe generator, consider the $t\ell$-dimensional (row) vector of bits

$$\boldsymbol{s}_{t,\ell,s} = (x_0, \ldots, x_{\ell-1}, x_s, \ldots, x_{s+\ell-1}, \ldots, x_{(t-1)s}, \ldots, x_{(t-1)s+\ell-1}).$$

From (1), each x_n can be expressed as a linear combination of $x_0, \ldots, x_{k-1}$, say

$$x_n = \sum_{i=0}^{k-1} b_{n,i} x_i \tag{4}$$

for some binary coefficients $b_{n,i}$. Therefore, $\boldsymbol{s}_{t,\ell,s}$ can be expressed as

$$\boldsymbol{s}_{t,\ell,s} = \boldsymbol{s}_0 B_{t,\ell,s}, \tag{5}$$

where $B_{t,\ell,s}$ is a $k \times t\ell$ binary matrix which can be written as follows: if x_n denotes the jth element of $\boldsymbol{s}_{t,\ell,s}$, then the jth column of the matrix is $(b_{n,0}, \ldots, b_{n,k-1})^T$. The next proposition gives a necessary and sufficient condition for equidistribution and for the sequence to be CF(t), in terms of the matrices $B_{t,\ell,s}$. Next, we will explain efficient ways of computing the $b_{n,i}$'s, and then give a different variant of Proposition 1 for the case of combined generators.

Proposition 1. *The sequence is (t, ℓ)-equidistributed if and only if the matrix $B_{t,\ell,s}$ has (full) rank $t\ell$. If $\ell_t = \lfloor k/t \rfloor < k/t \leq L$, then the sequence is also CF(t) if and only if the matrix $B_{t,\ell_t+1,s}$ has rank k.*

Proof. The matrix $B_{t,\ell,s}$ in the linear system (5) induces a linear mapping $\varphi_1 : \mathbf{F}_2^k \to \mathbf{F}_2^{t\ell}$. The dimension of the kernel of φ_1 is $k - k'$, where k' is the number of linearly independent rows in $B_{t,\ell,s}$. Therefore, each element of $\mathbf{F}_2^{t\ell}$ is the image of either 0 or $2^{k-k'}$ distinct elements of $\mathbf{F}_2^k$. Then, the sequence is (t, ℓ)-equidistributed if and only if each element of $\mathbf{F}_2^{t\ell}$ is the image of $2^{k-k'}$ distinct elements of $\mathbf{F}_2^k$, which occurs if and only if $2^{t\ell} \cdot 2^{k-k'} = 2^k$, i.e., $k' = t\ell$. Similarly, the matrix $B_{t,\ell_t+1,s}$ induces a linear mapping $\varphi_2 : \mathbf{F}_2^k \to \mathbf{F}_2^{t(\ell_t+1)}$ and the sequence is CF(t) if and only if φ_2 is one-to-one, which occurs if and only if the matrix $B_{t,\ell_t+1,s}$ has rank k. $\square$

To see how to compute the $b_{n,i}$'s efficiently, even when n is large, we can use a polynomial representation of $\mathbf{Z}_{\mathbf{m}}^{\mathbf{k}}$. Define $G_n(z) = z^n \bmod P(z)$, for all $n \geq 0$. This polynomial can be computed by a standard divide-to-conquer algorithm (see [3]).

Proposition 2. *The polynomial $G_n(z)$ can be written (uniquely) as*

$$G_n(z) = \sum_{i=0}^{k-1} b_{n,i} z^i, \tag{6}$$

where the coefficients $b_{n,i}$ are the same as in (4).

Proof. Suppose that $G_n(z)$ is given by (6). For any given initial state $\boldsymbol{s}_0 = (x_0, \ldots, x_{k-1})$, define the linear mapping ψ such that for any polynomial $P(z) = \sum_{i=0}^{k-1} c_i z^i$, one has $\psi(P) = \sum_{i=0}^{k-1} c_i x_i$. Then, if we define $y_n = \psi(G_n) = \sum_{i=0}^{k-1} b_{n,i} x_i$, it is easily seen that $y_n = x_n$ for $n < k$, that $\{y_n,\ n \geq 0\}$ follows the recurrence (1) for all $n \geq k$, and so $x_n = y_n$ for all $n \geq 0$. $\square$

So, the coefficients of the polynomials $G_0(z), \ldots, G_{\ell-1}(z), G_s(z), \ldots, G_{s+\ell-1}(z), \ldots, G_{(t-1)s}(z), \ldots, G_{(t-1)s+\ell-1}(z)$ are precisely the columns of the matrix $B_{t,\ell,s}$. These polynomials are easy to compute, especially when s is small, which is typically the case for single-component Tausworthe generators.

In fact, it turns out that when s is small, there is an even easier way to compute the $b_{n,i}$'s, which was pointed out to the author by Raymond Couture. Suppose that the initial state is $\boldsymbol{s}_0 = \boldsymbol{e}_i$, the ith unit vector in dimension k; then Equation (4) simplifies to $x_n = b_{n,i}$. But when s is small, computing all the elements of $\boldsymbol{s}_{t,\ell,s}$ for a given $\boldsymbol{s}_0$ can be done easily just by using the algorithm that implements the generator. Typically (e.g., with the algorithm QuickTaus that we gave previously), those elements are produced L at a time. In that case, for any T, by performing $T-1$ steps of the recurrence with $\boldsymbol{s}_0 = \boldsymbol{e}_i$, for $1 \leq i \leq k$, one readily obtains all the elements of all the matrices $B_{t,\ell,s}$, for $1 \leq t \leq T$ and $1 \leq \ell \leq k$. This information can be stored and each matrix $B_{t,\ell,s}$ can then be constructed as needed from these elements.

3.2. **Building the Binary Matrix for Combined Generators.** For combined generators, the value of s is not small in general. Fortunately, it is not necessary to compute that value: one can replace z^s by the right-hand-side of (3) in the definition of $G_n(z)$. Denote by $H_s(z)$ that right-hand-side, reduced modulo $P(z)$. Then, one has, for example, $G_{(t-1)s+\ell-1}(z) = z^{\ell-1}(H_s(z))^{t-1} \mod P(z)$.

There is however a still more efficient way of testing equidistribution in the case of combined generators, by decomposition, as we now explain. Consider the component j of the combined generator. By applying the previous discussion to that component, it follows that one can express $x_{j,n}$ as

$$x_{j,n} = \sum_{i=0}^{k_j-1} b_{j,n,i} x_{j,i} \tag{7}$$

for some binary coefficients $b_{j,n,i}$, and the vector of bits $\boldsymbol{s}_{j,t,\ell,s_j} = (x_{j,0}, \ldots, x_{j,\ell-1}, x_{j,s_j}, \ldots, x_{j,s_j+\ell-1}, \ldots, x_{j,(t-1)s_j}, \ldots, x_{j,(t-1)s_j+\ell-1})$ can be expressed as

$$\boldsymbol{s}_{j,t,\ell,s_j} = \boldsymbol{s}_{j,0} B_{j,t,\ell,s_j},$$

where B_{j,t,ℓ,s_j} is a $k_j \times t\ell$ binary matrix whose elements can be computed easily using the techniques that we discussed for the case of simple generators. The coefficients $b_{j,n,i}$ in (7) are precisely the coefficients of the polynomial $G_{j,n}(z) = z^n \bmod P_j(z)$, and can be efficiently computed, when s_j is small, by using the recurrence for component j. In particular, if each component satisfies the conditions of the algorithm of Section 2, then all the required bits can be quickly obtained using that algorithm.

Now, observe that

$$x_n \quad = \quad x_{1,n} + \cdots + x_{J,n} \;\; = \;\; \sum_{j=1}^{J} \sum_{i=0}^{k_j-1} b_{j,n,i} x_{j,i}$$

for each n. The definitions of u_n and $u_{j,n}$ imply that

$$x_{ns+i-1} \;\; = \;\; \sum_{j=1}^{J} x_{j,ns_j+i-1}.$$

It follows from these observations that

$$\boldsymbol{s}_{t,\ell,s} = \sum_{j=1}^{J} \boldsymbol{s}_{j,t,\ell,s_j} = \sum_{j=1}^{J} \boldsymbol{s}_{j,0} B_{j,t,\ell,s_j}.$$

This can be rewritten as

$$\boldsymbol{s}_{t,\ell,s} \equiv \tilde{\boldsymbol{s}}_0 \tilde{B}_{t,\ell,s},$$

where $\tilde{B}_{t,\ell,s}$ is the $k \times t\ell$ matrix defined as the vertical juxtaposition of B_{1,t,ℓ,s_1}, $\ldots, B_{J,t,\ell,s_J}$, while $\tilde{\boldsymbol{s}}_0$ is the k-dimensional row vector obtained by juxtaposing $\boldsymbol{s}_{1,0}, \ldots, \boldsymbol{s}_{J,0}$ horizontally. This leads to the following proposition.

Proposition 3. *The sequence is (t,ℓ)-equidistributed if and only if the matrix $\tilde{B}_{t,\ell,s}$ has (full) rank $t\ell$. If $\ell_t = \lfloor k/t \rfloor < k/t \le L$, then the sequence is also CF(t) if and only if the matrix $\tilde{B}_{t,\ell_t+1,s}$ has rank k.*

Proof. The proof is similar to that of Proposition 1, with B replaced by $\tilde{B}$. □

The advantage of using $\tilde{B}_{t,\ell,s}$ instead of $B_{t,\ell,s}$ is that the former can be constructed by working with the individual components of the generator separately. On the other hand, checking whether the matrix has full rank requires essentially the same amount of work for both $\tilde{B}_{t,\ell,s}$ and $B_{t,\ell,s}$. This can be achieved by triangulating the matrix (by Gaussian elimination) using exclusive-or operations between the rows or columns.

3.3. **Sufficient Conditions for ME and for ME-CF.** To verify whether a sequence is ME, one can compute ℓ_t for $t = 1, \ldots, k$, or t_ℓ for $\ell = 1, \ldots, L$, using Proposition 1 or 3. However, it is not necessary to compute ℓ_t for each of those k values of t, or t_ℓ for all L values of ℓ, because (t,ℓ)-equidistribution implies (t',ℓ')-equidistribution for all $t' \le t$ and $\ell' \le \ell$. Thus, if $\ell_{t'} = \ell_t$ for $t' < t$ and maximal resolution is reached in dimension t, then it is also reached in dimension t'. Likewise, if $t_{\ell'} = t_\ell$ for $\ell' < \ell$ and the generator has maximal dimension for resolution ℓ, then it also has maximal dimension for resolution ℓ'. Moreover, $(1, \min(k, L))$-equidistribution always holds by definition of P_t. This leads to the next proposition, which tells us for which values of t we need to compute ℓ_t, or for which values of

ℓ we need to compute t_ℓ. For simplicity, assume $L \geq \sqrt{k-1}$, which should always be the case in practice. Define

$$\begin{aligned}
\Phi_1 &= \{\max(2, \lfloor k/L \rfloor), \ldots, \lfloor \sqrt{k} \rfloor\}; \\
\Phi_2 &= \{t = \lfloor k/\ell \rfloor \mid \ell = 1, \ldots, \lfloor \sqrt{k-1} \rfloor\}; \\
\Psi_1 &= \{1, \ldots, \lfloor \sqrt{k} \rfloor\}; \\
\Psi_2 &= \{\ell = \lfloor k/t \rfloor \mid t = \max(2, \lfloor k/L \rfloor), \ldots, \lfloor \sqrt{k-1} \rfloor\}.
\end{aligned}$$

Proposition 4. *A maximal period sequence is ME if and only if $\Lambda_t = 0$ for all $t \in \Phi_1 \cup \Phi_2$. It is also ME if and only if $\Delta_\ell = 0$ for all $\ell \in \Psi_1 \cup \Psi_2$.*

Proof. Assume that $\ell_t = \ell_t^*$ for all $t \in \Phi_1 \cup \Phi_2$. Now, it suffices to show that $\ell_t = \ell_t^*$ for all $t \leq k$. Note that for $t = \lfloor k/L \rfloor$, one has $t \leq k/L$, and therefore $\ell_t = L$. Then, for $1 \leq t < \lfloor k/L \rfloor$, one also has $\ell_t = L$, which implies that if resolution $\ell_t = L$ is achieved in dimension $t = \lfloor k/L \rfloor$, it must also be achieved in all smaller dimensions. So, those smaller dimensions need not be included in Φ_1. Now, we examine the dimensions $t > \sqrt{k}$. Let t be an integer such that $\sqrt{k} < t \leq k$ and $t \notin \Phi_2$. Then, $\ell_t \leq \lfloor k/t \rfloor < \sqrt{k}$, so $\ell_t \leq \sqrt{k-1} \leq L$. Let $t' = \lfloor k/\ell_t \rfloor$. Then $t' \in \Phi_2$ and one has $t' \leq k/\ell_t$, which implies that $\ell_t \leq k/t'$, and so $\ell_t \leq \ell_{t'}$. Also, $t\ell_t \leq k$, which implies that $t \leq k/\ell_t$, so $t \leq \lfloor k/\ell_t \rfloor = t'$ and then $\ell_t \geq \ell_{t'}$. Therefore, $\ell_t = \ell_{t'}$. Then, since we have (t', ℓ')-equidistribution because of our initial assumption, we also have (t, ℓ)-equidistribution and this completes the proof of the first part. The proof of the second part is similar. □

The next proposition tells us for which values of t we need to check CF(t) to make sure that a given ME sequence is also CF. Define

$$\begin{aligned}
\Phi_3 &= \{t \mid 2 \leq t \leq k;\ \ell_t = \lfloor k/t \rfloor < k/t \leq L\}; \\
\Phi_4 &= \{t \in \Phi_3 \mid k \bmod (\ell_t + 1) \neq 0;\ \ell_{t-1} > \ell_t\}.
\end{aligned}$$

Proposition 5. *An ME sequence is also CF if and only if it is CF(t) for all $t \in \Phi_4$.*

Proof. The condition is clearly necessary; it remains to show that if CF(t) holds for all $t \in \Phi_4$ then CF(t) also holds for all $t \in \Phi_3$. Let $t \in \Phi_3 \setminus \Phi_4$. If $k \bmod (\ell_t + 1) = 0$, it means that $\ell_t + 1 = k/t' = \ell_{t'}$ for some $t' < t$. Then, since the sequence is ME, we must have $(t', \ell_t + 1)$-equidistribution, which implies that both matrices $B_{t',\ell_t+1,s}$ and $B_{t,\ell_t+1,s}$ have rank k, and therefore the sequence is CF(t). Otherwise (if $\ell_t + 1$ does not divide k), let $t' = \min\{t'' \mid \ell_{t''} = \ell_t\}$. Then, $t' \leq t < \lfloor k/\ell_t \rfloor = \lfloor k/\ell_{t'} \rfloor$ and $\ell_{t'} + 1 = \ell_t + 1$ does not divide k. Therefore, $t' \in \Phi_4$, and so we have CF(t'), that is, $B_{t',\ell_t+1,s}$ has rank k. This implies that $B_{t,\ell_t+1,s}$ also has rank k, that is, CF(t). □

4. Seeking Maximally Equidistributed Generators

We have written a computer program that seeks ME or near-ME combined sequences for a specified value of J, specified degrees $k_1, \ldots, k_J$ of primitive trinomials $P_1(z), \ldots, P_J(z)$, with $k_j \leq 32$ for each j, and $L = 32$. The program examines all combinations of values of q_j and s_j within specified ranges, among those that satisfy Condition 1, looking for ME or near-ME combined generators, and checking whether the ME ones are also CF, by using Propositions 3–5. When seeking near-ME generators, the user may specify upper bounds on the *dimension gap* Δ_ℓ for each ℓ and on the sum $\Delta = \sum_{\ell=1}^{L} \Delta_\ell$. The values of these bounds have a strong impact on the computational times, because a generator is rejected by the program as soon as Δ or one of the Δ_ℓ's exceeds its upper bound.

We now give examples of results obtained from this program. The program is written in C and is available from the author. The timings reported here are on a SUN SPARCstation 20 and the programs were compiled with "full optimization". Let us recall all the pairs (k, q) such that $25 \le k \le 32$, $0 < 2q < k$, and $P(z) = z^k - z^q - 1$ is a primitive trinomial. They are [14]: $(31, 3)$, $(31, 6)$, $(31, 7)$, $(31, 13)$, $(29, 2)$, $(28, 3)$, $(28, 9)$, $(28, 13)$, $(25, 3)$, $(25, 7)$.

Example 1. Let $J = 2$, $k_1 = 31$, and $k_2 = 29$. We performed an exhaustive search for ME generators ($\Delta = 0$) among all combined generators of that form with values of q_j and s_j satisfying Condition 1. A total of 2565 combinations were examined, which took approximately 17 seconds of cpu time, and no ME combination was found. We then performed another search with $\Delta \le 3$ and $\Delta_\ell \le 1$ for each ℓ. That took 25 seconds and we found a single combination, namely: $(q_1, q_2, s_1, s_2) = (3, 2, 22, 19)$, for which $\Delta_\ell = 1$ for $\ell = 6$, 15 and 20, and $\Delta_\ell = 0$ for all other values of ℓ. The three generators proposed in [11] turn out to have $\Delta \ge 4$.

Example 2. We made a similar search for the case $J = 2$, $k_1 = 29$, $k_2 = 28$, and with the constraint $\Delta = 0$. Here, 864 combinations were examined (this took 6.5 seconds) and the following ME-CF generator was found: $(q_1, q_2, s_1, s_2) = (2, 9, 18, 14)$.

Example 3. This search was for three-component combinations: $J = 3$, $k_1 = 31$, $k_2 = 29$, and $k_3 = 28$. The program performed an exhaustive search for ME generators ($\Delta = 0$) among the 82080 possible combinations with components that satisfy Condition 1. This took approximately 31 minutes of cpu. A total of 19 ME generators were found, among which the following three are ME-CF: $(q_1, q_2, q_3, s_1, s_2, s_3) = (13, 2, 3, 12, 4, 17)$, $(7, 2, 9, 24, 7, 11)$, and $(3, 2, 13, 20, 16, 7)$. These generators have period lengths $(2^{31}-1)(2^{29}-1)(2^{28}-1) \approx 2^{88}$. All their components satisfy the additional condition $L - k_j \le r_j - s_j$, except for the first component of the second generator. The characteristic polynomial of each combined generator has the form $P(z) = \sum_{i=0}^{k} c_i z^i$, where $k = 88$. Let $I_1 = \{i \mid c_i = 1\}$ denote the set of indices of the non-zero coefficients. Then, for the three ME-CF combinations that we found, one has $I_1 = \{0, 2, 3, 5, 13, 15, 16, 18, 28, 29, 30, 31, 32, 33, 34, 36, 41, 42, 43, 45, 57, 59, 60, 61, 63, 70, 88\}$, $I_1 = \{0, 2, 7, 11, 16, 18, 28, 29, 30, 31, 33, 35, 36, 37, 38, 40, 42, 45, 57, 59, 60, 61, 64, 69, 88\}$, and $I_1 = \{0, 2, 3, 5, 13, 15, 16, 18, 28, 29, 30, 32, 42, 44, 45, 46, 57, 59, 61, 73, 88\}$, respectively. The corresponding recurrences have 26, 24, and 20 non-zero coefficients (out of 88), respectively.

Example 4. The next search was for $J = 4$, $k_1 = 31$, $k_2 = 29$, $k_3 = 28$, $k_4 = 25$, and ME generators only. There are exactly 3 283 200 combinations that satisfy Condition 1 in this case. We performed an exhaustive search among those combinations and found 26 195 ME generators, 4744 of them being ME-CF. That took approximately 56 hours of cpu time. A list of those ME-CF generators is available from the author.

Figure 1 gives a portable computer code in the language C for the first ME-CF generator of Example 3, with period length $\approx 2^{88}$. This code implements the algorithm QuickTaus of Section 2.2 for each component and generates a $U(0, 1)$ random number. Before calling `taus88` for the first time, the integers `s1`, `s2`, and

```
unsigned long s1, s2, s3, b;

double taus88 ()
  {                   /* Generates numbers between 0 and 1. */
  b  = (((s1 << 13) ^ s1) >> 19);
  s1 = (((s1 & 4294967294) << 12) ^ b);
  b  = (((s2 <<  2) ^ s2) >> 25);
  s2 = (((s2 & 4294967288) <<  4) ^ b);
  b  = (((s3 <<  3) ^ s3) >> 11);
  s3 = (((s3 & 4294967280) << 17) ^ b);
  return ((s1 ^ s2 ^ s3) * 2.3283064365e-10);
  }
```

FIGURE 1. An implementation of a three-component generator in C.

`s3` must be initialized to any non-zero values. [1] On a SUN SPARCstation 20, it took approximately 0.9 seconds to generate one million random numbers with this code, compared with 4.8 seconds for the widely used 32-bit combined generator proposed in [4] (also implemented in C). On a PC-486 (33 MHz), with a different compiler, generating the same one million random numbers took 3.2 seconds for `taus88` and 7.6 seconds for the combined generator of [4]. Four-component ME-CF combined generators similar to that of Figure 1, with period $\approx 2^{113}$, are also easy to implement.

5. Conclusion

We showed how to find efficiently combined Tausworthe generators with optimal equidistribution properties in all dimensions. Our method is simple and faster than those previously available (e.g., much faster than the method described in [2]). It permits one to perform exhaustive searches for optimal parameters with a modest computing effort. ME and ME-CF combined generators (with trinomial-based components) turn out to be much easier to find when the number of components increases, because the number of possible combinations is then much higher, and also because the characteristic polynomials of the recurrences tend to have more non-zero coefficients when there are more components. Three combined Tausworthe generators with two components and period lengths 2^{60} were proposed in [11]. The three- and four-component ME-CF combinations that we found offer better alternatives, with longer period lengths and significantly better resolution. In terms of speed, these generators are also highly competitive compared to those which are currently available from software libraries.

Acknowledgments

This work has been supported by NSERC-Canada grant # OGP0110050 and FCAR-Québec grant # 93ER1654. I wish to thank Luc De Bellefeuille, who wrote

[1]Correction: In fact the most significant 31 bits of `s1`, 29 bits of `s2`, and 28 bits of `s3`, must be initialized to nonzero values. The initial values of the other bits are unused. So, `s1`, `s2`, and `s3` must be $\geq 2, 8, 16$, respectively. In practice, it is better to take larger (random) initial seeds.

the computer programs to perform the seeks, and Raymond Couture who suggested several improvements.

References

[1] A. Compagner, *The Hierarchy of Correlations in Random Binary Sequences*, Journal of Statistical Physics **63** (1991), 883–896.

[2] R. Couture, P. L'Ecuyer, and S. Tezuka. *On the Distribution of k-Dimensional Vectors for Simple and Combined Tausworthe Sequences*, Mathematics of Computation **60**, 202 (1993), 749–761 and S11-S16.

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

[4] P. L'Ecuyer, *Efficient and Portable Combined Random Number Generators*, Communications of the ACM **31**, 6 (1988), 742–749 and 774. See also the correspondence in the same journal, **32**, 8 (1989), 1019–1024.

[5] P. L'Ecuyer, *Testing Random Number Generators*, Proceedings of the 1992 Winter Simulation Conference, IEEE Press (1992), 305–313.

[6] P. L'Ecuyer, *Uniform Random Number Generation*, Annals of Operations Research **53** (1994), 77–120.

[7] J. H. Lindholm, *An Analysis of the Pseudo-Randomness Properties of Subsequences of Long m-Sequences*, IEEE Transactions on Information Theory **IT-14**, 4 (1968), 569–576.

[8] H. Niederreiter, *Random Number Generation and Quasi-Monte Carlo Methods*, SIAM CBMS-NSF Regional Conference Series in Applied Mathematics, vol. 63, SIAM, Philadelphia, 1992.

[9] R. C. Tausworthe, *Random Numbers Generated by Linear Recurrence Modulo Two*, Mathematics of Computation **19** (1965), 201–209.

[10] S. Tezuka, *Random number generation based on polynomial arithmetic modulo two*, IBM TRL Research Report, RT-0017, 1989.

[11] S. Tezuka and P. L'Ecuyer, *Efficient and Portable Combined Tausworthe Random Number Generators*, ACM Transactions on Modeling and Computer Simulation **1** (1991), 99–112.

[12] J. P. R. Tootill, W. D. Robinson, and D. J. Eagle, *An Asymptotically Random Tausworthe Sequence*, Journal of the ACM **20** (1973), 469–481.

[13] D. Wang and A. Compagner, *On the Use of Reducible Polynomials as Random Number Generators*, Mathematics of Computation **60** (1993), 363–374.

[14] N. Zierler and J. Brillhart, *On Primitive Tronomials (Mod 2)*, Information and Control **13** (1968), 541–554, and **14** (1969), 566–569.

Published in *Mathematics of Computation* **65**, 213 (1996), 203–213.
Errata for published version: In Eq.(10), $\mathbf{s}_{j,0}$ must be inside the summation.

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

E-mail address: lecuyer@iro.umontreal.ca