STATEMENT OF CONTRIBUTION

COMBINED MULTIPLE RECURSIVE RANDOM NUMBER GENERATORS

PIERRE L'ECUYER

Statement

A great many algorithms in operations research and other computational fields involve the generation of (often billions of) random numbers. Linear congruential generators (LCGs) are still widely used for that purpose. However, the relentless increase in available computing power has made those simple generators unreliable for several classes of applications. Other classes of generators have been proposed and studied, including combined LCGs, as well as generators based on linear recurrences of order larger than one. This paper studies the combination of generators of the latter class. It shows that such combination offers a good way of obtaining a fast and reliable generator, and improves significantly upon the combination of simple LCGs. We also provide a specific generator with its computer implementation.

Version: March 3, 1995

COMBINED MULTIPLE RECURSIVE RANDOM NUMBER GENERATORS

PIERRE L'ECUYER

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

We analyze the random number generators obtained by combining two or more multiple recursive generators. We study the lattice structure of such combined generators and argue that combination is a good way of obtaining robust generators, based on a recurrence with many nonzero coefficients, and which also possess a fast implementation.

Subject Classifications: Simulation, random number generation, linear congruential, lattice structure, multiple recursive, combined generators.

Area of review: Simulation.

Linear congruential random number generators (LCGs) with prime moduli smaller than 2^{31} have the merit of being easily implementable on 32-bit computers, but no longer satisfy the requirements of today's computer intensive simulations. Indeed, their period length could easily be exhausted in a few minutes of cpu time on a typical workstation. It is also now well-recognized that, for "statistical" reasons (see, e.g., Compagner 1991, L'Ecuyer 1994 for more details), the period length of a linear-type generator should be several orders of magnitude larger than what is actually used. There are also good reasons (e.g., variance reduction) for splitting the sequence of a random number generator into disjoint subsequences, to make several "virtual" generators out of the first one L'Ecuyer (1994), each of them having a long period and good properties. Because of those requirements, the availability of statistically robust generators with huge period lengths, say up to 2^{200} or so, is highly desirable.

One way of improving upon LCGs is to use multiple recursive generators (MRGs) (see L'Ecuyer 1990, L'Ecuyer, Blouin, and Couture 1993, L'Ecuyer 1994, Niederreiter 1992), which are based on a linear recurrence of higher order. More specifically, an MRG of order k is based on a kth-order linear recurrence of the form

$$\begin{aligned} x_n &= (a_1 x_{n-1} + \cdots + a_k x_{n-k} + b) \bmod m; \qquad (1) \\ u_n &= x_n/m, \end{aligned}$$

where m and k are positive integers, while b and each a_i belong to $\mathbb{Z}_m = \{0, 1, \ldots, m-1\}$. For reasons of efficiency, it has often been suggested to use only two non-zero coefficients a_i and $b = 0$ in (1). This gives a very fast generator whose period reaches $m^k - 1$ under verifiable conditions (L'Ecuyer 1990, L'Ecuyer, Blouin, and Couture 1993). However, when the number of non-zero coefficients in (1) is small compared to k, there are unfavorable limitations on the quality of lattice structure of the generator (see L'Ecuyer 995). In other words, these generators have structural defects, which are not necessarily catastrophic, but could conceivably show up in some computer-intensive simulations.

Another approach for increasing the period and improving the structure of the generator is combination. Combined LCGs, which add up the results of two or more LCGs with different moduli, have been studied by L'Ecuyer (1988), L'Ecuyer and Tezuka (1991), Wichmann and Hill (1982). They are equivalent (or approximately equivalent) to LCGs with large non-prime moduli. In other words, these combined LCGs can be viewed as efficient implementations of LCGs with huge moduli (L'Ecuyer and Tezuka 1991). One advantage of the combination approach proposed by L'Ecuyer (1988), compared to that of Wichmann and Hill (1982), is that the former adds "noise" to the lattice structure, i.e., shakes up the regularity of the points produced (L'Ecuyer and Tezuka 1991). However, to obtain a period length near m^k, we must combine at least k LCGs with distinct prime moduli close to m. This becomes inefficient as k increases. Other types of combinations, such as combined Tausworthe generators (Tezuka and L'Ecuyer 1991, Compagner 1991, Wang and Compagner 1993), have also been proposed and analyzed in the literature. For more details, see Couture and L'Ecuyer (1996), L'Ecuyer (1990), L'Ecuyer (1994) and the references cited there.

In this paper, we analyze what happens when we combine two or more MRGs. We show that the combined generator is equivalent (or approximately equivalent, depending on the type of combination), to an MRG with large modulus, equal to the product of the individual moduli. One important advantage of that combination is that the linear recurrence associated with the combined generator can have many non-zero coefficients. Another feature, for one of the combination types, is the noise added to the lattice structure, as with the combined LCGs. In the next section, we define the combined generators, derive their approximating MRGs, and characterize their period lengths. We then discuss the influence of combination on the lattice structure. Finally, we examine particular classes of combinations and suggest one specific generator. For more about the basic concepts of on finite fields, we refer the reader to Lidl and Niederreiter (1986).

1. The MRG Associated with a Combined Generator

Consider J MRGs ($J \geq 2$) such that for $j = 1, \ldots, J$, the jth recurrence has order k_j and is given by:

$$x_{j,n} \;=\; (a_{j,1}x_{j,n-1} + \cdots + a_{j,k_j}x_{j,n-k_j} + b_j) \bmod m_j. \tag{2}$$

We assume that the m_j's are pairwise relatively prime and that each recurrence is purely periodic. Let ρ_j denote the period length of the jth recurrence; that is, $x_{n+\rho_j} = x_n$ for all $n \geq 0$. Recall that if m_j is prime, it is easy to obtain ρ_j equal to $m_j^{k_j} - 1$: take $b_j = 0$ (a homogeneous recurrence) and select the coefficients $a_{j,i}$ in such a way that the characteristic polynomial of (2), defined as $f(x) = x^{k_j} - a_{j,1}x^{k_j-1} - \cdots - a_{j,k_j}$, is a primitive polynomial modulo m_j (Knuth 1981, Lidl and Niederreiter 1986). We recall that the polynomial $f(x)$ is primitive, for prime m_j, if and only if $m_j^{k_j} - 1$ is the smallest positive integer n such that $x^n \equiv 1 \pmod{f(x)}$. An algorithm for testing for primitivity modulo m_j is given in Knuth (1981), p.29. The only case where $b_j \neq 0$ seems to have practical interest is when $k_j = 1$ and m_j is a power of two; the period length can then reach m_j under certain conditions (see, e.g., Theorem 3.2.1.2.A of Knuth 1981).

Let $\delta_1, \ldots, \delta_J$ be arbitrary integers such that δ_j is relatively prime to m_j for each j. Define the two combined generators

$$z_n = \left(\sum_{j=1}^{J} \delta_j x_{j,n}\right) \bmod m_1; \qquad \tilde{u}_n = z_n/m_1 \tag{3}$$

and

$$w_n = \left(\sum_{j=1}^{J} \frac{\delta_j x_{j,n}}{m_j}\right) \bmod 1. \tag{4}$$

Let

$$\begin{aligned}
k &= \max(k_1, \ldots, k_j); & (5)\\
m &= \prod_{j=1}^{J} m_j\,; & (6)\\
b &= \left(\sum_{j=1}^{J} \frac{\delta_j b_j m}{m_j}\right) \bmod m\,; & (7)\\
n_j &= (m/m_j)^{-1} \bmod m_j \qquad \text{for } j = 1, \ldots, J; & (8)\\
a_i &= \left(\sum_{j=1}^{J} \frac{a_{j,i} n_j m}{m_j}\right) \bmod m \qquad \text{for } i = 1, \ldots, k; & (9)
\end{aligned}$$

where $a_{j,i} = 0$ for $i > k_j$, and where $(m/m_j)^{-1} \bmod m_j$ is the inverse of m/m_j modulo m_j (which exists, because the m_j's are assumed relatively prime). In other words, n_j is defined as the smallest positive integer which satisfies $n_j(m/m_j) \equiv 1 \pmod{m_j}$. It can be computed using the identity: $n_j = (m/m_j)^{m_j-2} \bmod m_j$ and a divide-to-conquer algorithm (Brassard and Bratley 1988, Knuth 1981, L'Ecuyer 1990), or by a variant of Euclid's algorithm, as explained in Knuth (1981). The divide-to-conquer algorithm computes the exponentiation modulo m_j using the following recursion:

$$x^n \bmod m_j = \begin{cases} x & \text{if } n = 1; \\ x \cdot x^{n-1} \bmod m_j & \text{if } n > 1,\ n \text{ even}; \\ x^{n/2} \cdot x^{n/2} \bmod m_j & \text{if } n > 1,\ n \text{ odd}. \end{cases}$$

Consider the following MRG, with composite modulus:

$$x_n = (a_1 x_{n-1} + \cdots + a_k x_{n-k} + b) \bmod m; \tag{10}$$

$$u_n = x_n/m. \tag{11}$$

In what follows, we show that (4) and (10–11) are equivalent, and that their period length ρ is equal to the least common multiple (lcm) of $\rho_1, \ldots, \rho_j$. We then give tight bounds on the difference between u_n and $\tilde{u}_n$. These bounds are close to zero when the m_j's are close to each other. All these results generalize those already given by L'Ecuyer and Tezuka (1991) for the case where $k = 1$ and $b = 0$.

Proposition 1. *If $(w_0, \ldots, w_{k-1}) = (u_0, \ldots, u_{k-1})$, then $w_n = u_n$ for all $n \geq 0$.*

Proof. By the same argument as in L'Ecuyer and Tezuka (1991), it is easily seen that

$$n_j (m/m_j)^2 \bmod m = m/m_j.$$

Then, since $(m/m_i)(m/m_j) \bmod m = 0$ for $i \neq j$, one obtains

$$\begin{aligned}
& m(a_1 w_{n-1} + \cdots + a_k w_{n-k} + b) \bmod m \\
= & \left[b + m \sum_{i=1}^{k} \left(\sum_{\ell=1}^{J} \frac{a_{\ell,i} n_\ell m}{m_\ell} \right) \left(\sum_{j=1}^{J} \frac{\delta_j x_{j,n-i}}{m_j} \right) \right] \bmod m \\
= & \left[b + \sum_{i=1}^{k} \sum_{j=1}^{J} n_j \left(\frac{m}{m_j} \right)^2 a_{j,i} \delta_j x_{j,n-i} \right] \bmod m \\
= & \left[b + \sum_{i=1}^{k} \sum_{j=1}^{J} \frac{m}{m_j} \delta_j a_{j,i} x_{j,n-i} \right] \bmod m \\
= & \left[\sum_{j=1}^{J} \left(\frac{m}{m_j} \delta_j \left(b_j + \sum_{i=1}^{k} a_{j,i} x_{j,n-i} \right) \bmod m_j \right) \right] \bmod m \\
= & \left(m \sum_{j=1}^{J} \frac{\delta_j x_{j,n}}{m_j} \right) \bmod m \\
= & m w_n.
\end{aligned}$$

Therefore, $\{m w_n,\ n \geq 0\}$ satisfies the same recurrence as $\{x_n,\ n \geq 0\}$, and that completes the proof. $\square$

The next lemma will be used in the proof of the proposition that follows. Define $c_j = (m/m_j)\delta_j \bmod m_j$.

Lemma 1. *One has* $x_n \equiv c_j x_{j,n} \pmod{m_j}$.

Proof. One has

$$x_n = m w_n = \left(m \sum_{\ell=1}^{J} \frac{\delta_\ell x_{\ell,n}}{m_\ell} \right) \bmod m, \tag{12}$$

which yields

$$x_n \equiv c_j x_{j,n} \pmod{m_j},$$

because $(m/m_\ell)\delta_\ell \bmod m_j = 0$ for $\ell \neq j$. □

Proposition 2. *The period of* $\{x_n,\ n \geq 0\}$ *is equal to* $\rho = \mathrm{lcm}(\rho_1, \ldots, \rho_J)$.

Proof. Let $s_{j,n} = (x_{j,n}, \ldots, x_{j,n+k_j-1})$ and $s_n = (x_n, \ldots, x_{n+k-1})$. Since each component is purely periodic, the combined generator is also certainly purely periodic, i.e., the initial state s_0 is eventually revisited. The period of $\{x_n,\ n \geq 0\}$ is the smallest ν such that $x_{\nu+n} = x_n$ for all $n \geq 0$, i.e., such that

$$s_\nu = s_0. \tag{13}$$

Clearly, since ρ is the least common multiple of the individual periods of the components, $\nu = \rho$ satisfies (13). It remains to show that no smaller ν satisfies (13). Suppose that such a ν exists. Then, from Lemma 1, one has that $c_j(x_{j,n} - x_{j,n+\nu}) \bmod m_j = 0$. From the assumption that δ_j is relatively prime to m_j and because the m_j's are pairwise relatively prime, it follows that c_j is invertible modulo m_j. This implies that $(x_{j,n} - x_{j,n+\nu}) \bmod m_j = 0$. Since this holds for all $n \geq 0$, it follows that ν must be a multiple of ρ_j, the period of $\{x_{j,n},\ n \geq 0\}$. This holds for all j. Therefore, ν must be equal to ρ. □

If each recurrence (2) is homogeneous ($b_j = 0$) and m_j is prime, then one can easily achieve $\rho_j = m_j^{k_j} - 1$ by selecting a primitive polynomial for each j. Then, each ρ_j is even, so $\rho \leq (m_1^{k_1} - 1) \cdots (m_J^{k_J} - 1)/2^{J-1}$. Another interesting possibility is to take a power of two for m_1, $k_1 = 1$, get $\rho_1 = m_1$, and then use distinct prime moduli for the remaining m_j's.

Note that the coefficients a_i in (10) do not depend on the choice of the δ_j's; only b does. Therefore, when the individual recurrences (2) are homogeneous, the recurrence for the combined generator is the same for whatever (nonzero mod m_j) choices for the δ_j's. However, changing the δ_j's will change in general the starting point $(x_0, \ldots, x_{k-1})$ in (10) and, as a result, will change the sequence produced. Note that there are m^k such starting points (including bad ones) or, equivalently, m^k possible states for the recurrence (10). Changing the starting point could have a non-negligible effect because those m^k states are usually partitioned into disjoint subcycles (plus perhaps some transient states), so changing the starting point could conceivably send us to a different subcycle. When two starting points belong to the same subcycle (i.e., are reachable from each other), we say that they (and the corresponding sequences) are *equivalent*.

The total number of states for the combined generator (including the trivial states) is equal to $\prod_{j=1}^{J} m_j^{k_j}$, since each component has $m_j^{k_j}$ possible states. If the k_j are not all equal, this could be much less than m^k. In that case, not all values of $(x_0, \ldots, x_{k-1})$ in (10) can be obtained as combinations of values of $(x_{j,0}, \ldots, x_{j,k-1})$ through (12). It turns out that the states $(x_0, \ldots, x_{k-1})$ which can be obtained as a combination are *recurrent* states for the recurrence (10), i.e., if the recurrence starts from such a state, it will eventually return to that state. The other states, which are not the result of a combination, are *transient*, that is, if any of them is the initial state for (10), then it will never be visited again by the recurrence. This is analyzed more deeply by Couture and L'Ecuyer (1996).

Example 1. Take $J = 2$, $m_1 = 5$, $k_1 = 1$, $m_2 = 3$, $k_2 = 3$. Select the multipliers so that each component has a full period, namely $\rho_1 = 4$ and $\rho_2 = 3^3 - 1 = 26$. Then, the combined generator has a total of $5 \times 27 = 135$ possible states (including the zeros), but its period is only $\rho = 52$. On the other hand, the recurrence (10) associated with the combined generator has order 3 and modulus 15; so its total number of states is $15^3 = 3375$. The recurrent states are the 135 states produced by the combination; the other 3240 states are all transient.

We now bound the difference (modulo 1) between u_n and $\tilde{u}_n$. The ϵ_n in (14) represents the distance between u_n and $\tilde{u}_n$ on the circle (or one-dimensional torus) obtained by joining the two extremities of the interval $[0, 1]$. Notice that $\tilde{u}_n$ must be a multiple of $1/m_1$, and therefore has less resolution than u_n, which is a multiple of $1/m$. However, $|\epsilon_n|$ is not bounded by $1/m_1$. As in L'Ecuyer and Tezuka (1991), define

$$\begin{aligned}
\Psi^+ &= \{j \mid 2 \le j \le J \text{ and } (m_j - m_1)\delta_j > 0\} \\
\Psi^- &= \{j \mid 2 \le j \le J \text{ and } (m_j - m_1)\delta_j < 0\}
\end{aligned}$$

$$\begin{aligned}
\Delta^+ &= \sum_{j \in \Psi^+} \frac{(m_j - m_1)(m_j - 1)\delta_j}{m_1 m_j} + \sum_{j \in \Psi^-} \frac{(m_j - m_1)\delta_j}{m_1 m_j} \\
\Delta^- &= \sum_{j \in \Psi^+} \frac{(m_j - m_1)\delta_j}{m_1 m_j} + \sum_{j \in \Psi^-} \frac{(m_j - m_1)(m_j - 1)\delta_j}{m_1 m_j}.
\end{aligned}$$

Proposition 3. *If* $(w_0, \ldots, w_{k-1}) = (u_0, \ldots, u_{k-1})$*, then*

$$\tilde{u}_n = (u_n + \epsilon_n) \bmod 1 \tag{14}$$

for all $n \ge 0$*, where*

$$\Delta^- \le \epsilon_n \le \Delta^+. \tag{15}$$

Proof. The proof mirrors that of Proposition 2 in L'Ecuyer and Tezuka (1991), and is omitted. □

2. Combining generators with a common modulus

In the preceding section, we assumed that the m_j's were pairwise relatively prime. Let us now consider a different situation, that where the m_j's in (2) are all the same, say $m_j = m$ for all j, but where the k_j's are distinct. We shall consider again the two combined generators (3) and (4). We note that in this section, k and m are still the order and modulus of the MRG associated with the combination, are defined differently as in (5–6).

Let m be a prime. For each j, we suppose that $b_j = 0$ and let $f_j(x) = x^{k_j} - a_{j,1}x^{k_j-1} - \cdots - a_{j,k_j}$ be the characteristic polynomial of the recurrence. We assume that $f_j(x)$ is a primitive polynomial modulo m, so that $\rho_j = m^{k_j} - 1$. Let

$$f(x) = x^k - a_1 x^{k-1} - \cdots - a_k = f_1(x) \cdots f_J(x) \bmod m$$

be the product of those characteristic polynomials, where $k = \prod_{j=1}^{J} k_j$, and $b = 0$. Consider now the recurrence (10) associated with that characteristic polynomial, again with $u_n = x_n/m$. We show that both combinations (3) and (4) follow exactly that recurrence. We also show that the period of the combined generator is bounded above by $\rho_1 \cdots \rho_J/(m-1)^{J-1}$, instead of $\rho_1 \cdots \rho_J/2^{J-1}$ as is the case for distinct prime moduli. Therefore, this method of combination with a large prime m appears unfavorable compared to that of the previous section.

Proposition 4. *Under the assumptions made in this section, if* $(w_0, \ldots, w_{k-1}) = (u_0, \ldots, u_{k-1}) = (\tilde{u}_0, \ldots, \tilde{u}_{k-1})$*, then* $w_n = u_n = \tilde{u}_n$ *for all* $n \geq 0$*.*

Proof. Since $m_j = m$ for all j, it is clear from their definitions that $\tilde{u}_n = w_n$ for all n. It remains to show that $\{x_n\}$ (in (1)) and $\{z_n\}$ obey the same linear recurrence. Observe that the minimal polynomial of the recurrence $\{\delta_j x_{j,n},\ n \geq 0\}$ is $f_j(x)$, the same as for $\{x_{j,n},\ n \geq 0\}$. The polynomials $f_j(x)$ are also irreducible, because they were assumed to be primitive. Since the k_j's are distinct, these polynomials must be relatively prime. Then, from Theorem 6.57 in Lidl and Niederreiter (1986), it follows that the sequence $\{z_n,\ n \geq 0\}$, which is the sum of linear recurrences with respective minimal polynomials $f_1(x), \ldots, f_J(x)$, is a linear recurrence with minimal polynomial $f(x)$. $\square$

Proposition 5. *Under the same assumptions as in the previous proposition, suppose that for all* j*,* $(x_{j,0}, \ldots, x_{j,k_j-1}) \bmod m \neq (0, \ldots, 0)$*. Then, the period length of* $\{x_n,\ n \geq 0\}$ *is equal to* $\rho = \mathrm{lcm}(m^{k_1} - 1, \ldots, m^{k_J} - 1)$*. Note that* $(m-1)$ *is always a common factor. The largest possible value of* ρ *is* $\rho = (m^{k_1} - 1) \cdots (m^{k_J} - 1)/(m-1)^{J-1}$*, and it could be reached only if the* k_j*'s are pairwise relatively prime.*

Proof. The first part follows from the proof of the previous proposition and Theorem 6.59 of Lidl and Niederreiter (1986). The second part is a consequence of Corollary 3.7 of Lidl and Niederreiter (1986). $\square$

3. The Lattice Structure

For any positive integer t, define

$$T_t = \{\boldsymbol{u}_n = (u_n, \ldots, u_{n+t-1}) \mid n \geq 0,\ s_0 = (x_0, \ldots, x_{k-1}) \in \mathbb{Z}_m^k\}. \tag{16}$$

This is the set of all possible overlapping t-tuples of successive values produced by (10–11), from all possible initial states. Consider also the shift of T_t defined by $T_t' = (T_t - (0, \ldots, 0, b, \ldots, b^{t-k})) \bmod 1$. In general T_t does not necessarily contain the zero vector, but T_t' does. Let L_t' be the integer lattice generated by T_t' and $\mathbb{Z}_m^k$, that is, the set of all linear combinations of elements of T_t' and $\mathbb{Z}_m^k$, with integer coefficients, and let $L_t = L_t' + (0, \ldots, 0, b, \ldots, b^{t-k})$ be the grid (shift lattice), which contains T_t.

The points of L_t lie in a set of equidistant parallel hyperplanes (Knuth 1981) and one would like that the distance d_t between those hyperplanes be relatively small, in order to avoid large slices of empty space. For historical reasons, computing d_t is called the *spectral test*. Another popular quality measure for a lattice is the Beyer quotient q_t (L'Ecuyer 1990, L'Ecuyer, Blouin, and Couture 1993), defined as the ratio of lengths of a shortest and longest vectors in a Minkowski reduced basis for the lattice, and which should be close to one. The computer programs described in L'Ecuyer and Couture (995) permit one to compute d_t and q_t in reasonably large dimensions, up to around 40 or more.

Note that d_t is the same for both L_t and L_t'. On the other hand, when $\rho < m^k$, the points visited from any given initial state form a strict subset of T_t, which might generate a strict subgrid of L_t.

When there are both transient and recurrent states, it is more appropriate to analyze the set $T_{r,t}$ of t-tuples which are recurrent, since only those states are obtained by the combination. One has $T_{r,t} \subseteq T_t$, and the inclusion is strict when the k_j's are not all equal. Let $L_{r,t}$ and $L_{r,t}'$ denote the grid and lattice associated with $T_{r,t}$ (the analogues of L_t and L_t'). Couture and L'Ecuyer (1996) explain how to construct a lattice basis for $L_{r,t}$ and give several results and special techniques for computing d_t efficiently in large dimensions for combined generators.

The points $\tilde{\boldsymbol{u}}_n = (\tilde{u}_n, \ldots, \tilde{u}_{n+t-1})$, $n \geq 0$, produced by the combined generator (3) no longer belong to the grids or lattices described above, because of the "noise" ϵ_n. If we equate (or join) the opposite faces of the t-dimensional unit hypercube $[0,1]^t$, we obtain the t-dimensional unit torus. Computing the Euclidean distances in that torus is equivalent to "neglecting" the modulo 1 operation in (14) (see L'Ecuyer and Tezuka 1991 for further discussion). We then obtain that the Euclidean distance between $\tilde{\boldsymbol{u}}_n$ and $\boldsymbol{u}_n$ in the unit torus is bounded by $\Delta\sqrt{t}$, where $\Delta = \max(|\Delta^+|, |\Delta^-|)$. Typically, the values of ϵ_n are also pretty much evenly distributed between Δ^- and Δ^+. As a result, when $\Delta\sqrt{t}$ is larger than d_t, the hyperplane structure usually becomes unrecognizable.

4. Examples

We now give specific numerical examples. The first two should not be taken as serious proposals for random number generators; their purpose is just to give concrete illustrations of the possible effect of combination. The last two examples are more realistic and could be used as actual random number generators. We have applied a battery of statistical tests to one of them (Example 4) and give a C program implementing it.

Example 2. Let $J = 2$, $m_1 = 103$, $k_1 = 1$, $a_{1,1} = 40$, $m_2 = 101$, $k_2 = 3$, and $(a_{2,1}, a_{2,2}, a_{2,3}) = (29, 14, -15)$. Then, each component has full period, that is $\rho_1 = 102$ and $\rho_2 = 101^3 - 1 = 1030300$, and the period of the combination is $\rho = \rho_1\rho_2/2 = 52545300$. The recurrence (10) associated with the combination has order $k = 3$, modulus $m = 10403$, multipliers $(a_1, a_2, a_3) = (4675, 721, 4429)$, and $b = 0$. The latter recurrence has 10403^3 possible states, 103×101^3 of which are recurrent. Table I shows the distances d_t between successive hyperplanes, in dimensions 4 to 10, for the lattice L_t generated by all the 10403^3 states (first column) and for the (sub)lattice $L_{r,t}$ generated by the recurrent states (second column). The latter is the proper one to analyze in this case, and clearly contains much fewer points than the former. One may also be interested by the sublattice generated by the ρ states visited over one of the two main cycles of the combined generator: here this sublattice turns out to be the same as $L_{r,t}$.

Table I

The Values of d_t for Example 2, for each component, for all states, and for the recurrent states of the combination

t	d_t			
	component 1	component 2	full	recurrent
4	0.30151	0.11547	0.00127	0.01048
5	0.30151	0.11547	0.00582	0.02767
6	0.57735	0.12500	0.01048	0.04560
7	0.57735	0.12500	0.02767	0.07161
8	0.57735	0.20000	0.04560	0.10370
9	0.57735	0.22361	0.07161	0.12039
10	0.57735	0.25820	0.10370	0.16667

Example 3. Let $J = 2$, $m_1 = 103$, and $m_2 = 101$ as in Example 2, but we now take $k_1 = k_2 = 2$. With $(a_{1,1}, a_{1,2}) = (21, -21)$ and $(a_{2,1}, a_{2,2}) = (27, -18)$, both components have full period, that is, $\rho_1 = 102^2 - 1 = 10608$ and $\rho_2 = 101^2 - 1 = 10200$. In this case, $\gcd(\rho_1, \rho_2) = 408$, so $\rho = \rho_1\rho_2/408 = 265200$. The recurrence (10) has order $k = 2$, modulus

Table II
The Values of d_t for Example 3

t	d_t
3	0.00285
4	0.00996
5	0.02429
6	0.05361
7	0.08058
8	0.10847
9	0.15811
10	0.15811

$m = 10403$, and all its 10403^2 states are recurrent. Therefore, the lattice $L_{r,t}$ is the same as L_t. Table II gives the values of d_t. Note that this generator has 408 main cycles of length 265200. We also analyzed the lattice (or grid) generated by some of those main cycles (i.e., with different initial states) and it turned out that it was the same as L_t in each case. In general, this need not always be the case: the lattice (or grid) generated by one main cycle could be a strict sublattice (or subgrid) of $L_{r,t}$.

Example 4. For a more realistic combined generator, let us take $J = 2$, $m_1 = 2^{31} - 1 = 2147483647$, $m_2 = 2145483479$, $k_1 = k_2 = 3$, $(a_{1,1}, a_{1,2}, a_{1,3}) = (0, 63308, -183326)$, and $(a_{2,1}, a_{2,2}, a_{2,3}) = (86098, 0, -539608)$. Each component has period $\rho_j = m_j^3 - 1$, and the combination has period $\rho = \rho_1\rho_2/2$, which is close to 2^{205}. The MRG associated with the combination has order 3, modulus $m = m_1 m_2 = 4607390686061167913$, and multipliers $a_1 = 2620007610006878699$, $a_2 = 4374377652968432818$, and $a_3 = 667476516358487852$.

Here, all states are recurrent and they generate the same lattice as that generated by each of the two main cycles. Table III gives the values of d_t and q_t for each of the two components, as well as for the combination. For comparison, the best simple LCGs with modulus $m = 2^{31} - 1$ cannot have a value of d_t smaller than 0.01 in dimension 5 and .20 in dimension 20 (approximately). The 32-bit combined generator proposed by L'Ecuyer (1988), whose period length is near 2^{61}, can be approximated by a LCG with $d_5 \approx 0.0002$, $d_{10} \approx 0.017$, and $d_{20} \approx 0.10$ (see L'Ecuyer and Tezuka 1991).

With $\delta_1 = -\delta_2 = 1$, one has $\Psi^+ = \{2\}$, Ψ^- is empty, and the bounds (15) become

$$4.34 \times 10^{-13} \le \epsilon_n \le 9.31 \times 10^{-4}.$$

In fact, over the entire period, the values of ϵ_n are distributed (practically) uniformly between those two bounds. The Euclidean distance between the points $\boldsymbol{u}_n$ and $\tilde{\boldsymbol{u}}_n$ in the t-dimensional

Table III
The generator proposed in Example 4:
d_t and q_t for each component and for the combination

t	component 1		component 2		combination		
	d_t	q_t	d_t	q_t	d_t	q_t	$\Delta\sqrt{t}$
4	5.16E-6	9.0E-5	1.83E-6	2.5E-4	1.1E-14	0.6585	1.86E-3
5	5.16E-6	0.1611	3.28E-6	0.5952	6.6E-12	0.7558	2.08E-3
6	2.45E-5	0.6807	2.45E-5	0.3948	4.8E-10	0.7315	2.28E-3
7	1.21E-4	0.5722	1.16E-4	0.5146	9.80E-9	0.7866	2.46E-3
8	3.74E-4	0.6424	4.07E-4	0.5930	9.55E-8	0.7167	2.63E-3
9	9.24E-4	0.6590	8.26E-4	0.7049	6.00E-7	0.7491	2.79E-3
10	1.58E-3	0.7746	2.12E-3	0.4970	2.25E-6	0.6667	2.95E-3
11	3.60E-3	0.6983	3.86E-3	0.6364	8.41E-6	0.7563	3.09E-3
12	4.41E-3	0.7343	5.67E-3	0.6674	2.66E-5	0.6676	3.23E-3
13	6.67E-3	0.7700	7.21E-3	0.7353	4.68E-5	0.7255	3.36E-3
14	8.18E-3	0.9083	1.03E-2	0.7439	1.05E-4	0.7362	3.48E-3
15	1.25E-2	0.8629	1.28E-2	0.5947	1.60E-4	0.8171	3.61E-3
16	1.60E-2	0.7156	1.78E-2	0.5895	2.68E-4	0.8671	3.73E-3
17	2.14E-2	0.7818	2.24E-2	0.5804	4.26E-4	0.8619	3.84E-3
18	2.24E-2	0.8576	2.32E-2	0.8028	7.05E-4	0.9026	3.95E-3
19	2.77E-2	0.9080	3.11E-2	0.7368	1.03E-3	0.8665	4.06E-3
20	4.08E-2	0.8399	3.23E-2	0.8468	1.32E-3	0.8062	4.17E-3

unit torus is bounded by $\Delta\sqrt{t} = 0.000931\sqrt{t}$, which is given in the last column of Table III. One can see that in dimensions up to 20 (and more), the lattice structure is "detroyed by the noise", in the sense that the bound $0.000931\sqrt{t}$ remains larger than the distance d_t between successive hyperplanes. In some larger dimension (around 40), that bound is getting close to d_t, which means that $\|\boldsymbol{u}_n - \tilde{\boldsymbol{u}}_n\|$ is then approximately uniformly distributed between zero and d_t, so the "empty slices" between the hyperplanes are nicely filled up.

Figure I gives an implementation of the combined generator (3) in the C language. Translation into other procedural languages such as FORTRAN, Pascal, Modula-2, and so on, is trivial. The two MRG components are implemented along the lines described by L'Ecuyer (1990) and L'Ecuyer, Blouin, and Couture (1993). The function `Random` returns an integer in the range $[0,\, 2^{31} - 2]$, while `Uniform01` returns a value (strictly) between 0 and 1 (assuming that the "`double`" floating-point real numbers are represented with at least 32 bits for their mantissa).

This code assumes that all integers in the range $[-2^{31},\, 2^{31} - 1]$ are well represented. The global variables `x10` to `x22` hold the generator's state and represent $x_{1,n}$, $x_{1,n+1}$, $x_{1,n+2}$, $x_{2,n}$, $x_{2,n+1}$, $x_{2,n+2}$, respectively. They must be initialized, before the first call, to values that

```
int  m1 = 2147483647, m2 = 2145483479,
     a12 =   63308,    q12 =  33921,    r12 =  12979,
     a13 = -183326,    q13 =  11714,    r13 =   2883,
     a21 =   86098,    q21 =  24919,    r21 =   7417,
     a23 = -539608,    q23 =   3976,    r23 =   2071,
     x10, x11, x12, x20, x21, x22;
double  Invmp1 = 4.656612873077393e-10;

int Random()
   {
   int h, p12, p13, p21, p23;
   /* Component 1 */
   h = x10 / q13;    p13 = -a13 * (x10 - h * q13) - h * r13;
   h = x11 / q12;    p12 = a12 * (x11 - h * q12) - h * r12;
   if(p13 < 0)  p13 = p13 + m1;  if(p12 < 0)  p12 = p12 + m1;
   x10 = x11;  x11 = x12;  x12 = p12 - p13;  if(x12 < 0) x12 = x12 + m1;
   /* Component 2 */
   h = x20 / q23;    p23 = -a23 * (x20 - h * q23) - h * r23;
   h = x22 / q21;    p21 = a21 * (x22 - h * q21) - h * r21;
   if(p23 < 0)  p23 = p23 + m2;  if(p21 < 0)  p21 = p21 + m2;
   x20 = x21;  x21 = x22;  x22 = p21 - p23;  if(x22 < 0) x22 = x22 + m2;
   /* Combination */
   if (x12 < x22) return (x12 - x22 + m1); else return (x12 - x22);
   }

double Uniform01()
   {
   int Z;
   Z = Random ();   if (Z == 0) Z = m1;   return (Z * Invmp1);
   }
```

Figure I

A C implementation of the combined generator of Example 4

satisfy $0 \leq x_{j,i} \leq m_j - 1$ for $j = 1, 2$ and $0 \leq i \leq 2$, and $x_{j,i} > 0$ for at least one i, for each j. For example, initializing each of those variables to a value between 1 and 2145483478 will do. These six initial values constitute the *seed*.

In terms of speed, we should expect this proposed generator to take approximately twice the time as the 32-bit combined LCG proposed by L'Ecuyer (1988) and used as the basis of the random number package of L'Ecuyer and Côté (1991), because here we have four modular products to compute instead of two, and some additional modular sums. We checked that empirically by running the two combined generators on a SUN SPARCstation 20 under Solaris. Both were implemented in C and compiled with the "cc" compiler at optimization level `-O2`. To generate one million random numbers, the combined generator of L'Ecuyer (1988) took 7.8 seconds, while the combined MRG of Figure I took 17.2 seconds. For both generators, these timings correspond to implementations where the constants such as `a12`, `q12`, `r12`, and so on, are first declared and then used in the code, as in Figure I, so that the procedure is in generic form, independent of the specific multipliers and moduli. We also tested versions where the specific constants were replaced directly by their numerical values in the code, to speed up the execution. Then, the timings we got where 4.8 and 9.5 seconds, respectively. Finally, we also tried the latter implementations compiled using the "`-fast`" option of the cc compiler and the two generators then took 1.9 and 3.4 seconds, respectively, to generate the 10^6 random numbers. To make sure that the compiler was not optimizing out the calls to the generators because the random numbers where not used, we added up the random numbers while they where generated and printed the sum.

Clearly, there is a price to pay in terms of speed to get the longer period length and (theoretically) better properties of the combined MRG. That price could be negligible when the random number generation takes only a small fraction of the total computing time. If a program takes hours of cpu of a fast computer and most of that time is for generating the random numbers, then the generator's speed may be critical, but its statistical robustness more so. In other words, the latter situation is likely to be one for which "classical" LCGs, with period length of around 2^{31} or 2^{32}, could produce wrong results, because the fraction of the period length that is used is too large. It takes less than half an hour of cpu time to exhaust the period of a LCG with modulus $2^{31}-1$ on our SPARCstation 20. The combined Tausworthe generators proposed by Tezuka and L'Ecuyer (1991) are also faster than that of Figure I, but have a shorter period length (near 2^{60}).

We applied a battery of empirical statistical tests to that generator: we ran the same 21 tests as in L'Ecuyer (1988), as well as the 10 tests used in L'Ecuyer (1992). The generator easily passed all those tests (the detailed results are available from the author).

The sequence produced by the generator can be split into disjoint subtreams by starting the generator from different initial states, spaced far apart in the original sequence. Based on that, one can build a package with several virtual generators, as in L'Ecuyer and Côté (1991). Those initial states can be computed easily if jumping ahead facilities are available for the individual MRG components; that is, if an efficient algorithm is available for computing the state of the MRG, say, ν steps ahead of the current one, for large values of ν. L'Ecuyer (1990), p.88 explains one way of doing that, based on the fact that the MRG (1) can be viewed as a LCG in matrix form, whose state is a k-dimensional vector and whose multiplier is a $k \times k$ matrix A. To jump ahead by ν values, just multiply the current state by A^ν, modulo m. The matrix $A^\nu \bmod m$ can be precomputed in time $O(\log \nu)$, using again the divide-to-conquer algorithm mentioned in Section 1.

Example 5. Let us now combine an MRG with a LCG with power-of-two modulus. We take $J=2$, $m_1 = 2^{32}$, $k_1 = 1$, $a_{1,1} = 738801091$, $b_2 = 1$, while $m_2 = 2^{31}-1 = 2147483647$, $k_2 = 3$, $(a_{2,1}, a_{2,2}, a_{2,3}) = (0, 377579228, -472831176)$, and $b_2 = 0$. The period lengths are $\rho_1 = 2^{32}$ for component 1, $\rho_2 = (2^{31}-1)^3 - 1$ for component 2, and $\rho = \rho_1\rho_2/2 \approx 2^{124}$ for the combination. Table IV gives the values of d_t and q_t for each component and for the combination (for the recurrent states).

Table IV
The generator of Example 5: d_t and q_t for each component and for the combination

t	component 1		component 2		combination	
	d_t	q_t	d_t	q_t	d_t	q_t
4	0.0112	0.4305	7.81E-7	5.9E-4	4.0E-10	0.8713
5	0.0249	0.5758	2.76E-6	0.8189	3.30E-8	0.7163
6	0.0347	0.8319	2.55E-5	0.6142	5.23E-7	0.8293
7	0.0700	0.7189	1.36E-4	0.4791	4.24E-6	0.6704
8	0.0839	0.7105	4.49E-4	0.6752	1.90E-5	0.7040
9	0.1163	0.6176	6.92E-4	0.8495	7.43E-5	0.6166
10	0.1250	0.7629	1.67E-3	0.4942	1.67E-4	0.6960
11	0.1543	0.7563	2.46E-3	0.7763	3.76E-4	0.6751
12	0.1667	0.7222	4.32E-3	0.7654	7.28E-4	0.7248
13	0.2500	0.7760	7.05E-3	0.5302	1.18E-3	0.8302
14	0.2500	0.7295	9.11E-3	0.7304	1.95E-3	0.7469
15	0.2500	0.7421	1.34E-2	0.7291	2.76E-3	0.8238
16	0.2887	0.6784	1.54E-2	0.8085	3.98E-3	0.7579
17	0.2887	0.7697	1.97E-2	0.8185	5.18E-3	0.7716
18	0.2887	0.8008	2.40E-2	0.8437	7.25E-3	0.8111
19	0.2887	0.7918	3.40E-2	0.7923	9.33E-3	0.7101
20	0.2887	0.8185	3.44E-2	0.7870	1.10E-2	0.7929

ACKNOWLEDGMENT

This work has been supported by NSERC-Canada grant # OGP0110050 and FCAR-Québec grant # 93ER1654. I wish to thank Raymond Couture, David Kelton, and the referees for their useful comments, Luc De Bellefeuille who helped writing the C program of Figure 1, and Jean-François Cordeau who helped performing the statistical tests.

REFERENCES

Brassard, G., and P. Bratley. 1988. *Algorithmics: Theory and Practice.* Prentice Hall.

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

Couture, R., and P. L'Ecuyer. 1996. Orbits and lattices for linear random number generators with composite moduli. *Mathematics of Computation*, **66**. To appear.

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

L'ECUYER, P. 1988. Efficient and portable combined random number generators. *Communications of the ACM*, **31**(6), 742–749 and 774. See also the correspondance in the same journal, 32, 8 (1989) 1019–1024.

L'ECUYER, P. 1990. Random numbers for simulation. *Communications of the ACM*, **33**(10), 85–97.

L'ECUYER, P. 1992. Testing random number generators. In *Proceedings of the 1992 Winter Simulation Conference*, 305–313. IEEE Press.

L'ECUYER, P. 1994. Uniform random number generation. *Annals of Operation Research*, **53**, 77–120.

L'ECUYER, P. 1995 ? Bad lattice structures for vectors of non-successive values produced by some linear recurrences. *ORSA Journal on Computing*. To appear.

L'ECUYER, P., F. BLOUIN., AND R. COUTURE. 1993. A search for good multiple recursive random number generators. *ACM Transactions on Modeling and Computer Simulation*, **3**(2), 87–98.

L'ECUYER, P., AND S. CÔTÉ. 1991. Implementing a random number package with splitting facilities. *ACM Transactions on Mathematical Software*, **17**(1), 98–111.

L'ECUYER, P., AND R. COUTURE. 1995 ? An implementation of the lattice and spectral tests for linear congruential and multiple recursive generators. Submitted.

L'ECUYER, P., AND S. TEZUKA. 1991. Structural properties for two classes of combined random number generators. *Mathematics of Computation*, **57**(196), 735–746.

LIDL, R., AND H. NIEDERREITER. 1986. *Introduction to Finite Fields and Their Applications*. Cambridge: Cambridge University Press.

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.

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.

WANG, D., AND A. COMPAGNER. 1993. On the use of reducible polynomials as random number generators. *Mathematics of Computation*, **60**, 363–374.

WICHMANN, B. A., AND I. D. HILL. 1982. An efficient and portable pseudo-random number generator. *Applied Statistics*, **31**, 188–190. See also corrections and remarks in the same journal by Wichmann and Hill, **33** (1984) 123; McLeod **34** (1985) 198–200; Zeisel **35** (1986) 89.