**Les cahiers du GERAD** ISSN: 0711-2440

# Efficient and Portable Combined Tausworthe Random Number Generators

S. Tezuka, P. L'Ecuyer

G–90–54

December 1990

# Efficient and Portable Combined Tausworthe Random Number Generators

Shu Tezuka
IBM Research
Tokyo Research Laboratory [1]

Pierre L'Ecuyer
Département d'I.R.O.
Université de Montréal [2]

November 1990

[1] 5-19 Sanbancho, Chiyoda-ku, Tokyo 102, Japan
[2] C.P. 6128, Succ. A, Montréal, H3C 3J7, Canada

**Abstract**

In this paper, we propose three combined Tausworthe random number generators with period length about $10^{18}$, whose $k$-distribution properties are good, and which can be implemented in a portable way. These generators are found through an exhaustive search for the combination with the best lattice structure in $GF\{2,x\}^k$, the $k$-dimensional vector space over the field of all Laurent series with coefficients in $GF(2)$. We then apply a battery of statistical tests to these generators for the comprehensive investigation of their empirical statistical properties. No apparent defect was found. In the appendix, we give a sample program in C for the generators.

**KEYWORDS:** Tausworthe sequences, spectral test, empirical statistical test

**Résumé**

Nous proposons trois générateurs combinés de type Tausworthe dont la période est approximativement de $10^{18}$, pour lesquels les vecteurs de $k$ valeurs successives sont bien distribués, et que l'on peut implanter facilement de façon portable. Ces générateurs sont le résultat d'une recherche exhaustive pour trouver les combinaisons ayant les structures de treillis de meilleure qualité dans $GF\{2,x\}^k$, l'espace vectoriel à $k$ dimensions défini sur le corps de séries de Laurent à coefficients dans $GF(2)$. Nous appliquons ensuite une batterie de tests statistiques à ces générateurs afin d'examiner leurs propriétés statistiques empiriques. Nous n'avons trouvé aucun défaut apparent. En annexe, nous donnons une implantation en langage C de ces générateurs.

# 1 Introduction

Combining two or more pseudorandom sequences has recently drawn much attention in the field of random number generation [2,6,8,11,12,18]. It has the potential of producing generators with larger period length, better randomness properties, and that are easy to implement. Combinations of linear congruential generators have been studied in [6,8,18]. It turns out that for the classes of combination suggested in [18] and [6], the combined generator is also linear congruential and almost linear congruential, respectively. Linear congruential generators have a lattice structure which can be characterized to some extent by the so-called Beyer and spectral tests [5,7]. It is well accepted that generators with bad (or too coarse) lattice structure must be avoided. Therefore, it is important to make sure that the combined generator (and not only each of its components) does not suffer from a bad lattice structure. Some pathological examples of combination are discussed in [8,16].

Tausworthe sequences, which do not possess the same kind of integer lattice structure, have been investigated as an alternative to linear congruential sequences. Recently, a theory for the randomness of these sequences, similar to that of linear congruential sequences, has been developed [14,15]. The theory shows that Tausworthe sequences have a lattice structure in the $k$-dimensional space over $GF\{2,x\}$, the field of all Laurent series over $GF(2)$. Consider for example the Tausworthe sequence whose two-dimensional plot is shown in Figure 1. Suppose we apply Marsaglia's lattice test[11], which examines the smallest integer lattice spanned by the set of all pairs of consecutive terms in the sequence (i.e. all integer linear combinations of these pairs). By hand calculation (see Appendix A), we can see easily that this lattice contains all points with integer coordinates.

Fast software implementations of Tausworthe generators are known, but only for special cases, e.g., trinomials of small degree. Unfortunately, these special cases have poor statistical properties [11,17]. Primitive trinomials of large degree (e.g., degree 521 [4]) have been suggested, but fast implementations without using extra array of memory are not available. André et al. [1] discuss how to find good Tausworthe sequences with respect to the two-dimensional discrepancy and suggest some. But again, the resulting sequences are slow to generate, since their characteristic polynomials have many nonzero coefficients.

In a companion paper [15], Tezuka proposed the XOR (bit-wise exclusive-or) combination of Tausworthe sequences and derived a sufficient condition for the combined sequences to have a good $k$-dimensional distribution property. A practical merit of this combination is that one can generate pseudorandom sequences with large period length, which have a good $k$-distribution property, in a portable and efficient way. The objective of this paper is twofold. First, we conduct an exhaustive search for the best combination of two Tausworthe sequences, where one is of period $2^{31}-1$ and the other, $2^{29}-1$, yielding a combined generator of period length $(2^{31}-1)(2^{29}-1) \approx 2^{60}$. Second, we perform empirical statistical tests on the resulting combined generators. The paper is organized as follows. Section 2 briefly overviews the definition of combined Tausworthe generators and a theorem for the good $k$-distribution of such sequences. In Section 3, we describe an exhaustive search conducted for finding combined Tausworthe sequences with good $k$-distribution properties. Three generators were retained. In Section 4, we perform a comprehensive study of these three generators on the basis of a

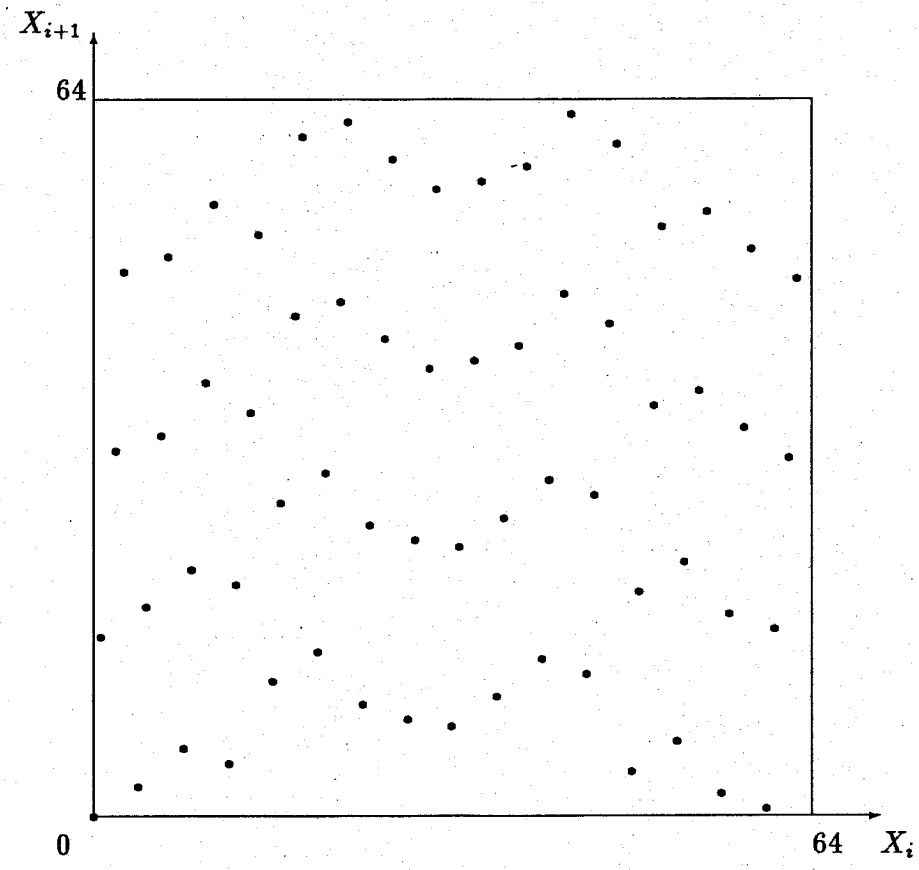


Figure 1: The pairs $(X_i, X_{i+1})$, for $i = 1, \ldots, 63$, produced by the Tausworthe sequence $X_i = \sum_{j=1}^{6} a_{4i+j} 2^{6-j}$, where $\{a_j\}$ follows the recurrence $a_j = (a_{j-5} + a_{j-6}) \bmod 2$.

battery of empirical statistical tests. Section 5 gives a quick comparison of the generators proposed here with the combination of linear congruential sequences.

# 2 Overview of combined Tausworthe sequences

## 2.1 LS2 generators and Tausworthe sequences

Let $GF\{2,x\}$ denote the field of all Laurent series of the form $S(x) = \sum_{j=-\infty}^{m} c_j x^j$, with $m$ integer and $c_j$ in $GF(2)$. Here we define an analogous version of linear congruential sequences in $GF\{2,x\}$. Let $\sigma$ be a mapping from $GF\{2,x\}$ to the real field, defined as

$$\sigma(S(x)) = S(2).$$

Then a pseudorandom sequence $u_i$, $i = 1, 2, ...$, in [0,1) is defined as

$$\begin{aligned} f_i(x) &= (g(x)f_{i-1}(x) + h(x)) \bmod M(x) \\ u_i &= \sigma(f_i(x)/M(x)), \end{aligned} \tag{1}$$

where $g(x), h(x), M(x)$ and $f_i(x)$ are polynomials in $GF\{2,x\}$. We denote this generator by the triplet $G = (g, h, M)$ and call it a LS2 generator. In practical situations, $u_i$ is expressed approximately by its truncated value, i.e. by summing from some constant $-L$.

A Tausworthe sequence is a special case of the above general class. Let $M(x)$ be a primitive polynomial of degree $p$ over $GF(2)$, $h(x) \equiv 0$, $g(x) = (x^s \bmod M(x))$, with $0 \leq s < 2^p - 1$, $gcd(s, 2^p - 1) = 1$, $m = -1$, and $L$ be the "word-size". Then, for $i = 1, 2, \ldots$, the Tausworthe sequence [13,15] can be written as

$$u_i = \sum_{j=1}^{L} a_{is+j} 2^{-j}, \tag{2}$$

where $\{a_j,\ j \geq 1\}$ is a binary sequence generated by the linear recurrence whose characteristic polynomial is $M(x)$. Note that the digital multi-step sequences over $GF(2)$ discussed in [1] are a special case of Tausworthe sequences, i.e., $0 < L = s \leq p$.

We now give an algorithm for the case where $M(x)$ is a primitive trinomial of the form $M(x) = x^p + x^q + 1$, with $q < p/2$, and $g(x) = x^s$, with $0 < s \leq p - q$. Each term of the sequence $\{u_i\}$ is expressed by its leading $L = p$ bits in this algorithm. This can be implemented easily, in a "portable" language that supports shifting and bitwise-XOR operations (like C), if $p$ [=L] is not larger than the computer's word-size.

**An algorithm for implementing (2) when $0 < s \leq p - q$:**

Step 0: A and B are $p$-bit words.
Step 1: B $\longleftarrow$ $q$ bit left shift of A
Step 2: B $\longleftarrow$ A XOR B
Step 3: B $\longleftarrow$ $p - s$ bit right shift of B
Step 4: A $\longleftarrow$ $s$ bit left shift of A
Step 5: A $\longleftarrow$ A XOR B
Step 6: Output A as the leading $p$ bits of $u_i$, return to Step 1.

The generators based on the above algorithm with $p$ not large, say $p \leq 64$, do not perform well in high dimensions, because linear dependency appears in the leading bits of the $k$-tuple $(u_i, ..., u_{i+k-1})$ when $k > \lfloor p/s \rfloor$ for a fixed $s$. (For further discussion, see Tootill et al. [17].) This is one reason why we investigate the combination of Tausworthe sequences in this paper.

## 2.2 Combination of Tausworthe sequences

Let $J \geq 2$. For each $j = 1, ..., J$, let $G_j = (g_j, 0, M_j)$ be a LS2 generator, where $g_j(x) = (x^{s_j} \bmod M_j(x))$, $M_j(x)$ is a primitive polynomial of degree $p_j$, and $gcd(s_j, 2^{p_j} - 1) = 1$. Let $\{u_i^{(j)},\ i \geq 1\}$ denote the corresponding sequence, whose period is $2^{p_j} - 1$. The combined sequence is defined by

$$V_i = u_i^{(1)} \text{ XOR } \cdots \text{ XOR } u_i^{(J)}. \tag{3}$$

Let $M(x) = m_1(x) \cdots m_J(x)$. For each $j$ in $\{1, \ldots, J\}$, define $M_{-j}(x) = M(x)/m_j(x)$ and let $n_j(x)$ be a polynomial such that $n_j(x)M_{-j}(x) \bmod M_j(x) = 1$. Let $g(x) = \sum_{j=1}^{J} g_j(x)n_j(x)M_j(x)$. The following theorem (see [15]) characterizes the combined sequence.

**Theorem 1** *The sequence $\{V_i\}$ corresponds to the LS2 generator $G = (g, 0, M)$ and its period equals the least common multiple of $(2^{p_1} - 1, \ldots, 2^{p_J} - 1)$.*

Tezuka [15] obtained a theorem that links the $k$-distribution of the sequences defined in (1) with the successive minima and reduced basis of a lattice in a vector space over $GF\{2, x\}$. Consider the $k$-tuples $(f_i(x)/M(x), ..., f_{i+k-1}(x)/M(x))$, $i = 1, 2, \ldots$, produced by (1). These are expressed by the grid (shifted lattice) $L_k + \lambda$, where $L_k$ is a lattice with basis

$$\begin{aligned} e_1 &= \frac{1}{M(x)}(1, g(x), g^2(x), \ldots, g^{k-1}(x)), \\ e_2 &= (0, 1, 0, \ldots, 0), \\ &\vdots \quad \vdots \\ e_k &= (0, 0, 0, \ldots, 1), \end{aligned}$$

and $\lambda = (0, 1, 1 + g(x), \ldots, 1 + g(x) + \cdots + g^{k-2}(x))(h(x)/M(x))$.

Define the norm of a vector $\alpha = (f_1, ..., f_k)$, where each $f_i$ is in $GF\{2, x\}$, as

$$|\alpha| = max_{1 \leq i \leq k}[deg(f_i)].$$

In this paper, the notions of reduced basis and successive minima of a lattice $L$ in a vector space over $GF\{2, x\}$ follow Lenstra's definitions [9]:

**Definition 1** *For $1 \leq j \leq k$, a $j$-th successive minimum $|b_j|$ of $L$ is recursively defined as the norm of a vector of a smallest norm in $L$ that is linearly independent of $b_1, b_2, ..., b_{j-1}$ over $GF\{2, x\}$, and the basis $b_1, b_2, ..., b_k$ is called a reduced basis of $L$.*

Let $l_k$ be the $k$-th successive minimum of the lattice $L_k$ associated with the combined generator $G = (g, 0, M)$. For each $i, j$ in $\{1, \ldots, J\}$, $i \neq j$, define $M_{-ij}(x) = M(x)/(m_i(x)m_j(x))$,

let $n_{ij}(x)$ be a polynomial such that $n_{ij}(x)M_{-ij}(x) \bmod M_i(x)M_j(x) = 1$, and let $g_{-j}(x) = \sum_{j=1}^{J}\sum_{i=1,\, j\neq j}^{J} g_j(x)n_{ij}(x)M_{-ij}(x)$.

Let $l_1^{(j)}$ be the first successive minimum of the lattice $L_k^{(j)}$ associated with the generator $G_j = (g_{-j}, 0, M_{-j})$. An *equidissection* of the $k$-dimensional unit hypercube into $2^{k\ell}$ cubic cells is defined as the set of all cubic cells in $[0,1)^k$ with side length of $2^{-\ell}$ and whose corners have coordinates that are all multiples of $2^{-\ell}$. That is

$$S_k(\ell) = \left\{(i_1 2^{-\ell}, (i_1+1)2^{-\ell}) \times \cdots \times (i_k 2^{-\ell}, (i_k+1)2^{-\ell}) \;\mid 0 \le i_1 < 2^\ell, \ldots, 0 \le i_k < 2^\ell\right\}.$$

Let $p = p_1 + \cdots + p_k$ be the degree of $M(x)$. The following theorem is proven in [15].

**Theorem 2** *If $l_1^{(j)} \ge l_k$ for $j = 1, ..., J$, and if the combined generator $G$ has the maximum possible period $(2^{p_1}-1) \times \ldots \times (2^{p_J}-1)$, then (1) the number of points in each cell of the equidissection $S_k(-l_k)$ is equal to an integer in the range $[2^{p+kl_k} - J, 2^{p+kl_k}]$, and (2) the number of cells containing $2^{p+kl_k}$ points is at least $2^{-kl_k} - D$, where $D = 2^p - (2^{p_1}-1) \times \ldots \times (2^{p_J}-1)$.*

Roughly speaking, the smaller the (negative) value of $l_k$ is, the better the $k$-distribution of the sequence becomes. Note that the constant $D$ in Theorem 2 is much less than $2^{-kl_k}$ if $-kl_k$ is near $p$ and $J$ is less than $p_j$ for each $j$. These conditions are usually met in practical situations.

# 3 Exhaustive search for the best combination

We combine two Tausworthe generators, each of them from a specific class. Generators of the first class are those of the form $G_1 = (x^{s_1}, 0, x^{31} + x^q + 1)$, with $0 < s_1 \le 31 - q$, for $q = 3, 6, 7, 13,$. Their period length is $2^{31} - 1$, a prime. Those of the second class have the form $G_2 = (x^{s_2}, 0, x^{29} + x^2 + 1)$, with $0 < s_2 \le 27$, and their period length is $2^{29} - 1 = 233 \times 1103 \times 2089$. Hence $gcd(2^{29} - 1, s_2) = 1$. Each of these generators can be implemented using the algorithm given in Section 2.1. We conducted exhaustive search for finding the best combined generator among all 2565 possible combinations of $G_1$ and $G_2$. For this search we used two types of criteria. The first criterion (to be minimized) is defined as

$$S_{15}(G) = \max_{2 \le k \le 15} ([60/k] + l_k),$$

where $l_k$ is the last successive minimum of the $k$-dimensional lattice associated with the combined generator $G$. This criterion is an analog of the one used in [3,6], which is based on the spectral test for linear congruential sequences. Its basic idea is to compare the value of $l_k$ to its theoretical upper bound. By using this, we found no generators with $S_{15}(G) = 0$ and 207 generators with $S_{15}(G) = 1$. For further selection, we tried to minimize

$$C_{15}(G) = \sum_{k=2}^{15} ([60/k] + l_k)$$

among those generators for which $S_{15}(G) = 1$. We obtained three combined generators with $C_{15}(G) = 2$. The values of the first and last successive minima for the selected combined generators, and for their components, are given in Tables 1 through 3.

A second criterion is based on the Beyer's ratio, which is the measure used in the lattice test for linear congruential sequences. This is defined as

$$L_{15}(G) = \max_{2 \le k \le 15} (l_k - l_1),$$

where $l_1$ is the first successive minimum of the lattice associated with the combined generator $G$ in $k$ dimensions. By using this, we found no generator with $L_{15}(G) = 0$ or 1, but 158 generators with $L_{15}(G) = 2$. For further selection, among those 158, we searched those minimizing

$$B_{15}(G) = \sum_{k=2}^{15} (l_k - l_1).$$

We obtained two generators with $B_{15}(G) = 10$, which are exactly those of Tables 1 and 3. The generator of Table 2 has $L_{15}(G) = 2$ and $B_{15}(G) = 11$. We see that the two criteria agree quite well.

As shown in Theorem 1, these generators can be written in the form (1) as follows: the generator in Table 1 is $(x^{59} + x^{56} + x^{54} + x^{53} + x^{49} + x^{48} + x^{47} + x^{46} + x^{44} + x^{42} + x^{39} + x^{38} + x^{36} + x^{35} + x^{34} + x^{33} + x^{31} + x^{30} + x^{29} + x^{26} + x^{25} + x^{24} + x^{22} + x^{21} + x^{18} + x^{17} + x^{13} + x^{12} + x^{11} + x^{10} + x^8 + x^3 + x^2, 0, x^{60} + x^{42} + x^{33} + x^{31} + x^{29} + x^{15} + x^{13} + x^2 + 1)$; the generator in Table 2 is $(x^{57} + x^{55} + x^{54} + x^{53} + x^{50} + x^{46} + x^{45} + x^{44} + x^{41} + x^{39} + x^{38} + x^{37} + x^{34} + x^{32} + x^{31} + x^{29} + x^{27} + x^{25} + x^{24} + x^{23} + x^{21} + x^{19} + x^{18} + x^{17} + x^{16} + x^{15} + x^{14} + x^{11} + x^9 + x^8 + x^7 + x^4 + x + 1, 0, x^{60} + x^{33} + x^{32} + x^{31} + x^{29} + x^5 + x^3 + x^2 + 1)$; the generator in Table 3 is $(x^{59} + x^{57} + x^{56} + x^{54} + x^{52} + x^{49} + x^{44} + x^{43} + x^{41} + x^{40} + x^{38} + x^{35} + x^{33} + x^{28} + x^{23} + x^{22} + x^{17} + x^{16} + x^{15} + x^{13} + x^{12} + x^9 + x^8 + x^5 + x^4 + x^3 + x^2 + 1, 0, x^{60} + x^{42} + x^{33} + x^{31} + x^{29} + x^{15} + x^{13} + x^2 + 1)$.

From Theorem 2, the $k$-distribution of the combined sequences can be summarized as follows: *For any* $k = 2, ..., 15$ *and* $j = 1, 2$, *we have* $l_1^{(j)} \ge l_k$ *and the period of the combined generator* $G$ *is equal to* $(2^{31} - 1) \times (2^{29} - 1)$; *therefore, the number of points in each cell of the equidissection* $S_k(-l_k)$ *is equal to an integer in the range* $[2^{60+kl_k} - 2, 2^{60+kl_k}]$, *and the number of cells containing* $2^{60+kl_k}$ *points is at least* $2^{-kl_k} - D$, *where* $D = 2^{31} + 2^{29} - 1$.

Table 1: The norm of the first and $k$-th successive minima of the lattices associated with the generators $G$, $G_1 = (x^{12}, 0, x^{31} + x^{13} + 1)$, and $G_2 = (x^{17}, 0, x^{29} + x^2 + 1)$, for dimensions $k = 2, ..., 15$.

| Dimension | $k$ | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Maximum resolution | $[60/k]$ | 30 | 20 | 15 | 12 | 10 | 8 | 7 | 6 | 6 | 5 | 5 | 4 | 4 | 4 |
| The combined generator | $-l_k$ | 30 | 19 | 15 | 12 | 10 | 8 | 7 | 6 | 6 | 5 | 5 | 4 | 4 | 3 |
| G | $-l_1$ | 30 | 21 | 15 | 12 | 10 | 9 | 8 | 7 | 6 | 6 | 5 | 5 | 5 | 5 |
| The generator | $-l_k^{(1)}$ | 12 | 7 | 6 | 5 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 |
| $G_1$ $(p_1 = 31)$ | $-l_1^{(1)}$ | 19 | 12 | 11 | 7 | 7 | 6 | 5 | 5 | 5 | 5 | 4 | 3 | 3 | 3 |
| The generator | $-l_k^{(2)}$ | 12 | 7 | 6 | 5 | 3 | 3 | 3 | 3 | 2 | 2 | 2 | 2 | 2 | 1 |
| $G_2$ $(p_2 = 29)$ | $-l_1^{(2)}$ | 17 | 12 | 10 | 6 | 6 | 5 | 4 | 4 | 3 | 3 | 3 | 3 | 3 | 2 |

Table 2: The norm of the first and $k$-th successive minima of the lattices associated with the generators $G$, $G_1 = (x^{21}, 0, x^{31} + x^3 + 1)$, and $G_2 = (x^{17}, 0, x^{29} + x^2 + 1)$, for dimensions $k = 2, ..., 15$.

| Dimension | $k$ | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Maximum resolution | $[60/k]$ | 30 | 20 | 15 | 12 | 10 | 8 | 7 | 6 | 6 | 5 | 5 | 4 | 4 | 4 |
| The combined generator | $-l_k$ | 29 | 20 | 15 | 12 | 10 | 8 | 7 | 6 | 5 | 5 | 5 | 4 | 4 | 4 |
| G | $-l_1$ | 31 | 20 | 15 | 12 | 10 | 9 | 9 | 7 | 7 | 6 | 5 | 5 | 5 | 4 |
| The generator | $-l_k^{(1)}$ | 10 | 10 | 7 | 4 | 4 | 3 | 3 | 3 | 3 | 2 | 2 | 2 | 2 | 2 |
| $G_1$ $(p_1 = 31)$ | $-l_1^{(1)}$ | 21 | 11 | 10 | 7 | 6 | 6 | 5 | 4 | 4 | 3 | 3 | 3 | 3 | 3 |
| The generator | $-l_k^{(2)}$ | 12 | 7 | 6 | 5 | 3 | 3 | 3 | 3 | 2 | 2 | 2 | 2 | 2 | 1 |
| $G_2$ $(p_2 = 29)$ | $-l_1^{(2)}$ | 17 | 12 | 10 | 6 | 6 | 5 | 4 | 4 | 3 | 3 | 3 | 3 | 3 | 2 |

Table 3: The norm of the first and $k$-th successive minima of the lattices associated with the generators $G$, $G_1 = (x^{13}, 0, x^{31} + x^{13} + 1)$, and $G_2 = (x^{20}, 0, x^{29} + x^2 + 1)$, for dimensions $k = 2, ..., 15$.

| Dimension | $k$ | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Maximum resolution | $[60/k]$ | 30 | 20 | 15 | 12 | 10 | 8 | 7 | 6 | 6 | 5 | 5 | 4 | 4 | 4 |
| The combined generator | $-l_k$ | 30 | 20 | 14 | 12 | 10 | 8 | 7 | 6 | 5 | 5 | 5 | 4 | 4 | 4 |
| G | $-l_1$ | 30 | 20 | 16 | 12 | 10 | 9 | 8 | 7 | 7 | 6 | 5 | 5 | 5 | 4 |
| The generator | $-l_k^{(1)}$ | 13 | 5 | 5 | 5 | 3 | 3 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 1 |
| $G_1$ $(p = 31)$ | $-l_1^{(1)}$ | 18 | 13 | 13 | 8 | 8 | 5 | 5 | 5 | 5 | 5 | 3 | 3 | 3 | 3 |
| The generator | $-l_k^{(2)}$ | 9 | 9 | 6 | 4 | 4 | 3 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 1 |
| $G_2$ $(p = 29)$ | $-l_1^{(2)}$ | 20 | 11 | 9 | 7 | 6 | 5 | 5 | 5 | 4 | 3 | 3 | 3 | 3 | 2 |

# 4 Empirical test results for the generators

We have submitted the three retained generators to the same battery of 21 statistical tests as in [6]. The final result of each test is the value $s$ of a Kolmogorov-Smirnov statistic $S$. A generator fails the test if the observed descriptive level $\delta = \Pr(S \geq s \mid H_0)$ is "too small", where $H_0$ denotes the (null) hypothesis that the generator is perfectly random. The results of the tests appear in Table 4, where $\delta_i$ represents the observed value of $\delta$ for the $i$-th combined generator. The test numbers correspond to those in [6]. For each test, the initial seeds for the two Tausworthe components were the binary representations of 12345 and 67890, respectively. The only "suspect" value was produced by the test number 20 (a collision test) for the first combined generator. Of course, this value could have been produced by chance and to detect that, we repeated the same test on the same generator, starting with different seeds. We used as starting seeds the final values at the end of the "first-trial" test. The result of the second trial was $\delta_1 = 0.1391$. We conclude that there is no apparent defect in the generators.

Table 4: Descriptive levels for the empirical statistical tests.

| Test | $\delta_1$ | $\delta_2$ | $\delta_3$ |
|---|---|---|---|
| 1 | .2403 | .3284 | .0850 |
| 2 | .3642 | .4169 | .2201 |
| 3 | .7886 | .9873 | .9192 |
| 4 | .9457 | .1646 | .4985 |
| 5 | .7775 | .7246 | .9266 |
| 6 | .4964 | .2750 | .4712 |
| 7 | .4017 | .6091 | .9271 |
| 8 | .2877 | .9440 | .5505 |
| 9 | .6561 | .6425 | .4988 |
| 10 | .7164 | .9449 | .5307 |
| 11 | .8686 | .7105 | .3142 |
| 12 | .3989 | .7195 | .8699 |
| 13 | .5621 | .9708 | .4851 |
| 14 | .4942 | .8660 | .2536 |
| 15 | .3471 | .8628 | .7277 |
| 16 | .3880 | .5238 | .8219 |
| 17 | .7324 | .5451 | .4932 |
| 18 | .1054 | .0683 | .7002 |
| 19 | .2989 | .2162 | .2220 |
| 20 | .0178 | .3219 | .6505 |
| 21 | .1785 | .4960 | .0859 |

# 5 Discussion and conclusion

Combined linear congruential generators, as proposed in [6,18], still possess a (sometimes approximate) integer lattice structure. As shown in [8], one can get rid of it in lower dimensions, with an appropriate choice of parameters. The combined Tausworthe generators proposed here do not have such a lattice structure. A portable implementation in C is given in Appendix B On a PC-386 (16 MHz) with a 80387 coprocessor, generating one million random numbers with our implementations took 110 seconds for the combined generator of L'Ecuyer [6] and 154 seconds for the combined Tausworthe implementation suggested in Appendix B. We wrote implementations in Modula-2, then in C, and both had roughly the same speed.

# Acknowledgements

This work has been supported by NSERC-Canada grant # A5463 and FCAR-Québec grant # EQ2831 to the second author. Pierre Audet helped implementing the generators and performing the statistical tests. Raymond Couture, Ben Fox and Masanori Fushimi gave valuable comments and discussions.

# References

[1] D.A. André, G.L. Mullen, and H. Niederreiter, " Figures of merit for digital multistep pseudorandom numbers", *Math. Comp.*, **54** (1990), 737–748.

[2] P. Bratley, B.L. Fox, and L.E. Schrage, *A Guide to Simulation*, 2nd ed., Springer-Verlag, 1987.

[3] G.S. Fishman and L.S. Moore, III, "An exhaustive analysis of multiplicative congruential random number generators with modulus $2^{31}-1$", *SIAM J. Sci. Stat. Comput.*, **7** (1986), 24–45.

[4] M. Fushimi, "Designing a uniform random number generator whose subsequences are k-distributed", *SIAM J. Computing*, **17** (1988), 89–99.

[5] D.E. Knuth, *The Art of Computer Programming: Vol. 2, Seminumerical Algorithms*, 2nd ed., Addison-Wesley, 1981.

[6] P. L'Ecuyer, " Efficient and portable combined random number generators", *Comm. ACM.*, **31** (1988), 742–749, 774.

[7] P. L'Ecuyer, "Random numbers for simulation", to appear in *Comm. ACM.*, (Oct. 1990).

[8] P. L'Ecuyer and S. Tezuka, " Structural properties for two classes of combined generators", in preparation, 1990.

[9] A.K. Lenstra, " Factoring multivariate polynomials over finite fields",*J. Comput. Syst. Sci.*, **30** (1985), 235–248.

[10] G. Marsaglia, "The structure of linear congruential sequences", in *Applications of Number Theory to Numerical Analysis*, Academic Press, New York, 1972, 249–285.

[11] G. Marsaglia, " A current view of random number generators", in *Computer Science and Statistics: Interface*, Elsevier Science Publishers B.V., North-Holland, New York, 1985, 3–10.

[12] G. Marsaglia, A. Zaman, and W. W. Tsang, "Towards a Universal Random Number Generator", *Stat. and Prob. Letters*, **8** (1990), 35–39.

[13] R.C. Tausworthe, " Random numbers generated by linear recurrence modulo two", *Math. Comp.*, **19** (1965), 201–209.

[14] S. Tezuka, "Walsh-spectral test for GFSR pseudorandom number generators", *Comm. ACM.*, **30** (1987), 731–735.

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

[16] S. Tezuka, "Analysis of L'Ecuyer's combined random number generator", *IBM TRL Technical Note*, RT-5014, 1989.

[17] J.P.R. Tootill, W.D. Robinson, and A.G. Adams, " The runs up-and-down performance of Tausworthe pseudo-random number generators", *J. ACM*, **18** (1971), 381–399.

[18] B.A. Wichmann and I.D. Hill, " An efficient and portable pseudorandom number generator", *Appl. Stat.*, **31** (1982), 188–190.

# Appendix A: Application of Marsaglia's lattice test to a Tausworthe sequence

As an example, consider the Tausworthe sequence used for Figure 1. It goes as follows: 32, 8, 6, 34, 41, 30, 40, 14, 36, 11, 55, 54, 38, .... According to Marsaglia's algorithm [11], we have $\alpha_1 = (32, 8), \alpha_2 = (6, 34), \alpha_3 = (41, 30), \alpha_4 = (40, 14), \alpha_5 = (36, 11), \alpha_6 = (55, 54)$. Then

$$D_1 = \begin{vmatrix} \alpha_2 - \alpha_1 \\ \alpha_3 - \alpha_1 \end{vmatrix} = \begin{vmatrix} -26 & 26 \\ 9 & 22 \end{vmatrix} = 26 \times 31 = 2 \times 13 \times 31 = 806,$$

$$D_2 = \begin{vmatrix} \alpha_5 - \alpha_4 \\ \alpha_6 - \alpha_4 \end{vmatrix} = \begin{vmatrix} -4 & -3 \\ 15 & 40 \end{vmatrix} = |(-160 + 45)| = 5 \times 23 = 115,$$

and $gcd(D_1, D_2) = 1$. Thus, the pairs of successive values from that sequence can generate all two-dimensional integer points modulo 64, by their integral linear combinations.

# Appendix B: A sample implementation in C.

In Figure 2, we give an implementation of a combination of two Tausworthe generators, in the programming language C. Each component is implemented according to the algorithm of Section 2.1. The variables `Q1`, `S1`, `P1mS1`, and `Mask1` [`Q2`, `S2`, `P2mS2`, and `Mask2`] hold the respective values of $q$, $s$, $p-s$, and $2^p-1$ for generator 1 [generator 2] in the combination, while `P1mP2` is the difference between the two values of $p$. At each step, the binary representation of `I1` corresponds to the "state" of the first component, and similarly for `I2` and the second component. Before calling the generator, `I1` and `I2` must be initialized to integers such that $0 <$ `I1` $\leq$ `Mask1` and $0 <$ `I2` $\leq$ `Mask2`. After initialization, each call to `CombTaus` produces the next value in the sequence $\{u_j\}$ of the combined generator. This code works on a 32-bit machine, i.e. if `int` represents a 32-bit integer. With some compilers (e.g., with most compilers for PC/DOS), `int` must be replaced by `long` to get 32-bit integers. The operators "`<<`", "`^`", and "`&`" in the code represent respectively left shift, bitwise XOR, and bitwise AND. The latter is used with a bit "mask" to make sure that only the appropriate number of bits is retained (see Step 0 of the algorithm in Section 2.1). At the end, `I2` is shifted for left alignment with `I1` before the bitwise XOR combination. The resulting "integer" is then "normalized" to a number between 0 and 1.

With our compiler on the PC, this code runs slightly faster (149 seconds instead of 154 for one million numbers) when all the variables, except `I1`, `I2`, and `b`, are replaced by their numerical values directly into the code. Here, we kept the variables mainly to ease code understanding, facilitate the coding of the other suggested combined generators for those who want to do it, and facilitate translation into another language.

```
int Q1=13, Q2=2, S1=12, S2=17, P1mS1=19, P2mS2=12, P1mP2=2;
unsigned int I1, I2, b, Mask1=2147483647, Mask2=536870911;
double Norm=4.656612873e-10;

double CombTaus ()
   {                        /* Generate numbers between 0 and 1.   */
   b = ((I1 << Q1) ^ I1) & Mask1;
   I1 = ((I1 << S1) ^ (b >> P1mS1)) & Mask1;
   b = ((I2 << Q2) ^ I2) & Mask2;
   I2 = ((I2 << S2) ^ (b >> P2mS2)) & Mask2;
   return ((I1 ^ (I2 << P1mP2)) * Norm);
   }
```

Figure 2: A sample C program for the combined Tausworthe generator in Table 1.