Journal of the Operations Research
Society of Japan
Vol. 37, No. 3, September 1994

A UNIFIED VIEW OF LONG-PERIOD RANDOM NUMBER GENERATORS

Shu Tezuka
IBM Research, Tokyo Research Laboratory

(Received February 15, 1993)

Abstract Two types of linear congruential random number generator are considered: the conventional one using integer arithmetic and another using polynomial arithmetic over finite fields. We show that most of the long-period random number generators currently used or recently proposed, which include multiple recursive generators, shift register generators, add-with-carry and subtract-with-borrow generators, Twisted-GFSR generators, Wichmann-Hill generators, and combined Tausworthe generators, can be viewed as producing truncated linear congruential sequences with large moduli in terms of integer or polynomial arithmetic. On this basis, we compare the above generators with respect to generation efficiency, lattice structure, and portability.

1 Introduction

With the great increases in the speed and power of microprocessors, Monte Carlo techniques for simulation and optimization problems have attracted increasing interests and assumed greater importance on microcomputers and workstations as well as on mainframe computers and supercomputers. Accordingly, reliable random number generators with long period lengths have received much attention from people working on large-scale Monte Carlo calculations. Generators with long periods that are currently in common use are Generalized Feedback Shift Register (GFSR), additive, and subtractive generators. Some people call these three types lagged Fibonacci generators with XOR, +, and −, respectively, where XOR is the bitwise exclusive-or operation. The structure and distribution properties of the first type in high dimensions have recently become well understood [3,10,14,15,17,18,19,20]. The last two are recommended by Knuth [4], who says, however, that very little has yet been proved about the randomness properties of these generators. They were also investigated by Marsaglia [10]. Besides these generators, *multiple recursive generators*, which are extended versions of linear congruential generators, are also popular [5]. From the viewpoint of efficiency and portability, Wichmann-Hill combined generators have also drawn considerable attention from practical users [6,25].

Recently, three types of long period random number generators have been proposed. The first, proposed by Marsaglia and Zaman [11], consists of *add-with-carry* and *subtract-with-borrow* generators, which can be viewed as improvements of the additive and subtractive generators in terms of memory efficiency; that is, their sequences have almost the maximum possible period of 2^{kw} using k w-bit words as an array of memory for the recurrence relation of order k, whereas the periods of the additive and subtractive generators using the same size of memory are far short of the maximum possible periods. An important consequence of this property is that the k-tuples of the consecutive terms of their sequences are almost equidistributed; that is, the sequences are provably uniform in high dimensions up to k. Another important property is that the sequences can be generated almost as fast as the additive and subtractive methods. Matsumoto and Kurita

[12] recently proposed a second type; called *Twisted GFSR* generators, which also have the same property as Marsaglia and Zaman's, and can be viewed as an improvement of GFSR generators. In this case the period attained is exactly $2^{kw} - 1$ by using k w-bit words; that is, the sequences are provably uniform in k dimensions. In addition, these generators can be implemented in such a way that they produce their sequences almost as quickly as the original GFSR sequences. Generators of the third type, called *combined Tausworthe* generators [2,21], use XOR combinations of simple Tausworthe sequences with different period lengths. From a theoretical viewpoint, these can be viewed as analogous with the Wichmann-Hill generators in terms of polynomial arithmetic modulo two.

The purpose of this paper is to give a unified framework for long-period random number generators, including the three new ones above as well as conventional ones such as multiple recursive, Wichmann-Hill, and GFSR generators. This unified view provides a theoretical basis for comparisons of most long-period generators commonly used with respect to generation efficiency, lattice structure, and portability. Section 2 describes most of the long-period random number generators currently used or recently proposed. In Section 3, we overview recent results [6,22] that add-with-carry and subtract-with-borrow generators and Wichmann-Hill combined generators can be formulated as the truncated version of linear congruential sequences with large integer moduli. In Section 4, we first define linear congruential sequences in terms of polynomial arithmetic over finite fields, and then show that multiple recursive generators, shift register generators, Twisted-GFSR generators, and combined Tausworthe generators can be classified as truncated versions of this class of linear congruential sequences. Section 5 compares the generators in terms of efficiency, lattice structure, and portability. Finally, we discuss future problems.

2 Descriptions of Long-Period Generators

In this section, we briefly describe most of the long-period random number generators that are widely used or have been recently proposed.

2.1 Multiple recursive generators

The multiple recursive generator modulo M, an integer, is written as follows: for $i = r, ...,$

$$x_i = a_{r-1}x_{i-1} + \cdots + a_0 x_{i-r} \pmod{M},$$

where the initial values $x_0, ..., x_{r-1}$ are not all-zero, and $a_0, ..., a_{r-1}$ are chosen so that the period length of x_i becomes large. Here the modulus M is usually a prime and is about the size of the computer word length, typically $M \approx 2^{32}$. Moreover, $a_i, i = 0, ..., r-1$ are chosen to satisfy $a_i(M \pmod{a_i}) < M$, so multiplication overflow can be avoided in software implementation [5]. If M is prime and the characteristic polynomial of $\{x_i\}$ is primitive over GF(M), it is obvious that the point set $(x_i, x_{i+1}, ..., x_{i+k-1}), i = 1, 2, ...$ is equidistributed when $k \leq r$. In k dimensions, where $k > r$, the point set is known to have a so-called lattice structure. The lattice basis has, for example, the following form when $k = r + 1$:

$$\begin{aligned} \mathbf{e}_1 &= \frac{1}{M}(1, 0, 0, \ldots, 0, a_0), \\ \mathbf{e}_2 &= \frac{1}{M}(0, 1, 0, \ldots, 0, a_1), \\ &\vdots \quad\quad \vdots \end{aligned}$$

$$\begin{aligned} \mathbf{e}_r &= \frac{1}{M}(0,0,0,\ldots,1,a_{r-1}), \\ \mathbf{e}_{r+1} &= (0,0,0,\ldots,0,1). \end{aligned}$$

The most preferable case for implementation is that in which the characteristic polynomial is a primitive trinomial $x^r + a_{r-s}x^s + a_0$, for $r > s$, modulo a prime M, namely,

$$x_i = a_{r-s}x_{i-r+s} + a_0 x_{i-r} \pmod{M},$$

because (i) the resulting period length is the maximum, $M^r - 1$, and (ii) the generation speed is fast, since one random number can be generated by two modulo-multiplications and one modulo-addition. As shown in [5], however, the lattice structure in dimensions higher than r is not good for those multiple recursive generators with primitive trinomials that are suitable for practical implementation.

Note that the *matrix generator* with prime moduli [14,15], which is suitable for implementation on a vector or parallel supercomputer, can be formulated as the parallel generation of a single multiple recursive sequence with different starting points provided that the period is maximum.

2.2 Shift register generators

There are two types of shift-register random number sequences: Tausworthe sequences and GFSR sequences. Let $M(z) = z^r + a_{r-1}z^{r-1} + \cdots + a_0$ be a primitive polynomial of degree r over $GF(2)$, and L be the "word-size." A binary sequence $x_i, i = r, r+1, \ldots$ follows the recurrence relation $x_i = a_{r-1}x_{i-1} + \cdots + a_0 x_{i-r} \pmod{2}$.

A Tausworthe sequence is thus defined as follows [16]:

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

where d is a constant with $0 < d < 2^r - 1$, $\gcd(d, 2^r - 1) = 1$. The algorithm below (see [21]) generates a Tausworthe sequence whose characteristic polynomial is a primitive trinomial of the form $M(z) = z^r + z^s + 1$, $s < r/2$, and $0 < d \leq r - s$. Each term of the sequence $\{u_i\}$ is expressed by its leading $L = r$ bits in this algorithm. This can be implemented easily, in a "portable" language that supports shifting and XOR operations (such as C), if r [=L] is not larger than the computer's word-size.

An algorithm for implementing (1) when $0 < d \leq r - s$**:**

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

A slight modification of the above can be adapted for generating a Tausworthe sequence whose characteristic polynomial is a primitive trinomial of the form $M(z) = z^r + z^s + 1$, $s < r/2$, and $d = r$ (Details are given in [1]).

A GFSR sequence is originally defined as follows [8]:

$$u_i = \sum_{j=1}^{L} x_{dj+i} 2^{-j}. \tag{2}$$

Note that $L \leq r$ (otherwise, a linear dependence relation appears between the column bits of u_i). Lewis and Payne suggested that d should be greater than $100r$. In addition, they employed a primitive trinomial as the characteristic polynomial of $\{x_i\}$ in order to realize a fast generation scheme for the sequence in the following way: Let $M(z) = z^r + z^s + 1$. The sequence can then be generated by the scheme

$$u_i = u_{i-r+s} \text{ XOR } u_{i-r}. \tag{3}$$

The lagged Fibonacci with XOR is a more general version of the sequences in (2) with primitive characteristic trinomials,

$$u_i = \sum_{l=1}^{L} x_{j_l+i} 2^{-l},$$

where $j_l, l = 1, ..., L$, are integers between 0 and $2^r - 1$. However, the recurrence relation (3) can still be exploited in this case. Today, people call this general version with any primitive characteristic polynomials a GFSR sequence [3,15,17].

The matrix representation of shift register sequences is very useful. Let C be the companion matrix of the polynomial $M(z) = z^r + a_{r-1}z^{r-1} + \cdots + a_1 z + a_0$, namely,

$$C = \begin{pmatrix} 0 & 0 & 0 & \cdots & a_0 \\ 1 & 0 & 0 & \cdots & a_1 \\ & \cdots & \cdots & \cdots & \\ & \cdots & \cdots & \cdots & \\ 0 & \cdots & 1 & 0 & a_{r-2} \\ 0 & 0 & \cdots & 1 & a_{r-1} \end{pmatrix},$$

and let $\alpha = (x_0, ..., x_{r-1})$ be a non-zero binary vector. The general GFSR is then written as follows:

$$\alpha G, \alpha CG, ..., \alpha C^i G, ...,$$

where G is an $r \times L$ matrix over $GF(2)$ whose l-th column vector, denoted by $G_l, l = 1, ..., L$, is uniquely determined by the equations,

$$x_{j_l+i} =< G_l, \alpha C^{i-1} >, \text{ for } i = 1, ..., r.$$

Here $< \alpha, \beta >$ means the inner-product of the binary vectors α and β over $GF(2)$.

2.3 Add-with-carry and subtract-with-borrow generators

Additive and *subtractive* generators are obtained by simply replacing the XOR operation of GFSR generators in (3) with the + or − operation. Marsaglia and Zaman [11] recently proposed new types of random number generators, *add-with-carry* and *subtract-with-borrow*, which can be viewed as improved variants of the additive and subtractive methods, defined as follows: the former is

$$x_i = x_{i-s} + x_{i-r} + c_i \pmod b.$$

where $r > s$ are lags, and $c_{i+1} = 0$ if $x_{i-s} + x_{i-r} + c_i < b$, or $c_{i+1} = 1$ otherwise. The latter is

$$x_i = x_{i-s} - x_{i-r} - c_i \pmod{b},$$

where $r > s$ are lags, and $c_{i+1} = 1$ if $x_{i-s} - x_{i-r} - c_i < 0$, or $c_{i+1} = 0$ otherwise.

The merit of this variant is that the maximum possible period becomes very large, namely, $b^r + b^s - 2$ for the add-with-carry (or $b^r - b^s$ for the subtract-with-borrow), whereas that of the additive and subtractive methods is about $b2^r$. In order to achieve such large periods, the parameters (r, s, b) should be chosen so that $b^r + b^s - 1$ is prime for add-with-carry (or $b^r - b^s + 1$ is prime for subtract-with-borrow) and b is a primitive root modulo $b^r + b^s - 1$ (resp. $b^r - b^s + 1$). However, since checking the primitivity with very large prime moduli is a difficult task in practice, Marsaglia and Zaman listed the parameters for their generators such that the order of b modulo $b^r + b^s - 1$ (resp. $b^r - b^s + 1$) is very large or near to the maximum. Usually, b is taken as large enough, namely, around 2^{32}, so the random number u_i in [0,1) is constructed as

$$u_i = x_i / b. \tag{4}$$

Their paper gives a list of several good parameters (r, s, b) for producing very large period sequences, based on selection through the so-called stringent statistical test. Among them, a component of the combined generator RANMAR, which is now very popular among computational physics researchers, is defined by $(r, s, b) = (43, 22, 2^{32} - 5)$. Its period is known to be maximum, i.e., $b^r - b^s \approx 2^{1376}$.

2.4 Twisted GFSR generators

Twisted GFSR generators were recently defined by Matsumoto and Kurita [12]:

$$\mathbf{v}_i = \mathbf{v}_{i-r+s} \text{ XOR } \mathbf{v}_{i-r}A, \text{ for } i = r, r+1, ...,$$

where $\mathbf{v}_i$ is a sequence of vectors in $GF(2^w)$, A is a $w \times w$ matrix over GF(2), and $r > s$. In particular, they analyzed the following special case, which is very useful for quick generation of the sequences:

$$\mathbf{v}_i = \mathbf{v}_{i-r+s} \text{ XOR } \mathbf{v}_{i-r}C^T, \tag{5}$$

where $\mathbf{v}_i$ is a sequence of vectors in $GF(2^w)$, C is a $w \times w$ companion matrix, and $r > s$. In this paper, we call this special case as Twisted GFSR with companion matrices. The conversion from a binary vector $(v_0, ..., v_{w-1})$ into a random number between 0 and 1 is as follows:

$$u = \frac{1}{2^w} \sum_{i=0}^{w-1} v_i 2^i.$$

The parameters r, s, and C are chosen so that the maximum period of the sequence v_i becomes $2^{wr} - 1$. In their paper, some parameters of Twisted GFSR with maximum periods are listed.

One of the advantages of this scheme is that the size r of an array is the minimum necessary to produce the period length $2^{wr} - 1$ with respect to the word-size w. In this sense, the scheme can be viewed as an improved version of the lagged Fibonacci scheme with XOR, because the latter produces a period $2^r - 1$ of a sequence with an identically sized array of r w-bit words. Another important merit is the fast generation algorithm. Let $\mathbf{v}_i = (v_{i,0}, ..., v_{i,w-1})$ and $\mathbf{a} = (a_0, ..., a_{w-1})$. The algorithm is then:

$$\text{if } v_{i-r,w-1} = 0 \text{ then } \mathbf{v}_i = \mathbf{v}_{i-r+s} \text{ XOR SR}(\mathbf{v}_{i-r})$$
$$\text{else } \mathbf{v}_i = \mathbf{v}_{i-r+s} \text{ XOR SR}(\mathbf{v}_{i-r}) \text{ XOR } \mathbf{a},$$

where SR is the one-bit right-shift operation.

2.5 Combined generators

In general, any combination of any random number generators can be called a *combined* generator. In this section, however, we restrict ourselves to the so-called *Wichmann-Hill* generators [25] and the recently proposed *combined Tausworthe* generators [2,21].

The Wichmann-Hill generator is defined as

$$U_i = X_i^{(1)}/m_1 + \cdots + X_i^{(J)}/m_J \pmod 1,$$

where $X_{i+1}^{(j)} = a_j X_i^{(j)} \pmod{m_j}$, for $j = 1, ..., J$, are linear congruential sequences with distinct primes moduli $m_j, j = 1, ..., J$. The period length of the sequence U_i becomes $(m_1 - 1) \times \cdots \times (m_J - 1)$ provided that the initial values $X_0^{(j)}$ are properly chosen.

Combined Tausworthe sequences are defined as follows:

$$U_i = u_i^{(1)} \text{ XOR } \cdots \text{ XOR } u_i^{(J)},$$

where for $j = 1, ..., J$, each sequence $u_i^{(j)}, i = 1, 2, ...$ is a Tausworthe sequence, and the characteristic polynomials of the sequences are all of distinct degree, so that the period of the sequence U_i becomes the least common multiple of the periods of $u_i^{(j)}, j = 1, ..., J$. By using the algorithm in Section 2.2 for each Tausworthe sequence, we can construct combined Tausworthe sequences quickly. The parameters of each component Tausworthe sequence for appropriate combinations are given in [21].

3 Overview of Recent Results

In this section, we overview recent results on Wichmann-Hill generators and add-with-carry and subtract-with-borrow generators. First, we consider linear congruential sequences with large moduli:

$$X_i = aX_{i-1} \pmod M.$$

A random number u_i in [0,1) is obtained through normalization by M as a fractional expansion with base b:

$$u_i = X_i/M = x_1^{(i)} b^{-1} + x_2^{(i)} b^{-2} + ...$$

where M is large and usually $b = 2$ on binary computers. In such a situation, we use the truncated version of u_i. Let $\tilde{u}_i$ be the truncation of u_i; that is, for some L,

$$\tilde{u}_i = x_1^{(i)} b^{-1} + x_2^{(i)} b^{-2} + ... + x_L^{(i)} b^{-L}.$$

If we form the k-dimensional vectors as

$$P_{k,i} = (u_i, ..., u_{i+k-1}),$$

and

$$\tilde{P}_{k,i} = (\tilde{u}_i, ..., \tilde{u}_{i+k-1}),$$

then the Euclidean distance between $P_{k,i}$ and $\tilde{P}_{k,i}$ is bounded by $b^{-L}k^{0.5}$, which is typically around 2^{-30}. Since it is well known that the points $P_{k,i}, i = 1, 2, ...$, form a lattice structure, in high dimensions $b^{-L}k^{0.5}$ could be small compared with the distance between hyperplanes. In that case, the hyperplane structure of $P_{k,i}, i = 1, 2, ...$, could give a good approximation of the behavior of the points $\tilde{P}_{k,i}, i = 1, 2, ...$

3.1 The modulus is a prime

A straightforward implementation of linear congruential sequences with large moduli is time-consuming because it involves multiple-precision modulo arithmetic. Recently, however, the following result was obtained [22]:

Proposition 1 *Let $M = b^r + b^s - 1$ (or $b^r - b^s + 1$) for add-with-carry (or subtract-with-borrow), and let b^* be the inverse of b modulo M, that is, $b^*b = 1 \pmod{M}$. Let $x_i, i = 1, 2, ...,$ be add-with-carry (or subtract-with-borrow) sequences. Then the sequence defined in (4) can be viewed as a truncated version of a linear congruential sequence $u_i = X_i/M, i = r, r+1, ...,$ with an appropriate X_0, where $X_i = b^* X_{i-1} \pmod{M}$.*

In other words, Marsaglia and Zaman's idea can be regarded as a breakthrough leading to the fast implementation of linear congruential sequences with large prime moduli.

For the case of (4), namely, $L = 1$, the truncated $\tilde{u}_i$ is given as

$$\tilde{u}_i = x_i/b.$$

We have

$$0 \le u_i - \tilde{u}_i = \sum_{j=2}^{\infty} x_{i-j+1} b^{-j} \le b^{-1}.$$

In other words, the distance between $\tilde{u}_i$ and u_i is less than the "resolution" of the add-with-carry generators. Therefore, the sequence $\tilde{u}_i = x_i/b, i = 1, 2, ...,$ from (4) is almost equivalent to $u_i, i = 1, 2, ...,$ which is a a linear congruential sequence from $X_i = b^* X_{i-1} \pmod{M}$, where $bb^* = 1 \pmod{M}$.

Finally, we should notice that the lattice structure of all these generators is always bad in dimensions higher than r. This is shown in the following way: without loss of generality, we consider only add-with-carry generators. From the definition of generators, the equations of hyperplanes on which all the points $(u_i, u_{i+1}, ...u_{i+r}), i = 1, 2, ...,$ lie are given by

$$u_i - u_{i-s} - u_{i-r} = j \text{ or } j + 1/b$$

for $j = -2, -1, 0, 1$. Since b is large enough, $1/b$ is negligible. So, for each j, a twin of hyperplanes corresponding to j and $j + 1/b$ can be approximated as one hyperplane. The distance between neighboring approximate hyperplanes is calculated from the coefficient of LHS, i.e., $1/\sqrt{1^2 + (-1)^2 + (-1)^2} = 1/\sqrt{3} = .577...$ This value means that the lattice structure is far from uniform distribution.

3.2 The modulus is a composite

We now consider the case in which M is a large composite integer. Here, we can use the Chinese Remainder theorem for large modulo arithmetic. Let $M = p_1^{e_1} \cdots p_J^{e_J}$. Then the calculation

$$X_i = aX_{i-1} \pmod{M}$$

can, for $j = 1, ..., J$, be reduced to

$$X_i^{(j)} = aX_{i-1}^{(j)} \pmod{p_j^{e_j}}.$$

If each recurrence for $j = 1, ..., J$ can be calculated quickly by an ordinary binary computer, X_i can also be obtained by using the Chinese Remainder theorem in the following way: Denote $m_j = p_j^{e_j}$. Define $M_{-j} = M/m_j$ and let n_j be an integer such that

$n_j M_{-j} \bmod m_j = 1$. Then X_i can be constructed as $X_i = \sum_{j=1}^{J} X_i^{(j)} n_j M_{-j} \pmod M$, that is,

$$U_i = \frac{X_i}{M} = \sum_{j=1}^{J} \frac{n_j X_i^{(j)}}{m_j} \pmod 1.$$

Unfortunately, this direct implementation of the Chinese Remainder approach looks rather time-consuming, and thus is not suitable for fast generation of random numbers.

The breakthrough came from rather a surprising direction. In 1986, Zeisel, followed by L'Ecuyer and Tezuka [6], found the following fact:

Proposition 2 *For each j in $\{1, \ldots, J\}$, define $M_{-j} = M/m_j$ and let n_j be an integer such that $n_j M_{-j} \bmod m_j = 1$. Let $a = \sum_{j=1}^{J} a_j n_j M_{-j}$. Then the Wichmann-Hill sequence*

$$U_i = \sum_{j=1}^{J} \frac{X_i^{(j)}}{m_j} = \frac{\bar{X}_i}{M} \pmod 1$$

where $\bar{X}_i = \sum_{j=1}^{J} X_i^{(j)} M_{-j}$, is formulated as a linear congruential sequence; that is, $\bar{X}_i$ satisfies $\bar{X}_i = a\bar{X}_{i-1} \pmod M$.

This is a special case in which M is square-free. It can easily be extended to the general M by letting $m_j = p_j^{e_j}$. In this case, the maximum possible period becomes the least common multiple of $\lambda(m_j), j = 1, ..., J$, where $\lambda(2^e) = 2^{e-2}$ if $e > 2$, or 2^{e-1} otherwise, and $\lambda(p^e) = (p-1)p^{e-1}$ for an odd prime p. This extended version also can be implemented in the same way as for the original Wichmann-Hill sequence.

4 Linear Congruential Sequences with Respect to Polynomial Arithmetic over Finite Fields

Here, we give a definition of linear congruential sequences with respect to polynomial arithmetic over $GF(q)$. Let $GF\{q, z\}$ denote the field of all Laurent series of the form $S(z) = \sum_{j=-\infty}^{m} c_j z^j$,where m is an integer, $c_j \in GF(q)$, and q is a power of prime p, i.e., $q = p^w$ (w is a positive integer). Here we define an analogous version of linear congruential sequences in $GF\{q, z\}$:

$$\begin{aligned} f_i(z) &= (g(z) f_{i-1}(z) + h(z)) \bmod M(z) \\ u_i(z) &= f_i(z)/M(z) = y_1^{(i)} z^{-1} + y_2^{(i)} z^{-2} + ..., \end{aligned} \tag{6}$$

where $g(z), h(z), M(z)$, and $f_i(z)$ are polynomials in $GF\{q, z\}$. We denote this generator by the triplet $G = (g, h, M)$ and call it an $LS(q)$ generator.

Consider the k-tuples $(f_i(z)/M(z), ..., f_{i+k-1}(z)/M(z))$, $i = 1, 2, \ldots$, produced by (6). These are expressed by the grid (shifted lattice) $\mathcal{L}_k + \lambda$, where $\mathcal{L}_k$ is a lattice with the basis

$$\begin{aligned} \mathbf{e}_1 &= \frac{1}{M(z)}(1, g(z), g^2(z), \ldots, g^{k-1}(z)), \\ \mathbf{e}_2 &= (0, 1, 0, \ldots, 0), \\ &\vdots \quad\quad \vdots \\ \mathbf{e}_k &= (0, 0, 0, \ldots, 1), \end{aligned}$$

and $\lambda = (0, 1, 1 + g(z), \ldots, 1 + g(z) + \cdots + g^{k-2}(z))(h(z)/M(z))$. Hereafter, we call $\mathcal{L}_k$ the associated lattice to $G = (g, h, M)$.

Define the norm of a vector $\alpha = (a_1(z), ..., a_k(z))$, where each $a_i(z)$ is in $GF\{q, z\}$, as

$$|\alpha| = \max_{1 \le i \le k} \deg(a_i).$$

In this paper, the notions of reduced basis and successive minima of a lattice $\mathcal{L}_k$ in a vector space over $GF\{q, z\}$ follow Lenstra's definitions [7]:

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

Note that Lenstra gave an efficient algorithm that calculates the reduced basis in $GF\{q, z\}$ [7], and showed that it is much faster than the approach based on Gaussian elimination.

4.1 q is a prime

Let q be a prime p. Then we define σ as a mapping from $GF\{p, z\}$ to the real field as follows:

$$\sigma^{(p)}(S(z)) = S(p) = \sum_{j=-\infty}^{m} c_j p^j.$$

We use $\sigma_L^{(p)}$, the truncated version of $\sigma^{(p)}$:

$$\sigma_L^{(p)}(S(z)) = \sum_{j=-L}^{m} c_j p^j.$$

Hereafter, we denote the truncated $LS(q)$ sequences by $LS_L(q)$ sequences. In practical situations, L depends on the size of p such that $p^L \approx 2^{32}$. In the following discussions, we consider two cases: (i) p is small, i.e., $L > 1$ and (ii) p is large enough, i.e, $L = 1$. First, we deal with the former.

4.2 q is a small prime

For simplicity, we consider the case in which $q = p = 2$. Recently [20], Tausworthe sequences have been shown as a special case of the above general class.

Proposition 3 *Tausworthe sequences can be formulated as $LS_L(2)$ sequences from $G = (g, h, M)$ such that $M(z) = z^r + a_{r-1}z^{r-1} + \cdots + a_0$, $g(z)$ is primitive modulo $M(z)$, $h(z) \equiv 0$, and L is the "word-size," where $M(z)$ is a characteristic polynomial for a Tausworthe sequence, i.e, a primitive polynomial over $GF(2)$.*

Next, we consider a similar formulation for the original GFSR sequences defined in (2). The result implies that the original GFSR sequences contain Tausworthe sequences as a proper subclass.

Proposition 4 *The original GFSR sequences in (2) can be formulated as $LS_L(2)$ sequences from $G = (g, h, M)$ such that $M(z)$ is an irreducible polynomial of degree r over $GF(2)$, $g(z)$ is primitive modulo $M(z)$, $h(z) \equiv 0$, and L is the "word-size."*

Proof: Let $F(z)$ be the characteristic polynomial of $x_i, i = 1, 2, ...$, and let $M(z)$ be that of the decimated sequence $x_{di}, i = 1, 2, ...$, with $\deg(M) = \deg(F)$. Note that $F(z)$ is a primitive polynomial, and $M(z)$ is irreducible. Let α be such that $F(\alpha) = 0$, then

$x_i = Tr(\gamma\alpha^i)$ for some γ, where the trace $Tr(\alpha)$ is defined as $\alpha + \alpha^2 + \cdots + \alpha^{2^{r-1}}$ in $GF(2^r)$. Note that $M(\alpha^d) = 0$. Since $M(z)$ is irreducible, α can be written as a linear combination of $1, \alpha^d, \alpha^{2d}, ..., \alpha^{(r-1)d}$, that is,

$$\alpha = \sum_{i=0}^{r-1} c_i \alpha^{di},$$

Thus $x_i = Tr(\gamma\alpha^i) = Tr(\gamma\alpha^{i-1}\alpha) = \sum_{j=0}^{r-1} c_j Tr(\gamma\alpha^{i-1}\alpha^{dj}) = \sum_{j=0}^{r-1} c_j x_{dj+i-1}$. By putting $g(z) = c_{r-1}z^{r-1} + \cdots + c_1 z + c_0$, we complete the proof.

Q.E.D.

Note that, in general, GFSR sequences cannot be written as $LS_L(2)$ sequences.

Following Tezuka and L'Ecuyer [21], Couture et al. [2] obtained a theorem that links the k-dimensional distribution of $LS_L(2)$ sequences with the successive minima and reduced bases of a lattice in a vector space over $GF\{2, z\}$. Before going into details, we need to explain how to measure the k-dimensional uniformity of the sequences. An *equidissection* of the k-dimensional unit hypercube into $2^{k\ell}$ cubic cells is defined as the set of all cubic cells in $[0,1)^k$ whose sides have a length of $2^{-\ell}$ and whose corners have coordinates that are all multiples of $2^{-\ell}$. That is,

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

Thus we have the following theorem [2]:

Theorem 1 *For $1 \le i \le k$, let l_i be the i-th successive minimum of the lattice associated to $G = (g, 0, M)$, where $M(z)$ is an irreducible polynomial of degree r over $GF(2)$ and $g(z)$ is primitive modulo $M(z)$. Define the quantity $d(\ell)$ as follows:*

$$d(\ell) = \sum_{i=1}^{k} (-\ell - l_i)^+,$$

where $(t)^+ = t$ if $t > 0$, or 0 otherwise. Then we have Table 1, which gives the number of cells denoted by $\#_\ell(n)$ for all values of n for which it could be non-zero, where n is the number of lattice points in a cell.

Table 1: Values of $\#_\ell(n)$ that could be non-zero for a given resolution ℓ

n	$\#_\ell(n)$
$2^{d(\ell)}$	$2^{r-d(\ell)} - 1$
$2^{d(\ell)} - 1$	1
0	$2^{\ell k} - 2^{r-d(\ell)}$

This theorem means that among $2^{\ell k}$ cells, $2^{r-d(\ell)} - 1$ cells each contains $2^{d(\ell)}$ lattice points, one cell contains $2^{d(\ell)} - 1$ points, and all the others are empty cells.

According to Tezuka [17,20], a pseudorandom sequence having a lattice structure in the vector space over $GF\{2, z\}$ is said to be k-distributed with ℓ-bit resolution when each cell of $S_k(\ell)$ contains the same number of lattice points. As shown in [2], we have the following corollary:

Corollary 1 *An $LS_L(2)$ sequence from $G = (g, 0, M)$ is k-distributed with ℓ-bit resolution if and only if $l_k \leq -\ell$, where $M(z)$ is irreducible over $GF(2)$ and $g(z)$ is primitive modulo $M(z)$.*

We shall comment on this result in connection with the spectral test. Mahler's theorem says that the i-th successive minimum of the prime lattice is equal to the $(k-i)$-th successive minimum of the dual lattice. Hence, the norm of the shortest vector of the dual lattice is exactly equivalent to the resolution of the sequences. This result corresponds to the conventional spectral test for linear congruential sequences, where the length of the shortest vector of the dual lattice is equivalent to the reciprocal of the maximum distance of the hyperplanes in which the points fall.

Next we consider combined Tausworthe sequences, which are analogous to Wichmann-Hill combined generators with respect to polynomial arithmetic. Let $m_j(z)$, $1 \leq j \leq J$ be primitive polynomials. For each j in $\{1, \ldots, J\}$, define $M_{-j}(z) = M(z)/m_j(z)$ and let $n_j(z)$ be a polynomial such that $n_j(z)M_{-j}(z) \bmod m_j(z) = 1$. The following result [21] characterizes the combined sequence:

Proposition 5 *Combined Tausworthe sequences can be formulated as $LS_L(2)$ sequences from $G = (g, h, M)$ such that $M(z) = m_1(z) \cdots m_J(z)$, $g(z) = \sum_{j=1}^{J} g_j(z) n_j(z) M_{-j}(z)$, $h(z) \equiv 0$, and L is the "word-size."*

The precise description of k-dimensional distribution of combined Tausworthe sequences can be obtained in terms of the successive minima of their associated lattice [2,21]. The discrepancy of simple Tausworthe sequences and general GFSR sequences has been analyzed in the last decade (see [14,15,18,19,23]).

4.3 q is a large prime

Let $x_i = y_1^{(i)}$ in (6). If $g(z) \equiv z$ and $h(z) \equiv 0$, then $x_{i+1} = y_1^{(i+1)} = y_2^{(i)}$. In this case, a random number is truncated as follows:

$$\tilde{u}_i = x_i / p.$$

Here, we should note that x_i satisfies the recurrence relation

$$x_i = a_{r-1} x_{i-1} + \cdots + a_0 x_{i-r} \pmod{p},$$

where $M(z) = z^r - a_{r-1} z^{r-1} - \cdots - a_0$. If $M(z)$ is a primitive polynomial over $GF(p)$, then the sequence x_i (or $\tilde{u}_i$), $i = 1, 2, \ldots$, has a period of $p^r - 1$. Hence we have

Proposition 6 *Multiple recursive sequences can be formulated as $LS_L(p)$ sequences from $G = (g, h, M)$, where p is a large prime, $g(z) \equiv z$, $h(z) \equiv 0$, $M(z)$ is primitive over $GF(p)$, and $L = 1$.*

In this case, as we noticed before, the exact (not approximate) integer lattice structure of the sequences can be studied in higher than r dimensions.

4.4 q is a prime power

Now, we consider an analogous version of linear congruential sequences in $GF\{q, z\}$, where q is a prime power. For reasons of simplicity and practical importance, we henceforth restrict ourselves to the case in which $q = 2^w$, where $w \approx 32$.

When q is large enough, the truncated linear congruential sequence in $GF\{q,z\}$ is given as follows: Let the truncated sequence $\tilde{u}_i(z), i = 1,2,...,$ be defined as

$$\tilde{u}_i(z) = y_1^{(i)} z^{-1},$$

where $y_1^{(i)}$ is the coefficient of z^{-1} in the formal Laurent series $f_i(z)/M(z)$ in (6). Consider the case in which $M(z) = z^r - m_{r-1}z^{r-1} - \cdots - m_0$, $g(z) \equiv z$, and $h(z) \equiv 0$. If we use the polynomial representation of $GF(2^w)$ based on an irreducible polynomial $Q(t) = t^w + a_{w-1}t^{w-1} + \cdots + a_0$, writing $y_i(t)$ for $y_1^{(i)}$, and $m_j(t)$ for m_j, $0 \le j < r$, then for the same reason as in the case of multiple recursive generators, the recurrence relation of $y_i(t)$ can be written as

$$y_i(t) = m_{r-1}(t)y_{i-1}(t) + \cdots + m_0(t)y_{i-r}(t) \pmod{Q(t)}. \tag{7}$$

Note that if $M(z)$ is primitive over $GF(2^w)$ the period of $y_i(t)$ becomes $2^{wr} - 1$. Since Theorem 2 in Couture et al. [2] is valid for any lattice in the vector space over $GF\{2,z\}$, we have

Theorem 2 *For multiple recursive generators in (7) with maximum period $2^{wr} - 1$, their point distribution is also given in Table 1 with r replaced by wr, where, for $1 \le i \le k$, l_i is the i-th successive minimum of the associated lattice.*

Let $y_i(t) = \sum_{j=0}^{w-1} v_{i,j}t^j$, where $\mathbf{v}_i = (v_{i,0},...,v_{i,w-1})$ is defined in (5). Since the multiplication by t modulo $Q(t)$ is equivalent to the multiplication by C^T in (5), where C is the companion matrix corresponding to $Q(t)$, we obtain

Proposition 7 *Twisted GFSR sequences with companion matrices can be formulated as $LS_L(q)$ sequences from $G = (g,h,M)$, where $q = 2^w$, $g(z) \equiv z$, $h(z) \equiv 0$, $M(z)$ is primitive over $GF(2^w)$, and $L = 1$, in other words, as multiple recursive sequences with respect to polynomial arithmetic modulo two, which is of the form*

$$\begin{aligned} y_i(t) &= y_{i-s}(t) + ty_{i-r}(t) \pmod{Q(t)}, \\ u_i(t) &= y_i(t)/t^w. \end{aligned}$$

In Table 2, we give the lattice structure of one of the Twisted GFSRs proposed in [12], where $Q(t) = t^{31} + t^{29} + t^{28} + t^{26} + t^{25} + t^{24} + t^{23} + t^{20} + t^{19} + t^{16} + t^{15} + t^{13} + t^{12} + t^{11} + t^{10} + t^8 + t^6 + t^5 + t^3 + t + 1, s = 2$, and $r = 13$. The period is $2^{403} - 1$, so equidistribution is guaranteed up to 13 dimensions. We investigated the lattice structure in dimensions 14 to 30. As shown in the table, the lattice structure is bad in all of these dimensions. For example, in $k = 14$ the number of empty cells, $2^{\ell k} - 2^{wr - d(\ell)}$, becomes $0, 2^{27}, 2^{42} - 2^{40}, ...,$ for $\ell = 1,2,3,4,....$ This means that even the most significant two bits of each term in the vectors $(\mathbf{v}_i,...,\mathbf{v}_{i+14}), i = 1,2,...,$ from this Twisted GFSR sequence are not equally distributed.

This phenomenon can be explained as follows: Let $\mathbf{v}_i = (v_{i,0},...,v_{i,w-1})$. In general, for any Twisted GFSR sequence there exists the following linear dependency relation:

$$v_{ij} = v_{i-s,j} + v_{i-r,j-1} + a_j v_{i-r,w-1} \pmod 2$$

for $1 \le j \le w-1$, and

$$v_{i0} = v_{i-s,0} + v_{i-r,w-1} \pmod 2,$$

for $j = 0$, since $Q(t)$ is an irreducible polynomial, i.e., $a_0 = 1$, where a_j is the coefficient of t^j in $Q(t)$. Note that the case of $j = w - 1$ means that there exists a linear dependency relation in the leading two bits of the consecutive $k(> r)$ vectors of $\mathbf{v}_i, i = 1, 2, \ldots$. This fully explains the results in Table 2. A more general result is found in [24], namely,

Theorem 3 *For any $k > r$, all Twisted GFSR sequences with companion matrices are k-distributed with at most two-bit resolution.*

Table 2: The successive minima in dimensions 14 to 30 for the Twisted GFSR with a period of $2^{403} - 1$

dim	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
$-l_1$	31	31	31	31	31	31	31	31	31	31	31	31	30	30	30	30	30
$-l_2$	31	31	31	31	31	31	31	31	31	31	31	30	30	30	30	30	30
$-l_3$	31	31	31	31	31	31	31	31	31	31	30	30	30	30	30	30	30
$-l_4$	31	31	31	31	31	31	31	31	31	30	30	30	30	30	30	30	30
$-l_5$	31	31	31	31	31	31	31	31	30	30	30	30	30	30	30	30	30
$-l_6$	31	31	31	31	31	31	31	30	30	30	30	30	30	30	30	30	30
$-l_7$	31	31	31	31	31	31	30	30	30	30	30	30	30	30	30	30	30
$-l_8$	31	31	31	31	31	30	30	30	30	30	30	30	30	30	30	30	30
$-l_9$	31	31	31	31	30	30	30	30	30	30	30	30	30	30	30	30	30
$-l_{10}$	31	31	31	30	30	30	30	30	30	30	30	30	30	30	30	30	29
$-l_{11}$	31	31	30	30	30	30	30	30	30	30	30	30	30	30	30	29	29
$-l_{12}$	31	30	30	30	30	30	30	30	30	30	30	30	30	30	29	29	29
$-l_{13}$	30	30	30	30	30	30	30	30	30	30	30	30	30	29	29	29	29
$-l_{14}$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$-l_{15}$	-	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$-l_{16}$	-	-	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$-l_{17}$	-	-	-	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$-l_{18}$	-	-	-	-	1	1	1	1	1	1	1	1	1	1	1	1	1
$-l_{19}$	-	-	-	-	-	1	1	1	1	1	1	1	1	1	1	1	1
$-l_{20}$	-	-	-	-	-	-	1	1	1	1	1	1	1	1	1	1	1
$-l_{21}$	-	-	-	-	-	-	-	1	1	1	1	1	1	1	1	1	1
$-l_{22}$	-	-	-	-	-	-	-	-	1	1	1	1	1	1	1	1	1
$-l_{23}$	-	-	-	-	-	-	-	-	-	1	1	1	1	1	1	1	1
$-l_{24}$	-	-	-	-	-	-	-	-	-	-	1	1	1	1	1	1	1
$-l_{25}$	-	-	-	-	-	-	-	-	-	-	-	1	1	1	1	1	1
$-l_{26}$	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1	1	1
$-l_{27}$	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1	1
$-l_{28}$	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1
$-l_{29}$	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1
$-l_{30}$	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1

5 Comparisons and Discussions

On the basis of the above results, we compare all the generators discussed in this paper with respect to generation efficiency, lattice structure, and portability.

5.1 Generation efficiency

In Table 3, we give a list of the times taken to generate one million random numbers for all the generators discussed in this paper, where typical parameters are chosen for each generator. The programs of the generators are all written as subroutines by using the C programming language, and run under AIX on the IBM PS/55 system (80386, 16MHz). The times do not include the initialization time. For comparison, a simple linear congruential generator with a modulus $2^{31} - 1$ took 16 seconds to generate a sequence of the same length. All the generators look fast enough for most of simulation purposes. Note

Table 3: The times (sec.) taken to generate one million random numbers by using the following typical generators: multiple recursive (MRG), GFSR, Twisted GFSR (TGFSR), lagged Fibonacci with $-$ (FIB($-$)), subtract-with-borrow (SWB), and Wichmann-Hill (WH) generators.

generators	MRG	GFSR	TGFSR	FIB($-$)	SWB	COMBT	WH
time (sec.)	36	18	25	21	24	19	58
memory (words)	5	521	25	607	43	2	2
period length	$\approx 2^{155}$	$2^{521} - 1$	$2^{800} - 1$	$\approx 2^{637}$	$\approx 2^{1376}$	$\approx 2^{60}$	$\approx 2^{60}$

that in the case of combined generators, in order to obtain much longer period lengths, we need to increase the number of components, and as a result the generation speed becomes slower.

In terms of memory efficiency, which is another important criterion for selecting good random number generators, lagged Fibonacci generators with $+, -$, and XOR are not recommended. Here, the memory means the area-size needed to store the data for the recurrence relations. For example, compare the TGFSR with $r = 25$, $w = 32$ and the GFSR with $r = 521$ used in Table 3. The period length of the TGFSR is $2^{800} - 1$, which is much larger than $2^{521} - 1$, the period of the GFSR, although the memory required for the TGFSR is about one-twentieth that required for the GFSR. Furthermore, the sequences generated by the TGFSR are 25-distributed with 32 bit resolution, whereas the GFSR produces only $16(= [521/32])$-distributed sequences with the same resolution. Similar arguments also hold for the comparisons of add-with-carry (or subtract-with-borrow) generators with additive (or subtractive) generators.

5.2 Lattice structure

First, we should keep in mind that any pseudorandom sequences that are deterministically generated suffer from a certain structure (or nonrandomness property). Therefore, we need to know the inherent structure of the sequences we use as well as possible, to avoid obtaining meaningless results in our simulation experiments. In this sense, linear congruential sequences of both the integer type and the polynomial type are well understood as regards their inherent structure, namely, the lattice structure.

As we saw, there are two types of lattice structure. One is the integer lattice, which corresponds to multiple recursive generators, add-with-carry and subtract-with-borrow generators, and Wichmann-Hill type generators. The relation between the uniform distribution and the integer lattice structure has been analyzed and explained [4,9,13]. The other type is the polynomial lattice, which corresponds to shift register generators, Twisted GFSR generators, and combined Tausworthe generators. In this case, the relation between

the uniform distribution and the lattice structure of sequences has recently been found and analyzed [2,17,20,21]. In both cases, spectral and lattice (or Beyer) tests have been developed for analyzing the uniform distribution of the sequences with respect to the lattice structure. As shown in the previous sections, for the typical generators such as multiple recursive generators with primitive trinomials, add-with-carry and subtract-with-borrow generators, and Twisted GFSR generators, the lattice structures in higher dimensions than the degree of the recurrence relation are not good. Only for Wichmann-Hill and combined Tausworthe generators, there exist several parameters which give good lattice structure even for high dimensions [6,21]. In this case, however, the sets of points from these sequences are only subsets of all the points in the lattice associated to the generators, formed by the multiplicative groups corresponding to the generators. This is the main difference between combined generators and the others.

In order to conduct the spectral or lattice test, we are bound to be involved in calculating the reduced basis (or the successive minima) of the lattice. In the case of the integer lattice, the problem of finding the shortest vector is believed to be NP-hard. However, recent progress has shown that it becomes feasible to calculate the reduced basis in up to 25 dimensions for cases with moduli up to nearly 2^{63}, which corresponds to the analysis of multiple recursive generators [5]. Add-with-carry and subtract-with-borrow generators correspond to the case of large moduli (a few hundred bits), where we need highly sophisticated computer programs to analyze their lattice structure, because the Cholesky decomposition involved becomes numerically unstable for the general case [22]. On the other hand, in the case of the polynomial lattice over finite fields we have Lenstra's polynomial-time algorithm [7], which proves to be faster than the algorithm based on Gaussian elimination. Actually, we can calculate the reduced basis in up to 50 dimensions in a reasonable time on workstations even for the case of moduli around 2^{500}. In this respect, it will be cheaper to analyze the high dimensional performance of Twisted GFSR generators rather than that of add-with-carry, subtract-with-borrow and multiple recursive generators.

5.3 Portability

The portability of an implementation depends on which programming language is used. For example, the C programming language includes the bitwise exclusive-or operation (XOR), so Tausworthe, GFSR, twisted GFSR, and combined Tausworthe generators are portable and easy to implement. However, this is not the case in Pascal or in standard FORTRAN. (Notice that most versions of FORTRAN in practical use have a built-in function for bitwise exclusive-or.) Add-with-carry, subtract-with-borrow, and multiple recursive generators can be implemented in a portable way for most of the programming languages popular in the scientific computing community.

6 Conclusions

For period lengths up to around 2^{100}, combined Tausworthe and Wichmann-Hill generators are recommended because the generation speed is fast, the memory size is small, and there are several generators with good lattice structures even for high dimensions. For very long periods, at present, we would recommend add-with-carry and subtract-with-borrow, multiple recursive with primitive trinomials, and Twisted GFSR generators. In spite of the good empirical performances reported by the originators, the lattice structures of the sequences from these generators are not good in dimensions higher than the degree of the

recurrence as we explained in the foregoing sections. Consequently, the current issue is how to find those generators with very long periods that have good lattice structure even in high dimensions and can be quickly generated.

In any case, all of the generators have only recently been proposed or come to be commonly used, so we need much more experience and knowledge about how to use them. There may be a trade-off between the efficiency of generation and the randomness properties (particularly the uniform distribution). These issues are important topics for future investigations.

References

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

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

[3] M. Fushimi and S. Tezuka, *The k-Distribution of Generalized Feedback Shift Register Pseudorandom Numbers*, Comm. ACM, 26, (1983), 516-523.

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

[5] P. L'Ecuyer and F. Blouin, *Linear Congruential Generators of Order $k > 1$*, Proc. of the 1988 Winter Simulation Conference, IEEE Press, (1988), 432-439.

[6] P. L'Ecuyer and S. Tezuka, *Structural Properties for Two Classes of Combined Random Number Generators,* Math. Comp., 57, (1991), 735-746.

[7] A.K. Lenstra, *Factoring Multivariate Polynomials over Finite Fields*, J. Comput. Syst. Sci., 30, (1985), 235-248.

[8] T.G. Lewis and W.H. Payne, *Generalized Feedback Shift Register Pseudorandom Number Algorithms,* Journal of ACM., 20, (1973), 456-468.

[9] G. Marsaglia, *Random Numbers Fall Mainly in the Planes*, In Proc. Nat. Acad. Sci. U.S.A. 60, (1968), 25-28.

[10] G. Marsaglia, *A Current View of Random Number Generators*, In Proc. Computer Science and Statistics: 16-th Symposium on the Interface. North-Holland, Amsterdam. (1985), 3-10.

[11] G. Marsaglia and A. Zaman, *A New Class of Random Number Generators*, Annals of Applied Probability, 1, (1991), 462-480.

[12] M. Matsumoto and Y. Kurita, *Twisted GFSR Generators*, ACM Trans. Modeling and Computer Simulation, 2, (1992), 179–194.

[13] H. Niederreiter, *Quasi-Monte Carlo Methods and Pseudorandom Numbers*, Bull. Amer. Math. Soc. 84, (1978), 957-1041.

[14] H. Niederreiter, *Recent Trends in Random Number and Random Vector Generation*, Annals of Operations Research, 31, (1991), 323-346.

[15] H. Niederreiter, *Random Number Generation and Quasi-Monte Carlo Methods*, CBMS-NSF Regional Conference Series in Applied Math., no. 63, SIAM, (1992).

[16] R.C. Tausworthe, *Random Numbers Generated by Linear Recurrence Modulo Two,* Math. Comp., 19, (1965), 201-209.

[17] S. Tezuka, *Walsh-Spectral Test for GFSR Pseudorandom Number Generators,* Comm. ACM., 30, (1987), 731-735.

[18] S. Tezuka, *On the Discrepancy of GFSR Pseudorandom Numbers,* J. ACM., 34, (1987), 939-949.

[19] S. Tezuka, *On Optimal GFSR Pseudorandom Number Generators,* Math. Comp., 50, (1988), 531-533.

[20] S. Tezuka, *Lattice Structure of Pseudorandom Sequences from Shift Register Generators,* Proc. of the 1990 Winter Simulation Conference, IEEE Press, (1990), 266-269.

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

[22] S. Tezuka and P. L'Ecuyer, *Analysis of Add-with-Carry and Subtract-with-Borrow Random Number Generators,* in Proc. of the 1992 Winter Simulation Conference, IEEE Press, (1992), 443-447.

[23] S. Tezuka and M. Fushimi, *Calculation of Fibonacci Polynomials for GFSR Sequences with Low Discrepancies*, Math. Comp. 60, 202(1993), 763–770.

[24] S. Tezuka and Y. J. Wong, *Ising Model Simulations with Two Classes of Random Number Generators*, submitted for publication.

[25] B.A. Wichmann and I.D. Hill, *An Efficient and Portable Pseudorandom Number Generator,* Applied Statistics, 31 (1982), 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.)

Shu Tezuka
IBM Research,
Tokyo Research Laboratory
1623-14, Shimotsuruma
Yamato, Kanagawa 242, Japan.