Journal of Computational and Applied Mathematics 236 (2011) 141–149

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

An efficient lattice reduction method for $\mathbf{F}_2$-linear pseudorandom number generators using Mulders and Storjohann algorithm

Shin Harase [*]

Graduate School of Mathematical Sciences, The University of Tokyo, 3-8-1 Komaba, Meguro-ku, Tokyo 153-8914, Japan

ARTICLE INFO

Article history:
Received 27 September 2010
Received in revised form 1 April 2011

MSC:
65C10
11K45

Keywords:
Random number generation
Uniformity
Lattice reduction algorithm

ABSTRACT

Recent simulations often use highly parallel machines with many processors, and they need many pseudorandom number generators with distinct parameter sets, and hence we need an effective fast assessment of the generator with a given parameter set. Linear generators over the two-element field are good candidates, because of the powerful assessment via their dimensions of equidistribution. Some efficient algorithms to compute these dimensions use reduced bases of lattices associated with the generator. In this article, we use a fast lattice reduction algorithm by Mulders and Storjohann instead of Schmidt's algorithm, and show that the order of computational complexity is lessened. Experiments show an improvement in the speed by a factor of three. We also report that just using a sparsest initial state (i.e., consisting of all 0 bits except one) significantly accelerates the lattice computation, in the case of Mersenne Twister generators.

1. Introduction

A recent trend in large-scale simulations is to use parallelism, based on many processors (or cores). In such simulations, a large number of pseudorandom number generators with distinct parameter sets are often required, to assign each parameter set to every core or every process. For this, an effective assessment of the quality of a generator with a given parameter is desired. Pseudorandom number generators based on linear recurrences over the two-element field are good candidates for this purpose (cf. Dynamic Creator of Mersenne Twister [1]), since they have effective assessment via their dimensions of equidistribution, which are values ensuring uniformity of high-dimensional distribution. This article proposes a fast algorithm to compute these dimensions, using a lattice reduction algorithm by Mulders and Storjohann [2] in the SIS method previously proposed by the author, Matsumoto, and Saito [3].

Let $\mathbf{F}_2 := \{0, 1\}$ be the two-element field, i.e., addition and multiplication are done modulo 2. We consider a pseudorandom number generator with a p-dimensional state space $S := \mathbf{F}_2^p$, an $\mathbf{F}_2$-linear state transition function $f : S \to S$, and an $\mathbf{F}_2$-linear output function $o : S \to O$ where $O := \mathbf{F}_2^w$ is the set of outputs (w intended for the word size of the machine). When we give an initial state $s_0 \in S$, the generator computes the next state by the recursion $s_{i+1} = f(s_i)$ $(i = 0, 1, 2, \ldots)$, and the output sequence is given by $o(s_0), o(s_1), o(s_2), \ldots \in O$. We identify O with the set of unsigned w-bit binary integers. This type of generator is called an $\mathbf{F}_2$-*linear generator*. For example, Mersenne Twisters [4,5] and WELL generators [6] belong to this class.

One of the merits of an $\mathbf{F}_2$-linear generator is that we can compute the dimension of equidistribution which measures high-dimensional uniformity of the sequence for the whole period. Since most significant bits (MSBs) in a word-size integer are more influential than the lower ones in a Monte Carlo simulation, we often use the *dimension of equidistribution of* v-*bit accuracy* $k(v)$ defined as follows (see surveys [7,8] for details).

* Tel.: +81 3 5465 7001; fax: +81 3 5465 7011.
E-mail addresses: harase@ms.u-tokyo.ac.jp, sharase@orange.ocn.ne.jp.

0377-0427/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.cam.2011.06.005

Definition 1. Consider an $\mathbf{F}_2$-linear generator as above. Let v be an integer with $1 \leq v \leq w$. Let $\mathrm{tr}_v : \mathbf{F}_2^w \to \mathbf{F}_2^v$ be the projection from w bits to the v MSBs, called the *truncation function*.

For a positive integer k, we define $o_v^{(k)}$ as the composition:

$$o_v^{(k)} : S \to (\mathbf{F}_2^v)^k, \qquad s_0 \mapsto (\mathrm{tr}_v \circ o(s_0), \mathrm{tr}_v \circ o(f(s_0)), \ldots, \mathrm{tr}_v \circ o(f^{k-1}(s_0))).$$

This map sends a state to the consecutive k-tuple output integers from the state, with only v MSBs extracted from each integer.

The generator is said to be *k-dimensionally equidistributed* if and only if the map $o_v^{(k)}$ is surjective. The maximum such k is called the *dimension of equidistribution with v-bit accuracy*, and denoted by $k(v)$.

The larger $k(v)$ for each of $1 \leq v \leq w$ is desirable. By dimension comparison, an obvious upper bound exists:

$$k(v)v \leq p, \quad \text{or equivalently } k(v) \leq \lfloor p/v \rfloor.$$

Definition 2. The gap

$$d(v) := \lfloor p/v \rfloor - k(v)$$

is called the *dimension defect at* v, and their sum

$$\Delta := \sum_{v=1}^{w} (\lfloor p/v \rfloor - k(v))$$

is called the *total dimension defect*. If $\Delta = 0$, the generator is said to be *maximally equidistributed*.

Usually, the above definition is adopted under the assumption that f has the maximal period $2^p - 1$. Then, every state except the zero state in S occurs exactly once in a period, and hence if one plots points in $[0, 1)^k$ using overlapping k-tuples from the outputs of the generator over a whole period, then each of 2^{kv} pieces of k-dimensional sub-cube obtained by dividing each axis into 2^v equal-length segments gets the same number of points (except the cube at the origin, which gets one less). This explains the terminology of the dimension of equidistribution. All generators treated here are assumed to satisfy the maximal-period condition.

We can compute $k(v)$ by linear algebra [9], since it is equivalent to the fullness of the rank of the representation matrix of $o_v^{(k)}$. However, a naive Gaussian elimination costs $O(p^3)$ bit operations for the rank computation, which is huge for large generators such as Mersenne Twister ($p = 19937$). Couture, L'Ecuyer, and Tezuka [10] and Tezuka [11] proposed much faster algorithms based on lattice structures over power series. Couture and L'Ecuyer [12] proposed an improvement by using the dual lattices and Lenstra's reduction algorithm [13]. Recently, the author, Matsumoto, and Saito [3] proposed a more efficient lattice computation method named SIS, based on manipulating the state space instead of the lattice points. The number of bit operations for SIS to obtain all $k(v)$ is approximately $2wp^2 + \frac{4}{3}w^3p + \frac{1}{4}w^4$. However, the computation of $k(v)$ is still time-consuming for large w (e.g., 64-bit or 128-bit generators [5,14]) because of presence of the terms of order w^3p and w^4.

In this article, we improve on SIS method by replacing Schmidt's lattice reduction with a more efficient lattice reduction algorithm based on [2,15], and show that this algorithm lowers the computational complexity. The number of bit operations is approximately $2wp^2 + \frac{1}{2}w^2p + \frac{1}{2}w^2(w+1)$.

As another direction, we propose to apply our algorithm to the sequences generated from 0-excess initial states (i.e., the state where all bits are 0 except one, which is a bad initialization used to assess the generators in [6]). In fact, this initialization considerably accelerates the lattice computation for generators with sparse transition function, such as Mersenne Twisters.

In Section 2, we recall the lattice method for computing $k(v)$. In Section 3, we propose a new lattice reduction algorithm based on [2,15], and analyze the computational complexity. In Section 4, we report the timing with or without 0-excess initial states.

2. Lattice method

2.1. Lattice structure

We briefly recall the lattice method for computing $k(v)$. Let K denote the formal power series field $K := \mathbf{F}_2((t^{-1})) = \left\{\sum_{j=j_0}^{\infty} a_j t^{-j} \mid a_j \in \mathbf{F}_2, j_0 \in \mathbf{Z}\right\}$. For $\alpha = \sum_{j=j_0}^{\infty} a_j t^{-j} \in K$, we define a standard norm by

$$|\alpha| := \begin{cases} \max\{-j \in \mathbf{Z} \mid a_j \neq 0\} & \text{if } \alpha \neq 0, \\ -\infty & \text{if } \alpha = 0. \end{cases}$$

For a vector $\gamma = (\alpha_1, \alpha_2, \dots, \alpha_v) \in K^v$, we define its norm by $\|\gamma\| := \max_{1\le i\le v} |\alpha_i|$. Note that $|\alpha|$ and $\|\gamma\|$ are often negative integers. To represent such a γ, we sometimes use a formal power series with vector coefficients. Namely, if $\alpha_i = \sum_{j=j_0}^{\infty} a_{i,j}t^{-j}$, then we denote

$$\gamma = (\alpha_1, \dots, \alpha_v) = \sum_{j=j_0}^{\infty} (a_{1,j}, \dots, a_{v,j})t^{-j}.$$

For $\gamma \neq (0, \dots, 0)$, we define its *coefficient vector at the leading term* $\pi(\gamma) \in \mathbf{F}_2^v$ by $\pi(\gamma) := (a_{1,-\|\gamma\|}, a_{2,-\|\gamma\|}, \dots, a_{v,-\|\gamma\|})$, so that $\gamma = \pi(\gamma)t^{\|\gamma\|} +$ (lower degree terms in t).

A subset $L \subset K^v$ is said to be an $\mathbf{F}_2[t]$-*lattice* if there exist K-linear basis $\omega_1, \omega_2, \dots, \omega_v$ of K^v such that L is their span over $\mathbf{F}_2[t]$, i.e., $L = \langle \omega_1, \omega_2, \dots, \omega_v\rangle_{\mathbf{F}_2[t]}$. We call such a set of vectors a *basis* of L. A basis $\omega_1, \dots, \omega_v$ of L is said to be a *reduced basis* if $\pi(\omega_1), \dots, \pi(\omega_v)$ are linearly independent over $\mathbf{F}_2$. Let us sort $\omega_1, \omega_2, \dots, \omega_v$ so that $\|\omega_1\| \le \|\omega_2\| \le \cdots \le \|\omega_v\|$. Then, the numbers $\nu_i := \|\omega_i\|$ $(i = 1, \dots, v)$ are uniquely determined by the lattice, and called *successive minima* (see [16, 17] for details).

Let $\mathrm{tr}_v : O = \mathbf{F}_2^w \to \mathbf{F}_2^v$ be the truncation function (Definition 1). For an $\mathbf{F}_2$-linear generator and an initial state $s_0 \in S$, we define its associated v-dimensional vector-valued *generating function* $\chi_v(s_0)$ as follows:

$$\chi_v(s_0) := \sum_{j=0}^{\infty} \mathrm{tr}_v(o(f^j(s_0)))t^{-1-j} = \mathrm{tr}_v(o(s_0))t^{-1} + \mathrm{tr}_v(o(s_1))t^{-2} + \cdots \in K^v. \tag{1}$$

We define $\Lambda_v \subset K^v$ by an $\mathbf{F}_2[t]$-linear span

$$\Lambda_v := \langle e_1, e_2, \dots, e_v, \chi_v(s_0)\rangle_{\mathbf{F}_2[t]}, \tag{2}$$

where $e_i \in K^v$ denotes the unit vector whose ith component is 1 and the other components are 0 $(i = 1, \dots, v)$. If we multiply $\chi_v(s_0)$ by the characteristic polynomial $P(t)$ of the transition function f, then we obtain a vector whose elements are all polynomial. Hence, Λ_v is an $\mathbf{F}_2[t]$-lattice. If $P(t)$ is irreducible, then Λ_v can be proved to be independent of the choice of the initial state $s_0 \neq (0, \dots, 0)$.

The following theorem asserts that we can obtain $k(v)$ by computing the successive minima of Λ_v, namely, by computing a reduced basis of Λ_v.

Theorem 1 ([10,11]). *Consider an $\mathbf{F}_2$-linear generator. Assume that the characteristic polynomial $P(t)$ of its transition function is irreducible. Take nonzero initial state $s_0 \in S$. Then, we have $k(v) = -\nu_v$, where ν_v is the vth successive minimum of Λ_v.*

2.2. Schmidt's reduction, inductive projection, and state representation

By Theorem 1, computations of $k(v)$ are reduced to computations of reduced bases of the lattices Λ_v. From now on, we treat the problem to compute a reduced basis of Λ_v for all $1 \le v \le w$. Couture–L'Ecuyer dual lattice method [12] computes a reduced basis of the dual basis of Λ_v inductively for $v = 1, \dots, w$. The author, Matsumoto, and Saito [3] proposed even faster method named SIS (for Schmidt's generating set reduction, inductive projection, and state representation), which we briefly recall.

Schmidt's generating set reduction (*SGR*), a variant of [18], is straightforward: for a given generating set $\{\omega_1, \dots, \omega_m\}$ of an $\mathbf{F}_2[t]$-lattice L, find any non-trivial linear relation over $\mathbf{F}_2$ among a subset of the coefficient vectors at the leading terms $\pi(\omega_i)$ $(1 \le i \le m)$. If there is none, the generating set is a reduced basis. If there is any linear relation, by using the vectors appearing in the linear relation, one can reduce the longest (with respect to the norm) by $\mathbf{F}_2[t]$-linear combination of the rest vectors. We iterate this reduction, until no linear relation exists. This works for a generating set of the lattice, while Lenstra's algorithm [13] works only for a basis (cf. its generalization for the generating set is in [19]).

The *inductive projection* is a way to compute reduced bases for all v. We compute a reduced basis $\{\omega_1, \dots, \omega_w\}$ of Λ_w by SGR. Then, we compute reduced bases of $\Lambda_{w-1}, \Lambda_{w-2}, \dots, \Lambda_1$, inductively using projection, as follows. Let ρ be the projection

$$\rho : K^{v+1} \to K^v, \quad (\alpha_1, \dots, \alpha_{v+1}) \mapsto (\alpha_1, \dots, \alpha_v),$$

which depends on v but we do not specify v since it is clear from the context. It is easy to see ((i) in Lemma 3 of [12])

$$\Lambda_v = \rho(\Lambda_{v+1}), \tag{3}$$

by looking at the definition (2) of Λ_v. This implies that if $\{\omega_1, \dots, \omega_{v+1}\}$ is a basis of Λ_{v+1}, then $\{\rho(\omega_1), \dots, \rho(\omega_{v+1})\}$ is a generating set of Λ_v. If the former is a reduced basis, then the vectors in the latter generating set are already short, and thus more easily converge to a reduced basis than starting from the defining generating set (2). Once we compute a reduced basis of Λ_w, we compute those of $\Lambda_{w-1}, \Lambda_{w-2}, \dots, \Lambda_1$, inductively as above. This is the *inductive projection*.

The *state representation* is a method to represent a lattice point in Λ_v by a state in S and some polynomial information. We define the action of t on $s \in S$ as $s \cdot t := f(s)$. Note that $\Lambda_v/(\mathbf{F}_2[t]^v)$ is an $\mathbf{F}_2[t]$-module. We have the following:

Lemma 2 (*Lemma 3.4. of [3]*)**.** *If the characteristic polynomial $P(t)$ is irreducible and χ_v in (1) is nonzero,*

$$\chi_v : S \to \Lambda_v/(\mathbf{F}_2[t]^v)$$

is an isomorphism as $\mathbf{F}_2[t]$-modules.

In this case, we have a decomposition

$$\Lambda_v = \mathbf{F}_2[t]^v \oplus \chi_v(S)$$

as an $\mathbf{F}_2$-linear space, and thus any element of Λ_v has a representation $poly + \chi_v(s)$ with a unique pair $(poly, s) \in \mathbf{F}_2[t]^v \times S$. Such a pair is said to be a (unique) *state representation* of $poly + \chi_v(s) \in \Lambda_v$. *poly* is called the *polynomial part*. In lattice reduction algorithms, only the addition of two lattice points and the multiplication to a lattice point by t suffice to complete the reduction. The addition is easy in the state representation. The multiplication by t is given by $(poly + \chi_v(s)) \cdot t = poly \cdot t + \mathrm{tr}_v(o(s)) + \chi_v(f(s))$. Thus, we may execute lattice computation.

3. Main result: PIS method

SGR algorithm seems comparable to or even more efficient than Lenstra's algorithm. More recently, Mulders and Storjohann [2] developed a faster lattice reduction algorithm. Wang, Zhu, and Pei [15] independently proposed a similar but somewhat specialized algorithm. Both of these can be called the *pivot reduction algorithm*. In this section, we propose to replace SGR in SIS with the pivot reduction algorithm, which we shall call *PIS*.

We follow the notation of [2] with some simplification.

Definition 3. For a nonzero vector $\gamma = (\alpha_1, \ldots, \alpha_v) \in K^v$, we define its *pivot index* (denoted by $I(\gamma) \in \{1, 2, \ldots, v\}$) by

$$I(\gamma) := \max\{i \mid |\alpha_i| = \|\gamma\|\}.$$

In other words, $I(\gamma)$ is the coordinate index of the rightmost nonzero component in the coefficient vectors at the leading term $\pi(\gamma) \in \mathbf{F}_2^v$.

Let L be an $\mathbf{F}_2[t]$-lattice spanned by a generating set $\omega_1, \ldots, \omega_m \in K^v$ ($m \geq v$), i.e., $L = \langle \omega_1, \ldots, \omega_m \rangle_{\mathbf{F}_2[t]}$. Let M be a generating set $\{\omega_1, \ldots, \omega_m\}$. From now on, we assume that M does not contain the zero vector. We shall define a *pivot reduction* on M, which gives another (smaller in a sense) generating set M' of the same lattice.

Lemma 3. *Assume that $I(\omega_i)$ for $\omega_i \in M$ are all different, i.e.,*

$$1 \leq k \neq l \leq m \Rightarrow I(\omega_k) \neq I(\omega_l). \tag{4}$$

Then, M is a reduced basis of L.

This follows because $\pi(\omega_i)$ are linearly independent by the definition of $I(\omega_i)$.

Definition 4 (*Pivot reduction*)**.** Assume $I(\omega_l) = I(\omega_k)$ for $1 \leq k \neq l \leq m$. By symmetry, we may assume $\|\omega_l\| \geq \|\omega_k\|$. The *pivot reduction* of ω_l by ω_k is to obtain a new generating set M' as follows. Put $\omega_l' := \omega_l - \omega_k \cdot t^{\|\omega_l\| - \|\omega_k\|}$. If $\omega_l' = (0, \ldots, 0)$, then put $M' := \{\omega_1, \ldots, \omega_{l-1}, \omega_{l+1}, \ldots \omega_m\}$ and renumber them (and hence m is decreased by one). If $\omega_l' \neq (0, \ldots, 0)$, then put $M' := \{\omega_1, \ldots, \omega_{l-1}, \omega_l', \omega_{l+1}, \ldots \omega_m\}$.

A pivot reduction decreases the "size" of M as follows.

Lemma 4 (*Lemma 2.2 of [2]*)**.** *Let M' be obtained from M by a pivot reduction of ω_l by ω_k. Let ω_l' be the vector as above. Then, we have either $\|\omega_l'\| < \|\omega_l\|$ or ($\|\omega_l'\| = \|\omega_l\|$ and $I(\omega_l') < I(\omega_l)$). Thus, every pivot reduction decreases the cardinality of M' (if $\omega_l' = (0, \ldots, 0)$), or $(\|\omega_l'\|, I(\omega_l')) < (\|\omega_l\|, I(\omega_l))$ holds with respect to the lexicographic order.*

The following lemma gives the relationship between the pivot reduction and the condition (4).

Lemma 5 (*Lemma 2.1. of [2]*)**.** *M does not satisfy the condition (4) if and only if we can apply a pivot reduction on M.*

Mulders and Storjohann lattice reduction algorithm [2] is to iterate pivot reductions on M until (4) holds. Since $\{(\|\omega\|, I(\omega)) \mid (0, \ldots, 0) \neq \omega \in L\}$ is well-ordered and bounded from below, this algorithm terminates and gives a reduced basis.

Next, we specialize this algorithm in a more efficient way, as proposed in Section IV of Wang et al. [15], with the following restriction on M. Assume $m = v + 1$, namely, the given generating set M of $L \subset K^v$ is $\{\omega_1, \ldots, \omega_{v+1}\}$. Assume that the following *triangular condition* for the first v vectors holds:

$$I(\omega_i) = i \quad \text{for } i = 1, \ldots, v. \tag{5}$$

Note that $\{e_1, \ldots, e_v, \chi_v(s_0)\}$ in (2) satisfies (5). The next procedure gives a sequence of pivot reductions that preserves this triangular condition to reach a reduced basis. We call this algorithm *Pivot Lattice Reduction* (*PLR*).

```
procedure Pivot Lattice Reduction (with triangular condition)
input: a generating set $\omega_1, \omega_2, \dots, \omega_{v+1}$ of $L$ with triangular condition (5).
output: a reduced basis $\omega_1, \omega_2, \dots, \omega_v \in L$.
begin
While $\omega_{v+1} \neq (0, \dots, 0)$ do
    (reduction step)
    Set $k \leftarrow I(\omega_{v+1})$ (i.e., the pivot index of $\omega_{v+1}$).
    If $\|\omega_{v+1}\| \geq \|\omega_k\|$
        Set $\omega_{v+1} \leftarrow \omega_{v+1} - \omega_k \cdot t^{\|\omega_{v+1}\|-\|\omega_k\|}$.
    else
        Swap $\omega_k$ and $\omega_{v+1}$.
        Set $\omega_{v+1} \leftarrow \omega_{v+1} - \omega_k \cdot t^{\|\omega_{v+1}\|-\|\omega_k\|}$.
    end if
end while
end
```

In this algorithm, $\omega_{v+1} \leftarrow \omega_{v+1} - \omega_k \cdot t^{\|\omega_{v+1}\|-\|\omega_k\|}$ is a pivot reduction. To keep the triangular condition (5), we reduce only the last vector ω_{v+1}, by swapping with ω_k if $\|\omega_k\| > \|\omega_{v+1}\|$ at the beginning of the reduction step. The algorithm terminates when and only when the reduction of ω_{v+1} is zero, and then the first v vectors are a reduced basis.

The following theorem shows the number of pivot reductions required to reach a reduced basis (similarly to Theorem 3.1 in [3]).

Theorem 6. *Let $\omega_1, \dots, \omega_{v+1}$ be a generating set of an $\mathbf{F}_2[t]$-lattice $L \subset K^v$ which satisfies the triangular condition (5). Let $\omega'_1, \dots, \omega'_v$ be the obtained reduced basis by PLR. Then, the number of pivot reductions in PLR is bounded from above by*

$$\left(\sum_{i=1}^{v+1} \|\omega_i\| - \sum_{i=1}^{v} \|\omega'_i\| - \|\omega^{\text{last}}\| + 1\right) v, \tag{6}$$

where ω^{last} denotes the last vector reduced to zero at the final step in PLR.

Proof. Again let $M := \{\omega_1, \dots, \omega_{v+1}\}$. We define the number *state size S^M* of M by

$$S^M := \sum_{i=1}^{v+1} (\|\omega_i\| v + I(\omega_i)). \tag{7}$$

Let N be a generating set obtained by applying some number of pivot reductions to M. If the cardinality of N is less than M, then N is a reduced basis, by the triangular condition (5). Assume that N consists of nonzero $v+1$ vectors, and define S^N as well as (7). Then, $S^M - S^N$ is said to be the *state drop* [2]. The number of pivot reductions from M to N is bounded above by the state drop, since a pivot reduction decreases the state size at least by one (see Lemma 4 or [2, Theorem 2.2]). Let N be the previous generating set to the last stage. Consequently, the pivot reduction to N at the $(v+1)$-st vector eliminates this vector to zero and gives a reduced basis. Hence $N = \{\omega'_1, \dots, \omega'_v, \omega^{\text{last}}\}$. By the triangular condition (5), $I(\omega_i) = I(\omega'_i) = i$ $(i = 1, \dots, v)$, which are canceled out in state drop. We have bounds $I(\omega_{v+1}) \leq v$ and $I(\omega^{\text{last}}) \geq 1$. Thus,

$$S^M - S^N \leq \left(\sum_{i=1}^{v+1} \|\omega_i\| - \sum_{i=1}^{v} \|\omega'_i\| - \|\omega^{\text{last}}\|\right) v + v - 1.$$

By adding 1 for the final reduction of ω^{last}, the theorem follows. □

Remark 1. This upper bound on the number of pivot reductions in PLR is exactly v times the number of reduction steps in SGR. Note that one reduction in SGR requires $O(v)$ times operations on vectors.

It is easy to check that the generating set obtained by the inductive projection (3) through PLR keeps the triangular condition (5), because if $\{\omega_1, \dots, \omega_{v+1}\}$ is a reduced basis of Λ_{v+1} obtained by PLR, then we have $I(\omega_i) = i$ for $i = 1, \dots, v+1$. Then, $\{\rho(\omega_1), \dots, \rho(\omega_{v+1})\}$ satisfies the triangular condition, so that we can inductively compute a reduced basis of Λ_v.

The state representation is applicable for PLR with no modification. Our proposal is a combination of PLR, inductive projection, and state representation, for computing all $k(v)$'s, named *PIS*.

We analyze the computational complexity for PIS. In a practical $\mathbf{F}_2$-linear generator, f and o can be computed by a few operations, often independently of the size of the state space. Hence, we assume that these operations are negligible from the total cost of the computation. For each $1 \leq v \leq w$, we consider the following average assumption similar to [3]. In Theorem 6, let L be Λ_v. Then, we have

$$\sum_{i=1}^{v} \|\omega_i'\| = -\dim(S), \tag{8}$$

by Lemma 2.4 in [3]. Since the last vector ω^{last} is reduced by one of $\omega_1', \ldots, \omega_v'$, we have the trivial lower bound

$$\|\omega^{\text{last}}\| \geq \nu_1, \tag{9}$$

where ν_1 is the first successive minimum of Λ_v. For simplicity, we make the assumption: $\|\omega^{\text{last}}\|$ is approximately equal to or greater than the average of $\|\omega_1'\|, \ldots, \|\omega_v'\|$, namely,

$$\|\omega^{\text{last}}\| \geq -\dim(S)/v. \tag{10}$$

This assumption fails in some cases (see the latter example in Section 4.1), but it holds (or holds with slight modification) for many applications (see Remark 3 and Section 4.1). From now on, in the computation of complexity, we always assume (10). The following theorem gives an upper bound of the total bit operations.

Theorem 7. *Under (10), PIS requires at most*

$$2w\dim(S)^2 + \frac{1}{2}w^2\dim(S) + \frac{1}{2}w^2(w+1)$$

bit operations to compute all $k(v)$, $w \geq v \geq 1$.

We need two more lemmas and corollaries to prove the theorem, corresponding to Theorems 4.1 and 4.2 in [3].

Lemma 8. *Under (10), the number of pivot reductions to obtain a reduced basis by PLR from the generating set in (2) is bounded by $(v+1)\dim(S)$.*

Proof. We apply Theorem 6. Since $\|e_i\| = 0$ and $\chi_v(s_0) \leq -1$, the first summation in (6) is less than or equal to -1. The second summation in (6) is $-\dim(S)$ by (8). The third term in (6) is bounded by $\dim(S)/v$ by (10). The result follows from a simple computation. □

Corollary 9. *Under the same assumption, the number of bit operations inside S (i.e., neglecting the polynomial part) in the PLR algorithm starting from the generating set in (2) is bounded by $(v+1)\dim(S)^2$, when using the state representation.*

Lemma 10. *Under (10), the number of pivot reductions in the PLR algorithm starting from $\{\rho(\omega_1), \ldots, \rho(\omega_{v+1})\}$, where $\omega_1, \ldots, \omega_{v+1}$ are the reduced basis of Λ_{v+1} previously obtained by PLR, has an upper bound $\dim(S) + v$.*

Proof. We apply Theorem 6. From $\|\rho(\omega_i)\| \leq \|\omega_i\|$ and (8), the first summation in (6) is $\sum_{i=1}^{v+1} \|\rho(\omega_i)\| \leq \sum_{i=1}^{v+1} \|\omega_i\| = -\dim(S)$. The second summation in (6) is $-\dim(S)$ by (8). The third term in (6) is bounded by $\dim(S)/v$ by (10). The result follows. □

Corollary 11. *Under the same assumption, the number of bit operations inside S for PLR to compute a reduced basis of Λ_v from that of Λ_{v+1} obtained by PLR is bounded by $\dim(S)^2 + v\dim(S)$, when using the state representation.*

Proof of Theorem 7. We count the number of bit operations in the polynomial part of the state representations in PLR. The operations on the polynomial part are necessary only when one reduces a vector with the non-trivial polynomial part in its state representation. At the beginning, $e_1, \ldots, e_w$ are such vectors. For every i, e_i gets the reduction at most i times, before its state representation has no polynomial part. Thus, the number of pivot reductions on the polynomial part in PLR is at most $w(w+1)/2$. Since the polynomial part consists of w bits, the total number of bit operations on the polynomial part in PLR is bounded by $w^2(w+1)/2$. Next, we count the number of bit operations in S of the state representation. Corollary 9 is applied for computing the first step $k(w)$, i.e., when $v = w$. Then, Corollary 11 is applied for computing $k(w-1), k(w-2), \ldots, k(1)$, i.e., for $v = w-1, w-2, \ldots, 1$, by using the inductive projection. Thus, to compute all $k(v)$, the number of bit operations is at most

$$\frac{1}{2}w^2(w+1) + (w+1)\dim(S)^2 + \sum_{v=1}^{w-1}(\dim(S)^2 + v\dim(S)),$$

and we obtain the theorem. □

Remark 2. We compare the computational complexity of PIS (Theorem 7) with SIS method. We assume (10) at the final step in SGR, as well. According to Theorem 4.3 of [3], SIS requires at most

$$2w\,\dim(S)^2 + \frac{4}{3}w^3\dim(S) + \frac{1}{4}w^4$$

bit operations for computing all $k(v)$, $w \geq v \geq 1$. Thus, PIS is superior to SIS in w. This is because SIS requires Gaussian elimination for finding a linear relation among $\pi(\omega_1), \ldots, \pi(\omega_v)$ over $\mathbf{F}_2$, which costs v^3 bit operations, at each reduction step for $k(v)$, while PIS has an automatic triangulation mechanism and no elimination is necessary. This cost is not negligible for large w.

Remark 3. In the preceding work by Couture and L'Ecuyer [12], the computational complexities were proved under the following condition: the minimum and the maximum of the successive minima of Λ_v have a difference of at most 1. This condition is said to be the *regularity*, and it means that all the successive minima are almost equal (or concentrate on the average). Then, from (9), we have $\|\omega^{\mathrm{last}}\| \geq \nu_1 \geq -\dim(S)/v - 1$. Under the regularity condition for each Λ_v, we can prove that PIS requires at most $2w\dim(S)^2 + w^2\dim(S) + \frac{1}{2}w^2(w+1)$ bit operations, because each number of pivot reductions in Lemmas 8 and 10 increases by at most v. Our assumption (10) can be considered as a weaker modified version of the regularity: the regularity requires that all nonzero vectors in Λ_v have the norms no less than $-\dim(S)/v - 1$, while (10) requires that one vector ω^{last} has the norm no less than $-\dim(S)/v$.

4. Numerical experiments

4.1. PIS versus SIS

We show some experiments to compare the following two methods:

1. PIS method (our proposal in Section 3).
2. SIS method (the method in [3]).

We apply these methods to some w-bit $\mathbf{F}_2$-linear generators (where w is 32 or 64), and measure the CPU time for computing $k(w), k(w-1), \ldots, k(2)$ in this order. All the tests are performed on 64-bit AMD-Athlon 64 4000+ and Linux operating system. The programs are implemented with C language and compiled by using gcc compiler version 4.4.1 with the -O2 optimization flag.

We first conducted an experiment with the 32-bit $\mathbf{F}_2$-linear generator WELL19937a' (a variant of WELL19937a [6] with the tempering improved by the author [20], and $\dim(S) = 19937$). This generator is maximally equidistributed. The first to third columns of Table 1 show the CPU time (in seconds). PIS is faster than SIS, and the difference increases when v becomes large, probably because of the Gaussian elimination of cost $O(v^3)$, which is avoided in PIS. The lattice Λ_v has the regularity for every $1 \leq v \leq 32$, except for $v = 6$. In PIS, the assumption (10) holds for five values of $1 \leq v \leq 32$, and a slightly weaker assumption $\|\omega^{\mathrm{last}}\| > -\dim(S)/v - 2$ holds for all v. In SIS, the assumption (10) holds for more than a half of $1 \leq v \leq 32$, and the weaker assumption also holds for all v. The experiments are also in accordance with the complexities obtained above.

To see the dependence on the size of w, we conducted an experiment with a 64-bit Mersenne Twister MT19937-64 [5] downloaded from (http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt64.html). This generator is a new version based on a three-term recurrence (not five-term) with $\dim(S) = 19937$. The right six columns of Table 1 show that PIS is about three times faster than SIS. MT19937-64 is far from maximally equidistributed (i.e., $\Delta = 7820$), and $k(v)$ attains the upper bound only for $v = 1, 2, 3, 4, 16, 32, 64$. For such v's, the lattice Λ_v has the regularity (except for $v = 3$), and the assumption (10) or a weaker assumption $\|\omega^{\mathrm{last}}\| > -\dim(S)/v - 2$ holds. For the other v's, the assumption does not generally hold in both PIS and SIS, but for all v, our experiments show that $\nu_1 > -2\dim(S)/v$ holds, and we have $\|\omega^{\mathrm{last}}\| > -2\dim(S)/v$ by (9). This lower bound changes only the constant coefficients in the order of Theorem 7 (and also in that of Remark 2). Actually, in the case where the generator is far from maximally equidistributed, our experiments show that PIS and SIS are even faster than the cases for which the generator is maximally equidistributed, since the number of reduction steps tends to be smaller (see Table 2 in Section 4.2).

4.2. Use of 0-excess states

As another direction for acceleration, we propose choosing the initial state $s_0 \in S$ as one of the most 0-excess states (e.g., consisting of all 0 bits except for one). The 0-excess states were proposed by Panneton et al. [6] as bad initializations for Mersenne Twisters such that the output sequence is sparse (namely much more 0 than 1) even after thousands of generations. Here, on the other hand, this phenomenon can be used to accelerate the reduction speeds. We designed an experiment comparing the timing of lattice reduction for randomly selected initial states with that for 0-excess initial states. We also conducted an experiment with the effect of 0-excess states to other 32-bit $\mathbf{F}_2$-linear generators, namely Mersenne Twister MT19937 [4] ($\dim(S) = 19937$) and WELL44497a' (a variant of WELL44497a [6] modified by the author [20] with

Table 1
The CPU time for computing $k(v)$ $(w \geq v \geq 2)$ of WELL19937a' and MT19937-64 (in seconds). They are listed in descending order with respect to v, according to the order of computation.

WELL19937a' ($w = 32$)			MT19937-64 ($w = 64$)					
	PIS	SIS		PIS	SIS		PIS	SIS
$k(32)$	0.275	0.451	$k(64)$	0.289	0.933	$k(32)$	0.004	0.009
$k(31)$	0.009	0.014	$k(63)$	0.000	0.000	$k(31)$	0.000	0.003
$k(30)$	0.009	0.014	$k(62)$	0.000	0.001	$k(30)$	0.001	0.002
$k(29)$	0.009	0.014	$k(61)$	0.000	0.001	$k(29)$	0.001	0.004
$k(28)$	0.008	0.014	$k(60)$	0.000	0.000	$k(28)$	0.002	0.004
$k(27)$	0.008	0.013	$k(59)$	0.000	0.001	$k(27)$	0.002	0.005
$k(26)$	0.008	0.013	$k(58)$	0.000	0.001	$k(26)$	0.002	0.007
$k(25)$	0.008	0.012	$k(57)$	0.000	0.000	$k(25)$	0.003	0.007
$k(24)$	0.010	0.012	$k(56)$	0.000	0.001	$k(24)$	0.004	0.007
$k(23)$	0.009	0.012	$k(55)$	0.000	0.001	$k(23)$	0.004	0.008
$k(22)$	0.009	0.012	$k(54)$	0.000	0.001	$k(22)$	0.004	0.008
$k(21)$	0.009	0.012	$k(53)$	0.000	0.001	$k(21)$	0.003	0.007
$k(20)$	0.009	0.012	$k(52)$	0.000	0.000	$k(20)$	0.001	0.003
$k(19)$	0.009	0.012	$k(51)$	0.000	0.001	$k(19)$	0.002	0.005
$k(18)$	0.009	0.012	$k(50)$	0.001	0.001	$k(18)$	0.003	0.006
$k(17)$	0.009	0.011	$k(49)$	0.000	0.001	$k(17)$	0.003	0.006
$k(16)$	0.008	0.012	$k(48)$	0.001	0.001	$k(16)$	0.004	0.007
$k(15)$	0.008	0.012	$k(47)$	0.000	0.001	$k(15)$	0.001	0.003
$k(14)$	0.008	0.011	$k(46)$	0.000	0.001	$k(14)$	0.003	0.005
$k(13)$	0.008	0.010	$k(45)$	0.000	0.001	$k(13)$	0.002	0.003
$k(12)$	0.009	0.010	$k(44)$	0.001	0.001	$k(12)$	0.003	0.005
$k(11)$	0.008	0.011	$k(43)$	0.000	0.001	$k(11)$	0.003	0.003
$k(10)$	0.009	0.010	$k(42)$	0.001	0.001	$k(10)$	0.003	0.004
$k(9)$	0.009	0.011	$k(41)$	0.001	0.002	$k(9)$	0.003	0.002
$k(8)$	0.008	0.010	$k(40)$	0.000	0.001	$k(8)$	0.003	0.003
$k(7)$	0.008	0.011	$k(39)$	0.001	0.002	$k(7)$	0.002	0.004
$k(6)$	0.008	0.010	$k(38)$	0.001	0.002	$k(6)$	0.003	0.003
$k(5)$	0.009	0.009	$k(37)$	0.001	0.002	$k(5)$	0.003	0.004
$k(4)$	0.009	0.010	$k(36)$	0.000	0.003	$k(4)$	0.003	0.004
$k(3)$	0.009	0.010	$k(35)$	0.001	0.004	$k(3)$	0.004	0.005
$k(2)$	0.009	0.009	$k(34)$	0.002	0.005	$k(2)$	0.005	0.005
Total	0.534	0.798	$k(33)$	0.002	0.005	Total	0.386	1.128

Table 2
The cumulative CPU time (in seconds) for computing all $k(v)(w \geq v \geq 2)$ of four $\mathbf{F}_2$-linear generators, by the two reduction algorithms and the two initializations. The column Δ shows the total dimension defect.

	PIS (0-ex.)	SIS (0-ex.)	PIS	SIS	Δ
MT19937-64 ($w = 64$)	0.105	0.197	0.386	1.128	7820
MT19937 ($w = 32$)	0.029	0.036	0.294	0.481	6750
WELL19937a' ($w = 32$)	0.525	0.786	0.534	0.798	0
WELL44497a' ($w = 32$)	2.536	3.091	2.591	3.193	0

$\dim(S) = 44497$). Table 2 gives a summary of the cumulative CPU time (in seconds) for computing all $k(v)(w \geq v \geq 2)$. WELL generators recover from 0-excess states quickly, and it is natural that the acceleration by 0-excess states of lattice reduction for WELL generators is not significant.

In contrast, let us choose some 0-excess initial states for Mersenne Twisters. Its state representation is very sparse. It is observed through the experiment that the state representation continues to be rather sparse even after a considerable number of pivot reductions. This results in a high probability to have a smaller norm vector after a pivot reduction (not the decrease of the pivot index). Consequently, the lattice reductions are significantly accelerated.

5. Conclusions

We propose PIS method for computing all $k(v)$, $w \geq v \geq 1$, which is an improvement of SIS in [3]. Our approach is to apply an efficient lattice reduction algorithm by Mulders and Storjohann with the triangular technique by Wang et al., and it is shown that this lowers the magnitude of computational complexity with respect to the word size w. The numerical experiments confirm that our improvement is practically effective, especially for large w. As another direction, use of 0-excess initial states significantly accelerates the lattice reductions in the case of Mersenne Twisters. This method is simple but very effective when we search for good $\mathbf{F}_2$-linear output functions for Mersenne Twisters (i.e., tempering in [21,4]), which is in particular useful for dynamic parameter searches [1].

Acknowledgments

The author wishes to express his gratitude to Professor Makoto Matsumoto at the University of Tokyo, for continuous support and encouragement. The author is greatly indebted to Dr. Mutsuo Saito at Hiroshima University for giving invaluable comments, especially, for pointing out an application to Dynamic Creator. The author is also grateful to the anonymous referees for many useful comments. This work was partially supported by Grant-in-Aid for JSPS Fellows 21·4427 and JSPS Core-to-Core Program 18005.

References

[1] M. Matsumoto, T. Nishimura, Dynamic creation of pseudorandom number generators, in: H. Niederreiter, J. Spanier (Eds.), Monte Carlo and Quasi-Monte Carlo Methods 1998, Springer-Verlag, Berlin, 2000, pp. 56–69.
[2] T. Mulders, A. Storjohann, On lattice reduction for polynomial matrices, J. Symbolic Comput. 35 (2003) 377–401.
[3] S. Harase, M. Matsumoto, M. Saito, Fast lattice reduction for $\mathbf{F}_2$-linear pseudorandom number generators, Math. Comp. 80 (2011) 395–407.
[4] M. Matsumoto, T. Nishimura, Mersenne Twister: a 623-dimensionally equidistributed uniform pseudo-random number generator, ACM Trans. Model. Comput. Simul. 8 (1998) 3–30.
[5] T. Nishimura, Tables of 64-bit Mersenne Twisters, ACM Trans. Model. Comput. Simul. 10 (2000) 348–357.
[6] F. Panneton, P. L'Ecuyer, M. Matsumoto, Improved long-period generators based on linear recurrences modulo 2, ACM Trans. Math. Software 32 (2006) 1–16.
[7] P. L'Ecuyer, F. Panneton, $\mathbb{F}_2$-linear random number generators, in: C. Alexopoulos, D. Goldsman, J.R. Wilson (Eds.), Advancing the Frontiers of Simulation: A Festschrift in Honor of George Samuel Fishman, Springer-Verlag, 2009, pp. 169–193.
[8] M. Matsumoto, M. Saito, H. Haramoto, T. Nishimura, Pseudorandom number generation: impossibility and compromise, J. Univ. Comput. Sci. 12 (2006) 672–690.
[9] M. Fushimi, S. Tezuka, The *k*-distribution of generalized feedback shift register pseudorandom numbers, Commun. ACM 26 (1983) 516–523.
[10] R. Couture, P. L'Ecuyer, S. Tezuka, On the distribution of *k*-dimensional vectors for simple and combined Tausworthe sequences, Math. Comp. 60 (1993) 749–761.
[11] S. Tezuka, The *k*-dimensional distribution of combined GFSR sequences, Math. Comp. 62 (1994) 809–817.
[12] R. Couture, P. L'Ecuyer, Lattice computations for random numbers, Math. Comp. 69 (2000) 757–765.
[13] A.K. Lenstra, Factoring multivariate polynomials over finite fields, J. Comput. System Sci. 30 (1985) 235–248.
[14] M. Saito, M. Matsumoto, SIMD-oriented fast Mersenne Twister: a 128-bit pseudorandom number generator, in: A. Keller, S. Heinrich, H. Niederreiter (Eds.), Monte Carlo and Quasi-Monte Carlo Methods 2006, Springer-Verlag, Berlin, 2008, pp. 607–622.
[15] L.-P. Wang, Y.-F. Zhu, D.-Y. Pei, On the lattice basis reduction multisequence synthesis algorithm, IEEE Trans. Inform. Theory 50 (2004) 2905–2910.
[16] K. Mahler, An analogue to Minkowski's geometry of numbers in a field of series, Ann. of Math. 42 (1941) 488–522.
[17] K. Mahler, On a theorem in the geometry of numbers in a space of Laurent series, J. Number Theory 17 (1983) 403–416.
[18] W.M. Schmidt, Construction and estimation of bases in function fields, J. Number Theory 39 (1991) 181–224.
[19] S. Paulus, Lattice basis reduction in function fields, in: P.J. Buhler (Ed.), Algorithmic Number Theory, in: Lecture Notes in Computer Science, vol. 1423, Springer-Verlag, Berlin, 1998, pp. 567–575.
[20] S. Harase, Maximally equidistributed pseudorandom number generators via linear output transformations, Math. Comput. Simulation 79 (2009) 1512–1519.
[21] M. Matsumoto, Y. Kurita, Twisted GFSR generators II, ACM Trans. Model. Comput. Simul. 4 (1994) 254–266.