Fast lattice reduction algorithms for optimizing $\mathbf{F}_2$-linear pseudorandom number generators

($\mathbf{F}_2$-線形擬似乱数発生法の最適化のための
高速格子簡約アルゴリズム)

原瀬　晋

Contents

Part I
Fast lattice reduction for $\mathbf{F}_2$-linear pseudorandom number generators

1 Introduction

Let $\mathbf{F}_2 := \{0, 1\}$ denote the two element field. Sequence generators based on $\mathbf{F}_2$-linear recursion are widely used in practical applications, in particular as pseudorandom number generators. Among the quality criteria of the generators, the notion of *the dimension of equidistribution with v-bit accuracy* is widely used as a most informative criterion for the higher dimensional uniformness of the distribution of the sequence (see [6]). When the state space is large, the computation of these dimensions is time consuming, and at the designing stage of the generator, it becomes a bottleneck in finding good parameters. Couture, L'Ecuyer and Tezuka [2] introduced a lattice basis reduction method to compute these dimensions, over the formal power series field $\mathbf{F}_2((t^{-1}))$. Couture and L'Ecuyer [1] improved Tezuka's resolution-wise lattice method [21] by using the dual lattice. The aim of this part is to propose a simpler and more efficient method. In §2, we briefly recall the notion of $\mathbf{F}_2$-linear generators and computation of the dimensions of equidistribution using lattices. In §3, we introduce a method to compute a reduced basis from a generating set, a method to compute the dimension of equidistribution with v-bit accuracy for $v = w, w-1, w-2, \ldots$ inductively in this order (w intended for the word size of the machine), and an efficient representation of the lattice elements (and operations on them) in terms of the state space. We give the computational complexity of the proposed method in §4. We compare the speeds of the state representation method and the dual basis method [1] in §5 using a C++ implementation.

2 Linear generator and lattice method

2.1 Dimension of equidistribution

We recall basic materials, see [6] and its references for the original definitions. An $\mathbf{F}_2$-linear sequence generator consists of a state space $S = \mathbf{F}_2^p$, an $\mathbf{F}_2$-linear state transition function $f : S \to S$, 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). Once an initial state $s_0 \in S$ is given, the generator computes the next state by the recursion $s_{i+1} = f(s_i)$ $(i = 0, 1, 2, \ldots)$ every time unit, and the output sequence is given by $o(s_0), o(s_1), o(s_2), \ldots \in O$. Throughout this part, $P(t)$ denotes the characteristic polynomial of f.

The dimension of equidistribution $k(v)$ of such a generator is defined as follows. We identify the output set $\mathbf{F}_2^w$ with the set of unsigned w-bit binary integers. Let us consider the most significant v bits (v MSBs) in the outputs. We regard this *to consider the output with v-bit accuracy.* This amounts to consider the composition $o_v : S \xrightarrow{o} \mathbf{F}_2^w \to \mathbf{F}_2^v$, where the latter map denotes taking the v MSBs. Define the k-tuple output function for any $k \geq 0$ by

$$o_v^{(k)} : S \to (\mathbf{F}_2^v)^k, \quad s_0 \mapsto (o_v(s_0), o_v(f(s_0)), \ldots, o_v(f^{k-1}(s_0))),$$

namely, $o_v^{(k)}(s_0)$ is the consecutive k-tuple of the outputs from the state s_0.

Definition 2.1. If $o_v^{(k)} : S \to (\mathbf{F}_2^v)^k$ is surjective, then the generator is said to be k-dimensionally equidistributed with v-bit accuracy. The largest value of k with this property is called the dimension of the equidistribution with v-bit accuracy, denoted by $k(v)$.

Since $o_v^{(k)}$ is linear, k-dimensional equidistribution means that every element in $(\mathbf{F}_2^v)^k$ occurs with the same probability, when the initial state s_0 is uniformly distributed over the state space. If the generator has the maximal period $2^p - 1$, then this amounts to saying that every kv-bit pattern occurs as consecutive overlapping k-tuples of v-bit integers equally often for the whole period, except the all-zero pattern which occurs once less often.

The larger $k(v)$ for each of $1 \leq v \leq w$ is desirable. By comparing the dimensions of S and $(\mathbf{F}_2^v)^k$, we have a trivial bound $p = \dim(S) \geq kv$, and hence $k(v) \leq \lfloor p/v \rfloor$. If the equality holds, then the generator is said to be *maximally equidistributed.*

We may compute $k(v)$ by checking the surjectivity by linear algebra [3]. For a large p, Couture et al. [2] and Tezuka [21] proposed much faster algorithms based on lattice structures over power series.

2.2 Lattice structure

We briefly recall the above-mentioned lattice method. 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 $\gamma = (\alpha_1, \alpha_2, \ldots, \alpha_v) \in K^v$, we define $||\gamma|| := \max_{1 \leq i \leq v} |\alpha_i|$. Note that $|\alpha|$ and $||\gamma||$ are often negative integers. Let $\alpha_i = \sum_{j=j_0}^{\infty} a_{i,j} t^{-j} \in K$. For $\gamma \neq 0$, we define its *coefficient vector at the leading term* $\pi(\gamma) \in \mathbf{F}_2^v$ by

$$\pi(\gamma) := (a_{1,-||\gamma||}, a_{2,-||\gamma||}, \ldots, a_{v,-||\gamma||}), \text{ so that } \gamma = \pi(\gamma) t^{||\gamma||} + \text{lower degree terms in } t.$$

A subset $L \subset K^v$ is called an $\mathbf{F}_2[t]$-*lattice* if it is the set of linear combination of $\omega_1, \omega_2, \ldots, \omega_v$ with coefficients in $\mathbf{F}_2[t]$, where $\omega_1, \ldots, \omega_v$ are linearly independent over K. Such a set of vectors is called a basis of L.

Theorem 2.2 (Lemma 1 of [9]). Let $\omega_1, \ldots, \omega_v$ be the points in an $\mathbf{F}_2[t]$-lattice $L \subset K^v$ with the following properties:

(1) ω_1 is a shortest nonzero vector in L;

(2) for $i = 2, \ldots, v$, ω_i is a shortest vector among the set of vectors ω in L such that $\omega_1, \ldots, \omega_{i-1}, \omega$ are linearly independent over K.

Then $\omega_1, \ldots, \omega_v$ form a basis of L.

Such a basis is called a *reduced basis* for L. It is not unique, but the numbers $\nu_i := ||\omega_i||$ are invariants of the lattice, called *successive minima* (see [8]). Let us consider an $\mathbf{F}_2$-linear generator. For a v-bit output sequence from an initial state $s_0 \in S$, let χ_v denote a (v-dimensional vector-valued) generating function in K^v:

$$\chi_v(s_0) := \sum_{j=0}^{\infty} o_v(f^j(s_0))t^{-1-j} = o_v(s_0)t^{-1} + o_v(s_1)t^{-2} + \cdots \in K^v. \tag{2.1}$$

This gives an $\mathbf{F}_2$-linear map $\chi_v : S \to K^v$. We define a lattice Λ_v in K^v as the set of $\mathbf{F}_2[t]$-linear combinations of $\chi_v(s_0)$ and the unit vectors, namely, the $\mathbf{F}_2[t]$-linear span

$$\Lambda_v := \langle \chi_v(s_0), e_1, e_2, \ldots, e_v \rangle_{\mathbf{F}_2[t]}, \tag{2.2}$$

where e_i is the vector whose i-th component is 1 and 0 for the other components ($i = 1, \ldots, v$). The vectors $\chi_v(s_0), e_1, e_2, \ldots, e_v$ form a generating set of Λ_v over $\mathbf{F}_2[t]$, but they are K-linearly dependent because they form a set of cardinality $v+1$. Still, these $v+1$ vectors generate a lattice: after multiplying each vector by the characteristic polynomial $P(t)$, every coordinate becomes a polynomial, and thus there is a basis consisting of v vectors.

Theorem 2.3 ([2, 21]). Assume that $P(t)$ is irreducible. Take nonzero $s_0 \in S$. Then, $k(v) = -\nu_v$ holds, where ν_v is the v-th successive minimum of the lattice Λ_v in K^v.

In [1], to obtain a reduced basis, the authors used the Lenstra reduction algorithm [7], which requires a basis of Λ_v as an initial input. Since $P(t)\chi_v(s_0) \in \mathbf{F}_2[t]^v$, one can define polynomials $(g_1(t), g_2(t), \ldots, g_v(t)) := P(t)\chi_v(s_0)$. Let $g_1(t)^{-1}$ denote a polynomial which is a multiplicative inverse to $g_1(t)$ modulo $P(t)$ (which exists if the MSB of the sequence is not constantly 0), and define a vector with polynomial components

$$\Psi := (g_1(t)^{-1} \cdot P(t)\chi_v(s_0)) \bmod P(t).$$

The first component of Ψ is 1, and the vectors $\Psi/P(t), e_2 \ldots, e_v$ form a basis of Λ_v. In applying the Lenstra basis reduction, to avoid the infinite formal power series

$\Psi/P(t)$, we multiply this basis by $P(t)$ to reduce to the polynomial computation. We apply the Lenstra's reduction to the polynomial vectors

$$(1, \bar{g}_2(t), \ldots, \bar{g}_v(t)), (0, P(t), \ldots, 0), \ldots, (0, \ldots, 0, P(t)) \in \mathbf{F}_2[t]^v, \tag{2.3}$$

where $\bar{g}_j(t) := g_1(t)^{-1} g_j(t) \bmod P(t)$ $(2 \leq j \leq v)$.

Later we use the following.

Lemma 2.4. Let $\nu_1, \ldots, \nu_v$ be the successive minima of Λ_v. We have

$$-\dim(S) = \sum_{i=1}^{v} \nu_i.$$

Proof. From (25) in [8] (or (3) in [1]), the lemma follows. □

2.3 Dual lattice

For a lattice $L \subset K^v$, its *dual lattice* L' is defined by

$$L' := \{h' \in K^v \mid h \cdot h' \in \mathbf{F}_2[t], \text{ for all } h \in L\},$$

where $h \cdot h' = \sum_{j=1}^{l} h_j(t) \cdot h'_j(t)$ (the scalar product) for $h = (h_1(t), \ldots, h_v(t))$ and $h' = (h'_1(t), \ldots, h'_v(t))$. The vectors

$$(P(t), 0, \ldots, 0), (-\bar{g}_2(t), 1, \ldots, 0, 0), \ldots, (-\bar{g}_v(t), 0, \cdots, 0, 1) \tag{2.4}$$

form (the so-called dual) basis of the dual lattice Λ'_v. The next theorem reduces the computation of successive minima of Λ_v to those of the dual.

Theorem 2.5 ([1]). Let $\nu_1, \nu_2, \ldots, \nu_v$ be the successive minima of Λ_v, and $\nu'_1, \nu'_2, \ldots, \nu'_v$ the successive minima of Λ'_v. We have, for $i = 1, 2, \ldots v$,

$$\nu_i + \nu'_{v-i+1} = 0.$$

A big advantage of using the dual in [1] is that we can use the reduced basis of Λ'_{v-1} to compute that of Λ'_v.

Assume $1 < v \leq w$. Let $\iota : K^{v-1} \to K^v$ be an inclusion by supplementing 0 at the vth coordinate, and $\rho : K^v \to K^{v-1}$ be the projection by deletion of the vth coordinate.

Theorem 2.6 ([1]). For $1 < v \leq w$, we have

(1) $\rho(\Lambda_v) = \Lambda_{v-1}$.

(2) $\Lambda'_v = \iota(\Lambda'_{v-1}) \oplus \langle(-\bar{g}_v(t), 0, \cdots, 0, 1)\rangle_{\mathbf{F}_2[t]}$,

where $\langle(-\bar{g}_v(t), 0, \cdots, 0, 1)\rangle_{\mathbf{F}_2[t]}$ denotes the space spanned by a single vector.

The first claim follows from the definition of Λ_v. The second follows from (2.4). Thus, for the dual lattice Λ'_v, we may choose its lattice basis as the union of a reduced basis of Λ'_{v-1} and the vector $(-\bar{g}_v(t), 0, \cdots, 0, 1)$. Consequently, if we compute reduced bases of $\Lambda'_2, \Lambda'_3, \ldots$ in this order, then we can use a reduced basis of Λ'_{v-1} in computing that of Λ'_v. Computational complexity given in [1] shows a significant advantage of this method, which we call *dual lattice method.*

3 Main result

3.1 Schmidt's generating set reduction

The Lenstra basis reduction requires a basis of the lattice. Although it is not difficult to obtain a basis from a set of generating vectors (cf. [5]), there is an even simpler reduction algorithm by Schmidt [20, P.200], which can be easily generalized to an algorithm to obtain a reduced basis from a generating set. We describe this generalized version, which we call *Schmidt's generating set reduction* (SGR).

procedure *Schmidt's generating set reduction*
input : a generating set $\omega_1, \omega_2, \ldots, \omega_m$ which spans a lattice L over $\mathbf{F}_2[t]$.
output : a reduced basis $\omega_1, \omega_2, \ldots, \omega_v \in L$.
begin
while $\pi(\omega_1), \pi(\omega_2), \ldots, \pi(\omega_m)$ are linearly dependent over $\mathbf{F}_2$ do
(reduction step)
Find a vector $(\alpha_1, \alpha_2, \ldots, \alpha_m) \in \mathbf{F}_2^v$ such that $\sum_{i=1}^m \alpha_i \pi(\omega_i) = (0, \ldots, 0)$.
Find an integer $i_{\max}$ such that $||\omega_{i_{\max}}|| = \max\{||\omega_i|| \mid 1 \leq i \leq m, \alpha_i \neq 0\}$.
Set $\omega_{i_{\max}} \leftarrow \sum_{i=1}^m \alpha_i t^{-||\omega_i|| + ||\omega_{i_{\max}}||} \omega_i$.
If $\omega_{i_{\max}} = 0$ then swap $\omega_{i_{\max}}$ and ω_m, and set $m \leftarrow m - 1$.

end while

Renumber $\omega_1, \omega_2, \ldots, \omega_m$ in such a way that $||\omega_1|| \leq ||\omega_2|| \leq \ldots \leq ||\omega_m||$.

end

Each reduction step decreases $\sum_i ||\omega_i||$, and since there is a shortest vector in a lattice, the algorithm terminates. Then, the number m of vectors is reduced to v, and $\pi(\omega_1), \ldots, \pi(\omega_v)$ are linearly independent. Such a basis is called a *reduced basis* in [20, P.199], and its equivalence to that in Theorem 2.2 follows from the uniqueness of their length ([20, P.201]).

Using SGR, we can use ρ and (1) in Theorem 2.6 to obtain a reduced basis of Λ_v from that of Λ_{v+1}: if $\omega_1, \ldots, \omega_{v+1}$ is a reduced basis of Λ_{v+1}, we obtain a generating set $\rho(\omega_1), \ldots, \rho(\omega_{v+1})$ of Λ_v, hence we may apply SGR. Since $\rho(\omega_1), \ldots, \rho(\omega_{v+1})$ are short, this lowers computational complexity significantly when one computes all $k(v)$ for $1 \leq v \leq w$ (see §4). We call this method *inductive projection*. Note that this method computes $k(w), k(w-1), \ldots, k(1)$ in this order, which is converse to the standard techniques (e.g., [1] [5]).

SGR in the inductive projection is proved to terminate when one vector is eliminated, as follows.

Theorem 3.1. Let $\omega_1, \ldots, \omega_{v+1}$ be a generating set of an $\mathbf{F}_2[t]$-lattice $L \subset K^v$. Suppose that $\pi(\omega_1), \ldots, \pi(\omega_{v+1})$ has rank v. When we apply the SGR algorithm to this generating set, then it terminates when the number of the vectors becomes v, namely, when a vector is reduced to zero.

Let $\omega'_1, \ldots, \omega'_v$ be the obtained reduced basis. The number of reduction steps in SGR required before the termination is bounded from above by

$$\sum_{i=1}^{v+1} ||\omega_i|| - \sum_{i=1}^{v} ||\omega'_i|| - ||\omega^{\text{last}}|| + 1, \tag{3.1}$$

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

In particular, let L be Λ_v in (2.2). Then, the value of (3.1) is bounded from above by $-||\omega^{\text{last}}|| + 1$, if $\omega_1, \ldots, \omega_{v+1}$ are the image by ρ of a reduced basis of the lattice Λ_{v+1} and the characteristic polynomial $P(t)$ is irreducible.

Proof. In SGR, in one reduction step, one vector is reduced by subtracting an $\mathbf{F}_2[t]$-linear combination of the other v vectors. Looking at the coefficient vectors at the leading term, this amounts to eliminating one $\mathbf{F}_2$ vector by subtracting an $\mathbf{F}_2$-linear combination of the other v vectors. The coefficients of the leading term of the reduced vector changes, but the other v vectors do not change. The reducibility implies that even if we throw away the reduced vector, still the rank of the coefficient vectors at the leading terms does not decrease. Thus, the rank of vectors $\pi(\omega_1), \ldots, \pi(\omega_{v+1})$ is always v. Consequently, if the reduced vector becomes zero, then the other v vectors have rank v at the leading terms, which means the termination.

In each reduction step, the sum $\sum_{i=1}^{v+1} ||\omega_i||$ is decreased at least by one. When one vector is reduced to zero, then this value becomes $-\infty$. We look at the last step of reduction. There is a vector ω^{last} that is reduced to zero, while the other v vectors are unchanged and become the reduced basis. At this stage, the above sum is $||\omega^{\text{last}}|| + \sum_{i=1}^{v} ||\omega_i'||$. Hence, the number of steps is bounded by their difference + 1, namely (3.1).

If the lattices are from an $\mathbf{F}_2$-linear generator, then $\sum_{i=1}^{v} ||\omega_i'|| = -\dim(S)$ holds by Lemma 2.4. If $\tilde{\omega}_i$ $(i = 1, \ldots, v+1)$ is a reduced basis of Λ_{v+1}, then

$$\sum_{i=1}^{v+1} ||\rho(\tilde{\omega}_i)|| \leq \sum_{i=1}^{v+1} ||\tilde{\omega}_i|| = -\dim(S),$$

hence the result. □

We check that in inductive projection, SGR satisfies the condition in Theorem 3.1. For $v = w$, we apply SGR to a generating set (2.2), whose cardinality is $v + 1$. In the induction step, we have $v + 1$ generators projected from a reduced basis, and SGR reduces them to v generators. In both cases, the coefficient vectors of the leading terms of the generating set have rank v as $\mathbf{F}_2$ vectors. Namely, $\pi(\chi_w(s_0)), \pi(e_1), \pi(e_2), \ldots, \pi(e_w)$ have rank w. In the induction step, let $\omega_1, \ldots, \omega_{v+1}$ be a reduced basis of Λ_{v+1}. Then $\pi(\rho(\omega_1)), \ldots, \pi(\rho(\omega_{v+1}))$ have rank v, since $\pi(\omega_1), \ldots, \pi(\omega_{v+1})$ have rank $v+1$ and the rank of $\pi(\rho(\omega_i))$'s is at least the rank of $\rho(\pi(\omega_1)), \ldots, \rho(\pi(\omega_{v+1}))$, which is v, and consequently the rank must be v. Thus, both cases satisfy the condition of Theorem 3.1.

Corollary 3.2. Under a heuristic assumption that on average $||\omega^{\text{last}}|| \geq -\dim(S)/v$ holds, the average number of the reduction steps in SGR to obtain a reduced basis of Λ_v from that of Λ_{v+1} is bounded from above by $\dim(S)/v + 1$.

Proof. The assumption is that, $||\omega^{\text{last}}||$ in the proof of the theorem is on average larger than or equal to the average of $||\omega_1'||, \ldots, ||\omega_v'||$. This is justified by the fact that ω^{last} is reduced by ω_i', hence has on average larger norm than the average norm of $||\omega_i'||$, which is $-\dim(S)/v$ by Lemma 2.4. □

Note that ω^{last} is reduced often by using the longest vector or the second longest vector among ω_i', hence the above bound $\dim(S)/v + 1$ seems over-estimated: SGR tends to stop in a smaller number of steps, which agrees with our experiments.

Remark 3.3. There is a modified Lenstra reduction algorithm [18] applicable to a generating set of a lattice, but its efficiency seems comparable to SGR. Wang and Zhu [23] and Wang, Zhu and Pei [24] applied SGR to compute the linear complexity of a multisequence. A more informative complexity, based on the successive minima obtained using SGR, is given by Wang and Niederreiter [22].

3.2 State representation

Another merit of the dual lattice method in [1] is that the space complexity is reduced. If we apply SGR to the generating set (2.2) polynomialized by multiplying by $P(t)$, each vector has components being polynomials of degree smaller or equal to $\deg(P(t)) = \dim(S)$. Thus, one vector requires $\dim(S) \times v$ bits of memory, and the generating set consumes $v(v+1)\dim(S)$ bits. We need to start from $v = w$, which costs a lot if $\dim(S)$ is large. On the contrary, in the dual lattice method, for $v = 1$ we have no reduction step (and $k(1) = \dim(S)$), and for $v = 2$ we need $2\dim(S)$ bits of memory for each of two vectors, and after a basis reduction, the components of the vectors in a reduced basis have degree $\dim(S)/2$ on average, thus $\dim(S)$ bits for one vector and $2\dim(S)$ bits for a reduced basis. In the same way, the reduced basis for Λ_v' consumes $v\dim(S)$ bits, which improves on $v(v+1)\dim(S)$ for the original lattice.

Instead of using the dual lattice, we propose a method to represent a vector in the lattice Λ_v by a state, which we call the *state representation*. Since one vector consumes only $\dim(S)$ bits of memory, memory efficiency is comparable to the dual lattice method (or better, since we need no assumption on the reducedness). Recall the map $\chi_v : S \to K^v$ defined in (2.1). Note that $K = \mathbf{F}_2[t] \oplus (\mathbf{F}_2[[t^{-1}]] \cdot t^{-1})$ as an $\mathbf{F}_2$-vector space, since any element of K is a sum of its polynomial part (namely, a linear combination of t^j with $j \geq 0$) and its fractional part (namely, an infinite linear combination of t^j with $j < 0$) in a unique way. Hence we have

$$K^v = \mathbf{F}_2[t]^v \oplus (\mathbf{F}_2[[t^{-1}]] \cdot t^{-1})^v.$$

The first direct summand is called the *polynomial part*, and the second is the *fractional part* which we denote by F^v. Let $F(\Lambda_v)$ be the fractional part $F^v \cap \Lambda_v$. Since Λ_v contains $\mathbf{F}_2[t]^v$, the polynomial part of any element in the lattice is in Λ_v, and so is the fractional part, namely:

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

as an $\mathbf{F}_2$-vector space. Assume that the characteristic polynomial $P(t)$ is irreducible. Then, the image of χ_v lies in $F(\Lambda_v)$. Note also that $\Lambda_v/(\mathbf{F}_2[t]^v)$ is an $\mathbf{F}_2[t]$-module.

Lemma 3.4. If the characteristic polynomial $P(t)$ is irreducible and χ_v is nonzero,

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

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

If $P(t)$ is irreducible, then this lemma implies that the fractional part of a lattice element has a unique representation by a state in S, and the sum and multiplication by t for lattice elements can be computed by those for the corresponding states. Thus, we can implement lattice reduction algorithms using operations on S. This is a key to reduce the space and time complexities by the state representation.

Proof. The action of t on $s \in S$ is defined by $t \cdot s := f(s)$. Since χ_v is linear, to show homomorphy, it suffices to show that

$$\chi_v(f(s_0)) \equiv t \cdot \chi_v(s_0) \bmod \mathbf{F}_2[t]^v.$$

But by the definition (2.1), $\chi_v(f(s_0)) = \sum_{j=0}^{\infty} o_v(f^{j+1}(s_0))t^{-1-j}$ and $t \cdot \chi_v(s_0) = \sum_{j=0}^{\infty} o_v(f^j(s_0))t^{-j}$, hence their difference is an $\mathbf{F}_2$ vector $o_v(s_0) \in \mathbf{F}_2[t]^v$.

We show isomorphy. Since $P(t)$ trivially acts on S, S is a $k := \mathbf{F}_2[t]/(P(t))$-module with k being a field. Since $\dim(S) = \deg(P(t))$, S is a one-dimensional k-vector space. On the other hand, $\Lambda_v/(\mathbf{F}_2[t]^v)$ is also a k-vector space, which is generated by a single element $\chi_v(s_0)$. Thus, χ_v is a k-linear map between two one-dimensional spaces, so χ_v is an isomorphism if nonzero. □

From now on, we assume irreducibility of $P(t)$. By the above lemma, we can represent the fractional part of an element of Λ_v as $\chi_v(s)$ in a unique way. Thus, any element of Λ_v has a unique representation as $poly + \chi_v(s)$ with polynomial part *poly* and the fractional part $\chi_v(s)$.

Definition 3.5. A pair of a polynomial vector *poly* and a state $s \in S$ is called the state representation of $poly + \chi_v(s) \in \Lambda_v$.

The addition of two representations is given by adding their polynomial parts, and by adding the states in the state space. Multiplication by t is given by

$$t(poly + \chi_v(s)) = (t \cdot poly + o_v(s)) + \chi_v(f(s)).$$

In applying SGR to (2.2), note that $e_1, \ldots, e_v$ are not in the image of χ_v, but once such a vector is reduced, then the result has only fractional part, having a representation $\chi_v(s)$. Thus, most computation can be done inside the state space.

There is a slightly improved version. In a lattice-reduction procedure, we need to compute the norm and the leading term of $\chi_v(s)$. A direct method is to compute $o_v(s), o_v(f(s)), o_v(f^2(s)), \ldots$ in this order, until one gets a nonzero vector. If $o_v(f^j(s))$ is the first nonzero vector, then this vector is the leading coefficient $\pi(\chi_v(s))$ and $||\chi_v(s)|| = -j - 1$ holds. This method is time-consuming if the norm is small, which is the case for the last steps of the reduction.

To avoid this, we adopted the following representation. Let s be a nonzero state that represents a lattice element $\chi_v(s)$. If $||\chi_v(s)|| = -n$, then we keep the pair $(n-1, f^{n-1}(s))$ as a representation of $\chi_v(s)$, instead of s. More precisely, consider the set $\tilde{S} := \{(m, s') \in \mathbf{Z} \times S \mid m \geq 0, ||\chi_v(f^{-m}(s'))|| = -m - 1\}$. The

above mapping $s \mapsto (n-1, f^{n-1}(s)) \in \tilde{S}$ gives the inverse to the mapping $\phi : \tilde{S} - \{0\} \to S - \{0\}; (m, s') \mapsto f^{-m}(s')$. Through this bijection, we use elements in $\tilde{S}$ as representations of lattice elements. It is easy to check that $||\chi_v(\phi(m, s'))|| = -m-1$ and $\pi(\chi_v(\phi(m, s'))) = o_v(s')$, so there is no need to search for the first nonzero term. One can check that in the reduction steps in SGR, we need only the norm and the leading term, hence this representation works. We leave it as an exercise to detail how to compute the sum and the multiplication by t in $\tilde{S}$.

We propose a combination of SGR, inductive projection, and state representation for computing all $k(v)$'s, which we call *SIS* for short. Thus, first SIS computes a reduced basis of Λ_w using SGR with state representation. By taking the projection, SIS computes a generating set of Λ_{w-1}, then reduces it to a reduced basis by SGR with state representation. SIS inductively computes reduced bases of Λ_w, Λ_{w-1}, Λ_{w-2}, ..., Λ_2, in this order (inductive projection). Theorem 2.3 gives $k(v)$ for $v = w, w-1, \ldots, 2$.

4 Computational complexities

In a practical $\mathbf{F}_2$-linear generator, f and o_v can be computed by a few operations, often independently of the size of the state space, which we assume is negligible from the total cost of the computation.

Theorem 4.1. The average number of bit operations to obtain the reduced basis by the SGR from the generating set in (2.2) is bounded by $(v+1)\dim(S)^2 + (v^3 + v^2)\dim(S)$, when using the state representation.

Proof. One step of the reduction in SGR consists of v^3 bit operations for Gaussian elimination to find a linear relation among $v+1$ $\mathbf{F}_2$ vectors, and v additions to reduce a vector. Each addition requires $\dim(S)$ bit operations in the state representation. Thus, one reduction step has $v\dim(S) + v^3$ bit operations. By Theorem 3.1 and Corollary 3.2, on average, the number of reduction steps does not exceed

$$||\chi_v(s_0)|| + ||e_1|| + \cdots + ||e_v|| - \sum_{i=1}^{v} ||\omega_i'|| + \dim(S)/v + 1.$$

From Lemma 2.4, $||\chi_v(s_0)|| \leq -1$ and $||e_i|| = 0$, it follows that this bound is equal to $\dim(S)(1 + 1/v)$. By multiplying, we have a complexity upper bound $\dim(S)(1+1/v)(v\dim(S)+v^3) = (v+1)\dim(S)^2 + (v^3+v^2)\dim(S)$. □

Theorem 4.2. The average number of bit operations for SGR algorithm to obtain a reduced basis of Λ_v from that of Λ_{v+1} has an upper bound $\dim(S)^2+(v^2+v)\dim(S)+v^3$, when we use the state representation.

Proof. By Corollary 3.2, SGR needs at most $\dim(S)/v+1$ reduction steps on average. As in the proof of Theorem 4.1, each reduction step has $v\dim(S)+v^3$ bit operations, hence we have $(\dim(S)/v+1)(v\dim(S)+v^3) = \dim(S)^2+(v^2+v)\dim(S)+v^3$. □

These theorems give an upper bound of computational complexity of SIS. Theorem 4.1 implies that the first step of SIS computing requires at most $w\dim(S)^2 + w^3\dim(S)$ bit operations. At the step of the inductive projection from Λ_v to Λ_{v-1} in SIS, Theorem 4.2 gives an upper bound of the complexity $\dim(S)^2+v^2\dim(S)+v^3$. By summing for $v = w-1, w-2, \ldots, 1$, $w\dim(S)^2 + \frac{1}{3}w^3\dim(S) + \frac{1}{4}w^4$ bit operations will suffice to compute the other $w-1$ values $k(w-1), k(w-2), \ldots, k(1)$. By summing, we have:

Theorem 4.3. SIS requires at most $2w\dim(S)^2+\frac{4}{3}w^3\dim(S)+\frac{1}{4}w^4$ bit operations to compute all $k(v)$, $w \geq v \geq 1$.

We compare this complexity with the following result for the dual lattice method described in §2.3. A lattice Λ_v is said to be *regular* if the minimum and the maximum of its successive minima have a difference of at most 1.

Theorem 4.4. [1, Theorem 2 and §4] Suppose that Λ'_{v-1} is regular. The number of bit operations for computing a reduced basis of Λ'_v from that of Λ'_{v-1} does not exceed

$$Cv(\dim(S)+v-1)^2, \quad v \geq 2,$$

where C is an absolute constant.

The number of bit operations for computing all $k(1), \ldots, k(w)$ does not exceed $C'\frac{w^2}{2}(\dim(S)+w-1)^2$ for an absolute constant C', under the regularity assumption for each lattice Λ'_v.

The comparison of the orders show that our SIS method is expected to be more efficient than this by a factor of w. (As pointed out by a referee, strictly speaking, these are only upper bounds and do not compare the efficiency). Note that there are differences in the estimation: our estimation does not assume the regularity on the lattices, but does depend on a heuristic argument on the average. Actually, the regularity implies that our estimation in Corollary 3.2 plus 1 gives an upper bound as a worst case analysis, since $||\omega^{\text{last}}|| \geq \nu_1 \geq -\dim(S)/v - 1$ if regular.

Remark 4.5. We are also interested in whether SGR is more efficient than Lenstra's algorithm or not, when used for dual lattice. According to our experiments, the answer is yes, but not that much, see the next section. We implemented a version of the dual lattice method, replacing the Lenstra algorithm with SGR.

There is one caution when SGR is used with the dual lattice method: to keep the efficiency, we need a triangulation process, as stated below. In the dual lattice method, let $\omega_1, \ldots, \omega_{v-1}$ be the computed reduced basis of Λ'_{v-1}. Let B be the square matrix of size $v-1$ whose j-th column is ω_j. As explained after Theorem 2.6, the vector ${}^t(-\bar{g}_v(t), 0, \cdots, 0, 1)$ is reduced by using $\iota(\omega_1), \ldots, \iota(\omega_{v-1})$ (called the first phase, see [1, Proof of Theorem 2]), then the Lenstra algorithm is applied to obtain a reduced basis of Λ'_v (the second phase). Let $\pi(B)$ be the square matrix whose j-th column is $\pi(\omega_j)$. In the first phase, in one reduction step, a linear equation $\pi(B)x = y$ with coefficients in $\mathbf{F}_2$ is solved, until the vector becomes non-reducible by $\iota(\omega_1), \ldots, \iota(\omega_{v-1})$ (where y may change at every step). If $\pi(B)$ happens to be triangular, then solving these linear equations is efficient.

If B is obtained by Lenstra algorithm, then $\pi(B)$ is triangular. If B is obtained by SGR, $\pi(B)$ may be not triangular, but it is easy to transform B to another reduced basis B' with $\pi(B')$ being triangular. We use SGR with this triangulation procedure, then the dual lattice method with SGR is often faster than that with Lenstra algorithm.

5 Speed comparison

The following computer experiments compare our SIS method and the dual lattice method. We assume $w = 32$, and choose three $\mathbf{F}_2$-linear generators. We measured the CPU time for computing each $k(v)$ for $2 \leq v \leq 32$, by using the following three methods:

(1) SIS method (our proposal in § 3).

(2) the dual lattice method with

  (a) SGR algorithm applied as in Remark 4.5.

  (b) Lenstra reduction algorithm (the method proposed in [1]).

We implemented these three methods in C++. In the SIS method, we compute $k(32)$ from (2.2), and then $k(31), \ldots, k(2)$ inductively. In the dual lattice method, we need to compute the characteristic polynomial $P(t)$ by Berlekamp-Massey algorithm [10], and then transform (2.3) into (2.4) by arithmetic operations modulo $P(t)$. We refer to this phase as the *precomputation* for the dual lattice method. For polynomial arithmetic in the precomputation, we used the library NTL (`http://www.shoup.net/ntl`). We compute $k(2)$ by applying SGR algorithm (a), or Lenstra's algorithm (b), to the basis (2.4), and then inductively compute $k(3), \ldots, k(32)$. The tests were performed on a computer with 64-bit AMD-Athlon 64 3200+ CPU and 2.0 GB of memory, on a Linux operating system. The programs were compiled using gcc compiler version 4 with the -O2 optimization flag.

We applied these methods to WELL44497a', which is a maximally equidistributed version of WELL44497a [17] by improving its tempering (see [4]). The generator has $\dim(S) = 44497$. The lattice Λ_v turns out to be regular for every $2 \leq v \leq 32$, except for $v = 7$. Table 1 gives the CPU time (in seconds) for computing a reduced basis. As predicted from the computational complexities (Theorems 4.1 and 4.2), in the SIS method, the CPU time for computing $k(32)$ is comparable to the sum of all the rest of the computations. Note that the consumed time for $k(31)$ to $k(2)$ is almost the same, as predicted from Theorem 4.2. In the dual lattice methods, computation of $k(2)$ is fast, and computation time of $k(v)$ increases, roughly

proportional to v, as predicted from Theorem 4.4. In these experiments, SGR is a little faster than Lenstra's algorithm.

The comparison of the CPU time is in accordance with the ratio v between the complexity of our method and that of the dual lattice method, in inductive computation of $k(v)$. In total, our method is much faster.

We also compared the timings for two other $\mathbf{F}_2$-linear generators with $\dim(S) = 19937$, namely WELL19937a' (a maximally equidistributed version of WELL19937a [17] introduced in [4], Λ_v being regular except for $v = 6$) and Mersenne Twister MT19937 [12] whose lattices are far from being regular (whose total dimension defect [17] Δ is 6750). Table 2 lists the total CPU time (in seconds) to compute all $k(v)$ $(2 \le v \le 32)$ by the three methods for the three generators. The first line lists the total time for each method applied for WELL44497a', copied from the last line of Table 1. The experiments on WELL19937a' show the same tendency. In MT19937, Lenstra's algorithm is faster than SGR.

Table 1: The CPU time for computing $k(v)$ $(2 \leq v \leq 32)$ of WELL44497a' (in seconds). For the SIS methods, they are listed in descending order with respect to v, according to the order of computation.

SIS		dual lattice		
	SGR		SGR	Lenstra
		precom.	1.829	1.845
$k(32)$	1.996	$k(2)$	0.064	0.064
$k(31)$	0.059	$k(3)$	0.138	0.140
$k(30)$	0.058	$k(4)$	0.212	0.217
$k(29)$	0.058	$k(5)$	0.286	0.298
$k(28)$	0.059	$k(6)$	0.364	0.379
$k(27)$	0.057	$k(7)$	0.445	0.468
$k(26)$	0.057	$k(8)$	0.527	0.553
$k(25)$	0.057	$k(9)$	0.612	0.646
$k(24)$	0.058	$k(10)$	0.700	0.739
$k(23)$	0.056	$k(11)$	0.791	0.836
$k(22)$	0.057	$k(12)$	0.879	0.940
$k(21)$	0.057	$k(13)$	0.977	1.047
$k(20)$	0.057	$k(14)$	1.071	1.157
$k(19)$	0.056	$k(15)$	1.183	1.266
$k(18)$	0.057	$k(16)$	1.284	1.383
$k(17)$	0.058	$k(17)$	1.386	1.491
$k(16)$	0.058	$k(18)$	1.505	1.619
$k(15)$	0.051	$k(19)$	1.623	1.761
$k(14)$	0.052	$k(20)$	1.742	1.895
$k(13)$	0.053	$k(21)$	1.857	2.008
$k(12)$	0.053	$k(22)$	1.985	2.148
$k(11)$	0.054	$k(23)$	2.114	2.288
$k(10)$	0.050	$k(24)$	2.247	2.468
$k(9)$	0.051	$k(25)$	2.365	2.595
$k(8)$	0.052	$k(26)$	2.516	2.739
$k(7)$	0.048	$k(27)$	2.682	2.929
$k(6)$	0.050	$k(28)$	2.824	3.098
$k(5)$	0.049	$k(29)$	2.969	3.248
$k(4)$	0.049	$k(30)$	3.117	3.443
$k(3)$	0.048	$k(31)$	3.308	3.630
$k(2)$	0.048	$k(32)$	3.456	3.806
total	3.622	total	49.053	53.139

Table 2: The cumulative CPU-time (in seconds) for computation of all $k(v)$ ($2 \leq v \leq 32$) of three $\mathbf{F}_2$-linear generators, by the SIS method and the dual lattice methods. The number in () shows the pre-computation time. The column Δ shows the total dimension defect.

generators	SIS	dual lattice		Δ
		SGR	Lenstra	
WELL44497a'	3.622	49.053(1.829)	53.139(1.845)	0
WELL19937a'	0.939	12.360(0.398)	13.476(0.392)	0
MT19937	0.529	9.399(0.403)	5.654(0.408)	6750

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

6 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 [13]), since they have effective assessment via their dimensions of equidistribution, which are values assuring uniformity of high-dimensional distribution. This part proposes a fast algorithm to compute these dimensions, using a lattice reduction algorithm by Mulders and Storjohann [15] in the SIS method in Part I.

Let $\mathbf{F}_2 := \{0, 1\}$ be the two-element field, i.e., addition and multiplication are done modulo two. 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 [12, 16] and WELL generators

[17] 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 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 [6, 14] for details).

Definition 6.1 (Definition 2.1 of Part I). 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:

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

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:

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

Definition 6.2. 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 maximal period $2^p - 1$. Then, every state except 0 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 [3], 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 [2] and Tezuka [21] proposed much faster algorithms based on lattice structures over power series. Couture and L'Ecuyer [1] proposed an improvement by using the dual lattices and Lenstra's reduction algorithm [7]. In Part I, we 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 [16, 19]) because of presence of the terms of order w^3p and w^4.

In this part, we improve on SIS method by replacing Schmidt's lattice reduction with a more efficient lattice reduction algorithm based on [15, 24], 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 [17]). In fact, this initialization considerably accelerates the lattice computation for generators with sparse transition function, such as Mersenne Twisters.

In Section 7, we recall the lattice method for computing $k(v)$. In Section 8, we propose a new lattice reduction algorithm based on [15, 24], and analyze the

computational complexity. In Section 9, we report the timing with or without 0-excess initial states.

7 Lattice method

7.1 Lattice structure

We briefly recall the lattice method for computing $k(v)$. Let K denote the formal power series field $K := \mathbf{F}_2((t^{-1})) = \{\sum_{j=j_0}^{\infty} a_j t^{-j} \mid a_j \in \mathbf{F}_2, j_0 \in \mathbf{Z}\}$. 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 $\gamma = (\alpha_1, \alpha_2, \ldots, \alpha_v) \in K^v$, we define its norm by $||\gamma|| := \max_{1 \leq i \leq v} |\alpha_i|$. 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

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

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

A subset $L \subset K^v$ is said to be an $\mathbf{F}_2[t]$-*lattice* if there exist K-linear basis $\omega_1, \omega_2, \ldots, \omega_v$ of K^v such that L is their span over $\mathbf{F}_2[t]$, i.e., $L = \langle \omega_1, \omega_2, \ldots, \omega_v \rangle_{\mathbf{F}_2[t]}$. We call such a set of vectors a *basis* of L. A basis $\omega_1, \ldots, \omega_v$ of L is said to be a *reduced basis* if $\pi(\omega_1), \ldots, \pi(\omega_v)$ are linearly independent over $\mathbf{F}_2$. Let us sort $\omega_1, \omega_2, \ldots, \omega_v$ so that $||\omega_1|| \leq ||\omega_2|| \leq \cdots \leq ||\omega_v||$. Then, the numbers $\nu_i := ||\omega_i||$ $(i = 1, \ldots, v)$ are uniquely determined by the lattice, and called *successive minima* (see [8, 9] for details). Note that here we modified the notion of a reduced basis defined in Theorem 2.2 in Part I by neglecting the orderings. If we sort the basis with respect to the norm, we recover the original definition (see the first paragraph in P.9).

Let $\mathrm{tr}_v : O = \mathbf{F}_2^w \to \mathbf{F}_2^v$ be the truncation function (Definition 6.1). 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:

$$\chi_v(s_0) := \sum_{j=0}^{\infty} \mathrm{tr}_v(o(f^j(s_0)))t^{-1-j} = \mathrm{tr}_v(o(s_0))t^{-1} + \mathrm{tr}_v(o(s_1))t^{-2} + \cdots \in K^v. \quad (7.1)$$

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

$$\Lambda_v := \langle e_1, e_2, \ldots, e_v, \chi_v(s_0) \rangle_{\mathbf{F}_2[t]}, \quad (7.2)$$

where $e_i \in K^v$ denotes the unit vector whose i-th component is $\hat{1}$ and the other components are 0 $(i = 1, \ldots, v)$. If we multiply $\chi_v(s_0)$ by the characteristic polynomial $P(t)$ of the transition function f, then all coefficients are polynomials. 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 $s_0 \neq 0$.

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 7.1 ([2, 21], and Theorem 2.3 in Part I). 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 v-th successive minimum of Λ_v.

7.2 Schmidt's reduction, inductive projection and state representation

By Theorem 7.1, 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 $1 \leq v \leq w$. Couture-L'Ecuyer dual lattice method [1] computes a reduced basis of the dual basis of Λ_v inductively for $v = 1, \ldots, w$. In Part I, we 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 [20], is straightforward: for a given generating set $\{\omega_1, \ldots, \omega_m\}$ of an $\mathbf{F}_2[t]$-lattice L, find any nontrivial linear relation over $\mathbf{F}_2$ among a subset of the coefficient vectors at the leading terms $\pi(\omega_i)$ $(1 \leq i \leq m)$. 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 [7] works only for a basis (cf. its generalization for the generating set is in [18]).

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

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

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 [1])

$$\Lambda_v = \rho(\Lambda_{v+1}), \tag{7.3}$$

by looking at the definition (7.2) of Λ_v. This implies that if $\{\omega_1, \ldots, \omega_{v+1}\}$ is a basis of Λ_{v+1}, then $\{\rho(\omega_1), \ldots, \rho(\omega_{v+1})\}$ 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 (7.2). Once we compute a reduced basis of Λ_w, we compute those of $\Lambda_{w-1}, \Lambda_{w-2}, \ldots, \Lambda_1$, 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 plus 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 7.2 (Lemma 3.4 of Part I). If the characteristic polynomial $P(t)$ is irreducible and χ_v 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

$$\Lambda_v = \mathbf{F}_2[t]^v \oplus \chi_v(S) \tag{7.4}$$

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 $poly + \chi_v(s) \in \Lambda_v$. The first direct summand in (7.4) 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.

8 Main result: PIS method

SGR algorithm seems comparable to or even more efficient than Lenstra's algorithm. More recently, Mulders and Storjohann [15] developed a faster lattice reduction algorithm. Wang, Zhu, and Pei [24] 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 [15] with some simplification.

Definition 8.1. 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 0. We shall define a *pivot reduction* on M, which gives another (smaller in a sense) generating set M' of the same lattice.

Lemma 8.2. Assume that $I(\omega_i)$ for $\omega_i \in M$ are all different, i.e.

$$1 \leq k \neq l \leq m \Rightarrow I(\omega_k) \neq I(\omega_l). \tag{8.1}$$

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 8.3. (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 $\omega_l' = 0$ 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 $\omega_l' \neq 0$ 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 8.4 (Lemma 2.2 of [15]). 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 $\omega_l' = 0$), 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 (8.1).

Lemma 8.5 (Lemma 2.1 of [15]). M does not satisfy the condition (8.1) if and only if we can apply a pivot reduction on M.

Mulders and Storjohann lattice reduction algorithm [15] is to iterate pivot reductions on M until (8.1) holds. Since $\{(||\omega||, I(\omega)) \mid 0 \neq \omega \in L\}$ 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. [24], 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:

$$I(\omega_i) = i \text{ for } i = 1, \ldots, v. \tag{8.2}$$

Note that $\{e_1, \ldots, e_v, \chi_v(s_0)\}$ in (7.2) satisfies (8.2). The next procedure gives a sequence of pivot reductions that preserves this triangular condition to reach to a reduced basis. We call this algorithm *Pivot Lattice Reduction* (*PLR*).

procedure Pivot Lattice Reduction (with triangular condition)
input: a generating set $\omega_1, \omega_2, \ldots, \omega_{v+1}$ of L with triangular condition (8.2).
output: a reduced basis $\omega_1, \omega_2, \ldots, \omega_v \in L$.
begin
While $\omega_{v+1} \neq (0, \ldots, 0)$ do
  (reduction step)
  Set $k \leftarrow I(\omega_{v+1})$ (i.e., the pivot index of ω_{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 ω_k and ω_{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 (8.2), 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 to a reduced basis (similarly to Theorem 3.1 in Part I).

Theorem 8.6. Let $\omega_1, \ldots, \omega_{v+1}$ be a generating set of an $\mathbf{F}_2[t]$-lattice $L \subset K^v$ which satisfies the triangular condition (8.2). Let $\omega'_1, \ldots, \omega'_v$ be the obtained reduced basis by PLR. Then, the number of pivot reductions in PLR is bounded from above by

$$(\sum_{i=1}^{v+1} ||\omega_i|| - \sum_{i=1}^{v} ||\omega'_i|| - ||\omega^{\text{last}}|| + 1)v, \tag{8.3}$$

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

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

by

$$S^M := \sum_{i=1}^{v+1}(||\omega_i||v + I(\omega_i)). \tag{8.4}$$

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 (8.2). Assume that N consists of nonzero $v+1$ vectors, and define S^N as well as (8.4). Then, $S^M - S^N$ is said to be the *state drop* [15]. 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 8.4 or [15, 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 $N = \{\omega'_1, \ldots, \omega'_v, \omega^{\text{last}}\}$. By the triangular condition (8.2), $I(\omega_i) = I(\omega'_i) = i$ $(i = 1, \ldots, v)$, which are canceled out in state drop. We have bounds $I(\omega_{v+1}) \leq v$ and $I(\omega^{\text{last}}) \geq 1$. Thus,

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

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

Remark 8.7. 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 (7.3) through PLR keeps the triangular condition (8.2), because if $\{\omega_1, \ldots, \omega_{v+1}\}$ is a reduced basis of Λ_{v+1} obtained by PLR, then we have $I(\omega_i) = i$ for $i = 1, \ldots, v+1$. Then, $\{\rho(\omega_1), \ldots, \rho(\omega_{v+1})\}$ 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 Part I. In Theorem 8.6, let L be Λ_v. Then, we have

$$\sum_{i=1}^{v} ||\omega_i'|| = -\dim(S), \tag{8.5}$$

by Lemma 2.4 in Part I. Since the last vector ω^{last} is reduced by one of $\omega_1', \ldots, \omega_v'$, we assume that $||\omega^{\text{last}}||$ is on average larger than or equal to the average of $||\omega_1'||, \ldots, ||\omega_v'||$, namely,

$$||\omega^{\text{last}}|| \geq -\dim(S)/v. \tag{8.6}$$

From now on, in the computation of complexity, we always assume (8.6). The following theorem gives an upper bound of the total bit operations.

Theorem 8.8. Under (8.6), 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 Part I.

Lemma 8.9. Under (8.6), the number of pivot reductions to obtain a reduced basis by PLR from the generating set in (7.2) is bounded by $(v+1)\dim(S)$.

Proof. In (8.3), by $||e_i|| = 0$ and $||\chi_v(s_0)|| \leq -1$, the first sum is ≤ -1. The second sum is $\dim(S)$ by (8.5). The third term is bounded by $\dim(S)/v$ by (8.6). The result follows from a simple computation. □

Corollary 8.10. 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 (7.2) is bounded by $(v+1)\dim(S)^2$, when using the state representation.

Lemma 8.11. Under (8.6), 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. In (8.3), by $||\rho(\omega_i)|| \leq ||\omega_i||$, the first sum is $\leq \sum_{i=1}^{v+1} ||\omega_i|| = -\dim(S)$ by (8.5). The second sum is again $\dim(S)$. The third term is bounded by $\dim(S)/v$. The result follows. □

Corollary 8.12. 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 8.8. 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 $w^2(w+1)/2$. Summing with Corollaries 8.10 and 8.12, we obtain the theorem. □

Remark 8.13. We compare the computational complexity of PIS (Theorem 8.8) with SIS method. We assume (8.6) at the final step in SGR, as well. According to Theorem 4.3 of Part I, 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 triangulization mechanism and no elimination is necessary. This cost is not negligible for large w.

9 Numerical experiments

9.1 PIS versus SIS

We show some experiments to compare the following two methods:

1. PIS method (our proposal in Section 8).

2. SIS method (the method in Part I).

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 [17] with the tempering improved by the author [4], and $\dim(S) = 19937$). This generator is maximally equidistributed. The left side of Table 3 shows 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. These 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 MT19936-64 [16] 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 side of Table 3 shows that PIS is about three times faster than SIS.

9.2 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, L'Ecuyer, and Matsumoto [17] 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 [12] ($\dim(S) = 19937$) and WELL44497a' (a variant of WELL44497a [17] modified by the author [4] with $\dim(S) = 44497$). Table 4 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.

10 Conclusions

We propose PIS method for computing all $k(v)$, $w \geq v \geq 1$, which is an improvement of SIS in Part I. 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 [11, 12]), which is in particular useful for dynamic parameter searches [13].

Table 3: 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'			MT19937-64					
	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 4: 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	0.105	0.197	0.386	1.128	7820
MT19937	0.029	0.036	0.294	0.481	6750
WELL19937a'	0.525	0.786	0.534	0.798	0
WELL44497a'	2.536	3.091	2.591	3.193	0

Acknowledgements

Firstly, the author wishes to express his gratitude to his thesis advisor, 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 the author invaluable comments, especially, for pointing out an application to Dynamic Creator.

The author would also like to thank Professor Shu Tezuka at Kyushu University for initiating the author into pseudorandom number generation, and Professor Masakazu Jimbo at Nagoya University who was a good advisor of the author's master's course at Keio University.

This work was partially supported by Grant-in-Aid for JSPS Fellows 21·4427, JSPS Grant-in-Aid for Scientific Research No. 21654004, No. 19204002, No. 21654017, and JSPS Core-to-Core Program No.18005.

Bibliography

[1] R. Couture and P. L'Ecuyer, *Lattice computations for random numbers*, Math. Comput. **69** (2000), no. 230, 757–765.

[2] R. Couture, P. L'Ecuyer, and S. Tezuka, *On the distribution of k-dimensional vectors for simple and combined Tausworthe sequences*, Math. Comput. **60** (1993), 749–761.

[3] M. Fushimi and S. Tezuka, *The k-distribution of generalized feedback shift register pseudorandom numbers*, Commun. ACM **26** (1983), no. 7, 516–523.

[4] S. Harase, *Maximally equidistributed pseudorandom number generators via linear output transformations*, Math. Comput. Simul. **79** (2009), no. 5, 1512–1519.

[5] P. L'Ecuyer and R. Couture, *An implementation of the lattice and spectral tests for multiple recursive linear random number generators*, INFORMS Journal on Computing **9** (1997), no. 2, 206–217.

[6] P. L'Ecuyer and F. Panneton, $\mathbb{F}_2$*-linear random number generators*, Advancing the Frontiers of Simulation: A Festschrift in Honor of George Samuel Fishman (C. Alexopoulos, D. Goldsman, and J. R. Wilson, eds.), Springer-Verlag, 2009, pp. 169–193.

[7] A. K. Lenstra, *Factoring multivariate polynomials over finite fields*, Journal of Computer and System Sciences **30** (1985), no. 2, 235 – 248.

[8] K. Mahler, *An Analogue to Minkowski's Geometry of Numbers in a Field of Series*, The Annals of Mathematics **42** (1941), no. 2, 488–522.

[9] ———, *On a theorem in the geometry of numbers in a space of Laurent series*, Journal of Number Theory **17** (1983), no. 3, 403 – 416.

[10] J. L. Massey, *Shift-register synthesis and BCH decoding*, IEEE Trans. Inform. Theory **IT-15** (1969), 122–127.

[11] M. Matsumoto and Y. Kurita, *Twisted GFSR generators II*, ACM Trans. Model. Comput. Simul. **4** (1994), no. 3, 254–266.

[12] M. Matsumoto and T. Nishimura, *Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator*, ACM Trans. Model. Comput. Simul. **8** (1998), no. 1, 3–30.

[13] ———, *Dynamic Creation of Pseudorandom Number Generators*, Monte Carlo and Quasi-Monte Carlo Methods 1998 (Berlin) (H. Niederreiter and J. Spanier, eds.), Springer-Verlag, 2000, pp. 56–69.

[14] M. Matsumoto, M. Saito, H. Haramoto, and T. Nishimura, *Pseudorandom Number Generation: Impossibility and Compromise*, J. Univer. Comput. Sci. **12** (2006), no. 6, 672–690.

[15] T. Mulders and A. Storjohann, *On lattice reduction for polynomial matrices*, J. Symb. Comput. **35** (2003), no. 4, 377–401.

[16] T. Nishimura, *Tables of 64-bit Mersenne twisters*, ACM Trans. Model. Comput. Simul. **10** (2000), no. 4, 348–357.

[17] F. Panneton, P. L'Ecuyer, and M. Matsumoto, *Improved long-period generators based on linear recurrences modulo* 2, ACM Trans. Math. Softw. **32** (2006), no. 1, 1–16.

[18] S. Paulus, *Lattice basis reduction in function fields*, Algorithmic Number Theory (Berlin) (P. J. Buhler, ed.), Lecture Notes in Computer Science, vol. 1423, Springer-Verlag, 1998, pp. 567–575.

[19] M. Saito and M. Matsumoto, *SIMD-oriented Fast Mersenne Twister: a 128-bit Pseudorandom Number Generator*, Monte Carlo and Quasi-Monte Carlo Methods 2006 (Berlin) (A. Keller, S. Heinrich, and H. Niederreiter, eds.), Springer-Verlag, 2008, pp. 607–622.

[20] W. M. Schmidt, *Construction and estimation of bases in function fields*, J. Number Theory **39** (1991), no. 2, 181 – 224.

[21] S. Tezuka, *The k-dimensional distribution of combined GFSR sequences*, Math. Comput. **62** (1994), no. 206, 809–817.

[22] L. Wang and H. Niederreiter, *Successive minima profile, lattice profile, and joint linear complexity profile of pseudorandom multisequences*, J. Complex. **24** (2008), no. 2, 144–153.

[23] L. Wang and Y. Zhu, *$F[x]$-lattice basis reduction algorithm and multisequence synthesis*, Sci. in China Ser. F **44** (2001), 321–328.

[24] L. Wang, Y. Zhu, and D.-Y. Pei, *On the Lattice Basis Reduction Multisequence Synthesis Algorithm*, IEEE Trans. Inform. Theory **50** (2004), no. 11, 2905–2910.