A. THE 64-BIT OUTPUT SEQUENCES OF SFMT19937

In this appendix, we examine the 64-bit integer output sequences of the SFMT19937 generator (or the double-precision floating-point numbers in \([0, 1)\) converted from them). In fact, the SFMT generators are optimized under the assumption that one will mainly be using 32-bit output sequences, so that the dimensions of equidistribution with \(v\)-bit accuracy for 64-bit output sequences are worse than those for 32-bit cases. We therefore study the structure of SFMT19937 and point out its weaknesses. We also apply empirical statistical tests to non-successive values of SFMT19937 and find that the generator fails them.

A.1. \(\mathbb{F}_2\)-Linear Relations of SFMT19937

In the case of \(\mathbb{F}_2\)-linear generators, there are always certain bits of output whose sum in \(\mathbb{F}_2\) becomes 0 in dimensions higher than \(k(v)\). Such relations are said to be \(\mathbb{F}_2\)-linear relations. When there exist \(\mathbb{F}_2\)-linear relations with small numbers of terms (e.g., \(\leq 6\)), use of the generator might carry risks in some situations [Matsumoto and Nishimura 2002]. We call the number of nonzero terms of an \(\mathbb{F}_2\)-linear relation the weight. In previous work, Harase [2014] showed that MT19937 has low-weight \(\mathbb{F}_2\)-linear relations in \((k(v) + 1)\)-dimensional output values and that, as a result, some non-random bit patterns are detectable in statistical tests. In this subsection, we investigate the \(\mathbb{F}_2\)-linear relations of SFMT19937 as well.

SFMT19937 is an \(\mathbb{F}_2\)-linear generator with \(w = 128\) and \(p = 19968\). The period is a multiple of \(2^{19937}\). Note that the state \(s_i \in S\) in Definition 2.1 has 31 more bits than 19937, so that the period may exceed \(2^{19937}\) (see also Proposition 1 in [Saito and Matsumoto 2008] for the periodicity). We set \(N := p/w = 156\). The state transition \(f : S \to S\) is expressed as \((w_i, \ldots, w_{i+N-1}) \mapsto (w_{i+1}, \ldots, w_{i+N})\) with the recursion

\[
[w_{i+N}] := w_i \tilde{A} + w_{i+M} \tilde{B} + w_{i+N-2} \tilde{C} + w_{i+N-1} \tilde{D}.
\]

Here \(w_0, w_1, \ldots\) are 128-bit integers, \(M \equiv 122\), and \(\tilde{A}, \tilde{B}, \tilde{C}, \tilde{D}\) are suitable \((128 \times 128)\)-matrices. SFMT19937 generates a 128-bit sequence \((w_0, w_1, w_2, \ldots)\), as in (2) of Definition 2.1. We denote \(w_i := (w_{i,0}, \ldots, w_{i,127}) \in \mathbb{F}_2^{128}\). When implementing this generator on 32-bit computers, the 128-bit integers \(w_i\) are divided into four 32-bit integers \(x_i[0], x_i[1], x_i[2], x_i[3]\) as follows:

\[
\begin{align*}
x_i[3] & \equiv w_{i,0}, \ldots, w_{i,31} \\
x_i[2] & \equiv w_{i,32}, \ldots, w_{i,63} \\
x_i[1] & \equiv w_{i,64}, \ldots, w_{i,95} \\
x_i[0] & \equiv w_{i,96}, \ldots, w_{i,127}
\end{align*}
\]

Note that the above four 32-bit integers are indexed starting from the least significant bits. Thus, the 32-bit output sequence of SFMT19937 is obtained as

\[
x_0[0], x_0[1], x_0[2], x_0[3], x_1[0], x_1[1], x_1[2], x_1[3], \ldots \in \mathbb{F}_2^{32}
\]
in this order. SFMT19937 is mainly optimized for 32-bit integer output (see [Saito and Matsumoto 2008] for details). On the other hand, the 64-bit integers of SFMT19937 (denoted by SFMT19937-64) are output as the concatenations

$$\langle x_0[1], x_0[0], x_0[3], x_0[2], x_1[1], x_1[0], x_1[3], x_1[2] \rangle, \ldots \in \mathbb{F}_2^{64}$$

(3)

in that order. The SFMT generator is \( \mathbb{F}_2 \)-linear as a 128-bit generator; but the 32- and 64-bit output sequences are no longer \( \mathbb{F}_2 \)-linear, according to Definition 2.1. To define \( k(v) \) for 32- or 64-bit output of the SFMT generator, we need to modify Definitions 2.1 and 2.2. For brevity, we omit the precise definition of \( k(v) \) for the SFMT generator (see [Saito and Matsumoto 2008] for details).

Table I summarizes the dimensions of equidistribution \( k(v) \). For SFMT19937-64, \( k(3) = \cdots = k(63) = 312 \) and \( k(64) = 310 \), so we omitted them from the table. (Recently, corrections to the tables of \( k(v) \) for SFMT generators have been reported; see http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/SFMT/index.html. Thus, we re-computed \( k(v) \) by using MTToolBox [Saito 2013].) Note that the \( k(v) \)'s are much lower than those for the 32-bit output sequences from SFMT19937. For example, \( k(19) = 624 \) for 32-bit output sequences.

Here we take a closer look at the most significant bits of (2). In fact, from Eq. (2) and Fig. 1 of [Saito and Matsumoto 2008], \( B \) and \( C \) are the right-shift (with some bit masks), so \( w_{i+M}B \) and \( w_{i+N-j}C \) do not affect the generation of the most significant bits. As a result, we can find the four-term \( \mathbb{F}_2 \)-linear relations on \( x_0[3], x_1[3], x_2[3], \ldots \) as

$$w_{i,j} + w_{i,j+8} + w_{i+155,j+18} + w_{i+156,j} = 0 \quad (j = 0, \ldots, 8).$$

(4)

In particular, the relation (4) implies that \( k(19) \) for SFMT19937-64 is at most 312, because such an \( \mathbb{F}_2 \)-linear relation destroys the surjectivity of the map in (2) of Proposition 2 of [Saito and Matsumoto 2008]. This is why the dimensions of equidistribution \( k(v) \) of SFMT19937-64 rapidly decrease for \( v \geq 19 \).

**Remark A.1.** For \( \mathbb{F}_2 \)-linear generators, \( \mathbb{F}_2 \)-linear relations coincide with the vectors in the (dual) lattices of [Couture and L’Ecuyer 2000] (with components in the polynomial ring \( \mathbb{F}_2[x] \)) associated to output sequences. Harase [2014] proposed a method to detect low-weight \( \mathbb{F}_2 \)-linear relations in \( (k(v) + 1) \)-dimensional output with \( v \)-bit accuracy, i.e., among \((k(v) + 1)v \) bits. By use of this method, we can also detect the \( \mathbb{F}_2 \)-linear relations on \( x_0[1], x_1[1], x_2[1], \ldots : \)

$$w_{i,64+j+8} + w_{i,64+j+16} + w_{i+154,64+j} + w_{i+156,64+j+8} = 0 \quad (j = 7, 10, 11).$$

(5)

for example.

**Remark A.2.** For the SFMT generator, when we convert 64-bit integers into double-precision floating-point numbers in \([0, 1)\), the dimensions of equidistribution drastically decrease compared with the conversion from 32-bit integers. Saito and Matsumoto 2008] for details).
sumoto [2009] developed the dSFMT generator, which is specialized for generating double-precision floating-point numbers based on the IEEE 754-2008 format (IEEE Standard for Binary Floating-Point Arithmetic (ANSI/IEEE Std 754-2008). For double-precision floating-point numbers, dSFMT is faster than SFMT and is also improved from the viewpoint of the dimensions of equidistribution with $v$-bit accuracy. Thus, as far as SIMD-oriented generators are concerned, for generating double-precision floating-point numbers, the dSFMT generator is preferable to conversion from the 64-bit integers of the SFMT generator (i.e., a function sfmt_genrand_res53(), which is obtained by dividing (3) by $2^{64}$). Note that dSFMT directly generates double-precision floating-point numbers with 52-bit accuracy and does not support 64-bit integer output sequences. Note also that the dSFMT generator is not maximally equidistributed ($\Delta = 2616$).

A.2. Birthday Spacings Tests for Non-Successive Values

L’Ecuyer and Touzin [2004] and L’Ecuyer and Simard [2014] reported that some multiple recursive generators based on sparse characteristic polynomials have a structural weakness and fail a standard statistical test (called the birthday spacings test; see Marsaglia 1985; Knuth 1997; L’Ecuyer and Simard 2001; 2007; Lemieux 2009]). Harase [2014] pointed out that 32-bit MT19937 fails this test for non-successive values in a similar fashion.

For the birthday spacings test, we select two positive integers $n$ and $d$ and generate $n$ “independent” points $u_0, \ldots, u_{n-1}$ in the $d$-dimensional hypercube $[0, 1)^d$. We partition the hypercube into $d^l$ cubic boxes of equal size by dividing $[0, 1)$ into $d$ equal segments. These boxes are numbered from 0 to $d^l - 1$ in lexicographic order. Let $I_1 \leq I_2 \leq \cdots \leq I_n$ be the numbers of the boxes where these points have fallen, sorted by increasing order. Define the spacings $S_j = I_{j+1} - I_j$, for $j = 1, \ldots, n - 1$, and let $Y$ be the total number of collisions of these spacings, i.e., the number of values of $j \in \{1, \ldots, n - 2\}$ such that $S_{j+i} = S_i$, where $S_{(1)}, \ldots, S_{(n-1)}$ are the spacings sorted by increasing order. We test the null hypothesis $H_0$ that the generator’s output is perfectly random. If $d^l$ is large and $\lambda = n^2 / (4d^n)$ is not too large, $Y$ is approximately a Poisson distribution with mean $\lambda$ under $H_0$. Further, we generate $N$ independent replications of $Y$, add them, and compute the $p$-value by using the sum, which is approximately a Poisson distribution with mean $NA$, under $H_0$. If $d = 2^r$, the $d$-dimensional output with $v$-bit accuracy is tested.

Next, for a sequence $u_0, u_1, u_2, \ldots \in [0, 1)$, we extract non-successive values and construct $d$-dimensional output vectors $u_i = (u_{(j_1+i)+j_1}, \ldots, u_{(j_i+1)+j_i})$ for $i = 0, \ldots, n - 1$ with a lacunary filter $I = \{j_1, \ldots, j_i\}$. We also drop the $\tau$ most significant bits, left-shift the others by $\tau$ positions, and return a floating-point number in $[0, 1)$. In other words, the output sequence is $2\tau u_i \mod 1$. If $\tau = 0$, this is the usual output sequence, and if $\tau > 0$, the least significant bits of the integer outputs are investigated. We use the birthday spacings tests implemented in the TestU01 package [L’Ecuyer and Simard 2014].

From the 64-bit integer output of SFMT19937-64, we generate double-precision floating-point numbers $u_0, u_1, \ldots \in [0, 1)$, which are obtained by the function sfmt_genrand_res53(). We select the lacunary filter $I = \{1, 311, 313\}$. Note that these indices are twice as much as in (4), because we split 128-bit integers (2) into two 64-bit integers and interchange them. We select the parameter set $(N, n, \tau, d, t) = (5, 20000000, 0, 2^{21}, 3)$, which comes from the parameter set of test No. 12 of Crush in TestU01. The second row of Table II shows the right $p$-values for five initial values. SFMT19937-64 decisively fails the birthday spacings tests. Further, we select the
Thus, according to 64-bit MELGs, the above weights are avoided so long as we focus on the consecutive 

\((v) + 1)^k\) bits, where \(k(v)\) and \((k(v) + 1)^k\) bits are taken from the consecutive \(k(v)\) and \((k(v) + 1)^k\) integers, respectively, by extracting the \(v\) most significant bits from each. For our 64-bit MELGs, we investigated \(F_2\)-linear relations among the above \((k(v) + 1)^k\) bits by using the method of Harase [2014]. We detected no low-weight \(F_2\)-linear relation (e.g., the number of weights \(\leq 20\)) among the above \((k(v) + 1)^k\) bits within the \(v\) most significant bits for \(1 \leq v \leq 32\). (In the case of MELG19937-64, the above weights are \(\geq 9500\) for \(1 \leq v \leq 32\).) Thus, according to [Matsumoto and Nishimura 2002], this result implies that the existence of a bad lacunary filter \(I\) for the 64-bit MELGs is avoided so long as we focus on the consecutive \((k(v) + 1)^k\) bits for \(1 \leq v \leq 32\), which correspond to just one dimension beyond \(k(v)\).

**REFERENCES**


**Table II.** \(p\)-Values on the Birthday Spacings Tests with Lacunary Filters \(I\) for SFMT19937-64

<table>
<thead>
<tr>
<th>(I)</th>
<th>1st (p)-value</th>
<th>2nd (p)-value</th>
<th>3rd (p)-value</th>
<th>4th (p)-value</th>
<th>5th (p)-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(I = {1, 311, 313} ) ((N = 5))</td>
<td>(1.2 \times 10^{-241})</td>
<td>(9.4 \times 10^{-281})</td>
<td>(6.5 \times 10^{-285})</td>
<td>(5.1 \times 10^{-274})</td>
<td>(1.3 \times 10^{-230})</td>
</tr>
<tr>
<td>(I = {0, 308, 312} ) ((N = 5))</td>
<td>0.09</td>
<td>0.60</td>
<td>3.1 \times 10^{-3}</td>
<td>4.0 \times 10^{-3}</td>
<td>5.7 \times 10^{-3}</td>
</tr>
<tr>
<td>(I = {0, 308, 312} ) ((N = 20))</td>
<td>(5.2 \times 10^{-3})</td>
<td>(7.5 \times 10^{-4})</td>
<td>(3.9 \times 10^{-4})</td>
<td>(8.0 \times 10^{-5})</td>
<td>(9.6 \times 10^{-4})</td>
</tr>
</tbody>
</table>

**Remark A.3.** As mentioned in Section 7 of [Saito and Matsumoto 2008], in the case of \(F_2\)-linear generators, the dimension of equidistribution \(k(v)\) with \(v\)-bit accuracy means that there is no constant \(F_2\)-linear relation among the \(k(v)\) \(v\)-bits, but there does exist an \(F_2\)-linear relation among the \((k(v) + 1)^k\) \(v\)-bits, where \(k(v)\) \(v\) and \((k(v) + 1)^k\) \(v\)-bits are taken from the consecutive \(k(v)\) and \((k(v) + 1)^k\) integers, respectively, by extracting the \(v\) most significant bits from each. For our 64-bit MELGs, we investigated \(F_2\)-linear relations among the above \((k(v) + 1)^k\) \(v\)-bits by using the method of Harase [2014]. We detected no low-weight \(F_2\)-linear relation (e.g., the number of weights \(\leq 20\)) among the above \((k(v) + 1)^k\) \(v\)-bits within the \(v\) most significant bits for \(1 \leq v \leq 32\). (In the case of MELG19937-64, the above weights are \(\geq 9500\) for \(1 \leq v \leq 32\).) Thus, according to [Matsumoto and Nishimura 2002], this result implies that the existence of a bad lacunary filter \(I\) for the 64-bit MELGs is avoided so long as we focus on the consecutive \((k(v) + 1)^k\) \(v\)-bits for \(1 \leq v \leq 32\), which correspond to just one dimension beyond \(k(v)\).
M. Saito and M. Matsumoto, 2009. A PRNG specialized in double precision floating point numbers using
DOI: http://dx.doi.org/10.1007/978-3-642-04107-5_38