A C++ Library to Test the Lattice Structure of Linear Generators and Search for Good Ones

Pierre L'Ecuyer

Université de Montréal · DIRO · GERAD GROUP FOR RESEARCH IN DECISION ANALYSIS · CIRRELT · IVADO

Joint work with **Christian F. Weiss**, Ruhr West University, Germany
and **Marc-Antoine Savard**, former M.Sc. student, Université de Montréal

MCQMC, Waterloo, Canada, August 2024

Lattice Tester and LatMRG Software

Lattice Tester is a C++ library to compute measures of uniformity (figures of merit) for lattices in the t-dimensional integer space $\mathbb{Z}^t$. Also basic tools to build a basis from generating vectors, construct a triangular basis, its m-dual basis, compute shortest vectors, etc. It uses the NTL library (Shoup, 2005).

LatMRG is a C++ library + executable programs to study the lattice structure of linear RNGs such as LCGs, MRGs, matrix LCGs, MWC, combined generators, etc. It can search for good parameters in terms of a given FOM, perhaps with constraints on the parameters, etc. It uses Lattice Tester.

Both are on GitHub, but still under construction. Each one has an extensive user's guide that describes the algorithms, the classes, and give examples.

A first version of LatMRG was written in Modula-2 around 1988–1990 and presented by L'Ecuyer and Couture (1997). A C++ version was started around 2002, modified by many people, but never really completed. We are now redesigning, rewriting most of the code, and writing detailed documentation.

Linear Random Numbers Generators Modulo a Large m

Linear Congruential Generator (LCG), with $0 < a < m$ integers:

$$x_n = a x_{n-1} \bmod m, \qquad u_n = x_n/m \in [0,1), \qquad \text{for } n \geq 0.$$

Multiple Recursive Generator (MRG):

$$x_n = (a_1 x_{n-1} + \ldots + a_k x_{n-k}) \bmod m, \qquad u_n = x_n/m \in [0,1),$$

where $x_j \in \mathbb{Z}_m$, $a_j \in \mathbb{Z}_m$, $a_k \neq 0$.

Matrix LCG:

State $\mathbf{x}_n = (x_{n,0}, \ldots, x_{n,k-1})^t$, transformed state $\mathbf{y}_n = (y_{n,0}, \ldots, x_{n,w-1})^t$,
matrices $\mathbf{A}$ $(k \times k)$ and $\mathbf{B}$ $(w \times k)$,

$$\mathbf{x}_n = \mathbf{A}\mathbf{x}_{n-1} \bmod m, \qquad \mathbf{y}_n = \mathbf{B}\mathbf{x}_n \bmod m, \qquad \mathbf{u}_n = \mathbf{y}_n/m \quad \text{(output vector)}.$$

Matrix LCG:

State $\mathbf{x}_n = (x_{n,0}, \ldots, x_{n,k-1})^t$, transformed state $\mathbf{y}_n = (y_{n,0}, \ldots, x_{n,w-1})^t$,
matrices $\mathbf{A}$ $(k \times k)$ and $\mathbf{B}$ $(w \times k)$,

$$\mathbf{x}_n = \mathbf{A}\mathbf{x}_{n-1} \bmod m, \qquad \mathbf{y}_n = \mathbf{B}\mathbf{x}_n \bmod m, \qquad \mathbf{u}_n = \mathbf{y}_n/m \quad \text{(output vector)}.$$

Can produce the sequence of random numbers $u_{0,0}, \ldots, u_{0,w-1}, u_{1,0}, \ldots, u_{1,w-1}, u_{2,0}, \ldots$.
Maximal period is $m^k - 1$, reached iff m is a prime number and the characteristic polynomial of $\mathbf{A}$ is a primitive polynomial modulo m.

Matrix LCG:

State $\mathbf{x}_n = (x_{n,0}, \ldots, x_{n,k-1})^{\mathrm{t}}$, transformed state $\mathbf{y}_n = (y_{n,0}, \ldots, x_{n,w-1})^{\mathrm{t}}$,
matrices $\mathbf{A}$ $(k \times k)$ and $\mathbf{B}$ $(w \times k)$,

$$\mathbf{x}_n = \mathbf{A}\mathbf{x}_{n-1} \bmod m, \qquad \mathbf{y}_n = \mathbf{B}\mathbf{x}_n \bmod m, \qquad \mathbf{u}_n = \mathbf{y}_n/m \quad \text{(output vector)}.$$

Can produce the sequence of random numbers $u_{0,0}, \ldots, u_{0,w-1}, u_{1,0}, \ldots, u_{1,w-1}, u_{2,0}, \ldots$.
Maximal period is $m^k - 1$, reached iff m is a prime number and the characteristic polynomial of $\mathbf{A}$ is a primitive polynomial modulo m.

The MRG is a special case with $w = 1$, $\mathbf{x}_n = (x_{n-k+1}, \ldots, x_n)$, and

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ a_k & a_{k-1} & \cdots & a_1 \end{pmatrix} \quad \text{and} \quad \mathbf{B} = \mathbf{e}_k = (0, \ldots, 0, 1).$$

MRG With Carry, Alias Multiply-With-Carry (MWC)

Let $b \geq 2$ and $a_0, \ldots, a_k$ be arbitrary integers, with $\gcd(a_0, b) = 1$, and $a_0^* = a_0^{-1} \pmod{b}$.
MWC generator of order k in base b: state is $(x_{n-k+1}, \ldots, x_n, c_n)$,

$$
\begin{aligned}
\tau_n &= a_1 x_{n-1} + \cdots + a_k x_{n-k} + c_{n-1}, \\
x_n &= a_0^* \tau_n \bmod b, \\
c_n &= (\tau_n - a_0 x_n) \operatorname{div} b \;=\; \lfloor (\tau_n - a_0 x_n)/b \rfloor, \\
\text{output: } u_n &= \sum_{\ell=1}^{\infty} x_{n+\ell-1} b^{-\ell} \qquad \text{(usually truncated).}
\end{aligned}
$$

MRG With Carry, Alias Multiply-With-Carry (MWC)

Let $b \geq 2$ and $a_0, \ldots, a_k$ be arbitrary integers, with $\gcd(a_0, b) = 1$, and $a_0^* = a_0^{-1} \pmod{b}$.
MWC generator of order k in base b: state is $(x_{n-k+1}, \ldots, x_n, c_n)$,

$$\begin{aligned} \tau_n &= a_1 x_{n-1} + \cdots + a_k x_{n-k} + c_{n-1}, \\ x_n &= a_0^* \tau_n \bmod b, \\ c_n &= (\tau_n - a_0 x_n) \operatorname{div} b \;=\; \lfloor (\tau_n - a_0 x_n)/b \rfloor, \\ \text{output: } u_n &= \sum_{\ell=1}^{\infty} x_{n+\ell-1} b^{-\ell} \qquad \text{(usually truncated).} \end{aligned}$$

Equivalent to an LCG with $m = -a_0 + \sum_{\ell=1}^{k} a_\ell b^\ell$ and multiplier $b^* \equiv b^{-1} \pmod{m}$.
Can take $b = 2^e$ for efficient implementation, and get m prime for long period.
In general, the maximal possible period is $m - 1$. Can approach b^{k+1}.
When $b = 2^e$ and $a_0 = 1$, the max period is only $(m-1)/2$.
For instance, with $e = 64$ and $k = 4$, one may get a period over 2^{300}.

Can be seen as an effective way to implement an LCG with a huge modulus m.

Combined Linear Generators

Combining two or more LCGs or MRGs by taking a linear combination of their output is an effective way to increase the period and improve the structure. Take C MRGs of the form

$$x_{c,n} = (a_{c,1}x_{c,n-1} + \cdots + a_{c,k}x_{c,n-k}) \bmod m_c \qquad \text{where } a_{c,k} \neq 0, \text{ for } c = 1, \ldots, C,$$

where $m_1 > \cdots > m_c$ are distinct primes. For arbitrary integers $\delta_1, \ldots, \delta_C$, a first combination is

$$z_n = (\delta_1 x_{1,n} + \cdots + \delta_C x_{C,n}) \bmod m_1, \qquad u_n = z_n/m_1,$$

and a second one is

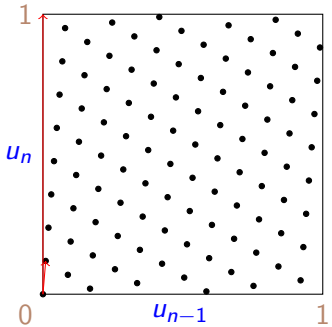
$$w_n = (\delta_1 x_{1,n}/m_1 + \cdots + \delta_C x_{C,n}/m_C) \bmod 1.$$

The two output sequences $\{u_n,\ n \geq 0\}$ and $\{w_n,\ n \geq 0\}$ are almost identical and the second is exactly equivalent to an MRG with modulus $m = \prod_{c=1}^{C} m_c$, and whose coefficients a_j can be computed explicitly. We can select the parameters by analyzing the structure of this equivalent MRG.

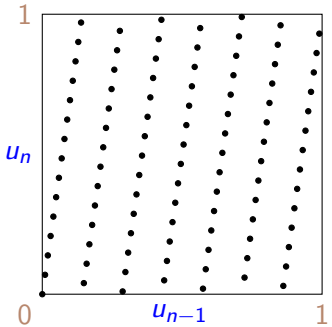
We can also combine MWC generators in the same way.

Lattice Structure

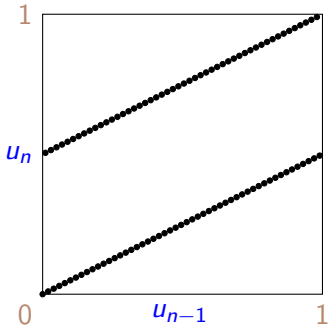
Example: Three LCGs with $m = 101$. Here we have $u_n = a u_{n-1} \bmod 1$.



$a = 12$



$a = 7$



$a = 51$

Lattices and their Dual

A lattice with basis $\mathbf{v}_1, \ldots, \mathbf{v}_t \in \mathbb{R}^t$, which is a set of t linearly independent vectors over $\mathbb{Z}$, can be written as

$$L_t = \{\mathbf{v} = \sum_{i=1}^{t} b_i \mathbf{v}_i : b_i \in \mathbb{Z}\}.$$

We call $\mathbf{V} = \begin{bmatrix} \mathbf{v}_1 & \cdots & \mathbf{v}_t \end{bmatrix}^t$ a basis matrix of L_t.

Lattices and their Dual

A lattice with basis $\mathbf{v}_1, \ldots, \mathbf{v}_t \in \mathbb{R}^t$, which is a set of t linearly independent vectors over $\mathbb{Z}$, can be written as

$$L_t = \{\mathbf{v} = \sum_{i=1}^{t} b_i \mathbf{v}_i : b_i \in \mathbb{Z}\}.$$

We call $\mathbf{V} = \begin{bmatrix}\mathbf{v}_1 & \cdots & \mathbf{v}_t\end{bmatrix}^t$ a basis matrix of L_t.

The dual lattice is

$$L_t^* = \{\mathbf{w} \in \mathbb{R}^t : \mathbf{w} \cdot \mathbf{v} \in \mathbb{Z} \text{ for all } \mathbf{v} \in L_t\}.$$

It is also a lattice with basis matrix $\mathbf{W} = (\mathbf{V}^{-1})^{\mathrm{t}}$, which is the dual of $\mathbf{V}$.

More generally, we say that $\mathbf{W}$ is m-dual to $\mathbf{V}$ if $\mathbf{V} \cdot \mathbf{W}^{\mathrm{t}} = m\,\mathbf{I}$. In LatMRG, m is always chosen so that all coordinates of $\mathbf{V}$ and $\mathbf{W}$ are integer, so they can be represented exactly.

Lattice Structure of MRGs

For any fixed $t > 0$, let

$$\Psi_t = \{(u_0, u_1, \ldots, u_{t-1}) \in [0,1)^t : \mathbf{x}_0 = (x_{-k+1}, \ldots, x_0) \in \mathbb{Z}_m^k\}.$$

Then $\Psi_t = L_t \cap [0,1)^t$ for a lattice

$$L_t = \{\mathbf{v} = \sum_{i=1}^{t} b_i \mathbf{v}_i : b_i \in \mathbb{Z}\}$$

where $\mathbf{v}_1, \ldots, \mathbf{v}_t \in \mathbb{R}^t$ are independent vectors that form a basis.
We saw pictures of this for $k = 2$ and $m = 101$.

Lattice Structure of MRGs

For any fixed $t > 0$, let

$$\Psi_t = \{(u_0, u_1, \ldots, u_{t-1}) \in [0,1)^t : \mathbf{x}_0 = (x_{-k+1}, \ldots, x_0) \in \mathbb{Z}_m^k\}.$$

Then $\Psi_t = L_t \cap [0,1)^t$ for a lattice

$$L_t = \{\mathbf{v} = \sum_{i=1}^{t} b_i \mathbf{v}_i : b_i \in \mathbb{Z}\}$$

where $\mathbf{v}_1, \ldots, \mathbf{v}_t \in \mathbb{R}^t$ are independent vectors that form a basis.
We saw pictures of this for $k = 2$ and $m = 101$.

More generally, for (lacunary) indices $I = (i_1, \ldots, i_s)$ with $1 \le i_1 < \cdots < i_s$, if

$$\Psi_I = \{(u_{i_1-1}, \ldots, u_{i_s-1}) \in [0,1)^s : (x_{-k+1}, \ldots, x_0) \in \mathbb{Z}_m^k\},$$

then $\Psi_I = L_I \cap [0,1)^s$ for a lattice L_I.

In our software, we multiply all the lattice vector coordinates by m, so they can always be represented exactly as integers. This gives rescaled lattices Λ_t and Λ_I.

1. If the lattice contains a very short nonzero vector, then since all multiples of this vector belong to the lattice, we get a very dense line of points and this bad. So we want the shortest nonzero vector in the lattice to be as large as possible.

2. The distance between the successive hyperplanes that contain all the points is 1 over the Euclidean length of the shortest nonzero vector in the dual lattice. So we want that length to be as large as possible, to avoid large empty slices of space.

3. If we take the ℓ_1 length instead, it gives the number of hyperplanes that contain all the points, and we also want it to be as large as possible.

Dual lattice to L_t, also m-dual to Λ_t, is

$$L_t^* = \{\mathbf{w} \in \mathbb{R}^t : \mathbf{w} \cdot \mathbf{v} \bmod 1 = 0 \text{ for all } \mathbf{v} \in L_t\} = \{\mathbf{w} \in \mathbb{R}^t : \mathbf{w} \cdot \mathbf{v} \bmod m = 0 \text{ for all } \mathbf{v} \in \Lambda_t\}.$$

If the rows of matrix $\mathbf{V}$ form a basis for Λ_t, then the rows of the matrix $\mathbf{W}$ for which $\mathbf{V} \cdot \mathbf{W}^\mathbf{t} = m\mathbf{I}$ are a basis for the m-dual.

Computing a Shortest Nonzero Vector in a Lattice

Given a basis $\mathbf{v}_1, \ldots, \mathbf{v}_t$, we can solve by a branch-and-bound (BB) procedure:

$$\min \|\mathbf{v}\|^2 \quad \text{subject to} \quad \mathbf{v} = \sum_{j=1}^{t} z_j \mathbf{v}_j \neq 0, \; z_j \in \mathbb{Z}.$$

Takes exponential time in t in worst case, but often works for t up to 50 or more in practice.

The BB requires either a triangular basis $\mathbf{V}$, or a Cholesky decomposition of the Gram matrix $\mathbf{V} \cdot \mathbf{V}^t$. Cholesky has numerical issues when t is large and some vectors $\mathbf{V}$ are large. For this, we use real numbers with very high precision when required. With a triangular basis there is no numerical issue since all entries are integers, but the bounds are much too wide.

Pre-reducing the basis alleviates numerical issues and makes the BB run much faster.

LLL reduction with factor δ, for $1/4 < \delta < 1$: Gives nearly-orthogonal basis vectors.

(k, δ)-BKZ reduction: Stronger reduction with $k > 2$, but requires more work,

In small dimensions, these reductions often already provide a shortest nonzero vector.

Building a basis for Λ_t or Λ_l

In each row, $v_{i,n} = a_1 v_{i,n-1} + \cdots + a_k v_{i,n-k}$. Standard approach for Λ_t:

$$\mathbf{V} = \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_t \end{pmatrix} = \begin{pmatrix} v_{1,1} & v_{1,2} & \dots & v_{1,t} \\ v_{2,1} & v_{2,2} & \dots & v_{2,t} \\ \vdots & \vdots & & \vdots \\ v_{k,1} & v_{k,2} & \dots & v_{k,t} \\ \vdots & \vdots & & \vdots \\ v_{t,1} & v_{t,2} & \dots & v_{t,t} \end{pmatrix} = \begin{pmatrix} \mathbf{1} & \mathbf{0} & \dots & \mathbf{0} & v_{1,k+1} & \dots & v_{1,t} \\ \mathbf{0} & \mathbf{1} & \dots & \mathbf{0} & v_{2,k+1} & \dots & v_{2,t} \\ \vdots & \vdots & \ddots & \vdots & \vdots & & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{1} & v_{k,k+1} & \dots & v_{k,t} \\ 0 & 0 & \dots & 0 & m & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 0 & \dots & m \end{pmatrix}$$

$$\mathbf{W} = \begin{pmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \vdots \\ \mathbf{w}_t \end{pmatrix} = \begin{pmatrix} m & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & m & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & m & 0 & \dots & 0 \\ -v_{1,k+1} & -v_{2,k+1} & \dots & -v_{k,k+1} & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ -v_{1,t} & -v_{2,t} & \dots & -v_{k,t} & 0 & \dots & 1 \end{pmatrix},$$

An alternative basis that requires less work: $y_n = a_1 y_{n-1} + \cdots + a_k y_{n-k}$.

$$\mathbf{V}^{(p)} = \begin{pmatrix} 1 & y_{k+1} & y_{k+2} & \cdots & y_{2k-1} & y_{2k} & \cdots & y_{t+k-1} \\ 0 & 1 & y_{k+1} & \cdots & y_{2k} & y_{2k-1} & \cdots & y_{t+k-2} \\ \vdots & \vdots & \ddots & & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 & y_{k+1} & y_{k+2} & \cdots & y_{t+1} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{1} & y_{k+1} & \cdots & y_t \\ 0 & 0 & \cdots & \cdots & 0 & m & \cdots & 0 \\ \vdots & \vdots & \ddots & \cdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \cdots & 0 & 0 & \cdots & m \end{pmatrix}$$

and a **valid m-dual basis** is

$$\tilde{\mathbf{W}}^{(p)} = \begin{pmatrix} m & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & m & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & m & 0 & \cdots & 0 \\ -y_{2k} & -y_{2k-1} & \cdots & -y_{k+1} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ -y_{t+k-1} & -y_{t+k-2} & \cdots & -y_t & 0 & \cdots & 1 \end{pmatrix}$$

Basis for Λ_I and its m-dual

For general $I = (i_1, \ldots, i_s)$, picking the s corresponding columns (for $t \geq i_s$) gives a set of t generating vectors, which can be reduced to a basis of s vectors.

1. Can be done by applying LLL to the generating vectors. Gives a basis with short vectors.

2. Can build a triangular basis, as follows. Let c be the gcd of all nonzero entries in first column. If all entries are 0, put $c = m$. By subtracting rows from another several times and exchanging rows, one can obtain c as first entry and zeros in the rest of first column. Then hide the first row and first column and repeat that recursively.

$$\begin{pmatrix} v_{1,i_1} & v_{1,i_2} & \ldots & v_{1,i_s} \\ v_{2,i_1} & v_{2,i_2} & \ldots & v_{2,i_s} \\ \vdots & \vdots & & \vdots \\ v_{t,i_1} & v_{t,i_2} & \ldots & v_{t,i_s} \end{pmatrix} \Rightarrow \begin{pmatrix} \mathbf{c} & \mathbf{v'_{1,2}} & \ldots & \mathbf{v'_{1,s}} \\ \mathbf{0} & v'_{2,2} & \ldots & v'_{2,s} \\ \vdots & \vdots & & \vdots \\ \mathbf{0} & v'_{t,2} & \ldots & v'_{t,s} \end{pmatrix} \Rightarrow \cdots \Rightarrow \begin{pmatrix} c_1 & v''_{1,2} & \ldots & v''_{1,s} \\ 0 & c_2 & \ldots & v''_{2,s} \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \ldots & c_s \end{pmatrix}$$

To obtain an m-dual basis, compute the m-inverse of this triangular basis. Fast and easy. Can reuse the same $\mathbf{V}$ for a large number of projections.

Important: The m-dual of a projection is not the same as the projection of the m-dual!

Sets / with very large lags

Suppose $i_1 = 1$ and some i_j are very large, e.g., $> 2^{50}$.
Then we cannot build all the columns from 1 up to i_j.
We want to be able to jump directly from column 1 to column i_j, in one step.

This can be done efficiently using a polynomial representation of the MRG:

$$p_n(z) = z p_{n-1} \bmod P(z)$$

where $p_n(z) = c_{n,0} + c_{n,1} z_1 + \cdots + c_{n,k-1} z^{k-1}$ represents the state at step n and $P(z) = z^k - a_1 z^{k-1} - \cdots - a_k$ is the characteristic polynomial of the recurrence.

There is a simple one-to-one linear map between the vector of coefficients $(c_{n,0}, c_{n,1}, \cdots, c_{n,k-1})$ and the MRG state vector $\mathbf{x}_n$.

To jump ahead by ν steps, transform the current state $\mathbf{x}_n$ into $p_n(z)$, compute $p_{n+\nu}(z) = z^\nu p_n(z) \bmod P(z)$, then transform to the state $\mathbf{x}_{n+\nu}$.

Example of a Figure of Merit (FOM)

Select d and $t_1, \ldots, t_d$, and let

$$
\begin{aligned}
M_{t_1,\ldots,t_d} &= \min_{1\le s\le d} \min_{I\in S_s(t_s)} \frac{\omega_I \ell_I}{\tilde{\ell}^*_s(m,k)} \qquad \text{where} \\
S_1(t_1) &= \{I = \{1,\ldots,s\} \mid d+1 \le s \le t_1\}, \\
S_s(t_s) &= \{I = \{i_1,\ldots,i_s\} \mid 1 \le i_1 < \cdots < i_s \le t_s\}, \\
\ell_I &= \text{length of shortest nonzero lattice vector in projection } I, \\
\tilde{\ell}^*_s(m,k) &= \text{upper bound on largest possible length.}
\end{aligned}
$$

This bound depends on the dimension s and the lattice density for this projection.
We assume that L_I has density $m^{\min(s,k)}$, so the rescaled Λ_I has density $m^{\min(s,k)-s} = m^{\min(0,k-s)}$, and its m-dual L_I^* has density $m^{-\min(s,k)}$.
If the FOM is for the dual, we take the m-dual for each projection.

Example of a Figure of Merit (FOM)

Select d and $t_1, \ldots, t_d$, and let

$$
\begin{aligned}
M_{t_1,\ldots,t_d} &= \min_{1\le s\le d} \min_{I\in S_s(t_s)} \frac{\omega_I \ell_I}{\tilde{\ell}^*_s(m,k)} \qquad \text{where} \\
S_1(t_1) &= \{I = \{1,\ldots,s\} \mid d+1 \le s \le t_1\}, \\
S_s(t_s) &= \{I = \{i_1,\ldots,i_s\} \mid 1 \le i_1 < \cdots < i_s \le t_s\}, \\
\ell_I &= \text{length of shortest nonzero lattice vector in projection } I, \\
\tilde{\ell}^*_s(m,k) &= \text{upper bound on largest possible length.}
\end{aligned}
$$

This bound depends on the dimension s and the lattice density for this projection.
We assume that L_I has density $m^{\min(s,k)}$, so the rescaled Λ_I has density $m^{\min(s,k)-s} = m^{\min(0,k-s)}$, and its m-dual L_I^* has density $m^{-\min(s,k)}$.
If the FOM is for the dual, we take the m-dual for each projection.
Each term in the FOM should be less than 1, unless we have the wrong density.
This can happen for the primal lattice when $k > 1$ and the points project on each other in a projection (smaller density).

Some Examples and Timings

Compute shortest vector in m-dual lattice for LCG with 50 different multipliers a, for prime modulus $m = 1099511627791$. Timings in microseconds.

Types: Int = NTL::ZZ, Real = double

Num. dimensions:	5	10	20	30	40
LLL5	793	2803	8779	12774	18754
LLL99999	700	3532	23331	60415	92778
BKZ99999-10	715	3720	31545	130793	369886
LLL99999+BB	830	4384	29142	289487	154419563
BKZ99999-12+BB	675	4262	37685	244102	21120263
BKZ999-12+BB	669	4158	37255	257545	20609422
L5+9+BKZ-10+BB	895	4653	34902	232759	18731232

Sums of square lengths of shortest basis vectors:

Num. dimensions:	5	10	20	30	40
LLL5	1867394	12208	2308	1898	1736
LLL99999	1835927	11669	1266	734	667
BKZ99999-10	1835927	11668	1254	695	595
All+BB methods	1835927	11668	1253	689	564

Total time to do the entire experiment with different number representations.

There were many more combinations of reduction methods.

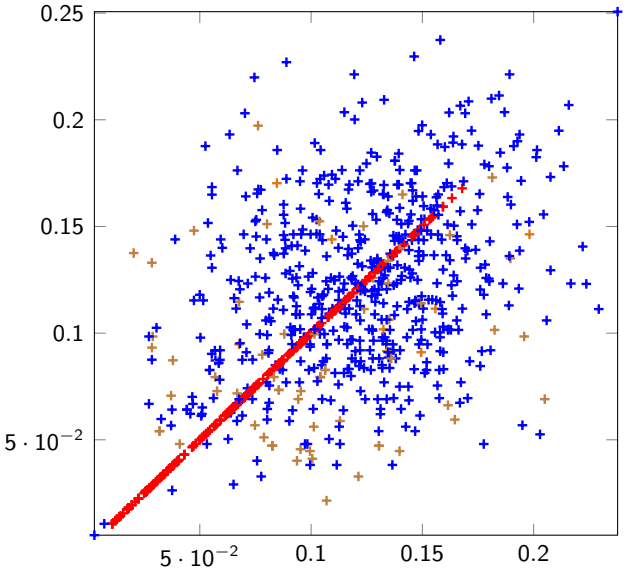
Code	<Int, Real>	$m = 1048573$		$m = 1099511627791$	
		primal	m-dual	primal	m-dual
LD	<long, double>	18.8	96.8	—	—
ZD	<ZZ, double>	17.2	106.1	269.2	418.8
ZX	<ZZ, xdouble>	81.9	562.0	1548.0	2502.0
ZQ	<ZZ, quad_float>	64.1	459.8	1177.3	1864.5
ZR	<ZZ, RR>	—	—	10358.4	17828.6

Comparing FOMs for primal and m-dual

For $m = 1048573$, we examine 1000 different multipliers a selected at random.
For each, we compute the FOM $M_{\mathbf{t}}$ for both the primal and the m-dual, and look at the dimension in which the worst-case is attained.
For $\mathbf{t}$, we try $\mathbf{t}_{16} = (16, 16, 12, 10)$ and $\mathbf{t}_{32} = (32, 32, 16, 12, 10)$.
Number of projections of each size: (12, 15, 55, 84) and (27, 31, 120, 220, 210).

		dimension					
		2	3	4	5	6	> 6
$\mathbf{t}_{16}$	proportion	90.3	331.3	506.0	6.02	6.02	60.3
	primal	359	433	207	1	0	0
	m-dual	370	469	161	0	0	0
$\mathbf{t}_{32}$	proportion	51.0	197.4	361.8	345.4	1.64	42.8
	primal	427	392	157	24	0	0
	m-dual	468	417	105	10	0	0

Scatter plot of the m-dual vs primal FOMs, for $\mathbf{t}_{32}$.
Red marks: worst projection is the same and is in two dimensions.



A random search for "good" LCGs

We search for good LCGs in terms of $M_{\mathbf{t}}$ for $\mathbf{t} = (32, 32, 16, 12, 10)$.
We examine $100,000$ candidates and want to find the best 3.

1. Naive search method: compute the FOM for each candidate.
2, 3. Keep a list of 3 and discard a candidate as soon as we know he misses the podium.
4. Two stages: first pass with LLL only and keep the best 50,
   then a second pass to find the 3 best from this list with $\mathbf{t}$.
5. Same as 4, except that in the first stage we use $\mathbf{t}_0 = (4, 32, 16, 12)$ instead of $\mathbf{t}$.

A random search for "good" LCGs

We search for good LCGs in terms of $M_{\mathbf{t}}$ for $\mathbf{t} = (32, 32, 16, 12, 10)$.
We examine $100,000$ candidates and want to find the best 3.

1. Naive search method: compute the FOM for each candidate.
2, 3. Keep a list of 3 and discard a candidate as soon as we know he misses the podium.
4. Two stages: first pass with LLL only and keep the best 50,
   then a second pass to find the 3 best from this list with $\mathbf{t}$.
5. Same as 4, except that in the first stage we use $\mathbf{t}_0 = (4, 32, 16, 12)$ instead of $\mathbf{t}$.

$m = 1099511627791$	primal		m-dual	
Method	CPU time	best FOM	CPU time	best FOM
1. BKZ+BB, naive	3532.4	0.241259	3334.9.0	0.223998
2. BKZ+BB, discard	8.9	0.264833	10.0	0.269388
3. LLL only, discard	8.5	0.264833	9.5	0.269388
4. Two stages, stage 1 with LLL, $\mathbf{t}$;	16.9		17.8	
stage 2 with BKZ+BB	0.03	0.264833	0.04	0.269388
5. Two stages, stage 1 with LLL, $\mathbf{t}_0$;	26.0		33.2	
stage 2 with BKZ+BB	0.06	0.264833	0.12	0.269388

Re-testing MRG32k3a

`MRG32k3a` is a popular 32-bits combined MRG proposed by L'Ecuyer (1999), based on the M_{t_1} criterion (only successive coordinates) for $t_1 = 32$. Was then tested for $t_1 = 48$.
Here we test it for multiple projections with lacunary indices, with criterion

$$M_{t_1,t_2,\ldots,t_s} = M_{40,100,100,50,50,25,25,20} \quad \text{with } d = 8.$$

t	Seq.	2	3	4	5	6	7	8
$i_{t-1} <$	40	100	100	50	50	25	25	20
Num. Proj.	37	100	4851	18424	211876	42504	134596	50388
$\min_I S_t(I)$	0.659	0.931	0.495	0.053	0.039	0.114	0.114	0.167
Worst set I	24	0,2	0,2,3	0, 39, 42, 44	0, 13, 33, 34, 39	0, 10, 12, 15, 18, 24	0, 6, 16, 18, 20, 21, 23	0, 1, 2, 5, 7, 10, 12, 19

When testing for so many projections (nearly half a million here), it is inevitable to have a few that are not "excellent". But we see here that things are really not so bad. The worst figure of merit is for a five-dim. projection which corresponds to the linear relation
$1234567u_0 + 2376763198u_5 - 45999765568u_8 = 0$.

Some relevant references

L'Ecuyer, P. (1999). Good parameters and implementations for combined multiple recursive random number generators. *Operations Research*, 47(1):159–164.

L'Ecuyer, P. and Couture, R. (1997). An implementation of the lattice and spectral tests for multiple recursive linear random number generators. *INFORMS Journal on Computing*, 9(2):206–217.

L'Ecuyer, P. and Couture, R. (2000). *LatMRG User's Guide: A Modula-2 software for the theoretical analysis of linear congruential and multiple recursive random number generators*. `http://www.iro.umontreal.ca/~lecuyer/myftp/papers/guide-latmrg-m2.pdf`.

L'Ecuyer, P. and Simard, R. (2014). On the lattice structure of a special class of multiple recursive random number generators. *INFORMS Journal on Computing*, 26(2):449–460.

L'Ecuyer, P. and Touzin, R. (2004). On the Deng-Lin random number generators and related methods. *Statistics and Computing*, 14:5–9.

L'Ecuyer, P., Wambergue, P., and Bourceret, E. (2020). Spectral analysis of the MIXMAX random number generators. *INFORMS Journal on Computing*, 32(1):135–144.

Shoup, V. (2005). *NTL: A Library for doing Number Theory*. Courant Institute, New York University, New York, NY. `http://shoup.net/ntl/`.