Improved Versions of the Lattice Tester and LatMRG Software Tools

Pierre L'Ecuyer, Université de Montréal, Canada

joint work with

Mamadou Thiongane, University Cheikh Anta Diop, Dakar, Senegal

Christian F. Weiss, Ruhr West University of Applied Sciences, Germany

MCM 2023 **Paris, France, June 2023**

Software libraries from my lab

Lattice Tester: Base tools to mesure the uniformity of integral lattices in general, in terms of spectral test in primal or dual lattice, with L^2 or L^1 norm, figures of merit that account for many projections, etc.

LatMRG: Software to analyze the lattice structure of linear random number generators (multiple recursive generators (MRGs), matrix MRGs, multiply-with-carry generators (MWC)) and search for good ones in terms of certain figures of merit. Since about 1986.

LatNet Builder: A tool to search for good good lattice rules, polynomial lattice rules, and digital nets in base 2, in terms of figures of merit specified by the user.

Both LatMRG and LatNet Builder use Lattice Tester.
These tools/libraries are in C++ and use the NTL library for arbitrary-precision calculations, and some polynomial, matrix, and lattice operations.

Software libraries from my lab

Lattice Tester: Base tools to mesure the uniformity of integral lattices in general, in terms of spectral test in primal or dual lattice, with L^2 or L^1 norm, figures of merit that account for many projections, etc.

LatMRG: Software to analyze the lattice structure of linear random number generators (multiple recursive generators (MRGs), matrix MRGs, multiply-with-carry generators (MWC)) and search for good ones in terms of certain figures of merit. Since about 1986.

LatNet Builder: A tool to search for good good lattice rules, polynomial lattice rules, and digital nets in base 2, in terms of figures of merit specified by the user.

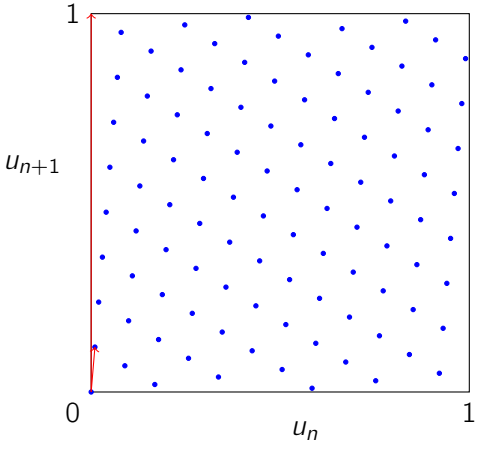
Both LatMRG and LatNet Builder use Lattice Tester.
These tools/libraries are in C++ and use the NTL library for arbitrary-precision calculations, and some polynomial, matrix, and lattice operations.

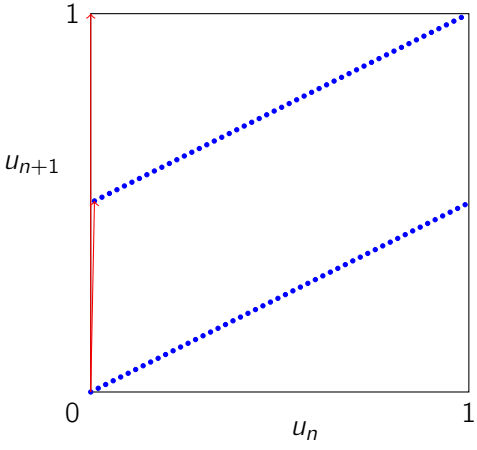
Other software on my table: **F2LinearGen**, **TestU01-64**, **SSJ**.

LatMRG: Lattice structure of linear generators

Linear congruential generator (LCG): $x_n = ax_{n-1} \bmod m$; $u_n = x_n/m$.

Examples with $m = 101$, for $a = 12$ (left) and $a = 51$ (right).





More general

Multiple recursive generator (MRG):

$$x_n = (a_1 x_{n-1} + \cdots + a_k x_{n-k}) \bmod m, \qquad u_n = x_n/m.$$

Matrix linear congruential generator:

$$\mathbf{x}_n = \mathbf{A}\mathbf{x}_{n-1} \bmod m, \qquad \mathbf{y}_n = \mathbf{B}\mathbf{x}_n \bmod m, \qquad \mathbf{u}_n = \mathbf{y}_n/m = (u_{nw}, \ldots, u_{nw+w-1})^{\mathrm{t}},$$

where $\mathbf{A}$ and $\mathbf{B}$ are $k \times k$ and $w \times k$ matrices.

Set of all possible vectors of t successive outputs,

$$\Psi_t = \{\mathbf{u} = (u_0, \ldots, u_{t-1}),\ \mathbf{x}_0 \in \mathbb{Z}_m^k\}.$$

We want the set Ψ_t to cover $[0,1]^t$ very evenly, for all t up to (say) 50 or so.

In fact, $\Psi_t = L_t \cap [0,1)^t$ where mL_t is an integral lattice in t dimensions ($\mathbb{Z}^t \subset L_t$).

More general

Multiple recursive generator (MRG):

$$x_n = (a_1 x_{n-1} + \cdots + a_k x_{n-k}) \bmod m, \qquad u_n = x_n/m.$$

Matrix linear congruential generator:

$$\mathbf{x}_n = \mathbf{A}\mathbf{x}_{n-1} \bmod m, \qquad \mathbf{y}_n = \mathbf{B}\mathbf{x}_n \bmod m, \qquad \mathbf{u}_n = \mathbf{y}_n/m = (u_{nw}, \ldots, u_{nw+w-1})^{\mathrm{t}},$$

where $\mathbf{A}$ and $\mathbf{B}$ are $k \times k$ and $w \times k$ matrices.

Set of all possible vectors of t successive outputs,

$$\Psi_t = \{\mathbf{u} = (u_0, \ldots, u_{t-1}),\ \mathbf{x}_0 \in \mathbb{Z}_m^k\}.$$

We want the set Ψ_t to cover $[0,1]^t$ very evenly, for all t up to (say) 50 or so.

In fact, $\Psi_t = L_t \cap [0,1)^t$ where mL_t is an integral lattice in t dimensions ($\mathbb{Z}^t \subset L_t$).

Combined linear generators (e.g., sum mod 1) also have a lattice structure.

Is it good or bad to have a lattice structure? Discuss.

Is it good or bad to have a lattice structure? Discuss.

How to measure the quality of the lattice structure?

Lattice rules used for quasi-Monte Carlo (QMC) have exactly the same type of structure. The most popular figure of merit (FOM) used to measure their quality and to select specific parameters is a weighted $\mathcal{P}_\alpha$.

This weighted $\mathcal{P}_\alpha$ is very relevant because it provides error bounds for QMC and variance bounds for RQMC, for certain classes of smooth functions. But its computation time increases at least linearly with the cardinality of Ψ_t, which is the number n of points in the lattice rule.

For RNGs, this cardinality is huge, usually more than 2^{200} and often much more, so $\mathcal{P}_\alpha$ cannot be used. Then what can we do? We use the spectral test instead.

A basis for the lattice L_t is a set of t independent vectors $\mathbf{v}_1, \ldots, \mathbf{v}_t$ such that

$$L_t = \sum_{j=1}^{t} \mathbb{Z}\mathbf{v}_j = \left\{ \mathbf{v} = \sum_{j=1}^{t} z_j \mathbf{v}_j \mid \text{ each } z_j \in \mathbb{Z} \right\}.$$

The dual basis is the set of (integer) vectors $\mathbf{w}_1, \ldots, \mathbf{w}_t$ such that $\mathbf{w}_i \cdot \mathbf{v}_i = \delta_{i,j} = \mathbb{I}[i = j]$.
It is a basis of the dual lattice L_t^*.

A basis for the lattice L_t is a set of t independent vectors $\mathbf{v}_1, \ldots, \mathbf{v}_t$ such that

$$L_t = \sum_{j=1}^{t} \mathbb{Z}\mathbf{v}_j = \left\{ \mathbf{v} = \sum_{j=1}^{t} z_j \mathbf{v}_j \mid \text{ each } z_j \in \mathbb{Z} \right\}.$$

The dual basis is the set of (integer) vectors $\mathbf{w}_1, \ldots, \mathbf{w}_t$ such that $\mathbf{w}_i \cdot \mathbf{v}_i = \delta_{i,j} = \mathbb{I}[i = j]$.
It is a basis of the dual lattice L_t^*.

In our software, we work with the rescaled lattice $\Lambda_t = mL_t$, so all coordinates of all vectors are integer and can be represented exactly. The lattices Λ_t and L_t^* are m-dual to each other.

A basis for the lattice L_t is a set of t independent vectors $\mathbf{v}_1, \ldots, \mathbf{v}_t$ such that

$$L_t = \sum_{j=1}^{t} \mathbb{Z}\mathbf{v}_j = \left\{ \mathbf{v} = \sum_{j=1}^{t} z_j \mathbf{v}_j \mid \text{ each } z_j \in \mathbb{Z} \right\}.$$

The dual basis is the set of (integer) vectors $\mathbf{w}_1, \ldots, \mathbf{w}_t$ such that $\mathbf{w}_i \cdot \mathbf{v}_i = \delta_{i,j} = \mathbb{I}[i = j]$.
It is a basis of the dual lattice L_t^*.

In our software, we work with the rescaled lattice $\Lambda_t = mL_t$, so all coordinates of all vectors are integer and can be represented exactly. The lattices Λ_t and L_t^* are m-dual to each other.

Spectral test. The (Euclidean) length ℓ_t of a shortest nonzero vector in the dual lattice is the inverse of the distance between successive hyperplanes that contain all the points of L_t. With the L^1-norm, the shortest dual vector length gives the minimal number of hyperplanes that contain all the points of Ψ_t. We may also look at the length d_t of the shortest vector in the primal lattice.

We want all these lengths to be not too small. But there are upper bounds on the best possible values: for a lattice of density η in t dimensions, $d_t \leq d_t^*(\eta) := \gamma_t^{1/2} \eta^{-1/t}$ where the γ_t are the Hermite constants (known exactly for $t \leq 8$ and approximately otherwise).

To compare across different densities and dimensions, we standardize the lengths to values in $(0, 1]$ by taking $d_t/d_t^*(\eta)$ in the primal lattice and $\ell_t/d_t^*(1/\eta)$ in the dual.

Non-successive outputs

In addition to vectors of successive outputs $(u_0, \ldots, u_{t-1})$, we can also look at vectors of outputs $(u_{i_1}, \ldots, u_{i_s})$ for arbitrary lags $I = \{i_1, \ldots, i_s\} \subseteq \{1, \ldots, t\}$. These vectors generate a lattice L_I which is the projection of L_t over the set of coordinates in I.

To define a figure of merit (FOM), we can select a class $\mathcal{J}$ of subsets I, compute the normalized uniformity measure for each, and take the (possibly weighted) worst case (minimum) as a figure of merit (FOM):

$$M_{\mathcal{J}} = \min_{I \in \mathcal{J}} \omega_I \ell_I / d^*_{|I|}(1/\eta_I).$$

The set $\mathcal{J}$ can contain pairs, triples, etc., whose indices are not no far apart, e.g.:

$$M_{t_1,\ldots,t_d} = \min \left[\min_{I \in S(t_1)} \omega_I \ell_I / \ell^*_I, \min_{2 \le s \le d} \min_{I \in S(s,t_s)} \omega_I \ell_I / \ell^*_I \right],$$

where

$$\begin{aligned} S(t_1) &= \{I = \{1, \ldots, s\} \mid d+1 \le s \le t_1\}, \\ S(s, t_s) &= \{I = \{i_1, \ldots, i_s\} \mid 1 = i_1 < \cdots < i_s \le t_s\}. \end{aligned}$$

Multiply-with-carry (MWC) generators

Let $b \geq 2$ and $a_0, \ldots, a_k$ arbitrary integers such that $\gcd(a_0, b) = 1$, $a_0^* = a_0^{-1} \bmod b$.
MWC of order k in base b: if current state is $\mathbf{x}_{n-1} = (x_{n-1}, \ldots, x_{n-k}, c_{n-1})$, we put

$$\tau_n = a_1 x_{n-1} + \cdots + a_k x_{n-k} + c_{n-1}, \qquad x_n = a_0^* \tau_n \bmod b, \qquad c_n = \lfloor (\tau_n - a_0 x_n)/b \rfloor,$$
$$\mathbf{x}_n = (x_n, \ldots, x_{n-k+1}, c_n), \qquad u_n = \sum_{\ell=1}^{\infty} x_{n+\ell-1} b^{-\ell}.$$

We take b as a power of 2 (for speed) and truncate the last sum to one term or two.

The MWC is equivalent to an LCG with modulus $m = -a_0 + \sum_{\ell=1}^{k} a_\ell b^\ell$ and multiplier $b^* = b^{-1} \bmod m$. It is as a clever way to implement an LCG with a very large modulus.

We can analyze their lattice structure by analyzing the corresponding (large) LCG.

Multiply-with-carry (MWC) generators

Let $b \geq 2$ and $a_0, \ldots, a_k$ arbitrary integers such that $\gcd(a_0, b) = 1$, $a_0^* = a_0^{-1} \bmod b$.
MWC of order k in base b: if current state is $\mathbf{x}_{n-1} = (x_{n-1}, \ldots, x_{n-k}, c_{n-1})$, we put

$$\tau_n = a_1 x_{n-1} + \cdots + a_k x_{n-k} + c_{n-1}, \qquad x_n = a_0^* \tau_n \bmod b, \qquad c_n = \lfloor (\tau_n - a_0 x_n)/b \rfloor,$$
$$\mathbf{x}_n = (x_n, \ldots, x_{n-k+1}, c_n), \qquad u_n = \textstyle\sum_{\ell=1}^{\infty} x_{n+\ell-1} b^{-\ell}.$$

We take b as a power of 2 (for speed) and truncate the last sum to one term or two.

The MWC is equivalent to an LCG with modulus $m = -a_0 + \sum_{\ell=1}^{k} a_\ell b^\ell$ and multiplier $b^* = b^{-1} \bmod m$. It is as a clever way to implement an LCG with a very large modulus.

We can analyze their lattice structure by analyzing the corresponding (large) LCG.

The largest possible period length is $m - 1$, reached when m is prime and b^* is primitive mod m. Goresky and Klapper (2003) have listed such max-period generators, but their lattice structure was never examined.

If $a_0 = 1$, then $b^* = (m+1)/b$, $x_n = \tau_n \bmod b$ and $c_n = \tau_n \operatorname{div} b$. Easier to implement, but then the max period is $(m-1)/2$, attained iff $(m-1)/2$ is odd and 2 is primitive mod m. This holds for instance when $(m-1)/2$ is prime. We want to search for good parameters of this type, say for $b = 2^{32}$ and $b = 2^{64}$.

If $a_0 = 1$, then $b^* = (m+1)/b$, $x_n = \tau_n \bmod b$ and $c_n = \tau_n \operatorname{div} b$. Easier to implement, but then the max period is $(m-1)/2$, attained iff $(m-1)/2$ is odd and 2 is primitive mod m. This holds for instance when $(m-1)/2$ is prime. We want to search for good parameters of this type, say for $b = 2^{32}$ and $b = 2^{64}$.

Marsaglia and Zaman (1991) initially proposed add-with-carry and subtract-with-borrow generators, which are MWC with $a_0 = \pm a_r = \pm a_k = 1$ for some $r < k$ and the other $a_j = 0$. But then it was shown that the nonzero points (u_{n-k}, u_{n-r}, u_n) for the equivalent LCG lie in only two parallel planes!
Better MWC generators can be constructed by selecting more general coefficients.

Goal: search for MWC constructions that are fast and have good uniformity properties.

If $a_0 = 1$, then $b^* = (m+1)/b$, $x_n = \tau_n \bmod b$ and $c_n = \tau_n \operatorname{div} b$. Easier to implement, but then the max period is $(m-1)/2$, attained iff $(m-1)/2$ is odd and 2 is primitive mod m. This holds for instance when $(m-1)/2$ is prime. We want to search for good parameters of this type, say for $b = 2^{32}$ and $b = 2^{64}$.

Marsaglia and Zaman (1991) initially proposed add-with-carry and subtract-with-borrow generators, which are MWC with $a_0 = \pm a_r = \pm a_k = 1$ for some $r < k$ and the other $a_j = 0$. But then it was shown that the nonzero points (u_{n-k}, u_{n-r}, u_n) for the equivalent LCG lie in only two parallel planes!
Better MWC generators can be constructed by selecting more general coefficients.

Goal: search for MWC constructions that are fast and have good uniformity properties.

These RNGs can be very fast on recent hardware, by taking either $b = 2^{32}$ or $b = 2^{64}$. In particular, most GPUs support only 32-bit integers, but one can multiply two 32-bit integers and extract the low32 and high32 parts separately, in one instruction. This can compute x_n and c_n efficiently when $b = 2^{32}$.

What LatMRG is supposed to do

Testing a given generator. We may want to check if the RNG has maximal period, or apply the spectral test in some way, compute a FOM, etc.

Searching for good parameters. For a given type of construction, given modulus m and order k, perhaps with constraints on the coefficients, we want to search for RNGs that minimize a given FOM.

There is a very flexible executable program that can analyze an RNG or make a search. The input parameters are given in a `xml` file.

At a lower level, LatMRG is a C++ library that contains a few dozen classes that can construct the appropriate lattices for selected classes of linear RNGs, and search for good RNGs. It uses NTL and Lattice Tester.

History of LatMRG

First version written in Modula-2 around 1988. (Earlier Pascal program in 1985.)
We used our own package (SENTIERS) for arithmetic with arbitrary large integers.
Paper on LatMRG in INFORMS JOC (1997). Was used for many of my RNG papers.

History of LatMRG

First version written in Modula-2 around 1988. (Earlier Pascal program in 1985.)
We used our own package (SENTIERS) for arithmetic with arbitrary large integers.
Paper on LatMRG in INFORMS JOC (1997). Was used for many of my RNG papers.

Modula-2 compilers became too hard to find. Richard Simard started translating in C++ around 2000, and several students did further pieces. SENTIERS was replaced by NTL.
We use templates for flexible types of integers and real numbers.

History of LatMRG

First version written in Modula-2 around 1988. (Earlier Pascal program in 1985.)
We used our own package (SENTIERS) for arithmetic with arbitrary large integers.
Paper on LatMRG in INFORMS JOC (1997). Was used for many of my RNG papers.

Modula-2 compilers became too hard to find. Richard Simard started translating in C++ around 2000, and several students did further pieces. SENTIERS was replaced by NTL.
We use templates for flexible types of integers and real numbers.

In 2014, David Munger split the old LatMRG in two parts: the current Lattice Tester (initially called `latcommon`) which provides facilities to analyze arbitrary integer lattices, and the new LatMRG built over Lattice Tester to test the lattice structure of linear RNGs.
Lattice Tester is also used by LatNet Builder.

Several students made changes to all this software until recently. Some algorithms have changed along the way. When I got into the code recently, I found it rather messy.

I am currently working on Lattice Tester and LatMRG intensively.

Lattice Tester

Basic objects: Integral lattices and their m-dual (also integral).

Main tasks:

A. Basis Construction. Given a set of vectors, find a basis for the generated lattice.

B. Find m-Dual Basis. Given a lattice basis, compute the corresponding m-dual basis.

C. Lattice Basis Reduction. Given a basis, find another basis whose vectors are nearly orthogonal or as short as possible in some sense. LLL-reduction, BKZ-reduction, Minkowski-reduction, etc.

D. Shortest Vector. Find shortest nonzero vector in the lattice, with L^2 or L^1 norm.

Design: Lattice Tester is a C++ library that contains several dozen classes to manipulate and reduce lattice bases, find shortest nonzero vectors, compute FOMs, etc. Also a lot of basic facilities of all kinds to operate on flexible (integer and float) types, vectors, matrices, interface NTL, etc.

Alternative algorithms

Example: When computing the FOM $M_{t_1,\ldots,t_d}$ for the dual lattice, for a large collection of projections, one must construct a basis for the projection from a set of generating vectors, then compute the m-dual basis, then find a shortest vector in it, for each projection.

We have a method that constructs a triangular basis, and a faster method that constructs an LLL-reduced (but not triangular) basis. But computing the m-dual is faster for the triangular basis than for the other one.

Timings for different methods, in basic clock units, for LCG with $m = 1048573$:

dim:	10	20	30	40	50
UppTri	859	857	1439	2064	**2776**
LLL, $\delta = 0.5$	62	201	424	653	**1001**
LLL, $\delta = 0.8$	212	1489	2977	4671	5342
LLL, $\delta = 0.9999$	115	1291	3088	4388	8208
mDualUT	212	1022	2971	6173	**10785**
mDual	1813	3444	8105	18546	**29830**

Also, computing a shortest vector using the branch-and-bound algorithm is faster if we apply more intensive pre-reductions via LLL or BKZ, but the latter takes time.
The optimal compromise depends on the dimension: in higher dimensions, we should work more on pre-reductions.

Timings for different methods, in basic clock units, for LCG with $m = 1073741827$:

pre-red.:	None	pairwise	LLL, 0.5	LLL, 0.8	LLL, 0.99999	BKZ
dim.						
10	2279295	103743	260	**238**	518	401
20	5937701	1934427	2523	**1331**	1595	3100
30			584351	90625	**16362**	18358
40			239600527	15685520	**2797533**	3390653

With no (or too weak) pre-reductions, the BB takes much more time and often just fails.

XML Input for LatMRG Executable

Conclusion

- This software is for (few) specialists who construct and analyze RNGs, not really for the general users of RNGs. I would like it to be useful for them.
- I am open to suggestions on how to make it better.
- If you want to contribute, let me know!

References

- P. L'Ecuyer and R. Couture, "An Implementation of the Lattice and Spectral Tests for Multiple Recursive Linear Random Number Generators", INFORMS J. on Computing, 9, 2 (1997), 206–217.
- P. L'Ecuyer and R. Couture, "LatMRG User's Guide: A Modula-2 software for the theoretical analysis of linear congruential and multiple recursive random number generators", `https://www-labs.iro.umontreal.ca/~lecuyer/myftp/papers/guide-latmrg-m2.pdf`, 2000.
- P. L'Ecuyer, P. Marion, M. Godin, and F. Puchhammer, "A Tool for Custom Construction of QMC and RQMC Point Sets," Monte Carlo and Quasi-Monte Carlo Methods 2020, A. Keller, Ed., Springer-Verlag, 2022, 51–70.
- P. L'Ecuyer and D. Munger, "Algorithm 958: LatticeBuilder: A General Software Tool for Constructing Rank-1 Lattice Rules", ACM Transactions on Mathematical Software, 42, 2 (2016), Article 15.
- P. L'Ecuyer, P. Wambergue, and E. Bourceret, "Spectral Analysis of the MIXMAX Random Number Generators," INFORMS Journal of Computing, 32, 1 (2020), 135–144.

Github sites:

https://github.com/umontreal-simul/ (current distributions)

https://github.com/pierrelecuyer/ (versions under reconstruction)